Helsinki University of Technology Laboratory for Theoretical Computer Science
Technical Reports 22
Teknillisen korkeakoulun tietojenkäsittelyteorian laboratorion tekninen raportti 22
Espoo 2007 HUT-TCS-B22

SECURITY TOPICS AND MOBILITY MANAGEMENT IN HIERARCHICAL AD HOC NETWORKS (SAMOYED): FINAL REPORT

Maarit Hietalahti, Mikko Särelä, Antti Tuominen, and Pekka Orponen

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY
TECHNISCHE UNIVERSITÄT HELSINKI
UNIVERSITE DE TECHNOLOGIE D'HELSINKI

Helsinki University of Technology Laboratory for Theoretical Computer Science
Technical Reports 22
Teknillisen korkeakoulun tietojenkäsittelyteorian laboratorion tekninen raportti 22
Espoo 2007 HUT-TCS-B22

SECURITY TOPICS AND MOBILITY MANAGEMENT IN HIERARCHICAL AD HOC NETWORKS (SAMOYED): FINAL REPORT

Maarit Hietalahti, Mikko Särelä, Antti Tuominen, and Pekka Orponen

Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory for Theoretical Computer Science

Teknillinen korkeakoulu
Tietotekniikan osasto
Tietojenkäsittelyteorian laboratorio

Distribution:
Helsinki University of Technology
Laboratory for Theoretical Computer Science
P.O.Box 5400
FI-02015 TKK, FINLAND
Tel. +358 9 451 1
Fax. +358 9 451 3369
E-mail: lab@tcs.tkk.fi
URL: http://www.tcs.tkk.fi/

© Maarit Hietalahti, Mikko Särelä, Antti Tuominen, and Pekka Orponen

ISBN 978-951-22-9186-1
ISSN 0783-540X

Multiprint Oy
Espoo 2007

ABSTRACT: This report presents a summary of the technical results of project SAMOYED (2003–2006). This three-year research project considered topics in hierarchical ad hoc networks, especially ad hoc access networks, with a focus on mobility management and security issues. Results of the project include techniques for establishing and maintaining connectivity in ad hoc access networks; a novel method for dynamic local clustering and cluster-based routing in ad hoc networks; and requirements analysis and design of security architectures for clustered ad hoc networks. This report consists of a general overview of the topics studied in the project, and reprints of five of the project's publications.

KEYWORDS: ad hoc networks, mobility management, hybrid networks, ad hoc access networks, session continuity, global connectivity, multi-homing, Host Identity Protocol, tactical networks, clustering, cluster-based routing, security architectures

Contents

1 Introduction and Overview **1**
- 1.1 Project synopsis . . . 1
- 1.2 The ad hoc network environment . . . 1
- 1.3 Hierarchical ad hoc networks . . . 2
- 1.4 Research issues . . . 2
- 1.5 Hybrid networks . . . 3
- 1.6 Clustering and routing . . . 3
- 1.7 Security . . . 3
- 1.8 Project publications . . . 4

2 Hybrid Networks **5**

3 Clustering and routing **7**
- 3.1 A novel clustering method . . . 7
- 3.2 Cluster-based routing . . . 7
  - Intra-cluster routing . . . 8
  - Inter-cluster routing . . . 8
  - Route discovery . . . 9
  - Data traffic . . . 9

4 Security **10**
- 4.1 Routing attacks . . . 10
  - Protecting against routing attacks . . . 11
  - Secure routing: Surveys and comparisons . . . 11
- 4.2 Stimulating cooperation . . . 11
  - Reputation systems . . . 12
  - Payment systems . . . 12
  - The value of cooperation mechanisms . . . 12
- 4.3 Managing trust relations, keys and certificates . . . 13
  - Managing PKI in ad hoc networks . . . 13
  - Example: certificate repositories . . . 13

References **16**

Appendices **18**

1 INTRODUCTION AND OVERVIEW

1.1 Project synopsis

SAMOYED was a three-year research project considering topics in hierarchical ad hoc networks, especially ad hoc access networks. The focus of the work was on mobility management and security issues. Funding for the project was provided, at the level of two full-time researchers, by the Finnish National Technology Agency TEKES (80%), L. M. Ericsson (10%), and the Finnish Defence Forces (10%). The project commenced in September 2003 and was completed in December 2006. The site of research was the Laboratory for Theoretical Computer Science at the Helsinki University of Technology TKK.

The research was supervised by Prof. Pekka Orponen, and the core research team consisted of M.Sc. (Tech.) Maarit Hietalahti and M. Sc. (Tech.) Mikko Särelä. During Maarit Hietalahti's maternity leave in the first half of 2006, the team was augmented by Stud. Tech. Antti Tuominen. During the project, Mikko Särelä completed an extended research visit to the Calit2 research centre at UCSD (12/2005–07/2006).

Other researchers whose work significantly contributed to the project were M. Sc. (Tech.) Tuulia Kullberg, M. Sc. (Tech.) Stefano Marinoni (also part-time employed by the project 06–12/2005), D. Sc. (Tech.) Satu Elisa Schaeffer and Doc. Pekka Nikander.

1.2 The ad hoc network environment

Ad hoc networks consist of (usually mobile and wireless) nodes that create and maintain their intercommunication links without the help of a pre-existing infrastructure. On top of this transitory physical layer, network services such as routing are provided. Lack of infrastructure means a lack of central entities such as fixed routers, name servers, etc. Additionally, parties involved in a communication across a network might not have any common history, which complicates the provision of services requiring trust or continuity

Maintaining connections in an ad hoc network is difficult, because the links are unreliable and the network topology is dynamic. Also the devices forming a network are often small and portable, with a limited battery-life. Therefore, they do not have much memory or computational power and they might not be tamper-resistant. Connections are formed by routing messages from point to point via other peer devices and not through dedicated router networks.

An ad hoc device can lose its connection to the rest of the network for several reasons: it can move out of the network's reach, it can run out of batteries or be compromised, or something similar can happen to the other devices that are connecting this device to the rest of the network. In general, node movements in a network can be erratic and unpredictable. Hence, the network topology may be very irregular and change rapidly. Links are few, so the network may even be partitioned occasionally.

1.3 Hierarchical ad hoc networks

Much of existing ad hoc network research focuses on the flat network model, where no assumptions are made regarding the network's structure and the movement of nodes with respect to this structure. Consequently, it is often claimed that ad hoc networks will experience severe scaling problems when the number of nodes increases.

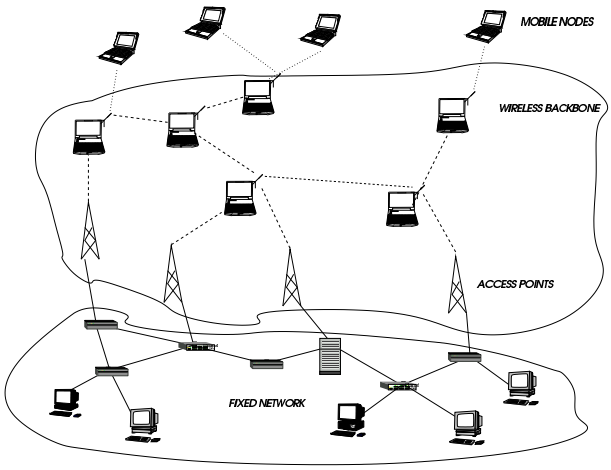


Figure 1: An ad hoc access network

In many real-life situations, however, networks have some kind of natural hierarchical structure that can be used to support their management. E.g. one of the most promising applications of ad hoc networks is to use them for providing connections of mobile nodes to a fixed network, as a so called *ad hoc access network* (Figure 1). Here the mobile nodes are organised so as to maintain contact to a supporting fixed network, and the static access points of the fixed network consequently induce a natural clustering among the mobile nodes. Similarly in e.g. a military network, a certain amount of structure based on troop organisation and movements can be expected. In such situations not only node locations, but also their movements may be correlated.

1.4 Research issues

General research issues in ad hoc networks arise from their characteristics that unlike in fixed networks, the connections are dynamic and unstable, nodes do not have common history, their computational capability is limited etc. The lack of stable infrastructure entails that mobility management, congestion control, routing, quality of service issues, security etc. must be handled locally and adaptively. Standard methods require global information on network state which is difficult to collect and rapidly outdated.

Hierarchical structure in a network may facilitate efficient protocols for these tasks, in particular if supported by connections to a fixed wired network. Work in the SAMOYED project focused on three specific aspects of hierarchical ad hoc networks: hybrid networks, especially mobility management in ad hoc access networks; dynamic local clustering and cluster-based routing in ad hoc networks; and security architectures for clustered ad hoc networks.

1.5 Hybrid networks

Hybrid networks consist of interconnected infrastructured and infrastructureless parts. These parts may be wired (and possibly large such as the Internet), wireless mesh networks, or wireless ad hoc networks. The main task is to combine these very different networks into an integrated system where each node can reach the other hosts, services and data they need and are entitled to. The most researched problem in this area is how to provide Internet access to nodes in an ad hoc network and vice versa. Such networks are often called *ad hoc access networks* in the literature.

Results of the SAMOYED project in this area include techniques for maintaining session continuity in ad hoc access networks [12], a multi-homed solution [20] to the global connectivity problem using the Host Identity Protocol [17], and designs for applying the Host Identity Protocol to tactical ad hoc networks [21]. An overview of this research area is presented in Section 2.

1.6 Clustering and routing

Hierarchical structures can be used to assist in the operation of an ad hoc network, e.g. by creating a hierarchical addressing scheme combined with node level mobility schemes such as Mobile IP [18]. Routing algorithms can take advantage of the hierarchical, clustered structure by using a different routing scheme inside a cluster and outside it.

A number of cluster-based routing algorithms for ad hoc networks have been discussed in the literature, e.g. [2] and [9]. Cluster-based routing methods approaches can be more efficient than flat topology based, as shown e.g. in [24]. Less work has, however, so far been done in the area of hierarchical network management techniques.

In the SAMOYED project, a novel dynamic cluster-forming protocol was developed [22]. The protocol is very simple, has low signalling overhead, uses only information locally available at the nodes, and produces dense and stable clusterings that are nevertheless responsive to node mobility. After validation by simulation studies, the clustering method was further developed into a prototype Linux implementation, together with an OLSR-based hierarchical routing scheme capable of taking advantage of the concomitant cluster structure. This research is discussed in Section 3.

1.7 Security

Distributed security is a difficult issue: reliance on centralised trusted entities must be minimised; trust relations must be independent of the place and network topology in which they were formed; and validation of trust relations must be local.

Hierarchical structure may simplify security schemes, if intra-cluster connections are more stable and/or reliable than inter-cluster ones. Protocols supporting the forming of groups, e.g. group key establishment, may be useful in this setting. On the other hand, in a clustered network the clusterhead nodes may create single weak points; this issue should be addressed together with mobility management.

In the area of network security, the research focus of the SAMOYED project was on trust management and security architectures in a distributed, but hierarchical ad hoc network environment. This work is summarised in the thesis [4], and briefly surveyed in Section 4 of this report.

1.8 Project publications

The results of the project have been disseminated in the conference papers [3, 12, 20, 21, 22] and theses [4, 13, 16, 19]. Copies of publications [12, 20, 21, 22] and [4] are included as appendices to this report.

Acknowledgments

The SAMOYED project team wishes to thank the many people whose help and participation was essential to the project's progress. We are especially grateful to our coworkers and coauthors Ms. Tuulia Kullberg, Mr. Stefano Marinoni, Doc. Pekka Nikander and Dr. Satu Elisa Schaeffer, as well as the project members' thesis supervisors and instructors Prof. Hannu H. Kari, Doc. Pekka Nikander (again) and Prof. Kaisa Nyberg. Finally, many thanks are due to members of the project's management team, Mr. Risto Määttä, Dr. Raimo Vuopionperä, Ms. Heli Kukko and Ms. Tiina Nurmi, for their consistent support and advice.

2 HYBRID NETWORKS

The first research focus area of the SAMOYED project concerned the construction and utilisation of hybrid networks. Such networks may arise, e.g., in military settings, disaster recovery, and novel grass-root end user networks. Examples include a navy network, where the topology across ships is dynamic and changing as ships move, but within ships relatively static, though it may experience sudden changes in battle due to destruction of equipment; and disaster relief communications, where some existing network infrastructure may remain, some be built on site, and some parts of the network may exist as ad hoc networks. Of special interest in the project were the issues of establishing and maintaining network connectivity in ad hoc access networks.

The problem of Internet access for ad hoc network hosts comprises several subproblems:

- locating an Internet Gateway
- gateway registration and AAA
- address configuration
- route maintenance to the Internet Gateway
- end-to-end mobility management.

The first problem for an ad hoc network node to solve is how to locate an Internet Gateway. The Globalv6 [26] draft proposes that for IPv6 based networks either the Internet Gateway periodically sends advertisements to the ad hoc network of the Internet access provided, or responds to requests from ad hoc nodes. These advertisements and responses contain the network prefix the ad hoc network node must use.

Address configuration is another problem in ad hoc networks. If the Internet Gateways support NAT, then it is possible to use only ad hoc network local addresses within the ad hoc network. Support for NAT must be advertisable. The possible future MANET prefix could be used as an indication of NAT support, as it would not be globally routable.

If the gateway does not support NAT, then ad hoc nodes must configure global addresses to communicate with nodes in the Internet. This requires learning the prefix, configuring the address and detecting address collisions.

In the case of several Internet Gateways within the ad hoc network, the ad hoc network is, in theory, a multi-homed network in the Internet. Each node could, if the addressing problems are solved, be reached from several places in the Internet topology.

Widespread use of ingress filtering makes it practically mandatory to route packets through a chosen Internet Gateway, though. On the other hand, routes in ad hoc networks are dynamic and, for some routing protocols, may change without the explicit knowledge of the endpoints. From the Internet endpoint's view, the name (IP address) of the other endpoint should not change without explicit name changing signalling (e.g. Binding Update in Mobile IPv6 [10] or readdressing signalling in Host Identity Protocol [17]. To solve this problem, the Globalv6 draft recommends using the routing header with the address of

the Internet Gateway for packets destined to the Internet, in order to force them through the chosen Internet Gateway.

The Host Identity Protocol [17] can be used to enable ad hoc network nodes to switch the Internet Gateway they are using. As it supports multihoming, an ad hoc network node can obtain several points of entry to the Internet and be reachable from all of them at the same time. Potentially, this can be used to make repairing a connection faster, as a new global address is already available when an old address dies (due to gateway losing route to ad hoc network node).

In a scenario with several separate fixed networks connected by ad hoc networks, one of the main problems is to achieve connectivity. If the networks are truly separate, then one cannot assume a common addressing structure, nor can one assume that there are no naming conflicts, unless the address space is large enough (and randomisation used) to make that highly improbable. This kind of network, created as it grows from the parts it contains should have a mechanism for different parts finding out each other and a common naming format that can be used for inter-network connections.

In the publications appended to this report, article [12] develops techniques for maintaining session continuity in ad hoc access networks, article [20] presents a multi-homed solution to the global connectivity problem using the Host Identity Protocol, and article [21] discusses the advantages of the Host Identity Protocol in the setting of tactical ad hoc networks.

3 CLUSTERING AND ROUTING

The second research focus of the SAMOYED project was dynamic local clustering of nodes and cluster-based routing in ad hoc networks.

Hierarchical, cluster-based routing can reduce routing table sizes when using proactive (table-based) routing protocols, as compared to flat host routes. The motivation for clustering is to reduce and localise signalling messages, as well as to maintain groups of nodes which can be addressed hierarchically, as opposed to flat addressing.

In a highly dynamic environment, maintaining hierarchy is a challenge. To be effective, a clustering algorithm should be able to minimise changes to the cluster (subnet) composition. Topology changes within a cluster are not very important, but because of hierarchical addressing, changes to cluster memberships are. Moving from one cluster to another affects the node's address. Frequent address changes are undesirable even if the address is only used for routing, and even more so if it is also used for identification.

3.1 A novel clustering method

A new locally computable clustering method was introduced and studied in the SAMOYED project [22]. Local computability removes the need to disseminate cluster information beyond the nodes' immediate neighbourhoods. A node can decide to join, leave, or create a cluster based on the information received from its one-hop neighbours only. The clustering protocol maximises, by local computations, for each cluster C of nodes a global fitness function $f(C)$ conceived as the product of the internal density of the cluster (large number of intra-cluster connections relative to the cluster size $|C|$) and its "introversion" (high fraction of all connections for C are internal to C). The explicit expression for the resulting objective function is:

$$\begin{aligned} f(C) &= \frac{\deg_{\text{int}}(C)}{\binom{|C|}{2}} \cdot \frac{\deg_{\text{int}}(C)}{(\deg_{\text{int}}(C)+\deg_{\text{ext}}(C))} \\ &= \frac{2\deg_{\text{int}}(C)^2}{|C|\,(|C|-1)(\deg_{\text{int}}(C)+\deg_{\text{ext}}(C))}. \end{aligned}$$

Here $\deg_{\text{int}}(C)$ and $\deg_{\text{ext}}(C)$ denote the number of intra-cluster and inter-cluster connections of node set C, respectively.

Maximising the fitness function $f(C)$ can be performed by purely local computations, and produces dense and stable clusters, thereby minimising address changes.

3.2 Cluster-based routing

Clustering alone does not provide routing information; a routing protocol must be used. Since most mobile ad-hoc network routing protocols employ some sort of beacon messages for neighbour sensing, one can embed all the cluster information in these beacon messages. This way clustering does not add to signalling overhead.

Cluster-based routing can be divided into two layers: intra-cluster and inter-cluster routing. Since these two can operate independent of each other,

one can choose different routing protocols for them, or use the same protocol for both, if desired. It may be approriate to choose different protocols, since requirements for intra- and inter-cluster routing may well be different. Also, route optimisation may be easier separately.

Intra-cluster routing

Proactive (table-based) routing gains the most from clustering. Reactive (on-demand) routing establishes routes when they are needed, so in most cases the number of routes maintained by a single node is relatively small. Proactive protocols try to maintain routes to all nodes in the network at all times. With clustering, the network is partitioned into smaller pieces. So, even if routes are maintained to all nodes within a given cluster, the number of routes per node stays relatively small.

For the purposes of the SAMOYED project, OLSR was chosen as the intra-cluster routing protocol. OLSR works well with relatively small and dense clusters, such as produced by the clustering method considered. OLSR could easily be used as the only protocol for both intra- and inter-cluster routing, but here it is used only for routing within a cluster.

By way of hearing neighbour cluster routing beacons, a cluster can determine all its neighbouring clusters. Neighbour clusters can be added to routing tables as prefix routes. This will be useful in the route discovery phase.

Inter-cluster routing

The main benefit of clusters, as to routing, is the ability to have hierarchical addressing. One can point to a cluster by just its ID. Instead of next-hop routing (i.e. sending a packet towards the final destination by forwarding it to the next hop on the path), one can have next-cluster routing. Or if using source routing (i.e. sending packets with full path information), one can use cluster IDs as intermediate destinations. Obviously, the inter-cluster routing protocol can disregard any topology changes that occur within clusters. This is a great advantage, since it will likely result in longer route lifetimes.

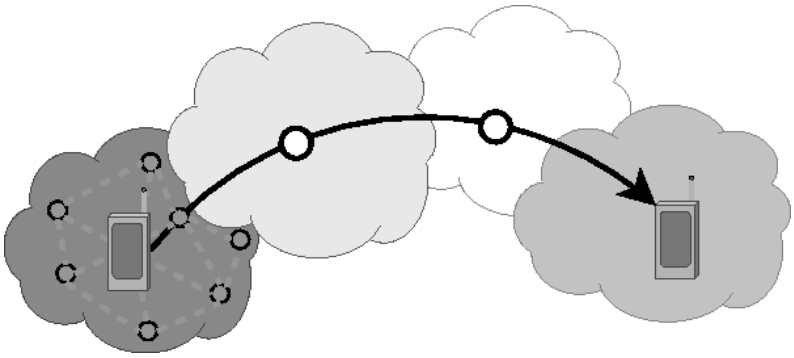


Figure 2: Inter-cluster route with intermediate clusters

Fig. 2 illustrates routes known to a node. In the local cluster, all routes are known. A route to a node in another cluster contains just a list of the intermediate clusters to get to the final destination. No information about the topology internal to the intermediate clusters is needed.

Route discovery

When a proactive routing protocol is used, each node has routes to other nodes. In this case route discovery consists more of a mapping of identity to an address. If node identity is (part of) the address, route discovery is essentially superfluous. In current IP networks, an address has the dual nature of being a topology based routing token as well as an ID for the node.

Reactive routing protocols use route discovery whenever a new route is needed. If the route is previously unknown, or it has expired, a route request is issued. Normally route requests are flooded in the network. Since one is here using a reactive protocol for inter-cluster routing, route requests are not flooded inside a cluster, but rather forwarded to each neighbouring cluster. Neighbouring clusters are recorded as prefix routes (discovered from beacons) in the routing table, as previously described.

As a result of the intra-cluster routing protocol operation, a node receiving a route request from another cluster knows whether the requested destination is in the current cluster. If it is, the node can just send a route reply to the requester. Otherwise, the node can forward the request to the neighbouring clusters.

The original requester will add the newly discovered route to its routing table when it receives the route reply. If multiple routes are discovered, the best one should be used. Route metrics for evaluating the best route may be just about anything. Hop-count distance is commonly used, although in this case, inter-cluster routes should not record hop counts, but rather cluster counts.

Data traffic

A node sending data packets to another node in the same cluster does not need any extra procedures. It just sends data towards the final destination. If the destination is in a neighbouring cluster, and the source node has succesfully discovered the route, sending data works the same way as in a local cluster. When the data must go through intermediate clusters, the source node must include the cluster ID path to the final destination.

4 SECURITY

The third research domain of the SAMOYED project was security architectures for clustered ad hoc networks. Research in this area is fully reported in the thesis manuscript [4] appended to this report. Here we present a summary of the main issues.

In mobile wireless ad hoc networks, nodes form connections among themselves without the help of pre-existing routers or other similar services. The networks are self-organised, so that users need not be concerned with network management. The same applies for security operations, apart from managing the (physical) security of the ad hoc devices and what the equipments' access control demands (remembering passphrases, for example). Overlay networks and sensor networks are generally out of the scope of this work, although some solutions are applicable also in those fields.

Reliability of communications is an especially important security issue in ad hoc networks. Cooperation of the nodes is necessary in order to have packets forwarded. One also needs to consider notions of distributed security. No particular node can be expected to be reachable at all times. Therefore, the need for on-line contacts to central entities has to be minimised. In addition, nodes should be able to perform the computations needed for most critical network operations by using only the information provided within their neighbourhoods (locally computable solutions). The goal is to enable communications in a secure way, from enabling packet forwarding and secure routing to establishing trust relations and encryption keys.

4.1 Routing attacks

In ad hoc networks where nodes themselves act as routers, compromised nodes can seriously hinder the operations of the network by not cooperating in, or abusing the routing of messages. If routing tables are forged, packets will not reach their destinations. The same can happen if message headers are tampered with. Routing information may be *spoofed*, *altered* or *replayed*. A malicious node can use the faults in a routing protocol to attract much traffic from a particular area, thus creating a *sinkhole*. It can *selectively drop* only certain packets and thus focus the attack on a selected part of the network or selected messages. This is especially effective when the malicious node is in a sinkhole.

In a *Sybil attack*, a malicious node gathers several identities for posing as a group of many nodes instead of one. This is targeted to undermine solutions that rely on the cooperation of multiple nodes.

In a *wormhole attack*, the idea is to distort routing with the use of a low-latency out-of-bound channel to another part of the network where messages are replayed. In a *HELLO flood attack*, a malicious node can send, record or replay HELLO-messages with high transmission power. If a protocol uses *link-layer acknowledgements*, these acknowledgements can be forged, so that other nodes believe a weak link to be strong or disabled nodes alive. These attacks can be used to create a sinkhole, or to otherwise confuse routing.

The above attacks are described in [11, 1, 6].

Protecting against routing attacks

Several solutions have been proposed for protecting against above mentioned routing attacks. For example, article [11] suggests link layer encryption and authentication with a common symmetric key for preventing most outsider attacks. Replay attacks are often dealt with by an increasing counter; time stamps are also possible. Protecting against insider attacks is challenging. An insider cannot be prevented from participating in the operations of a network. It is suggested in [11] that nodes should share individual unique symmetric keys with the base station and that the number of neighbours per node should be limited (a node cannot form symmetric keys with too many other nodes).

HELLO flood attacks are suggested to be dealt with by verifing the bi-directionality of the link. For wormhole attacks, article [11] suggests geographic routing which brings forth another problem: should one trust the advertised location information? Another solution [7] is to compare the time it takes for a packet to travel from the sender's geographical location to the time stamp attached to the message.

In multipath routing, several paths are used simultaneously for communication, for example by sending partial messages via different paths. When redundancy is added to data, the division can be done so that the full message can be reconstructed from even partially received data. The main objective is tolerating packet loss, rather than detecting and isolating malicious nodes.

When isolation of malicious nodes is needed, a reputation system (See subsection 4.2) can be combined to routing. Isolation is done by accepting communications only from nodes with valid credentials. However, monitoring other nodes' behavior becomes very inefficient when nodes have high mobility.

Secure routing: Surveys and comparisons

A good survey on secure routing in ad hoc networks is presented in [5]. More specific security surveys can be found, for example, in [27] and [15].

4.2 Stimulating cooperation

Availability of network services is an important part of security in ad hoc networks. Having one's packets forwarded to their destination does not only depend on securing the routing against malicious attackers, but also on the cooperation of other nodes on the route.

In open networks, where nodes are individuals acting for their own best interest, the nodes are thought to be selfish, due to their limited battery-life. They are likely to save their limited energy and they may not be willing to forward packets on the benefit of others, unless otherwise motivated. Therefore, packet forwarding has to be stimulated, assuming that the goal of the nodes is to send as much of their own packets as they can, nevertheless making sure that packets from other nodes will also be forwarded. The same problem has consequences also when nodes are not independent, but part of a common organisation, as conserving energy is a common topic in every ad hoc network that is composed of small devices with limited batteries. Solutions that aim to divide the packet-forwarding load fairly between the nodes will probably also be useful in saving the total energy of the network.

Reputation systems

In a *reputation system* other nodes form an opinion on a node's behaviour. This opinion has a direct effect on how high a priority the node's packets will receive, and how much packets will be routed via the node. A node's assessed behaviour can include its:

- will to forward other nodes' packets,
- will to take part in other common activities,
- not disrupting communications,
- not compromising common secrets.

A reputation system may give an inside attacker efficient means for making DoS attacks using falsified reports of bad behaviour. Sometimes it is difficult to sort out the malicious nodes from their victims. Reputation systems should protect themselves against such misuse.

A reputation system is being used, for example, in article [28] in connection with the AODV routing protocol. The nodes need a valid token in order to have their messages forwarded. The neighbourhood verifies a token, monitors the node's behaviour and decides whether the token should be renewed. The neighbours sign a new token with the system secret using a threshold scheme. Collaboration among the attackers is assumed to be limited to fewer than a threshold value of k attackers per neighbourhood. There is a decreasing overhead over time: the lifetime of a token is extended every time it is renewed. The solution is localised. However, this system may be vulnerable to the Sybil attack, and the neighbourhood is expected to be very stable, renewing the same node's token repeatedly. Therefore it becomes inefficient with high node mobility.

Payment systems

This is a system where nodes trade tokens, or essentially any sort of payments, against packet forwarding services. Usually, tokens are collected for data forwarding only, not for route discovery or control messaging. A node might participate in route discovery, but forward data selectively, thus creating a *grey hole*. This is not explicitly punished, because such behaviour causes the node to lose income. A payment system avoids judgements on a node's behaviour. Therefore it also avoids the sometimes complicated management of reputation and trust issues. The sender will pay tokens for the forwarding nodes in the path, hence the route should be previously known, or estimated in some way, so that the trading can take place. This is obviously easier with proactive routing protocols.

Trading tokens is similar to virtual currency systems, which usually means that extra management of the tokens on the market is needed. More discussion of the advantages and disadvantages of different virtual currency systems can be found in the economics literature.

The value of cooperation mechanisms

Article [14] estimates the throughput with different forwarding probabilities in a sparse network in conjunction with on-demand routing. It is assumed that

all nodes have the same forwarding/dropping ratio. Individual participation is estimated with forwarding probability pairs (with and without a cooperation method), and the resulting global throughput is calculated. It was found that in medium- and large-scaled networks with long routes, the effect of increased participation is low. However, in small ad hoc networks with short average route length and certain forwarding probability pairs, there is improvement in the overall throughput.

Promoting cooperation has received much attention in the field of game theory. A model for reciprocal behaviour, Tit-for-tat (TFT), and its more generous version, Generous tit-for-tat (G-TFT) are studied in [23]. The work demostrates that under an energy constraint G-TFT promotes cooperation if every node of the network conforms to it (Nash equilibrium).

In order to apply a reputation system in a self-organised fashion, automatic detection and evaluation of attacks is needed. For example, article [8] defines a taxonomy of basic and anomalous events in routing, and applies these to AODV with the help of a finite state automaton. Automatic attack detection is a complicated task; some methods may be found in the literature on intrusion detection.

4.3 Managing trust relations, keys and certificates

This work concentrates on the relationships between devices. Many types of trust relations can be formed in ad hoc networks. Building security associations between nodes can be done with the help of some initial basis of trust, a certificate authority (CA), or pre-distributed keys or shared secrets.

Whatever the initial trust relationship between nodes is, there are many constructs in the literature for maintaining old and forming new trust relations. There is a lot of literature on trust relations that is not related to hierarchical ad hoc networks, but can be applied also in this context.

Managing PKI in ad hoc networks

A Public-Key Infrastructure (PKI) provides identifiers for each node (public-private key pairs) and a way to authenticate these identifiers (certificate of a node's public key, signed by a CA, whose public key is known to everyone).

A PKI may be previously constructed. This is possible for a network having a single administrative unit. Some solutions are presented in the literature for bootstrapping a PKI. A PKI is relatively distributed: public keys and certificates can be stored and used locally. Connections to a central entity, the CA, are sometimes needed, for example for revocation of certificates and adding new nodes.

Example: certificate repositories

In [25] an interesting distributed public-key management system is presented. Every node can issue certificates. A node keeps an updated *certificate repository* and a non-updated certificate repository. The problem is to find a valid certificate chain from the repositories. The updated repositories are preferred, but if a certificate path can only be found using non-updated data, an update is requested on-line.

REFERENCES

[1] J. Douceur. The Sybil attack. In *Proceedings of the International Workshop on Peer–to–Peer Systems (IPTPS)*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260, Cambridge, MA, USA, March 2002. Springer.

[2] Z. Haas and M. Pearlman. Providing ad hoc connectivity with the reconfigurable wireless networks. In *Proceedings of the SIGCOMM Conference on Communications Architectures, Protocols and Applications*, Vancouver, Canada, September 1998. ACM.

[3] M. Hietalahti. Cooperation in clustered ad hoc networks. In *Proceedings, 5th Scandinavian Workshop on Wireless Ad-hoc Networks (ADHOC)*, Stockholm, Sweden, May 2005. Poster.

[4] M. Hietalahti. Requirements for a security architecture for clustered ad hoc networks. Manuscript of Lic.Sc. (Tech.) thesis, Helsinki University of Technology TKK, Department of Computer Science and Engineering, June 2007. 4+69 pp.

[5] Y.-C. Hu and A. Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy*, pages 28–39, May/June 2004.

[6] Y.-C. Hu, A. Perrig, and D. B. Johnson. Wormhole detection in wireless ad hoc networks. Technical Report TR01-384, Department of Computer Science, Rice University, December 2001. 15 pp.

[7] Y.-C. Hu, A. Perrig, and D. B. Johnson. Packet leashes: A defence against wormhole attacks in wireless ad hoc networks. In *Proceedings, IEEE Conference of the Computer and Communications Societies (INFOCOM)*, volume 3, pages 1976–1986, San Francisco, CA, USA, April 2003.

[8] Y. Huang and W. Lee. A cooperative intrusion detection system for ad hoc networks. In *Proceedings, Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, pages 135–147, Fairfax, VA, USA, October 2003. ACM Press.

[9] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen. Scalable routing strategies for ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1369–1379, August 1999.

[10] D. Johnson, C. Perkins, and J. Arkko. Mobility support in IPv6. Request for Comments RFC 3775, IETF Network Working Group, Internet Engineering Task Force, June 2004. 163 pp.

[11] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad Hoc Networks, Special Issue on Sensor Network Applications and Protocols*, 1(2-3):293–315, September 2003.

[12] T. Koponen, P. Eronen, and M. Särelä. Resilient connections for SSH and TLS. In *Proceedings, USENIX Annual Technical Conference*, pages 329–340, Boston MA, USA, June 2006. USENIX Association.

[13] T. Kullberg. The effect of the access point selection method on reachability between a mobile ad hoc node and a fixed node. Master's thesis, Helsinki University of Technology TKK, Department of Computer Science and Engineering, May 2005. 9+60 pp.

[14] B. Lamparter, M. Plaggemeier, and D. Westhoff. Estimating the value of co–operation approaches for multi–hop ad hoc networks. *Ad Hoc Networks*, 3(1):17–26, January 2005.

[15] P. Lungaro. Cost/performance trade–offs in "two–layer" ad hoc multihop cellular access systems. In *Proceedings, 4th Scandinavian Workshop on Wireless Ad Hoc Networks (ADHOC)*, Stockholm, Sweden, May 2004. 7 pp.

[16] S. Marinoni. Performance of wireless ad hoc routing protocols – a simulation study in realistic environments. Master's thesis, Helsinki University of Technology TKK, Department of Computer Science and Engineering, May 2005. 11+80 pp.

[17] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host Identity Protocol. Internet Draft draft-ietf-hip-base-08, Internet Engineering Task Force, June 2007. 105 pp.

[18] C. Perkins. IP mobility support. Request for Comments RFC 2002, Internet Engineering Task Force, October 1996. 78 pp.

[19] M. Särelä. Measuring the effects of mobility on reactive ad hoc routing protocols. Master's thesis, Helsinki University of Technology TKK, Department of Computer Science and Engineering, April 2004. 14+60 pp.

[20] M. Särelä. Multi-homed Internet access in ad hoc networks using Host Identity Protocol. In *Proceedings, 4th ESF Workshop on Middleware for New and Emerging Mobile Applications (MiNEMA)*. Departamento de Informática, Universidade de Lisboa, July 2006. 5 pp.

[21] M. Särelä and P. Nikander. Applying Host Identity Protocol to tactical networks. In *Proceedings, Military Communications Conference (MILCOM)*, volume 2, pages 834–840, Monterey CA, USA, November 2004. IEEE Communications Society.

[22] S. E. Schaeffer, S. Marinoni, M. Särelä, and P. Nikander. Dynamic local clustering for hierarchical ad hoc networks. In *Proceedings, IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), International Workshop on Wireless Ad-hoc and Sensor Networks (IWWAN) subtrack*, volume 2, pages 667–672. IEEE Communications Society, June 2006.

[23] V. Srinivasan, P. Nuggehalli, C. Chiasserini, and R. Rao. Cooperation in wireless ad hoc networks. In *Proceedings, IEEE Conference of the Computer and Communications Societies (INFOCOM)*, volume 2, pages 808–817, San Francisco, CA, USA, April 2003.

[24] J. Sucec and I. Marsic. Clustering overhead for hierarchical routing in mobile ad hoc networks. In *Proceedings, IEEE Conference of the Computer and Communications Societies (INFOCOM)*, volume 3, pages 1698–1706, New York, USA, June 2002.

[25] S. Čapkun, L. Buttyán, and J.-P. Hubaux. Self–organized public–key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):52–64, Jan–Mar 2003.

[26] R. Wakikawa, J. T. Malinen, C. Perkins, A. Nilsson, and A. J. Tuominen. Global connectivity for IPv6 mobile ad hoc networks. Internet Draft draft-wakikawa-manet-globalv6-05.txt, Internet Engineering Task Force, March 2006. 28 pp.

[27] W. Wang, Y. Lu, and B. Bhargava. On security study of two distance vector routing protocols for mobile ad hoc networks. In *Proceedings, 1st IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 179–186, Dallas–Fort Worth, TX, USA, March 2003.

[28] H. Yang, X. Meng, and S. Lu. Self–organized network layer security in mobile ad hoc networks. In *Proceedings, ACM Workshop on Wireless Security (WiSe)*, pages 11–20, Atlanta, Georgia, USA, September 2002.

APPENDICES

Reprints of the following project publications are appended to this report:

[12] T. Koponen, P. Eronen, M. Särelä, "Resilient connections for SSH and TLS". *Proceedings, The 2006 USENIX Annual Technical Conference (Boston MA, USA, June 2006)*, 329–340. © 2006 USENIX Association, Berkeley CA, USA. Reprinted with permission.

[20] M. Särelä, "Multi-homed Internet access in ad hoc networks using Host Identity Protocol". *Proceedings, 4th ESF Workshop on Middleware for New and Emerging Mobile Applications (MiNEMA'06, Lisbon, Portugal, July 2006)*. Departamento de Informática, Universidade de Lisboa, 2006.

[21] M. Särelä, P. Nikander, "Applying Host Identity Protocol to tactical networks". *Proceedings, IEEE Military Communications Conference (MILCOM'04, Monterey CA, USA, November 2004)*, Vol. 2, 834–840. © 2004 IEEE Communications Society, New York NY, USA. Reprinted with permission.

[22] S. E. Schaeffer, S. Marinoni, M. Särelä, P. Nikander, "Dynamic local clustering for hierarchical ad hoc networks". *Proceedings, IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'06), International Workshop on Wireless Ad-hoc and Sensor Networks (IWWAN'06) subtrack (New York NY, USA, June 2006)*, Vol. 2, 667–672. © 2006 IEEE Communications Society, New York NY, USA. Reprinted with permission.

[4] M. Hietalahti, Requirements for a Security Architecture for Clustered Ad Hoc Networks. Manuscript of Lic.Sc. (Tech.) Thesis. Helsinki University of Technology TKK, Department of Computer Science and Engineering, August 2007. 4+69 pp. *Note:* Work described in Chapter 5.4 of the thesis was supported by project "Ad-Hoc Networks", funded by the Finnish Defence Forces.

HELSINKI UNIVERSITY OF TECHNOLOGY LABORATORY FOR THEORETICAL COMPUTER SCIENCE
TECHNICAL REPORTS

ISBN 978-951-22-9186-1
ISSN 0783-540X

Resilient Connections for SSH and TLS

Teemu Koponen
Helsinki Institute for Information Technology
teemu.koponen@hiit.fi

Pasi Eronen
Nokia Research Center
pasi.eronen@nokia.com

Mikko Särelä
Helsinki University of Technology
Laboratory for Theoretical Computer Science
id@tcs.hut.fi

Abstract

Disconnection of an SSH shell or a secure application session due to network outages or travel is a familiar problem to many Internet users today. In this paper, we extend the SSH and TLS protocols to support resilient connections that can span several sequential TCP connections. The extensions allow sessions to survive both changes in IP addresses and long periods of disconnection. Our design emphasizes deployability in real-world environments, and addresses many of the challenges identified in previous work, including assumptions made about network middleboxes such as firewalls and NATs. We have also implemented the extensions in the OpenSSH and PureTLS software packages and tested them in practice.

1 Introduction

An increasing number of Internet hosts are mobile and equipped with more than one network interface. Simultaneously, operation of mobile hosts has become more continuous: the hosts have long uptimes, and the applications do not need to be closed when the host enters a "suspended" state. However, in the today's Internet, applications experience this combination of improved connectivity and operation as a less stable networking environment. This is mainly because transport layer connections break more frequently due to changes in IP addresses, network failures, and timeouts during disconnected or suspended operation.

It is often desirable to hide these disruptions from the end user. For instance, a user should be able to suspend a laptop, move to a different location, bring up the laptop, and continue using the applications that were left open with minimal inconvenience. In other words, the system should provide *session continuity* over the disruptions in network connectivity (cf. Snoeren's analysis of the session abstraction [23]).

Traditionally, session continuity has been considered as a part of mobility, and has been handled in the data link layer (e.g., wireless LAN or GPRS handover mechanisms) or in the network layer (e.g., Mobile IP). However, there are a number of reasons why providing session continuity higher in the protocol stack is desirable:

Long disconnection periods: while network-layer mobility mechanisms can deal with changing IP addresses, they cannot help the transport layer to overcome likely timeouts during long disconnections. Moreover, how exactly should long disconnections be handled often depends on the application in question.

No network infrastructure: in today's Internet it is common that clients are mobile but servers are not. In this kind of environment, session continuity can be provided without requiring the deployment of additional fixed infrastructure (such as Mobile IP home agents).

Applications get upgraded: it is often claimed that mobility has to be low in the stack to enable it for a large number of different applications. However, we hypothesize that it is often actually easier to deploy resilient mechanisms built into applications. After all, the applications get upgraded all the time and processes for that exist; but installing and configuring a Mobile IP implementation is beyond capabilities of most users and system administrators.

Limited end-to-end connectivity: mobility mechanisms implemented in the network or transport layer may not work across various types of middleboxes that are present in the network. For instance, if a firewall near a client allows only outbound TCP connections, Mobile IP does not work. Session continuity mechanisms integrated into applications make the least number of assumptions about the network between the endpoints.

These arguments suggest that the session layer is the lowest layer to implement *resilient connections* that can span several sequential transport layer (TCP) connections, and thus, survive not only changes in IP addresses, but also relatively long periods of disconnection.

In this paper, we extend two common secure session layer protocols to support resilient connections: Secure SHell (SSH) Transport Layer Protocol [28, 29] and Transport Layer Security (TLS) [3].[1] We have implemented these extensions in two open-source software packages: OpenSSH, the most popular SSH implementation [16], and PureTLS, a Java TLS library [20].

Our main contributions are as follows. First, we have developed resiliency extensions for the common TLS and SSH protocols that largely avoid the deployability problems associated with previous proposals. Second, we have analyzed the challenges faced when implementing this kind of extensions to legacy software packages that were not designed with resiliency in mind. In particular, different styles of handling concurrency and I/O have large implications for the implementations: OpenSSH uses asynchronous (select-based) I/O with a process for each client, while PureTLS uses synchronous I/O with threads.

The rest of the paper is structured as follows. In Section 2, we introduce the SSH and TLS protocols and previous work on resilient connections. Our design principles, described in Section 3, attempt to address deployment challenges we have identified in the existing proposals. In Section 4, we introduce our extensions to the SSH and TLS protocols. Section 5 describes our prototype implementations, which are then evaluated in Section 6. Finally, Section 7 summarizes our conclusions and discusses remaining open issues.

2 Background and related work

The Secure Shell (SSH) is a protocol for secure login and other network services [28]. It consists of three main sub-protocols: the SSH transport layer protocol, user authentication protocol, and connection protocol. The SSH transport layer protocol is the lowest layer, and is responsible for authenticating the server and providing an encrypted and integrity-protected channel for the other sub-protocols. The user authentication protocol authenticates the client, while the connection protocol multiplexes several logical connections (such as interactive terminal sessions, X11 window system forwarding, and TCP/IP port forwarding) over a single transport layer connection.

Transport Layer Security (TLS) is a session layer protocol providing encrypted and authenticated communication session for communication between two applications [3]. It consists of two major parts: the TLS record protocol provides a secure communication channel to upper layers, and is responsible for encryption and integrity protection of data. The TLS handshake protocol provides the key material and authentication for the TLS record protocol; this usually involves X.509 certificates and a key exchange based on RSA encryption. The two remaining components of TLS, the alert and change cipher spec protocols, are beyond the scope of this paper.

The benefits of providing session continuity above the transport layer have been recognized before; for instance, Duchamp [4] and Snoeren [24] provide several arguments in its favor. There is a large number of proposals that provide resilient connections above the transport layer but below the application layer protocol: Persistent connections [32], Mobile TCP socket [18, 19], MobileSocket [15], SLM or Session Layer Mobility [11], Reliable sockets [30], Migrate [23], Robust TCP connections [5], NapletSocket [33], Channel-based connectivity management [26], and Dharma [13], to mention just a few examples.

The common part of most of these proposals is a library placed above the transport layer but below the sockets API used by the application. The library presents a single unbroken communication channel to the application, hiding transport layer disruptions from the applications. The library is responsible for the signaling required to manage the multiple TCP connections, and also buffers application data so it can be retransmitted over a new TCP connection if necessary (this is required since most operating systems do not allow applications to access the TCP buffers).

However, implementing resilient connections in the "sockets API" layer has a number of drawbacks.

- The proposals typically use out-of-band signaling: a separate TCP connection (or UDP-based "session") coordinates multiple TCP connections. This can lead to deployment problems if, e.g., a firewall allows the port used by the application itself, but not the port used for resiliency signaling. An important reason for out-of-band signaling is the lack of an extension negotiation mechanism in the sockets API layer; however, such a mechanism is essential for incremental deployment. While some proposals (such as Zandy's reliable sockets [30]) do actually implement the initial resiliency signaling in-band, they rely on obscure TCP semantics with questionable deployability properties. However, even these solutions change to out-of-band signaling after the connection setup (e.g., due to TCP's head-of-line blocking issues).

- A separate key exchange is required to protect the signaling messages (if the messages are protected at all). This introduces additional overhead.

- While a separately delivered dynamically linked library that "hijacks" the operations of the normal

[1] Note that despite their names, both protocols are strictly above the transport layer (TCP) in the protocol stack, and thus calling them session-layer protocols is more accurate.

socket calls is a good approach for research, it creates deployment problems if it is does not come bundled and tested with the software with which it is to be used. Deploying such separate component is likely to get less-than-enthusiastic response from, e.g., corporate IT departments who would have to deploy and manage this component in mission-critical environments.

Proposals to implement the session continuity even higher in the protocol stack than the session layer exist. If an application protocol connection setup is a light-weight operation (e.g., HTTP GET), it's not necessary to extend any protocol to implement reconnections. An application just reconnects and at the same time minimizes the visibility of the reconnection to its user. For example, most modern mail user agents operate in this manner.

An application protocol connection setup may consume an considerable amount of resources, however, and thus, several application protocols have been extended to provide session continuity. For instance, REX, an SSH-like remote execution utility [9], the XMOVE extension to the X Window System [25], the REST extension to FTP [6], and SIP [22] all allow continuing a session even if a transport layer connection is disrupted for some reason. These extensions are typically very specific to the application in question; in contrast, our TLS extensions would work with any application-layer protocol run over TLS.

Session continuity for interactive terminal sessions can also be provided by decoupling the terminal seen by applications from the remote terminal session, as done in, for instance, Screen [7] and Bellovin's Session Tty Manager [1]. However, these approaches still require the user to manually establish a new SSH connection and reattach the terminal session.

Proposals that operate in the transport layer (e.g., Huitema's Multi-Homed TCP [8]) are beyond the scope of this paper. As they require modifications to the operating system's TCP/IP stack, and may not work with existing middleboxes, we do not consider them easily deployable.

3 Design principles

Based on the existing work, we have set our design principles to emphasize deployability.

No network changes: no extra requirements for the network or middleboxes between two communicating hosts should be set. As an example, the extensions must not require any additional configuration in firewalls.

Incremental deployment: the extensions should provide functionality once both connection end-points support it. The extensions should also interoperate with legacy end-points without the extensions.

Limited end-point changes: the extensions should require only modifications in TLS and SSH implementations, but no operating system changes or additional software components. The latter includes, e.g., dynamic libraries interposed between the application and the operating system.

In terms of functionality, our design principles were:

Disconnections may last long: the extension should deal gracefully with long periods of disconnection. The maximum supported disconnection period is a local policy issue, and not a protocol issue. Thus, the protocol extensions should not limit the duration of disconnections.

No handover optimization: the extensions are not optimized for fast handovers. This is mainly because we believe the default disconnection to be relatively long (from tens of seconds to hours).

In addition to the protocol extensions, there are certain implementation aspects to be considered. Server side concurrency is the most important one. The mechanisms used to implement concurrency often depend on the operating system and programming language used. Obviously, our extensions should not prevent typical server implementation strategies such as "a process for each client" (either forked on demand or beforehand), "a thread for each client", or select-style asynchronous I/O.

4 Protocol extensions

In this section we describe the extensions made to the SSH and TLS protocols. Since the extensions have much in common, we present the shared features first, followed by the SSH and TLS specific details.

4.1 Common features

In-band signaling: deployability concerns in practice mandate the use of in-band signaling. In other words, information required by the extensions is sent as part of normal SSH and TLS messages, and all TCP connections are initiated by the client. This ensures that resilient connections do not introduce any additional requirements for the network between the client and the server.

Extension negotiation: incremental deployment requires interoperability with endpoints that do not support these extensions, and thus, their use has to be negotiated. Fortunately, both SSH and TLS have mechanisms for negotiating protocol features when the connection is set up.

Securing signaling: when the client creates a new TCP connection to the server, it has to somehow indicate that it wants to continue a previous session, and prove that it is indeed the same client as previously. Thus, we need a way to identify an existing session (any public and unique information exchanged during the session setup

will do) and way to authenticate the signaling. Since both SSH and TLS establish session keys between the client and the server, the authentication is relatively easy to do.

Buffer management: both SSH and TLS operate over TCP, which provides a reliable lossless connection channel. However, when the TCP connection breaks, the TCP socket buffers may contain data that was not yet received by the other endpoint, and thus, the data has to be retransmitted when a new TCP connection is created. Since operating systems typically do not allow access to the TCP buffers, separate buffers have to be maintained in the application.

Previously, two different approaches have been used for managing these buffers: either data is removed from the buffer only when an explicit session layer acknowledgement is received (e.g., MobileSocket by Okoshi et al. [15]), or the buffer size is limited to the size of TCP buffers (e.g., Zandy's reliable sockets [31]). We chose the former approach: the endpoints send acknowledgements regularly (say, after receiving 64 kB from the peer). While the nodes may know their own TCP buffer sizes, the network may also contain transport layer proxies that buffer data: for instance, TLS is often run through web proxies using the "CONNECT" method [12]. Thus, while explicit acknowledgements add some overhead, only they ensure that the extensions work properly in existing network environments. This corresponds to the "end-to-end argument" by Saltzer et al. [21]: since parts of TCP may be implemented by the communication system itself, end-to-end reliability can be correctly implemented only above TCP.

Closing: SSH and TLS connections are both tightly bound to an underlying TCP connection. The resiliency extensions render the situation more complex: if the TCP connection breaks, the server should wait for the client to reconnect again. Thus, the protocol should have an explicit "close" message to be used when the endpoints actually want to close the session permanently. Fortunately, both the SSH transport layer protocol and TLS have this kind of messages. However, we discovered that OpenSSH did not actually send the close message, since previously there was no need to differentiate between a gracefully closed session and a broken TCP connection.

4.2 Extending SSH

Resilient connections for SSH could be implemented either in the SSH transport layer protocol or the connection protocol. In the end, we decided to implement our extension in the SSH transport layer protocol, since this seemed to be simpler, and had more in common with the TLS extensions described in the next section.

The SSH protocol suite is extensible: in the transport layer protocol, the client and the server negotiate the algorithms that will be used during the session. However, while the algorithms are negotiable, the negotiation does fix algorithm categories. Thus, we had to re-use an existing category to negotiate the resiliency support: the client and the server announce their support for this extension by including a Key EXchange (KEX) algorithm named "resilient" as the least-preferred algorithm. If both endpoints supports this extension, they enable buffering of data and sending of explicit acknowledgements. The acknowledgement is a new SSH message type that contains the sequence number of the next expected record.

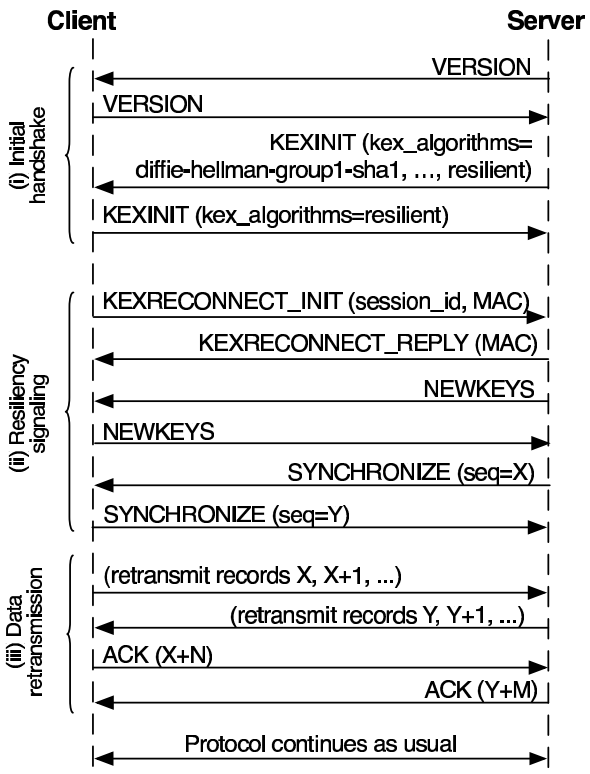


Figure 1: Reconnecting an existing SSH session.

Modeling the resiliency extension as a special key exchange algorithm also simplifies things when the client wants to reconnect; i.e., continue the same session over a different TCP connection. The exchange is shown in Figure 1. The client indicates that it wants to continue a session by listing "resilient" as the only supported key exchange algorithm. The client then sends a message containing a session identifier and a Message Authentication Code (MAC); the server responds with its own MAC. The MACs prove that the parties are still the same as in the original connection, and are calculated over the VERSION and KEXINIT messages (which include nonces to prevent replays). The MAC is calculated using a separate key used only for the KEXRECONNECT messages, and is derived during the initial handshake at the same time

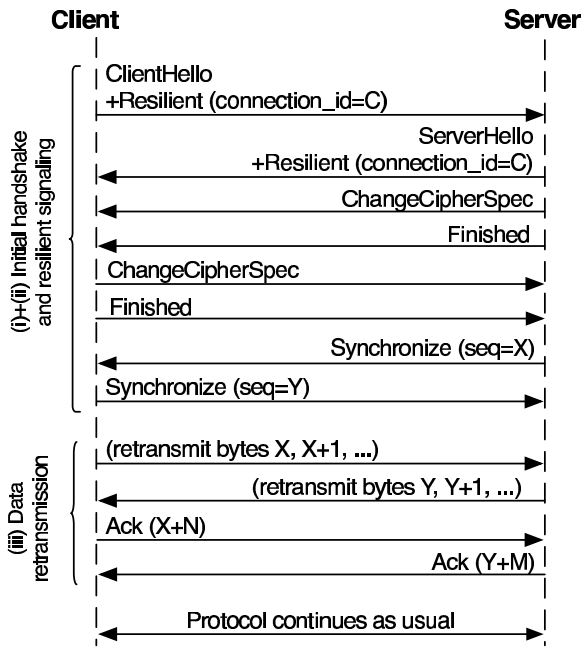


Figure 2: Reconnection procedure in TLS.

as the encryption and integrity protection keys.

After this, the endpoints take the cryptographic keys into use, send a "synchronize" message indicating where the previous connection was broken, and retransmit lost data from the buffers.

The SSH transport layer supports payload compression. While the transport layer protocol implements the compression, compression is done before encryption. Thus, the compression belongs to the topmost part of the transport layer. We decided to hide the connection disruptions from the compression engine to maintain the compression engine's state intact. Re-establishing the compression state would only decrease the compression performance during reconnections.

4.3 Extending TLS

The resiliency extension to TLS is negotiated using the TLS extension mechanism [2] in ClientHello/ ServerHello messages. Similarly as in the SSH case, if both endpoints support this extension, they start buffering data and sending acknowledgement messages. In the TLS case, the acknowledgement messages contain the number of application data bytes received instead of the TLS record sequence numbers. We chose this approach since the reconnection handshake is based on the abbreviated TLS handshake which resets the sequence numbers back to zero.

The reconnection exchange is shown in Figure 2. The client indicates that it wants to continue an existing connection by including a connection identifier in the ClientHello message. The ClientHello/ServerHello messages are followed by an abbreviated handshake (based on the normal TLS session resumption handshake) which verifies that the parties have remained the same and establishes fresh session keys.

After this, the endpoints tell how much data needs to be retransmitted, and retransmit the lost data, if any.

It is important to note that while the cryptographic handshake re-uses an existing TLS feature called "session resumption", there is an important difference. TLS session resumption is a feature of the TLS handshake protocol which caches the results of expensive public-key operations. It is a performance optimization and is independent of the actual data transfer (the TLS record protocol). Thus, it does not enable a client to continue an existing connection that was for some reason broken.

4.4 Security analysis

Making SSH and TLS sessions resilient to disconnections could introduce new security vulnerabilities. However, we believe that the extensions presented in this paper provide the same level of security as the situation when new SSH and TLS sessions are initiated to handle disconnections. In this section, we provide a high-level analysis of our protocol extensions. A complete security analysis of our protocol is beyond the scope of this paper.

In our extensions, all messages are authenticated using shared keys created during the initial SSH or TLS protocol exchange. Thus, an attacker cannot spoof or modify the reconnect messages. Replay attacks are not possible, since the first SSH and TLS key exchange messages include fresh nonces that are covered by a MAC later during the handshake.

Since the extensions require the endpoints to buffer data that has not been acknowledged, the amount of resources needed by a single SSH or TLS session is increased. Thus, the work required for a denial of service attack against a server (by creating a large number of sessions) may be less than in normal SSH or TLS. However, in most cases the buffers are likely to represent only a small share of the resources, and thus, denial of service resistance is not significantly changed.

5 Implementation considerations

In this section, we analyze the implications of resilient connections for SSH/TLS client and server side implementations.

5.1 When to reconnect and which interface to use?

Resiliency against connection disruptions brings a new challenge to client and server side implementations of both protocols. On the client side, the challenge is to determine when to start the reconnection procedure. Some options include the following:

1. A manual request; e.g., a user could click a "reconnect now" button in the application user interface.
2. Automatically when the device is brought up from a "suspended" power management state.
3. Whenever the current TCP connection is broken.
4. Whenever a more preferred network interface is available.
5. Probably several more options exist.

In addition to deciding when to reconnect, there may be multiple interfaces available: which of these should be used to establish the connection?

To ensure easy deployability, the solution should depend only on tools and APIs commonly available on the deployment environment and not require any additional software on the client machine.

Therefore, we decided to simply rely on the operating system's source address selection. In other words, we leave it to the operating system to decide which local interface should be used when a TCP connection is established, and initiate reconnection when the operating system's decision changes, or the current TCP connection is broken.

This raises the question of how to notice that the OS's source address selection policy has changed. In Windows, the Winsock API has a feature ("SIO_ROUTING_INTERFACE_CHANGE" socket option; see [14]) that allows the application to be notified of changes. BSD-based Unixes have "PF_ROUTE" routing sockets [27] and Linux has Netlink sockets [10] that also allow monitoring of routing table changes.

In the end, we implemented two different approaches. For OpenSSH, we used a routing socket to monitor routing table changes. In PureTLS, we settled for polling the OS in regular intervals to see if the preferred interface has changed. The polling can be done, for instance, by creating a "connection-mode" UDP socket, and reading the local address using the getsockname() API call (note that no UDP packets are actually sent). The approach was preferable in Java, since it avoided the need to have native and platform-specific code.

On the server side, a certain level of uncertainty is imminent too. For a server, the challenge is to determine the time to discard a session that waits for its client to reconnect. The difference to the client side is that the server must make the decision completely without the help of a user. Our vision is that the time a server is willing to keep resources allocated for a session without a connected client is a local policy issue. Different users may have different timeouts as well as servers with different loads may have different timeouts. For instance, one could assume a shared server is willing to maintain sessions shorter period of time than a server solely used by a single user. For the prototype implementations, we implemented a configurable server-wide timeout.

5.2 Server side concurrency

A common server design strategy is to create a new process or thread for each new client connection. While this often simplifies the server design, in this context concurrency becomes a complicating factor, since it results in a situation where the client's original session and reconnection request are handled by two different processes or threads.

A server designer has two options to choose from: either the new process finds the corresponding old process and passes the new TCP connection to the old process, or the other way around. Regardless of the choice, the server must maintain a table mapping sessions to processes for inter-process (or inter-thread) communication. In our implementations, the new process passes the new connection to the old process. Before passing the connection, the new process validates reconnection attempts. The validation requires contacting the old process, as the new process has no other access to the session keys. Once the new process has passed the connection to the old process, it exits.

Two reasons made us to choose the new process/thread to pass its state to the old process/thread. First, the new process has simply less state to pass: in practice, passing a file descriptor of a transport connection and sequence numbers to synchronize is sufficient. Second, besides the amount of state, the new process has state information that is easier to transfer. The old process can have such state that is impossible to pass across process boundaries. As an example, consider a TLS server process that creates child processes. In majority of platforms, it is impossible to pass child processes from a process to another—which would be a requirement if the old process passed its state to the new process.

In a multi-threaded server, implementing state passing is straightforward. However, if a server is implemented using concurrent processes, the above indicates that the server requires certain Inter-Process Communication (IPC) facilities:

1. An inter-process message channel to validate recon-

nection requests using keys stored in the old process.

2. An inter-process message channel to pass sequence numbers for synchronization.

3. A file descriptor passing mechanism to transfer a transport connection (socket) from the new process to the old process.

The required IPC facilities are realistic on most modern platforms, but they have often platform-specific features. Therefore, while the resiliency extensions are unlikely to prevent porting a server implementation to another platform, IPC mechanisms may add an extra twist to the porting process.

5.3 Atomic reconnections

Reconnection attempts must be atomic: the protocol state machine of an old connection must not become corrupted if an attempt fails. As discussed above, we designed the server implementations in a way that the new process transfers its state to the old process only after a reconnection request is determined to be valid. The approach has a positive side effect: the server side reconnection handling becomes atomic from the old process' point of view. If a reconnection request is invalid, the old process sees nothing.

Client implementations required similar atomic reconnection attempts: either a reconnection attempt succeeds or no state is affected. Unfortunately, implementing this in a general case can be challenging, as we learned in a hard way. A normal client implementation can modify global variables and data structures while connecting, and if connecting fails, it simply exits. Being a perfectly valid approach without resiliency extensions, this becomes challenging when the client implementation should behave in a deterministic manner in the case of connection failures.

Our observation was that it is tempting to modify a client implementation to behave as a multi-process or multi-threaded server: a fresh client process or thread attempts to reconnect and only once it succeeds, it passes its state to the old process or thread. In this way, the client implementation may dirty the new process or thread state but the attempt still remains atomic from the old process' point of view. As we implemented our clients in this way, we found out an unfortunate side effect: clients implemented in aprocess model require similar IPC facilities as the servers do. Our OpenSSH client was implemented as a multi-process and PureTLS client as multi-threaded client.

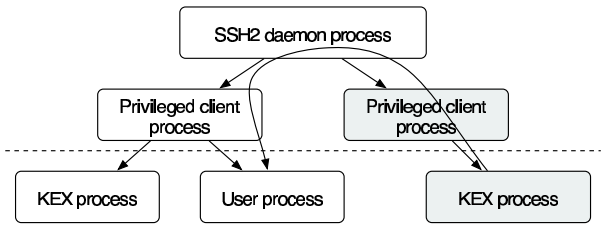


Figure 3: OpenSSH separates privileged processes (above dashed line) and less privileged processes (below dashed line). Grey boxes depict the processes processing a reconnection request.

5.4 Interface to higher layers

SSH transport layer protocol and TLS are not useful alone; they are always used together with some higher layer protocol. TLS is used with many different applications, while in the SSH case, the higher layer protocols are the SSH connection and user authentication protocols.

In general, we would like to change the interface offered by TLS and SSH transport layer protocol as little as possible. However, some changes and/or enhancements may be desirable. For instance, in the TLS case, some applications may be interested in knowing when a connection is no longer working, when a reconnection has happened, or even initiating reconnection.

Another set of issues arises from the fact that the IP addresses and port numbers used by the TCP connections may change when reconnecting. If the application uses these values for some other purpose than just sending packets, it may want to know when they change. For instance, OpenSSH can be configured to allow connections only from certain IP addresses. Similarly, a Java application can retrieve the addresses using Socket object methods such as getInetAddr(), and use them for, e.g., access control. Thus, it would be useful to have callbacks that allow the application logic to be notified when the addresses change.

These changes in the higher layer interfaces may require small modifications to OpenSSH and applications that use PureTLS. However, we have not yet implemented or further explored these modifications in the current versions of our prototypes.

5.5 OpenSSH

OpenSSH implements privilege separation to limit the effects of possible programming errors [17]. In the privilege separation, a privileged server daemon process uses less privileged processes to interface with clients. Less privileged processes then communicate with the privileged process through a monitor that protects the priv-

ileged process. Figure 3 depicts how OpenSSH forks (straight arrows) a separate process to do the key exchange and user authentication. Once the KEX process is done, the OpenSSH server forks yet another process to actually serve the client. Only the last process runs under the user's identity.

The OpenSSH privilege separation requires state serialization and passing across process boundaries: different processes perform the key exchange and the actual connection serving. After the key exchange, the KEX process serializes its key material together with information about the agreed algorithms and passes the state to its privileged parent process. The parent process then forks the actual connection serving process and passes the state further there.

It turned out that for both the OpenSSH server and client implementations, privilege separation facilities based on Unix socket pairs were enough to provide the required IPC facilities once they were extended to pass file descriptors. No additional authentication between processes was necessary either; the Unix socket pairs are invisible beyond a process and its child processes. On the server side, facilities provide atomic reconnections and the transfer of a new connection to an old process. On the client side, they only guarantee atomic reconnections as discussed earlier.

On the server side, we decided to transform the main daemon process into a message broker as it was the only common factor between all processes. The curved arrow in Figure 3 depicts how a connection together with synchronization information actually travels through several processes via the main daemon process, from the new process eventually to the old process.

5.6 PureTLS

In PureTLS, most of the implementation complexity comes from the requirement to keep the objects visible to the application (such as Socket, InputStream and OutputStream instances) unchanged over reconnections.

For Socket, this required creating an additional layer of indirection: a new Socket instance that forwards the method calls to the "real" underlying socket. Fortunately, PureTLS already contained this kind of indirection layer, and only small modifications were needed to allow changing of the underlying socket on-the-fly.

6 Evaluation

In this section we present measurement results of reconnection transactions for both protocols and discuss the complexity of implementations. In this paper, we did not focus on the performance optimizations. Instead, the

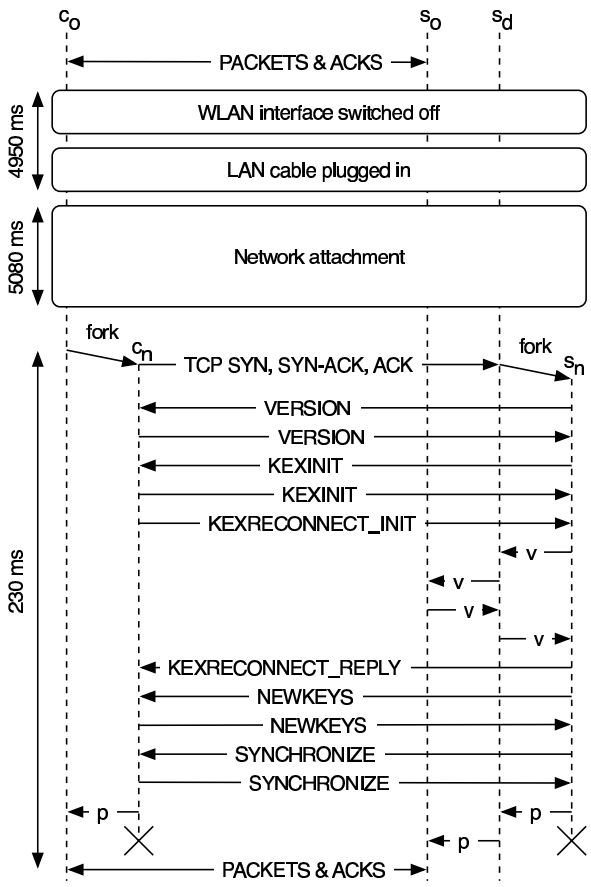


Figure 4: SSH processes involved in reconnecting a session.

main purpose of our evaluation is to show that our prototypes work and their performance is adequate for the intended use (relatively long disconnections).

6.1 Measurements

One of our main assumptions behind the design principles was that typical connection disruptions last a relatively long time. Therefore, we constructed one such scenario: a user manually switches from Wireless LAN to wired Ethernet. While the access to Internet from both networks goes through different NAT boxes, the user still expects his connections to survive from a changing IP address and NAT box. We conducted a set of measurements to validate the hypothesis and measure the actual expected length of typical reconnections.

In our scenario, the user downloads a large file from a remote server, either over SFTP or TLS. First, a user's laptop is attached to a wireless access point, but then the user decides to connect it to a fixed LAN to access remote services not available through the restricted public WLAN. Switching the access point requires, besides

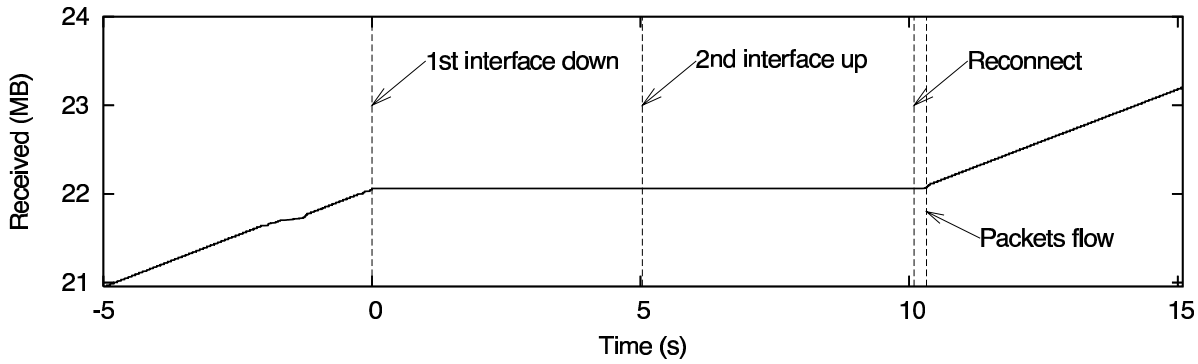


Figure 5: Progress of an SFTP transfer before, during, and after reconnection.

plugging an Ethernet cable, also turning off the WLAN interface; otherwise the laptop operating system keeps the WLAN interface as its primary interface.

We expect that typical disconnections would last significantly longer than the couple of seconds in these measurements. For example, we have used the extensions to keep SSH sessions alive while a suspended laptop is carried from the office to home.

OpenSSH measurements

In the OpenSSH tests, the remote download server was on the Internet and the round-trip time to the server was 10 ms through both WLAN and LAN. The SFTP client run on Mac OS X 10.4, while the SSH server was run on Linux.

As described in Section 5.1, the client can start reconnection not only when the TCP connection breaks, but also when the preferred source address has changed. Our OpenSSH extension uses a separate process to monitor the routing tables of the operating system. Once this process realizes that the route to the server has changed, it sends a signal to other processes that handle the actual reconnection.

Figure 4 represents the reconnection from a process viewpoint. On the client and server sides, temporary processes (c_n and s_n, respectively) handle the reconnection and old processes (c_o and s_o) receive the new transport connection only when it is time to resend lost packets. The main daemon process (s_d) only brokers messages between processes. In the figure, arrows titled as 'v' depict the inter-process validation messaging, while 'p' arrows depict the actual state passing.

Figure 5 shows the number of bytes an SFTP client has received as a function of time. The test user turned off the WLAN interface at time zero. The user quickly plugged a wired LAN cable in; finding the cable and inserting it to the laptop took less than three seconds. While it did not take that long to request an IP address (in the figure interface is up once it has an IP address), the graph illustrates how long it actually took before the network attachment was completely over from the SFTP point of view. Before the SFTP client receives a signal from the routing table monitoring daemon, 5 seconds has passed since the wired LAN interface came up. The actual reconnection then takes only about 200 ms before the file download continues.

PureTLS measurements

In the PureTLS tests, the client run on Linux and the server on Windows XP; the round-trip time to the server was around 1 ms.

Figure 6 shows the number of bytes received as a function of time. In this case, the network disruption lasted 5.5 seconds, and recovering from it took about 0.5 seconds. The differences compared to the OpenSSH case are explained mainly by how the reconnection is triggered (see Section 5.1).

Acknowledgment overhead

Figures 5 and 6 show only the downlink traffic and do not contain the additional network traffic caused by the session layer acknowledgements. However, this traffic is tolerable, and does not necessarily generate additional IP packets since the SSH/TLS ACKs can fit in the same packets as TCP ACKs.

Our OpenSSH acknowledgment implementation was suboptimal, since it acknowledges every received SSH transport layer message. While a single SSH ACK payload consumed only 5 bytes of space (packet type and 32-bit sequence number), the minimum cipher block sizes and MAC together increased the total size of ACK messages; a single ACK, with default OpenSSH configuration, consumed 32 bytes in total. Despite this suboptimal implementation, the extra traffic caused by the ACKs was more than acceptable, since the SSH transport layer messages can be up to 32 kilobytes: the ACK traffic amounted to less than 0.6% of the whole bandwidth. The PureTLS implementation does not acknowledge all records, but instead attempts to send ACKs at the same time as application data; the overhead figures were comparable to the OpenSSH case.

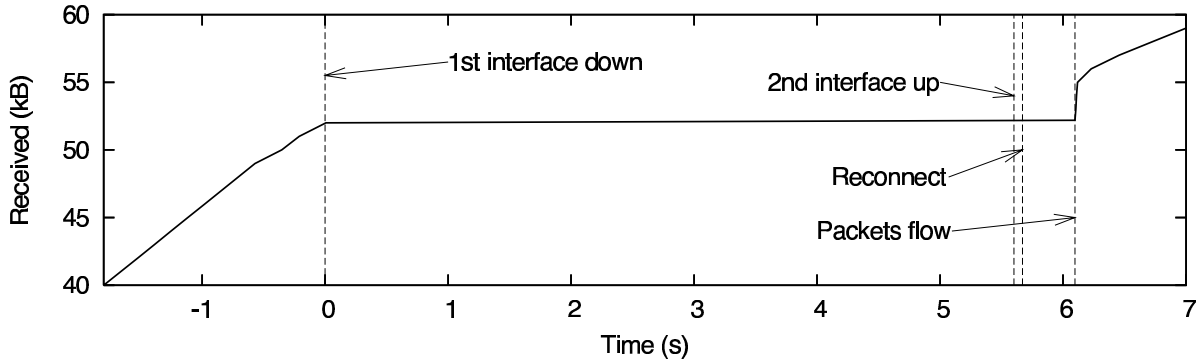


Figure 6: Progress of an TLS transfer before, during, and after reconnection. The temporary rapid speed-up after reconnection is caused by an implementation anomaly: the file transfer application uses a significantly smaller block size than the retransmissions done by PureTLS.

6.2 Implementation complexity

In-band resiliency signaling helps deployment, but it has an obvious extra cost: server and client applications must be modified. Next, we will briefly discuss the required implementation effort in the light of experiences from OpenSSH and PureTLS.

Our OpenSSH extension required about 2,200 lines of code. Over half of this code is related to passing state information and socket handles between the different processes. On the other hand, implementing explicit acknowledgments and buffering required relatively little effort, since much of existing OpenSSH functions were reusable as such. The OpenSSH implementation required roughly a man month of efficient work time, which included one complete refactoring.

The PureTLS extension was slightly simpler (about 1,000 lines of code) since Java's inter-thread communication facilities were much easier to use.

7 Conclusions

True *session continuity* in today's Internet requires not only handovers, but also gracefully handling long periods of disconnected operation. The contribution of our paper is three-fold. First, we have identified design principles that emphasize *deployability* and address some of the challeges in previous work. Second, following these principles, we have extended the SSH and TLS protocols to support resilient connections. Third, we have analyzed implementation issues faced when adding the functionality into two existing software packages.

Our three design principles are as follows: mechanisms for providing session continuity (a) should not place additional requirements for the network, (b) must allow incremental deployment, both providing benefits to early adopters and interoperating with legacy endpoints, and (c) should not require changes in operating system or third party libraries.

Our experience with the SSH and TLS extensions indicates that these design principles mandate certain protocol features, the most important one being in-band signaling. Furthermore, the protocol needs to support extension negotiation and explicit close messages, and has to be extended with explicit acknowledgements for transferred data.

The required extensions to the TLS and SSH protocols were relatively simple. In our case, we embedded the resiliency negotiation into the initial connection setup messages in a backwards compatible manner. In addition, both protocols execute mutual authentication while reconnecting simply by proving the possession of the shared secret of a suspended session.

In the implementations, handling the server side concurrency was clearly the most challenging part. The process (or thread) that is handling the reconnection request must find the corresponding old process, since only the old process can validate the request. After a successful validation, the new process must pass the connection state and the TCP socket to the old process. This translates into inter-process or inter-thread communication mechanisms. Moreover, while dividing functionalities between the new and old processes, we found out that a new process should prepare a reconnection attempt and only alter the state of the old process after the reconnection attempt has succeeded. This simplified the implementations considerably.

While our paper has focused on addressing deployment challenges, deployability remains a difficult concept. Much of existing work on mobility has focused on issues easy to measure and compare, such as handover performance. Deployability in general, as well as approaches to compare it, have received less attention, and clearly, more work is needed to better understand how

different protocol design choices affect deployability.

Acknowledgments

The authors would like to thank N. Asokan, Dan Forsberg, Andrei Gurtov, Tobias Heer, Janne Lindqvist, and Pekka Nikander for their comments and suggestions. Helpful comments were also provided by Tero Kivinen who said he had planned a similar extension to SSH several years ago (however, no additional information is available about this work). Finally, we thank the anonymous reviewers and our shepherd, Stefan Saroiu, for their insightful comments.

References

[1] Steven M. Bellovin. The "Session Tty" Manager. In *Proceedings of the Summer 1988 USENIX Conference*, pages 339–354, San Francisco, June 1988.

[2] Simon Blake-Wilson, Magnus Nystrom, David Hopwood, Jan Mikkelsen, and Tim Wright. Transport Layer Security (TLS) Extensions. RFC 3546, IETF, June 2003.

[3] Tim Dierks and Christopher Allen. The TLS protocol version 1.0. RFC 2246, IETF, January 1999.

[4] Daniel Duchamp. The discrete Internet and what to do about it. In *2nd New York Metro Area Networking Workshop*, New York, NY, September 2002.

[5] Richard Ekwall, Pétér Urbán, and André Schiper. Robust TCP connections for fault tolerant computing. *Journal of Information Science and Engineering*, 19(3):503–516, May 2003.

[6] Robert Elz and Paul Hethmon. Extensions to FTP. Work in progress (IETF Internet-Draft, draft-ietf-ftpext-mlst-16), September 2002.

[7] GNU Screen. http://www.gnu.org/software/screen/, 2006.

[8] Christian Huitema. Multi-homed TCP. Work in progress (draft-huitema-multi-homed-01), May 1995.

[9] Michael Kaminsky, Eric Peterson, Daniel B. Giffin, Kevin Fu, David Mazières, and M. Frans Kaashoek. REX: Secure, Extensible Remote Execution. In *Proceedings of the 2004 USENIX Annual Technical Conference*, Boston, MA, June–July 2004.

[10] Andi Kleen et al. rtnetlink, NETLINK_ROUTE – Linux IPv4 routing socket. Linux Programmer's Manual, man page rtnetlink(7), 2000.

[11] Björn Landfeldt, Tomas Larsson, Yuri Ismailov, and Aruna Seneviratne. SLM, a framework for session layer mobility management. In *Proceedings of the 8th International Conference on Computer Communications and Networks (ICCCN '99)*, Boston, MA, October 1999.

[12] Ari Luotonen. Tunneling TCP based protocols through Web proxy servers. Work in progress (draft-luotonen-web-proxy-tunneling-01), August 1998.

[13] Yun Mao, Björn Knutsson, Honghui Lu, and Jonathan M. Smith. Dharma: Distributed home agent for robust mobile access. In *Proceedings of IEEE INFOCOM 2005*, Miami, FL, March 2005.

[14] Microsoft Corporation. Windows sockets 2. MSDN Library, http://msdn.microsoft.com/library/en-us/winsock/winsock/windows_sockets_start_page_2.asp, 2006.

[15] Tadashi Okoshi, Masahiro Mochizuki, Yoshito Tobe, and Hideyuki Tokuda. MobileSocket: Toward continuous operation for Java applications. In *Proceedings of the 8th International Conference on Computer Communications and Networks (ICCCN '99)*, Boston, MA, October 1999.

[16] OpenBSD project. OpenSSH. http://www.openssh.org/, 2006.

[17] Niels Provos, Markus Friedl, and Peter Honeyman. Preventing privilege escalation. In *Proceedings of the 12th USENIX Security Symposium*, Washington, DC, August 2003.

[18] Xun Qu, Jeffrey Xu Yu, and Richard P. Brent. A mobile TCP socket. Technical Report TR-CS-97-08, Computer Sciences Laboratory, RSISE, The Australian National University, 1997.

[19] Xun Qu, Jeffrey Xu Yu, and Richard P. Brent. A mobile TCP socket. In *Proceedings of the IASTED International Conference on Software Engineering*, San Francisco, CA, November 1997.

[20] Eric Rescorla. Claymore PureTLS. http://www.rtfm.com/puretls/, 2006.

[21] Jerome H. Saltzer, David P. Reed, and David D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[22] Henning Schulzrinne and Elin Wedlund. Application-layer mobility using SIP. *ACM SIGMOBILE Mobile Computing and Communications Review*, 4(3):47–57, July 2000.

[23] Alex Snoeren. *A Session-Based Approach to Internet Mobility*. PhD thesis, Massachusetts Institute of Technology, 2002.

[24] Alex C. Snoeren, Hari Balakrishnan, and M. Frans Kaashoek. Reconsidering Internet mobility. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, Schloss Elmau, Germany, May 2001.

[25] Ethan Solomita, James Kempf, and Dan Duchamp. XMOVE: a pseudoserver for X window movement. *The X Resource*, 11, July 1994.

[26] Jun-Zhao Sun, Jukka Riekki, Marko Jurmu, and Jaakko Sauvola. Channel-based connectivity management middleware for seamless integration of heterogeneous wireless networks. In *Proceedings of the the 2005 Symposium on Applications and the Internet (SAINT '05)*, Trento, Italy, January–February 2005.

[27] Sun Microsystems. route – kernel packet forwarding database. Solaris 10 Reference Manual Collection, man page route(7P), 2003.

[28] Tatu Ylönen and Chris Lonvick. The Secure Shell (SSH) protocol architecture. RFC 4251, IETF, January 2006.

[29] Tatu Ylönen and Chris Lonvick. The Secure Shell (SSH) transport layer protocol. RFC 4253, IETF, January 2006.

[30] Victor C. Zandy and Barton P. Miller. Reliable network connections. In *Proceedings of the 8th annual International Conference on Mobile Computing and Networking (MobiCom '02)*, Atlanta, GA, September 2002.

[31] Victor Charles Zandy. *Application Mobility*. PhD thesis, University of Wisconsin-Madison, 2004.

[32] Yongguang Zhang and Son Dao. A "persistent connection" model for mobile and distributed systems. In *Proceedings of the 4th International Conference on Computer Communications and Networks (ICCCN '95)*, Las Vegas, NV, September 1995.

[33] Xiliang Zhong, Cheng-Zhong Xu, and Haiying Shen. A reliable and secure connection migration mechanism for mobile agents. In *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW '04)*, Tokyo, Japan, March 2004.

Multi-homed Internet Access in Ad Hoc Networks using Host Identity Protocol

Mikko Särelä
Theoretical Computer Science
Helsinki University of Technology
Espoo, Finland
Mikko.Sarela@tkk.fi

Abstract

An ad hoc access network may have several Internet Gateways that are located in different parts of the Internet topology. In this paper, we present a multi-homed solution to the global connectivity problem using the Host Identity Protocol (HIP). The solution, taking advantage of the multi-homing capabilities of HIP, allows a MANET node to maintain several locators e.g. IP addresses to the corresponding node and to automatically switch between them, when any of them becomes unusable. The main benefit comes from the fact that the return routability signaling about the locators can be done well in advance. In addition, the Internet Gateways can act as NAT devices between the Internet and the ad hoc network and allow the ad hoc network routing to utilize the cryptographically secure host identifiers HIP provides.

1 Introduction

Some nodes in mobile ad hoc networks (MANETs) may have access to a fixed network infrastructure, such as the Internet. As these gateways to the Internet may reside in different organizations, such a MANET should be considered to be a potentially multi-homing access network from the Internet architectural perspective. In the literature, these networks are often called either ad hoc access networks, or hybrid networks, to distinguish them from pure ad hoc networks with no fixed network infrastructure. In this paper, we will refer to them as ad hoc access networks.

The problem of Internet access for ad hoc network hosts consists of several subproblems: finding an Internet Gateway (GW), registration with the Internet Gateway, access control, authorization, and address configuration, route maintainance to the Internet Gateway, passing data traffic between the MANET and the Internet, reachability for the MANET nodes from the Internet, and the end-to-end mobility management.

In this paper, we present a method for Internet access in ad hoc networks based on the Host Identity Protocol [7, 6] that solves the problems of registration, passing data traffic, and reachability. Our contribution is to solve the global route maintainance in a multi-homing fashion so that several routes (Internet Gateways) may be ready for use in a single session simultaneously.

The rest of the paper is structured as follows. In Section 2 we introduce previous work on global access in ad hoc access networks and then continue with the Host Identity Protocol in Section 3. We then describe our solution in Section 4 and finally conclude in Section 5.

2 Background

Globalv6 [14] proposes the necessary requirements for Internet access from ad hoc networks. Internet Gateways either send advertisements, or respond to requests and inform a MANET node of a network prefix it must use. The draft recommends using the routing header with the address of the Internet Gateway for packets destined to the Internet in order to force them through the chosen Internet Gateway.

Numerous proposals use Mobile IP to make changing Internet Gateways possible e.g. [1, 13, 15]. Miller et al. [5] extended DHCP to ad hoc network usage to solve the address configuration problem in IPv4. It is also possible to hide the mobility from the MANET nodes using mobile gateways [16, 17] as is done with mobile networks.

In a performance study, it was found that link breaks were the main factor affecting transport protocol performance during micro mobility [9]. Link breaks are a major reason for switching Internet Gateways and thus, there is a need for finding faster and more efficient ways of switching from one Internet Gateway to another when the route breaks. Nilsson et al. [10, 11] integrate micro mobility and macro mobility in ad hoc access networks using HAWAII [12] and Mobile IP to obtain faster handovers for micromobility.

3 Host Identity Protocol

Host Identity Protocol introduces host identities, which create a new flat and cryptographically secure name space. Host identifiers are public cryptographic keys. They replace the end-to-end connectivity purpose of IP addresses in upper layers leaving IP addresses as topological locators used for routing. Transport layer connections are composed of host identities and port numbers.

Since public keys need to be long for security reasons, using them directly is problematic. Instead, 128-bit host identity tags (HIT) are used. They are created from the public key using a cryptographic hash function. This saves bandwidth and has the added benefit that a Host Identity looks exactly like an IPv6 address, and thus no changes to transport layer applications are required.

The HIP protocol [7] consists of a base exchange, mobility signaling [2], and some additional messages. The purpose of the base exchange is to create assurance that the hosts indeed possess the private keys corresponding to their identities. This is accomplished with a two round trip Diffie Hellman key exchange protocol, which creates a pair of IPSec security associations between end-points based on the Host Identifiers.

During base exchange both hosts also create a mapping from IP addresses to HITs and vice versa for both in-coming and out-going traffic such that transport layer sees HITs and IP layer sees IP addresses in packets.

The base exchange,shown in Figure 1, consists of I_1, R_1, I_2, R_2 messages. When the responder receives an I1 packet, it selects a suitable R1 packet from a pool of precomputed messages. The R1 message contains a puzzle that the initiator has to solve. The same message also initiates the Diffie-Hellman exchange. It contains the responder's host identity public key, together with the Diffie-Hellman public key and other Diffie-Hellman parameters. From the traffic analysis point of view, it is important to notice that the responder is not able to form the session key before the I2 packet arrives. Therefore, the responder's host identity public key is currently transmitted in clear.

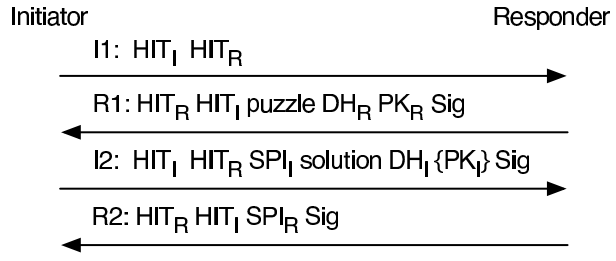


Figure 1. HIP base exchange

Upon receiving R1, the initiator solves the puzzle, computes a session key, and sends I2. I2 includes the puzzle solution, Diffie-Hellman parameters, SPI, and the initiator's host identity public key. The host identity public key is encrypted using the session key.

The responder verifies that the puzzle is correctly solved, creates the session key, authenticates the initiator, and creates session state. The final message, R2, contains the responder's SPI and a signature. The signature allows the initiator to complete the authentication procedure.

The use of cryptographically secure Host Identifiers clarifies the meaning of IP addresses by making them pure topological labels. This makes it easier for hosts to signal new interfaces or changes to address(es) in a secure manner. HIP provides a generalization called *locator* for describing new interfaces and changes to addresses.

3.1 Mobility and Multihoming

Once the HIP base exchange has been completed and the security associations are in place, the end-points can inform their peers about other interfaces they can be reached from and the current IP addresses assigned to the interfaces. This is done using the *locator* parameter in the HIP re-addressing protocol [2]. The protocol consists of Re-address (REA) and New SPI (NES) packets.

A host can use the HIP re-addressing protocol to inform its peer of multi-homing by using the *locator* parameter [2] in either R1 or I2 packets. *Locator* is a name that controls how the packet is routed through the network [2]. The name may contain for example an IPv6 address, an IPSec Security Parameter Index (SPI) value, and other information that is needed to route a packet. When a host wishes to inform its peer that it has new interfaces, or that one of its current interfaces has moved, it can do so using the HIP Readdressing protocol. The data structure containing the *locator* also states its lifetime (seconds) and status (*unverified*, *active*, and *deprecated*).

When a node changes its location in the network topology, it sends a Re-address (REA) packet from its new address. The peer optionally responds with a New-SPI (NES) packet, containing a new SPI that is used to verify that the mobile is indeed in the claimed location. The node then responds with an ESP message to the new SPI, thus verifying that it is indeed reachable from the new location. In addition the architecture supports the concept of Rendezvous Server [6], which is used for initial rendezvous, simultaneous movement, and location privacy.

3.2 Delegation

A *locator* through which a node can be reached does not have to describe an address where the node actually resides.

It may also contain the IP address of a middle-box, which then forwards the data traffic to the actual destination. The HIP Rendezvous Server is an example of such a middle-box, as is the Mobile IP Home Agent.

In addition to being reachable from said location, a node may also delegate the right to make signaling to the middle-box [8]. The key here is the authorization certificate that the node gives to the middle-box, which describes the rights that the middle box is given. We propose to use an authorization certificate that allows the middle-box to inform corresponding nodes of new *locator(s)* and to manage them.

This requires the middle-boxes to make address translations for packets that pass them. In HIP, IPSec Security Parameter Index (SPI) value can be used as an index for the end-point identifier together with the destination IP address [18], which makes it possible to make changes to the IP addresses and port numbers without affecting the end-to-end connectivity. The SPI multiplexed Network Address Translation (SPINAT) device establishes state during the IPSec control plane signaling.

4 Multihomed ad hoc access networks

We propose the use of Host Identity Protocol for enabling global connectivity for MANET nodes. Many ad hoc routing protocols, such as AODV [] use flat routing identifiers. Thus, HITs can be directly used as routing identifiers in such ad hoc routing protocols. The MANET node first registers with one of more Internet Gateways as specified in globalv6 [14].

An Internet Gateway, which does network address translation (e.g. SPINAT), must be able to advertise that capability to the MANET. We propose that this will be done by advertising the MANET network prefix that is not globally routable. This way the MANET node will know that it does not need a globally routable IP address to reach Internet and can omit the address configuration and trust that the gateway will translate the MANET local source address to a globally routable address.

With HIP the packets are recognized with IPSec SPI value and destination address. A SPINAT [18] enabled gateway will translate the IP packet header so that the MANET local routing identifier of the MANET node is changed to the globally routable IP address and vice versa. If the gateway does not support NAT, then it will advertise a globally routable network prefix and the host will then configure an address to itself as described in the globalv6 draft [14].

The registration to the HIP enabled Internet Gateway is done with the base exchange. As the base exchange may carry many types of secure signaling, it may be possible to combine AAA signaling with it for regulating Internet access. The gateway then creates an entry to its database about the MANET host. After registering, the MANET node may setup sessions with Internet nodes.

The MANET node may register with multiple Internet Gateways simultaneously and then announce all the different addresses to the corresponding node via *locator* parameter. Thus, the communication may go through any of the Internet Gateways that the MANET node has registered with. To make this possible, the MANET node has to inform the corresponding node of these potential addresses and for security reasons, the reachability of the MANET node from those addresses needs to be verified.

As the Internet Gateways are doing NAT to the packets that pass through them, the MANET node does not necessarily know the addresses through which it is reachable. Instead of informing corresponding node of new *locators* and managing them itself, it gives the Internet Gateway an authorization certificate, which authorizes the Internet Gateway to to do that on its behalf, thus delegating the right to signaling to the Internet Gateway.

It then sends a session context to the gateways, which contains information about current ongoing sessions and its existing Rendezvous Services [4]. Using this information, the Internet Gateway is able to inform the Rendezvous Service and the corresponding nodes of a new *locator*. This can be done using the HIP re-addressing protocol. The corresponding node can ascertain that the Internet Gateway has the right to manage the *locator* by examining the delegation certificate attached to the re-addressing message.

The *locator* contains the IP address of the gateway and the IPSec SPI value to use for that destination. The corresponding node verifies that the MANET node is reachable at the new *locator* via the return routability mechanism in the re-addressing signaling.

When each Internet Gateway has informed the corresponding node of the new *locator* for the MANET node, the corresponding node will have an entry list in its database for that session containing the IP addresses, SPI values, port numbers, the used Host Identity, the potential delegation certificate, and security associations (SAs) to use for each combination of source address and destination address. One of the *locators* is marked as the preferred *locator*.

A single SA pair can be used for all the different *locator* combinations as long as only one route is used at a time. Simultaneously used routes need separate SAs to avoid violating the IPSec anti-replay window [3]. Using only one SA may be preferable, as the maximum number of different routes is the number of source node *locators* multiplied by the number of destination node *locators*, which can be relatively high in case when both nodes are mobile and not each potential route (through any pair of *locators*) may end up being used, even if each *locator* is used at some point.

The information that the MANET node (MN), the Corresponding Node (CN), and the GWs have after the initial system setup is shown in Table 1. The table shows that both

Table 1. The information that MN, CN, and GWs have during communication session. All nodes use this information to match the incoming packets and the GW uses the information it has to change the src/dst addresses and SPI values in the packets.

	DST	SPI_{out}	SRC	SPI_{in}	Routing Header	Active	Preferred	
MN	IP_{CN}	SPI_{MN}^{GW1}	HIT_{MN}	SPI_{GW1}^{MN}	HIT_{GW1}	yes	yes	
MN	IP_{CN}	SPI_{MN}^{GW2}	HIT_{MN}	SPI_{GW2}^{MN}	HIT_{GW2}	yes	no	
CN	IP_{MN}	SPI_{CN}^{GW1}	IP_{GW1}	SPI_{GW1}^{CN}	-	yes	yes	
CN	IP_{MN}	SPI_{CN}^{GW2}	IP_{GW2}	SPI_{GW2}^{CN}	-	yes	no	
GW	CN	SPI_{out}^{CN}	SPI_{in}^{CN}	MN	SPI_{out}^{MN}	SPI_{in}^{MN}	Cert	Active
GW1	IP_{CN}	SPI_{CN}^{GW1}	SPI_{CN}^{GW1}	HIT_MN	SPI_{GW1}^{MN}	SPI_{MN}^{GW1}	C(MN,GW1)	yes
GW2	IP_{CN}	SPI_{GW2}^{CN}	SPI_{CN}^{GW2}	HIT_MN	SPI_{GW2}^{MN}	SPI_{MN}^{GW2}	C(MN,GW2)	yes

routes (GW1 and GW2) are currently active and the GW1 is the currently preferred route to the MN. The SAs between the MN and the GW and the SA key material have been omitted from the table as there is no difference to typical IPSec operation.

The MN has a SA with both Internet Gateways and with the CN. Both table entries for the CN also contain the routing header information to be used and the key material. The key material is the same for all routes between the MN and the CN. The CN has the IP addresses and the SPI values for both GWs, through which the MN can be reached. The GW has a separate set of SPI values for packets going from it to the CN and those going to the MN. Thus, the GW may change both the addresses and the SPI values in the packet passing through it.

When the route between MANET node and an Internet Gateway breaks, the MANET node can switch to another Internet Gateway and send the corresponding node a message informing the new Internet Gateway (*locator*) as the preferred route to itself. At the same time, the Internet Gateway that notices that it has lost its route to the MANET node, will send a message to the correspondent node using a new *locator* status field *inactive*.

The *inactive* status field indicates that the *locator* cannot currently be used to reach the destination (but may be usable again in near future). The new status field is intended primarily for *locator*s that reside somewhere else in the Internet topology than the actual end point and need to forward traffic to the actual destination. The *inactive* status means that the *locator* cannot currently be used, but may become usable again in the future. *Inactive* status means that the correspondent node may keep the *locator* and the SA associated with it for future use, or may decide to remove them.

That way the corresponding node will know to stop using that *locator*. If the route between Internet Gateway and MANET node is re-established, the gateway can send a Readdressing message announcing the *locator* as *active*. The optional test for return routability is not required, because the return routability test has already been done to the *locator*, when it was first received. The purpose of return routability test is to protect against redirection attacks in which the traffic is targeted toward a Denial of Service target. The explicit authorization certificate for the signaling ensures that the activation message can only be sent by the node whose reachability from the location has already been verified.

The described approach also works for the case, in which both communicating nodes can be found in the same MANET. In this case, both nodes also have an ad hoc routing identifier they can use for communications, i.e. a *locator* through which they are directly reachable. Thus, the nodes can have the direct route through the ad hoc network and also other routes through the Internet. Having the additional routes through the gateway helps, when the ad hoc network splits. If both nodes still have a route to at least one Internet Gateway, they can immediately switch their communications to go through the Internet Gateways.

5 Conclusions

We have presented a method for multi-homed Internet access from ad hoc networks using Host Identity Protocol. The solution is based on delegating limited signaling rights to the Internet Gateways. By introducing the *inactive* state to *locators*, a *locator* can be shut down and brought back up with a single message to the correspondent node.

In addition, the proposed solution allows the ad hoc routing protocol to use cryptographically secure routing identifiers, e.g. HITs instead of IP addresses, which should make deployment of secure ad hoc routing protocols easier. This can be achieved, because Host Identity Layer acts as the new end-to-end layer and allows ad hoc network routing and Internet routing to be kept in separate realms. The secure

registration between Internet Gateway and MANET node via base exchange node makes it possible for the ISP to identify and authorize hosts in ad hoc access network and secure deployment of AAA services.

Global reachability from ad hoc networks can be achieved with Host Identity Protocol with minimal changes to the protocol, namely the support for mobility contexts and delegation of signaling. Previous research has found that route breakages are the main reason for poor performance in ad hoc access networks and thus, the goal of this work is to minimize the time it takes to have another route up and ready for use. As the secondary routes are already established before the breakage, they can be immediately taken into use as one route fails.

In future, our purpose is to test the hypothesis that having a multiple points of contact simultaneously affects communications performance by building a simulation implementation and comparing the performance of HIP based Internet access to mobile IP based access using a network simulator. Many parts of the protocol, e.g. the method to determine a broken route between Internet Gateway and MANET node, need to be decided in more detail.

A node would benefit from knowing more knowledge of the condition in the possible paths between it and the corresponding node. Especially the security, bandwidth, delay, jitter, and reliability of possible routes would be beneficial. How to determine these conditions on a network path is an interesting problem that we leave for future. It may also be possible to leverage several paths simultaneously, if problems related to unsynchronized arrival of the packets can be solved.

Acknowledgements

The author would like to express his gratitude to Pekka Nikander, Teemu Koponen, Anders Nilsson, and the anonymous reviewers for their suggestions and valuable comments on various phases of the paper. This research has been funded by Finnish National Technology Agency, Ericsson, and Finnish Defence Forces. The author was a visiting scholar to CALIT2 in University of California, San Diego during the final phases of the work.

References

[1] J. Broch, D. Maltz, and D. Johnson. Supporting hierarchy and heterogenous interfaces in multi-hop wireless ad hoc networks. In *Proceedings of the 4th International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'99)*, pages 370–375, Jun 1999.

[2] T. Hendersson (editor). End-host mobility and multihoming with host identity protocol. Internet draft:draft-ietf-hip-mm-03, Feb 2006.

[3] S. Kent. IP encapsulating security payload (ESP). Rfc, IETF Network Working Group, December 2005.

[4] J. Laganier and L. Eggert. Host identity protocol (HIP) rendezvous extension, October 2005. Internet Draft, Work in progress draft-ietf-hip-rvc-04.

[5] M. J. Miller, W. D. List, and N. H. Vaidya. A hybrid network implementation to extend infrastructure reach. Technical report, University of Illinois at Urbana-Champaign, Jan 2003.

[6] R. Moskowitz and P. Nikander. Host identity protocol architecture. Internet draft:draft-moscowitz-hip-arch-03.txt (work in progress), Aug. 2005.

[7] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. The host identity protocol. Internet draft:draft-moscowitz-hip-base-05.txt (work in progress), Mar 2006.

[8] P. Nikander and J. Arkko. Delegation of signalling rights. In *Security Protocols, 10th International Workshop*, LNCS 2845, pages 203–212, Cambridge, UK, Apr 2002. Springer, 2003.

[9] A. Nilsson. Performance of routing and wireless aware transport layer connections in micro mobility ad hoc networks. In *Proceedings of the 6th World Wireless Congress*, May 2005.

[10] A. Nilsson, A. Hamidian, and U. Krner. Micro mobility and internet access performance for TCP connections in ad hoc networks. In *Nordic Teletraffic Seminar 17*, August 2004.

[11] A. Nilsson and U. Krner. Micro mobility performance in internet access ad hoc networks. In *World Wireless Congress (WWC04)*, May 2004.

[12] R. Ramjee, T. L. Porta, S. Thuei, K. Varadhan, and S. Wang. HAWAII: A domain-based approach for supporting mobility in wide-area wireless networks. In *IEEE Intl Conference on Network Protocols*, 1999.

[13] Y. Sun, E. M. Belding-Royer, and C. E. Perkins. Internet connectivity for ad hoc mobile networks. *International Journal of Wireless Information Networks*, 9(2):75–88, Apr 2002.

[14] R. Wakikawa, J. T. Malinen, C. Perkins, A. Nilsson, and A. J. Tuominen. Global connectivity for IPv6 mobile ad hoc networks. Internet draft: draft-wakikawa-manet-globalv6-04.txt (work in progress), November 2002.

[15] R. Wakikawa, J. T. Malinen, C. E. Perkins, A. Nilsson, and A. J. Tuominen. Internet connectivity for ad hoc networks. *Wireless Communication and Mobile Computing Journal*, 2:465–482, Aug 2002.

[16] R. Wakikawa, H. Matsutani, R. Koodli, A. Nilsson, and J. Murai. Mobile gateway: Integration of MANET and NEMO. In *ACM Mobihoc Poster*, Mar 2004.

[17] R. Wakikawa, H. Matsutani, R. Koodli, A. Nilsson, and J. Murai. Mobile gateways for mobile ad-hoc networks with network mobility support. In *Proceedings of the 4th International Conference on Networking, ICN05*, Apr 2005.

[18] J. Ylitalo, P. Salmela, and H. Tschofenig. SPINAT: Integrating IPSec into overlay routing. In *First International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm'05)*, September 2005.

APPLYING HOST IDENTITY PROTOCOL TO TACTICAL NETWORKS

Mikko Sarela
Theoretical Computer Science
Helsinki University of Technology
Espoo, Finland

and

Pekka Nikander
Nomadiclab
Ericsson Research IP Networks
Jorvas, Finland

ABSTRACT

In this paper, we describe the current status of the Host Identity Protocol and discuss how it could be applied to tactical networks, including mobile ad hoc networks. The Host Identity Protocol (HIP) is a protocol proposal at the IETF for separating the end-point identifier and locator nature of IP addresses. It introduces a new name space, consisting of public cryptographic keys, and uses these keys to identify hosts. All applications deal with the public keys instead of IP addresses; with a backward compatibility layer, most current applications will continue to work unchanged. A new layer in the kernel dynamically maps the public keys in outgoing packets into IP addresses, and vice versa for incoming packets.

INTRODUCTION

The term "tactical network" generally refers to a communications network employed in a military setting. There is increasing interest in using Internet-based protocols as the foundation for future tactical networks. While there are certain cost benefits to this approach (equipment choices, lower training and operations costs), the generally available standard Internet protocols may not satisfy the communications requirements of tactical networks in terms of security, mobility, and protocol performance.

One concept common among Internet users is the notion that their computer is identifiable by an IP address. This has certainly been true for most users connected by wires, via a single interface, to the network. However, the situation becomes more complicated when a device has more than one network interfaces. In a mobile setting with possibly spotty radio performance, it may be increasingly common for devices to use more than one interface, to improve network availability. Moreover, when a device moves around, it typically needs to obtain a new IP address to conform to the locally-available address prefix, since IP addresses are hierarchical and aggregated by the prefix. Once devices have more than one IP address, and once IP addresses become dynamic, it becomes increasingly hard and less secure to rely on the assumptions that IP addresses have a static, one-to-one mapping with a particular computer.

In this paper, we describe how the Host Identity Protocol (HIP) [7], a new architecture and protocol for IP-based networks, may improve the situation for IP-based tactical networks that are faced with these types of mobility and multi-homing scenarios. In general, we suggest that additional layers of abstraction between the network layer and application layer can allow hosts to better adapt to changing networking conditions. In this paper, we only concentrate on a few aspects, namely mobility, multi-access, and security, leaving considerations such as congestion control and transitory connectivity for future work.

The rest of this paper is organized as follows. First, in the next Section, we briefly describe the problem at hand. The following four Sections briefly describe the HIP architecture and based exchange, HIP based mobility and multi-homing, HIP based access control and untraceability, and bridging IP addressing realms with HIP. In the last two sections, we suggest how HIP could be applied to tactical networks, and provide some conclusions.

TACTICAL AD HOC NETWORKS

In general, NATO requirements suggest that tactical networks should be

- designed for joint combined operations at the battle field,
- easy to install and maintain within different network scenarios, and
- backward connected to legacy WAN systems. [4]

Tactical networks consist of a combination of semi-static, slowly moving, and rapidly moving devices. There is a desire to secure the networks to prevent eavesdropping, and typically multiple independent levels of security are provided. There are also some conflicting desires on host identification. On one hand, there is a desire to be able to identify computers in the network in a manner that cannot be spoofed, for

the purposes of access controls and traffic prioritization. On the other hand, there is a desire to prevent eavesdroppers from discerning the whereabouts of the important nodes. Therefore, the system must employ strong identity authentication in combination with obfuscation techniques.

Problems in current practice

There are several problems in current commercial Internet technologies that need to be resolved. First, in the current systems there is a strong tendency to use IP addresses as endpoint identifiers, and make authorization decisions based on the IP addresses of the peers. This clearly breaks down in both mobile environment, and in multihoming environment (which is increasingly of interest to tactical hosts who want path diversity), and is basically difficult to deal with from a preplanning or provisioning standpoint, because one cannot perform dynamic address allocation.

Second, home-agent based solutions to mobility, such as Mobile IP [11] and Mobile IPv6 [5], are fragile. In fact, the return routability test required by the commercial Mobile IP route optimization solutions brings this fragility to route optimisation, as the home agent needs to be reachable at least time to time. A more direct authentication of hosts for mobility purposes is desired.

Third, while it is desirable to allow IP address based access control in order to support current system, it would be desirable to provide access control based on strong cryptography. Preferably, such a system not only allows access control of hosts or servers, but also access control as to who is even allowed to have a packet floating around on a particular network segment.

Fourth, many of the current security protocols open a direct venue for CPU exhaustion denial-of-service attacks by sending in garbage.

Finally, there is the desire to limit the possibilities for traffic analysis even by legitimate parties. Information about the current IP addresses (and therefore the location) of important units should not be visible to parties that are not involved in direct communication with them.

HOST IDENTITY PROTOCOL (HIP)

The Host Identity Protocol (HIP) [6, 7] separates location and identity by defining a new *Host Identity* namespace between the transport and internetworking (IP) layers. Figure 1 provides a comparison between the current and HIP architectures. In the current architecture IP addresses represent both location (for routing) and identity along with port numbers through sockets (for processes).

The new HIP architecture is depicted on the right side of the figure 1. The transport layer sockets are now named with separate host identities, which the Host Identity layer translates to one or more IPv4 or IPv6 addresses. This binding between Host Identities and IP addresses is simultaneously dynamic and one-to-many, providing for mobility and multihoming, respectively. Both of these features make IP level traffic analysis protection easier to achieve.

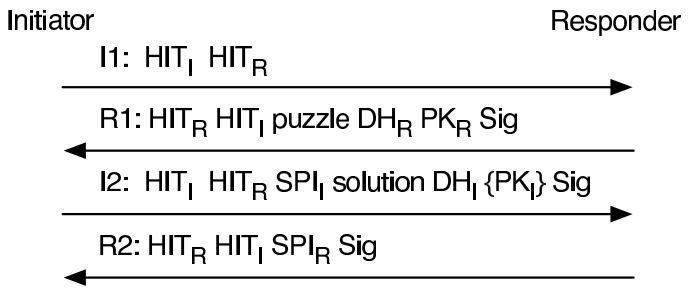


Figure 2: HIP base exchange

Each host generates one or more public/private key pairs to provide identities for itself. The public keys act as *Host* or *End-Point Identifiers*. A host can prove that it corresponds to the Host Identity by signing some data with the (non-disclosed) private key. All other parties can use the Host Identity (a public key) to authenticate the host.

A *Host Identity Tag* (HIT) is a 128-bit representation for a host identifier. It is created by taking a cryptographic hash of the public key. There are two advantages of using a hash over using the public key as such. First, its fixed length makes protocol coding easier. Second, it presents a consistent format for protocols, independent of the public key technology.

The introduction of new cryptographical end-point identifiers clarifies the role of IP addresses. When HIP is used, IP addresses become pure topological labels, naming locations in the Internet. An end-point may change its IP address without breaking connections. Thus, the relationship between location names and identifiers becomes dynamic.

HIP base exchange

The Host Identity Protocol (HIP) [6] consists of a two-round-trip, end-to-end Diffie-Hellman key exchange protocol (called base exchange), a mobility management protocol, and some additional messages. The purpose of the HIP base exchange is to create assurance that the peers indeed possess the private key corresponding their host identifiers. Additionally, the exchange creates a pair of IPSec Encapsulated Security Payload (ESP) security associations (SAs), one in each direction.

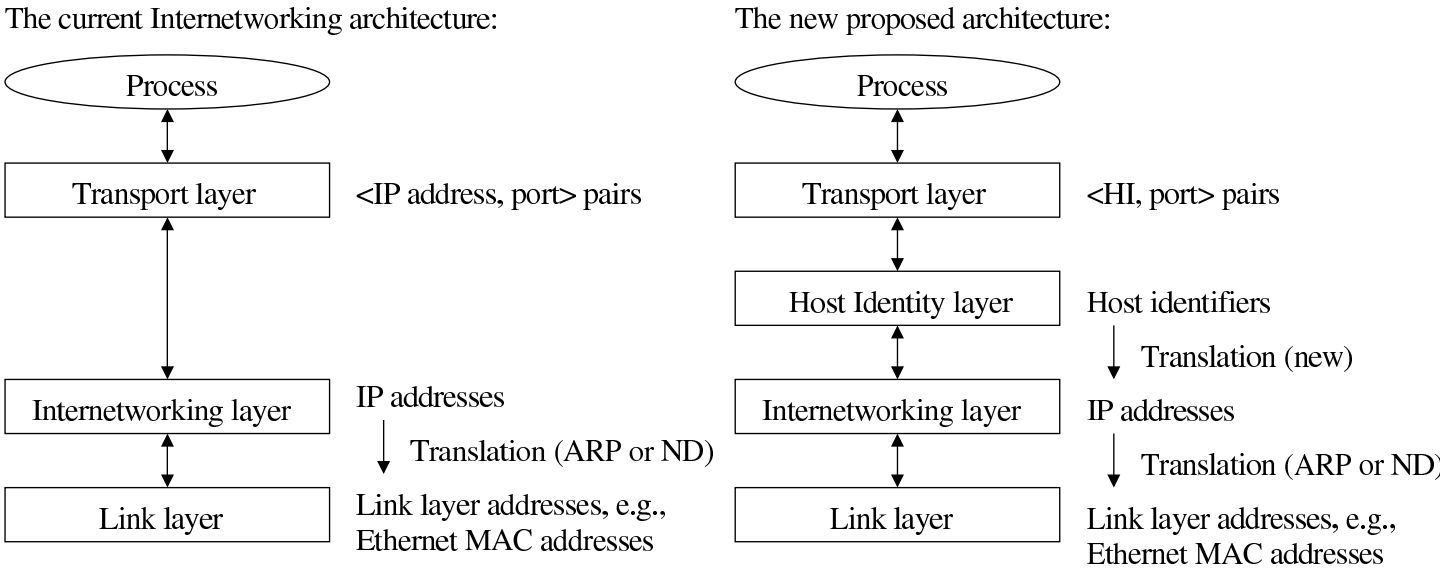


Figure 1: The current Internetworking and the proposed new architectures

The base exchange consists of messages I1, R1, I2 and R2. The HIP base exchange is illustrated in Figure 2. Each HIP message consists of fixed fields, including the HITs of an initiator and a responder, followed by a number of variable length parameters. The first packet, I1, contains only the fixed header, i.e., the HITs. If the initiator does not know the responder's HIT, it may leave that field empty. If so, the reponder is free to select among any of its identities.

When the responder receives an I1 packet, it selects a suitable R1 packet from a pool of precomputed messages. As DoS resistance has been one of the main design goals in HIP, the responder maintains a pool of pre-computed and signed R1 packets, allowing it to pick a pre-computed message instead of constructing one. To facilitate this, the initiator's HIT is not included in the R1 signature.

The R1 message contains a puzzle that the initiator has to solve. The same message also initiates the Diffie-Hellman exchange. It contains the responder's host identity public key, together with the Diffie-Hellman public key and other Diffie-Hellman parameters. From the traffic analysis point of view, it is important to notice that the responder is not able to form the session key before the I2 packet arrives. Therefore, the responder's host identity public key is currently transmitted in clear.

Upon receiving R1, the initiator solves the puzzle, computes a session key, and sends I2. I2 includes the puzzle solution, Diffie-Hellman parameters, SPI, and the initiator's host identity public key. The host identity public key is encrypted using the session key.

The responder verifies that the puzzle is correctly solved, creates the session key, authenticates the initiator, and creates session state. The final message, R2, contains the responder's SPI and a signature. The signature allows the initiator to complete the authentication procedure.

The HIP negotiation results in the parties having an authenticated Diffie-Hellman secret, $\mathrm{KEY_{DH}}$. The HITs and the Diffie-Hellman secret are used to generate key-material in the following way:

$$\begin{aligned} \mathrm{KEY}_1 &= \mathrm{SHA1}(\mathrm{KEY_{DH}}|\mathrm{HIT_{smaller}}|\mathrm{HIT_{larger}}|1) \\ \mathrm{KEY}_2 &= \mathrm{SHA1}(\mathrm{KEY_{DH}}|\mathrm{KEY}_1|2) \\ \mathrm{KEY}_n &= \mathrm{SHA1}(\mathrm{KEY_{DH}}|\mathrm{KEY}_{n-1}|n) \\ \mathrm{KM} &= \mathrm{KEY}_1|\mathrm{KEY}_2|...|\mathrm{KEY}_n \end{aligned}$$

The actual keys, used in encryption and integrity protection, are derivered serially from this key-material. It is important to notice that both of the peers must know both the HITs and the shared Diffie-Hellman secret before they become able to encrypt or decrypt anything. Since the HITs are sent as plain text in the base exchange messages, this is not a problem in the current HIP protocol. However, in [15] it is shown how to *blind* the HITs; see also the Section on Untraceability. The blinding could play an essential role in traffic analysis protection.

New semantics for IPSec

It is important to notice that HIP does not change the IP or IPsec packet structure. However, it modifies

the details of packet handling within the end-nodes. On the other hand, at the logical level, the new name space imposes changes to the *logical* packet structure. That is, each packet must logically include both the end-point identifiers and IP addresses of the sender and recipient. However, when IPsec is used, the Security Parameter Index (SPI) values can be used as *indices for end-point identifiers*, resulting in packets that are syntactically identical to those used today.

Since the packets are integrity protected with ESP, the recipient is always able to verify that a received packet was sent by the peer, no matter what the source and destination addresses are. Thus, by binding the IPsec security associations to public keys instead of IP addresses, the destination address becomes purely routing information. Only during the base exchange, when the hosts have not authenticated each other, and during re-addressing, does the source address play a substantial role. Once the peer hosts have secure bindings between the public keys and IP addresses, the source address is no more needed by the hosts, and its function reduces to carrying information about the topological path the packet has taken [2].

MULTI-ADDRESSING AND MOBILITY

Once the HIP base exchange has been completed and the security associations are in place, the end-points can inform their peers about the interfaces they have and the current IP addresses assigned to the interfaces. This is useful, when a host has either multiple addresses, or when a host has moved into a new location and received a new IP address. The mechanism is defined in the HIP re-addressing protocol [9]. The protocol proposal consists of Re-address (REA) and New SPI (NES) packets.

With a REA packet, the mobile node informs its peer about its IP addresses. The peer optionally responses with a NES packet, containing a new SPI, that is used to verify that the mobile node is indeed in the claimed location. The third message, ESP to the new SPI, acts as a response to the NES. The purpose of the NES/ESP message pair is to prevent legitimate mobile nodes from inducing flooding attacks. The NES/ESP check is optional based on the level of mutual trust in the network.

The REA packet contains information about interfaces and corresponding IP addresses. It includes a signature. The optional NES packet is used to implement a reachability test procedure for each IP address (corresponding to the Return Routability (RR) test in Mobile IPv6). Each end-point has complete freedom to select which interfaces and IP addresses to announce to the peer. All that the peer needs to know is that the announcing end-point is indeed reachable through the claimed IP addresses. Note that the above approach allows hosts to move around, change IP addresses, and have multiple active IP addresses, without inhibiting the ability of peer hosts to authenticate to whom they are talking.

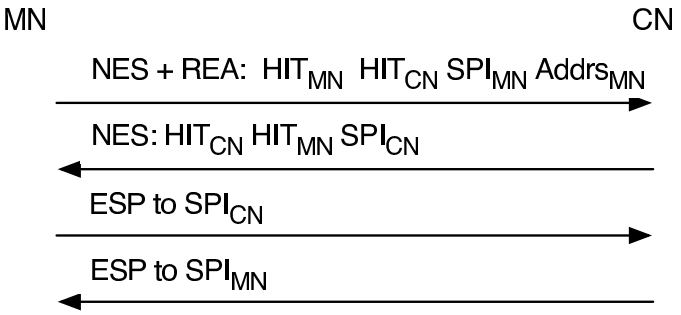


Figure 3: HIP mobility exchange

HOST IDENTIFIERS, ACCESS CONTROL, AND UNTRACEABILITY

Host Identities are not present in every data packet, and the data packets are merely integrity protected, not authenticated. It is likely too burdensome from a computational standpoint to sign every data packet. Nevertheless, there exists a mechanism whereby firewalls and other devices that perform access control can authenticate data flows and regulate which flows (and thereby which hosts) are allowed to access a particular network segment.

The key to this approach is to construct HIP-aware firewalls that observe the HIP base exchange and re-addressing exchanges. These HIP exchanges have been explicitly designed to allow firewalls and other middleboxes to observe the required fields. These firewalls can authenticate the (signed) HIP control packets, and then observe which IP addresses and SPIs the protocol negotiates to include. Thereafter, the addresses and SPIs serve as a proxy for the HITs in the subsequent data packets. This approach is much more flexible than relying on IP addresses for access controls, as is typically done, although since the HIT name space is flat, there is no opportunity to aggregate hosts behind a single prefix.

In [15], Ylitalo et. al. introduce a technique called *BLIND* where the real identity of HIP hosts can be completely hidden from eavesdroppers while still retaining the identity authentication properties of the protocol. The idea is based on using temporary, obfuscated Host Identity Tags (HITs) in the place of the permanent, well known ones. Since the goal is to make the temporary HITs non-sensible to eavesdroppers, any nodes that need to be able to detect the real identity of the communicating nodes must be *preconfigured* with the iden-

tities of the potential peer hosts. While this may be a problem in a commercial open network, this is typically not a problem for firewalls or end-nodes in a military setting, where the identities must be preconfigured anyway. The temporary HITs can be changed into different ones whenever a host moves, making tracking virtually impossible.

The basic idea in [15] is to replace the real HITs with a hash of the real HIT and a random nonce. The resulting temporary HIT and the nonce are carried in the initial protocol messages. All nodes that have the real HIT in their possession can find it by a simple iterative search, while nodes that do not possess the real HIT face a computationally impossible problem. For nodes configured with a large number of potential HITs, the initial packets can carry a hint, thereby reducing the required search time.

In summary, when the BLIND approach is used, it is possible to achieve the conflicting goals of strong, cryptographic identity authentication while protecting the identities from eavesdropping outsiders.

HOST IDENTIFIERS, NAT, AND EPHEMERAL IP ADDRESSES

The purpose of network address translation (NAT) is to bridge different IP addressing domains. The most common need for NAT is the use of private IP address space (because of a shortage of IPv4 addresses), but there are also other motivations, like address stability. Basically, any NAT approach makes it possible for a middle box to change the IP addresses of in a packet without breaking end-to-end communications. In standard NAT today, the transport layer identifiers, i.e., ⟨IP-address, port⟩ pairs, are used as static identifiers. However, this is problematic because the transport level identifiers ⟨IP address, port⟩ and network layer addresses ⟨IP address⟩ are smeared together.

When location names and host identifiers are separated, as is done in HIP, the new global name space can be used for static transport layer identifiers. As a result, there are several advantages for using Host Identity name space with NAT. First, a NAT device can easily identify connections using the Host Identities. This means that it becomes possible to inititate connections through a NAT device in both directions[1]. Second, the introduction of a name space allows IP address changes even between IPv4 and IPv6, because higher level protocols use Host Identities rather than IP addresses.

A HIP enabled NAT device translates IP addresses, using the HITs as identifiers for the connection state. However, the HITs are not present in the regular traffic packets between two HIP hosts. Instead, the IPsec SPI is used as an index to the NAT state. If it uniquely identifies the state, as can be fairly easily arranged it may take the place of HITs for handling regular data packets. However, since there may be several HITs behing a single public IP address, the NAT device must learn the SPI values during the initial HIP base exchange, or during mobility signaling. Using 32-bit SPI values instead of 16-bit port numbers also increases the number of connections that can be maintained using a single IP address.

In [14], Ylitalo et. al. propose a new NAT concept called SPINAT: SPI multiplexed NAT. It works in the same way as a regular NAT-PT but uses SPI numbers instead of port numbers. A SPINAT device learns the SPIs and HITs by inspecting HIP base exchange and/or HIP mobility signaling. It can do this securely as there are signatures present in the packets. If a given SPI value is already in use, the SPINAT device may securely replace it with a unique one. Alternatively, if it has multiple public IP addresses, it can assign conflicting SPIs on different public IP addresses, and use the ⟨address, SPI⟩ pair as an index to the translation state.

The SPINAT technique does not require any tunneling headers. The advantage in packet size compared to the current Mobile IP based solution is substantial.

If we compare the HIP based NAT mechanism to routing, there are some similarities. A HIP based NAT device changes IP addresses while using the upper layer state as an index, just like a router changes link layer addresses using the IP address as an index. The difference is in how the state is created: in the case of IP layer routing, the forwarding state is created as a result of running routing protocols, while in HIP "routing" the state is created by inspecting HIP control packets.

HIP IN TACTICAL ENVIRONMENTS

To utilize HIP in tactical environments, we propose an approach based on the following principles.

- Utilize HITs as host identifiers, allowing usage of current IP address based access control mechanisms with strong security controls. To prevent location tracking, combine this with the BLIND approach [15].

- Use a public key infrastructure (PKI) for identities that can divide participants of the network into different trustworthiness classes (for example, our own troops of different kind, allies, and neutrals who need to access different parts of our network). Such a PKI must support fast revocation, must be decentralized, and must tolerate network partition. While leaving the design of such a PKI for future

[1] This requires that the NAT device is able to map the HIT to a private IP address. This is likely to be the typical case when HIP is used with NAT.

work, we envision that it could be based on a partitioning tolerant Distributed Hash Table (DHT) design.

- Use HIT based IPv6-like ad hoc routing in small networks and within a single cluster, solving the ad hoc network addressing and Duplicate Address Detection (DAD) problems. In larger and more stable networks traditional IPv4 and IPv6 addressing and routing can be used.

- Use the SPINAT approach [14] to pass packets between addressing domains. In this context, an addressing domain may be an ad hoc network (using HITs as addresses), a cluster in a larger ad hoc network, or any other independently managed network. This allows HIT based ad hoc domains and more traditional IP address based domains to be combined.

- Use the signalling delegation approach by Nikander et. al. [8] to reduce mobility signalling within an addressing domain.

While the details of the approach need more work, especially in the PKI area, the foundation appears to be solid. Using HITs as host identifiers has been shown to work [10]. Using HITs instead of IP addresses in an ad hoc network is straightforward as the typical ad hoc routing protocols assume pre-defined, unstructured, stable address space [12]. The SPINAT approach is very similar to the IPNL approach [3] by Francis et. al. while using ESP for tunneling and HIP for soft state management in the middle boxes. Finally, the signalling delegation approach [8] is a straightforward application of the more generic trust management approaches, including SDSI/SPKI and KeyNote2 [1].

The approach is demonstrably less fragile than Mobile IP [10]. In particular, no fixed home agents are needed. To facilitate fast movement and to solve the simultaneous movement problem, a Forwarding agent can be used to keep track the current IP addresses of a mobile host [10]. As discussed recently at the 59th IETF meeting [13], basically any node can act as a forwarding agent for other nodes that it has a connection with. This can act as a starting point for designing a robust rendezvous infrastructure that works well even under network partitioning and intermittend connectivity.

CONCLUSIONS

The Host Identity Protocol (HIP) is a promising new protocol proposal currently under discussion at the IETF. Additionally, a number of research projects are considering HIP as an architectural component. There are currently five publicly known implementations of the HIP base protocol, three of which are distributed under open source or compatible licenses (OSS); see Table 1.

Table 1: HIP implementations

Boeing Phantom Works	Linux	
Ericsson Research Nomadiclab	FreeBSD	OSS
Helsinki University of Technology	Linux	OSS
Indranet technologies	Python	OSS
Sun Research Grenoble	Solaris	

In summary, it can be seen that HIP can solve the problems identified above, namely:

- host identifiers can be used with strong security guarantees instead of IP addresses, thereby allowing IP addresses to change over time without disrupting communications;

- notifying a peer of an IP address change due to mobility can be done directly with no communications through a home network;

- HIP allows firewalls to cryptographically authenticate which hosts have packets on a given network segment;

- HIP has been designed to minimize vectors for denial-of-service attacks;

- since intermediary routers and firewalls can change the IP addresses, tracking IP addresses brings relatively little benefit to an eavesdropper. The focus is moved to end-point-identifiers, such as public keys and HITs; and

- extensions to HIP should allow hosts to protect their identities from eavesdroppers while still authenticating themselves to each other.

ACKNOWLEDGEMENTS

The authors would like to express their gratitude to Mats Naslund and Satu Virtanen, and especially to Thomas R. Henderson, for their constructive comments and suggestions on various versions of this paper.

References

[1] T. Aura. *Distributed access-rights management with delegation certificates*, pages 211–235. Number 1603 in LNCS. Springer, 1999.

[2] C. Candolin and P. Nikander. IPv6 Source Addresses Considered Harmful. In *Proc. NordSec 2001*, Nov. 2001. Sixth Nordoc Workshop on Secure IT Systems, Lyngby, Denmark.

[3] P. Francis. Ipnl: A NAT-extended Internet architecture. In *Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 69–80. ACM Press, New York, NY, USA, 2001.

[4] ISSC NATO Open Systems Working Group. *Section 3.1.1. General Requirements*, chapter 3. Dec. 2003.

[5] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. Internet Draft, work in progress, June 2003.

[6] R. Moskowitz and P. Nikander. Host Identity Protocol. Internet Draft, work in progress, June 2003.

[7] R. Moskowitz and P. Nikander. Host Identity Protocol Architecture. Internet Draft, work in progress, May 2003.

[8] P. Nikander and J. Arkko. Delegation of signalling rights. In *Security Protocols*, LNCS, 2003. Cambridge Security Protocols Workshop, April 2002.

[9] P. Nikander and J. Arkko. End-Host Mobility, Multi-Homing and NAT Travelsal with Host Identity Protocol. Internet Draft, work in progress, May 2003.

[10] P. Nikander, J. Ylitalo, and J. Wall. Integrating Security, Mobility, and Multi-Homing in a HIP Way. In *Proc. Network and Distributed Systems Security Symposium*, Feb. 2003. NDSS'03, San Diego, CA, USA.

[11] C. Perkins. IP Mobility Support. RFC 2002, 1996.

[12] C. Perkins. *Ad Hoc Networking*. Addison-Wesley, 2000.

[13] T. Shepard. Some thoughts on hip rendezvous. In *Proceedings of the 59th IETF Meeting*, Mar. 2003.

[14] J. Ylitalo and P. Nikander. Spinat: Spi multiplexed nat for secure and efficient mobility. Unpublished manuscript.

[15] J. Ylitalo and P. Nikander. Blind: A complete identity protection framework for end-points. In *Security Protocols*, LNCS, 2004. Cambridge Security Protocols Workshop, April 2004.

Dynamic Local Clustering for Hierarchical Ad Hoc Networks

Satu Elisa Schaeffer
Laboratory for Theoretical Computer Science
Helsinki University of Technology
P.O. Box 5400, FI-02015 TKK, Finland
elisa.schaeffer@tkk.fi

Stefano Marinoni
Laboratory for Theoretical Computer Science
Helsinki University of Technology
P.O. Box 5400, FI-02015 TKK, Finland
stefano.marinoni1@studenti.unimi.it

Mikko Särelä
Laboratory for Theoretical Computer Science
Helsinki University of Technology
P.O. Box 5400, FI-02015 TKK, Finland
mikko.sarela@tkk.fi

Pekka Nikander
Nomadiclab
Ericsson Research
Jorvas, Finland

Abstract

Hierarchical, cluster-based routing greatly reduces routing table sizes compared to host-based routing, while reducing path efficiency by at most a constant factor [9]. More importantly, the amount of routing related signalling traffic is reduced [7, 11, 19]. On the other hand, address changes caused by nodes changing their cluster produces address management traffic. In this paper, we present a new local clustering method that produces dense and stable clusters, thereby minimizing address changes and allowing better and more stable network conditions for ad hoc routing.

1 Introduction

When clustering is introduced to an ad hoc routing system, locally computable clustering is a necessity in order to avoid generation of excess control traffic. In the ideal case, each arriving node is able to determine the appropriate cluster simply by consulting its immediate neighbors, who will not need to communicate further to determine the best cluster. Proposals for and analysis of cluster-based routing in dynamic networks include [10, 19].

Within a clustered network, routing can be divided into two subproblems: finding a route of clusters to the destination node and finding a route within each cluster either to the next cluster or to the destination node within the cluster. If two previously disconnected clusters become connected or vice versa, the inter-cluster routing is affected. Desirably inter-cluster connectivity changes are rare and nodes only switch from one cluster to another in order to minimize intra-cluster routing and maintenance costs. Avoiding cluster changes helps stabilize routing by cluster hops in comparison to routing based on individual links.

It is common for many clustering algorithm proposals that nodes are at most two hops away from the members of their corresponding clusters [2, 4, 5, 6, 12]. Methods differ for example in the usage of cluster heads and possible cluster overlaps. Ohta et al. [16] propose a clustering algorithm similar to the one presented in this paper, where the clusters are chosen from neighboring ones, bounding the size of each cluster. Our contribution is in choosing the clusters based on a particular method for local graph clustering that helps achieve dense clusters [18]. Our clustering protocol does not impose explicit constraints on the cluster diameter and hence the intra-cluster hop counts are not limited. The goal is to produce such a clustering where topology changes are concentrated *inside* clusters and changes in inter-cluster connectivity are avoided.

We aim at clusters with high local density and only few links to the rest of the network desirable as they simplify the routing task. Link state algorithms, such as OLSR [3], require dense and relatively small networks in order to be efficient [17] and perform well for intra-cluster routing with dense and stable clusters. Inter-cluster routing, on the other hand, may well use on-demand routing protocols that construct routes based on cluster hops and gain the advantage of more stable routes, as the clustering hides many route-breaking topology changes that occur within single clusters.

2 Cluster fitness

In this paper, we model ad hoc networks as dynamic graphs, consisting of nodes and edges (bidirectional links). The focus is on the clustering protocol. We use a graph-theoretical fitness measure [18] to locally select the cluster of an arriving node. We adopt the following notation to define the fitness measure used: in a graph $G = (V, E)$, a cluster candidate is a set of nodes $C \subseteq V$, and the set of edges of the subgraph induced by C is $E_c = \{(m, n) \in E \mid m, n \in C\}$. The *size* of the cluster is the number of nodes included in the cluster, denoted by $|C|$. The (local) *density* $\delta_\ell(C)$ of a cluster C is $|E_c| / \binom{|C|}{2}$ for clusters with more than one node and zero otherwise. The density of the entire graph $\delta(G)$ is simply $|E| / \binom{|V|}{2}$. Clusters for which $\delta_\ell(C) \gg \delta(G)$ can be considered good. The *relative density* $\delta_r(C)$ [14] is defined in terms of the *internal degree* $\deg_{\text{int}}(C) = |E_c|$ and *external degree*

$$\deg_{\text{ext}}(C) = |\{(m, n) \in E \mid m \in C, n \in V \setminus C\}| \quad (1)$$

of a cluster candidate C as the fraction of the internal degree of the total number of edges incident on the cluster. It is commonly acknowledged that a good graph cluster should have many edges connecting the included nodes to each other, and as few as possible connecting the cluster to the rest of the graph, and hence, high relative density [8, 14]. We want each node to be connected to each member of their cluster by at least one path *within* the cluster, preferably directly linking to many cluster members, and linking to only few nodes outside its cluster. The first criterion is fulfilled if only connected subgraphs are considered as cluster candidates. We choose to optimize the product of the relative and local densities to achieve clusters that fulfill the other two criteria:

$$f(C) = \frac{2 \deg_{\text{int}}(C)^2}{|C|(|C| - 1)(\deg_{\text{int}}(C) + \deg_{\text{ext}}(C))}. \quad (2)$$

With respect to this measure, a good cluster is both dense and "introvert", and the combination avoids counterintuitive clusterings produced by optimizing either one of the two density measures alone.

3 Clustering protocol

The clustering algorithm initiates, e.g. after the node first wakes up, by probing the neighborhood with a CLUSTER REQUEST message to which all neighboring nodes respond with a CLUSTER REPLY. The response message consists of the node identifier, cluster identifier and three integers: the number of nodes in the cluster $|C|$, the internal degree $\deg_{\text{int}}(C)$ of the cluster, and the external degree $\deg_{\text{ext}}(C)$ of the cluster. If no responses arrive, the node creates a singleton cluster and becomes its cluster head. This allows the clustering to initialize in a distributed fashion.

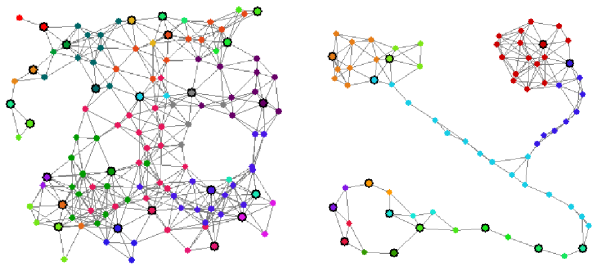


Figure 1. Stationary nodes with fixed range have been added one by one, with existing nodes updating their clusters (indicated with colors) after the newcomer selects a cluster. Cluster heads have a black border. On the right, a more anomalous network structure.

If replies do arrive within a beacon frame, the node chooses among the neighboring clusters by optimizing the change in cluster fitness, choosing the cluster for which its join would cause the highest increase (or smallest decrease). The node declares its selection by broadcasting a CLUSTER JOIN message containing the cluster identifier of the chosen cluster. Upon the creation of a singleton cluster, the node sends a CLUSTER JOIN message containing the cluster identifier it chose.

The CLUSTER REQUEST and CLUSTER REPLY messages are then used periodically to maintain up-to-date neighborhood information and to make decisions of leaving and joining clusters. Generally, a node only performs a cluster switch (through a join operation) when it is *quality-increasing*: a node i executing the cluster-selection protocol switches from its current cluster C_i to another cluster C if the sum of the cluster fitnesses C_i and C grows as i switches from C_i to C.

In addition, we impose upper and lower bounds on the cluster sizes so that nodes primarily choose clusters that are within the bounds. If there are no such neighboring clusters, a node prefers clusters below the lower bound, and in their absence, then will create a new cluster. No node may join a cluster whose size is at or above the upper bound.

A node stays in the same cluster until it either announces a join to another cluster or the cluster splits. The periodic cluster request and subsequent cluster join messages are the basic cluster maintenance mechanisms. We utilize the cluster-head status of a node in coping with cluster splits and having the cluster heads periodically broadcast keep-alive messages that are flooded only within the respective clusters. The lack of a keep-alive message indicates to a node that it has disconnected from its cluster head and it

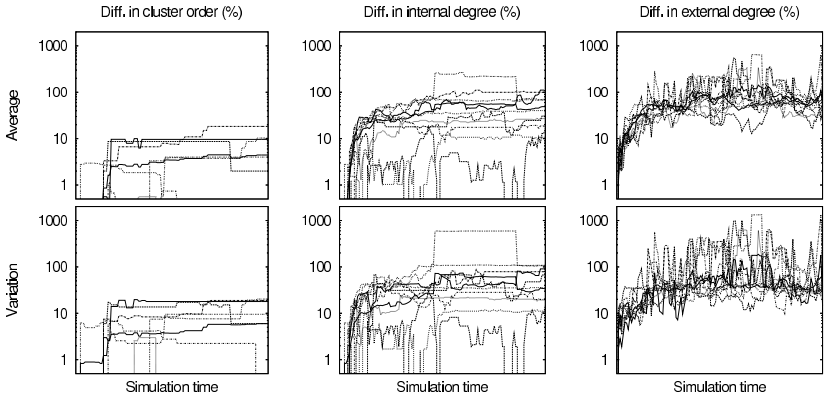


Figure 2. Differences in $|C|$ (left), $\deg_{\text{int}}(C)$ (middle), and $\deg_{\text{ext}}(C)$ (right) over a set of $N = 10$ runs of $D = 250$ seconds. Each line relates to a single run. The upper plots show the average difference $Avg_p = 100 * 1/30 \sum_{i=1}^{30} \frac{abs(E_p(i) - V_p[C(i)])}{V_p[C(i)]}$ over time, for $p \in \{$size,in,out$\}$, and the lower plots the corresponding deviation.

must reinitiate the cluster selection protocol.

4 Experiments

We sketched a small-scale simulator to visualize clusterings [20] (examples shown in Figure 1). We also built an `ns-2` implementation [13] of the algorithm for larger scale experiments. Our experiments with simulation tools are promising: the clusters achieve a proper sense of locality in space and their structure corresponds well to the intuitive global clusterings of the network.

In the `ns-2` simulations, we used networks of 30 nodes in a one square-kilometer area. The minimum cluster order was set to five and the maximum to eight nodes; the simulator was very slow for larger networks. Each node probed its neighborhood, with a range of 250 meters, on five-second intervals and the cluster heads broadcasted a status message for intra-cluster flooding on five-second intervals.

4.1 Effects of outdated information

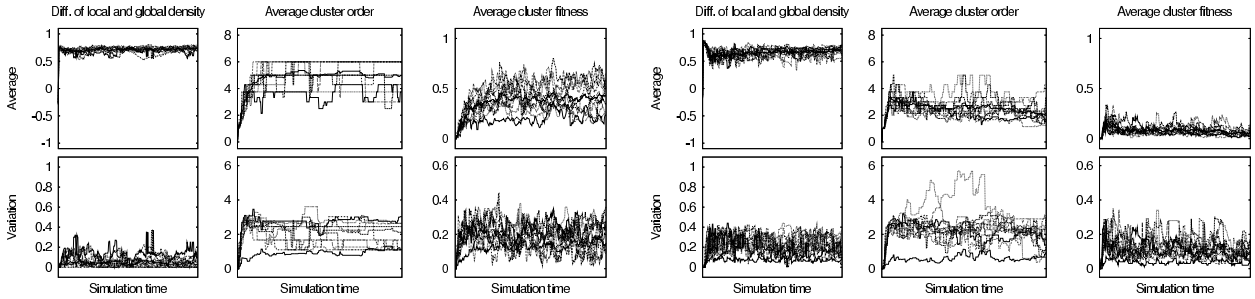
Observing the behavior of the clustering method on the simulators, it also seems feasible to approximate the fitness function using estimates of $|C|$, $\deg_{\text{int}}(C)$ and $\deg_{\text{ext}}(C)$. Such "lazy updates" would allow for a more relaxed control traffic within the cluster, as not all nodes need to be immediately aware of newcomers, departing nodes, or changes in edges. The effects of outdated information can be deduced from the fitness function (Equation 2); the magnitude of the difference between the actual value, and the assumption made at a single node depends on the rate of change in the clustering, as well as the frequency with which updated information is propagated in the network.

We traced a set of `ns-2` runs and computed at each time step the true values of the above measures and compared those to the "belief" of each node, calculating the distance in percentage of the real value. Formally, in every instant of time, every node i belongs to a precise cluster $C(i)$. This cluster has its order $V_{\text{size}}[C(i)]$, its internal degree $V_{\text{in}}[C(i)]$ and its external degree $V_{\text{out}}[C(i)]$; they are the actual values. Likewise, at every instant of time, each node i, holds its estimated values respectively as $E_{\text{size}}(i)$, $E_{\text{in}}(i)$, $E_{\text{out}}(i)$.

Figure 2 shows that the estimate for cluster size does not diverge over time and the internal degree often "restores" the correct value, but the estimates for the external degree remain far from the true value. However, we seem to achieve a practical clustering even with the problems in determining the external degree. With additional control overhead, the accuracy could be improved.

One reason for the problematic estimation of the external degree is that in cluster splits, there is a risk that the original cluster will not notice the departure of some nodes. In situations where splits are frequent and the departing nodes will often become completely detached from the old cluster, not even remaining in the neighborhood, the cluster heads should send out time-stamped beacon messages containing the cluster member list that are propagated by broadcast within the respective clusters, and the member nodes respond (through a broadcast tree formed by the order in which the nodes received the beacon message from each other) by stating which of those members are currently their neighbors and how many other neighbors they have.

Such a mechanism allows for the entire cluster to main-



Group-mobility model

Random way-point model

Figure 3. The difference of the average local density and the global density, cluster order, and cluster fitness averaged over the set of clusters for each time step in $\{0, 2, 4, \ldots, 600\}$ (average over the 10 runs drawn thick).

tain a more up-to-date view on the cluster topology. The cluster head should not send out a new beacon before it receives the replies to the previous ones; the waiting time should be reset upon the arrival of a reply and the computation of the current values should only be done after a timeout occurs with no further reply arrivals. If however the cluster head receives a reply *after* the timeout, it should increase the waiting time for the next beacon round. A mechanism for reducing the time if all replies arrive quickly could also be included. Note that by adding a hop counter to the beacon messages, incremented by each forwarding node, nodes can include the value of the counter upon their first reception of the message to their replies and thus inform the cluster head of their "effective" distance from the cluster head; this information could also be used to adjust the waiting time at the cluster head.

As described above, cluster formation is based on an exchange of simple messages that contain the cluster identifier and three integers: the size of the existing cluster, the internal degree of the cluster, and the external degree of the cluster. If a link state routing protocol is used within the cluster, the nodes can use link state information to produce the current values, and do not need to exchange any extra messages for intra-cluster information. Using these figures, together with information about the new or deleted edges, each node under consideration is able to estimate the cluster quality for each cluster candidate.

For moderately sized clusters (at most 256 nodes) and 64-bit cluster identifiers, all of the required information can be fit into 16 bytes. This could be included in existing link-layer frames, IP layer address resolution, neighbor-discovery messages, routing messages, or in Wireless LAN beacon frames.

4.2 Cluster quality

We studied the quality of the clusterings produced by a series of `ns-2` simulations, studying cluster density, fitness, and stability as the main indicators. We ran $N = 10$ simulations with 30 nodes. The mobility models utilized were reference-point group mobility (GM) model with nodes moving in small groups, random direction (RD) model, random walk model (RW), and random way-point (RWP) model [1, 15].

In all our scenarios the nodes move with speed uniformly distributed in $[0, 15]$ m/s after an initial period of $[0, 5]$ seconds. In GM, each individual node moves as in RWP, but within a restricted area of 200m^2 surrounding the group imaginary reference point, while reference points also move as in RWP, but within the whole simulation area. For RW, nodes change direction on one-second intervals.

We report averages, and variations of some measured indicators; formally, we denote the average of a set of k values $\{y_1, y_2, \ldots, y_k\}$ as $\mathrm{Avg}\,[y_i]_1^k = \frac{1}{k}\sum_1^k y_i$, and the variation as:

$$\varrho\,[y_i]_1^k = \sqrt{\frac{\sum_{i=1}^{k}\left(\mathrm{Avg}\,[y_i]_1^k - y_i\right)^2}{k}} \qquad (3)$$

where, k denotes the cluster count at a certain time step, and $y_i = m(C_i)$ is an instantaneous measure (concerning the cluster C_i) for a certain metric m. With regard to our clustering algorithm, we considered particularly significant to monitor the overall conditions (average over all clusters) in terms of density, order and fitness. Hence, the metrics measured over a period of $D = 600$ seconds were the following, with a measurement taken for each time step $t \in \{0, 2, 4, \ldots, 600\}$: the local density of the clusters versus the density of the graph $(\delta_\ell\,(C_i) - \delta\,(G))$, the cluster order$|C_i|$, and the cluster fitness $f\,(C_i)$. For the density difference, the range is $[-1\,, 1]$ and a positive value indicates

Table 1. Measures of graph (Equation 4) and cluster stability (Equation 5) for the mobility models (MM), averaged over $N = 10$ experiments of duration $D = 600$ seconds.

MM	$\tilde{\mathcal{B}}/D$	$\tilde{\mathcal{B}}_{\text{int}}/D$	$\tilde{\mathcal{E}}/D$	$\tilde{\mathcal{E}}_{\text{int}}/D$	$\tilde{\mathcal{T}}/D$	$\tilde{\mathcal{T}}_{\text{int}}/D$	$\tilde{\mathcal{Q}}$	$\tilde{\mathcal{F}}$	$\tilde{\mathcal{S}}$	$\tilde{\mathcal{T}} \cdot \tilde{\mathcal{S}}$
GM	1.53	0.25	1.55	0.21	3.08	0.46	0.03	0.01	0.04	77
RD	1.04	0.43	1.06	0.20	2.10	0.63	0.10	0.08	0.18	227
RW	1.13	0.47	1.15	0.42	2.28	0.90	0.04	0.02	0.07	89
RWP	1.50	0.59	1.51	0.26	3.01	0.86	0.09	0.07	0.16	289

that dense subgraphs have been selected as clusters; if the value is close to one, almost all links present in the graph are internal to some cluster. For the cluster order the range is [0,8], its value over time it is a first indicator of cluster stability as stable clusters must have few fluctuations. The fitness varies in $[0, 1]$ with values close to one indicating optimal clusters.

The results are shown in Figure 3; results for RD, and RW mobility models were similar and omitted. All mobility models produced clusters with much higher local density than the density of the entire graph. Unexpectedly, the group mobility model produced large clusters with very high density, whereas group mobility scenario had consistently much better fitness than in other mobility models. Both random way-point and random direction acted similarly, producing small, but dense clusters.

4.3 Cluster stability

We also studied the *stability* of the graph and cluster topologies (results are shown in Table 1), recording the total amount of link breakages $\mathcal{B}_i$ and new link establishments $\mathcal{E}_i$ and the average topology change rate $\mathcal{T}/D$ by considering the graph variations occurred per second in experiment i, $i \in \{1, 2, \ldots, N\}$:

$$\tilde{\mathcal{B}} = \text{Avg}\left[\mathcal{B}_i\right]_1^N, \tilde{\mathcal{E}} = \text{Avg}\left[\mathcal{E}_i\right]_1^N, \tilde{\mathcal{T}} = \tilde{\mathcal{B}} + \tilde{\mathcal{E}}. \quad (4)$$

We additionally recorded the number of topology changes that were *internal* to clusters, denoting these by $\mathcal{B}_{\text{int}}$, $\mathcal{E}_{\text{int}}$, and $\mathcal{T}_{\text{int}}$, respectively [1]. Cluster stability was measured by the number of cluster changes, distinguishing between two categories: $\mathcal{Q}_i$ is the number of quality-increasing cluster switches and $\mathcal{F}_i$ is the number of switches due to a cluster split;

$$\tilde{\mathcal{Q}} = \text{Avg}\left[\frac{\mathcal{Q}_i}{\mathcal{B}_i + \mathcal{E}_i}\right]_1^N, \tilde{\mathcal{F}} = \text{Avg}\left[\frac{\mathcal{F}_i}{\mathcal{B}_i + \mathcal{E}_i}\right]_1^N \quad (5)$$

We denote $\tilde{\mathcal{S}} = \tilde{\mathcal{Q}} + \tilde{\mathcal{F}}$; note that as $\tilde{\mathcal{T}}$ is the average number of topology changes, $\tilde{\mathcal{S}} \cdot \tilde{\mathcal{T}}$ is the average amount of cluster changes in a single simulation run.

[1] Inter-cluster topology changes could be deduced as a difference between $\mathcal{T}$ and $\mathcal{T}_{\text{int}}$

The results in Table 1 show that group mobility model and random walk show have the most stable clustering structure of the four, although the reasons differ. Random walk creates mainly local movements, which means that the overall topology of the graph will tend to stay the same with small variations. It has as low rate of topology changes as random direction, but causes much less changes in the clustering structure. This is due to the local movements of nodes in random walk vs. global movements of nodes in random direction. Group mobility model creates global movements, but with certain groups of nodes staying close to each other. This causes a high rate of changes in the topology, but low rate of changes in the clustering. The random-direction model also produces global movements.

We experienced very few clusters splits and changes in general. Overall, the rate of changes in clustering is small. Group mobility and random walk cause changes in clustering in 4% and 6% of the cases where topology changes and random direction and random way-point models in 16% and 18% respectively.

5 Conclusions

We introduced a new local measure for clustering quality, and outlined a simple protocol for local cluster management. The simulations show that the clustering algorithm is capable of creating a clustering structure, which hides most of the topology changes within the network and thus making inter-cluster routing task easier. The algorithm is capable of capturing the structure that may exist in the movements of nodes. This is especially marked by the group mobility model having the highest rate of topology change, while having least changes in clustering both per topology change and per unit of time. The algorithm also managed to form clusters with high local density, allowing us to partition the network into smaller subnetworks which are easily managed by proactive routing algorithms such as OLSR [3] that are designed especially for dense networks with small diameter.

As future work we plan to study how the proposed clustering algorithm could be used to further optimize routing and address management. On top of a base-layer cluster-

ing, we could form a hierarchy of clusterings with a very similar cluster-formation protocol, relying on routing the higher-level cluster requests to the cluster heads. Such a layering would however introduce additional duties to the cluster heads, but is an interesting area for further work.

Acknowledgments

The first author was supported by the Academy of Finland (under grants 202205 and 206235), the Nokia Foundation, and the Rotary Foundation.

References

[1] T. Camp, J. Boleng, and V. A. Davies. A survey of mobility models for ad hoc network research. *Wireless Communication and Mobile Computing*, 2(5):483–502, Sept. 2002.

[2] C.-C. Chiang, H.-K. Wu, W. Liu, and M. Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. In *Proceedings Of IEEE SICON'97*, pages 197–211, Apr. 1997.

[3] T. H. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR). Technical Report RFC 3626, Internet Engineering Task Force, Reston, VA, USA, 2003.

[4] A. Ephremides, J. E. Wieselthier, and D. Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. *Proceedings of the IEEE ?*, 75:56–73, Jan. 1987.

[5] M. Gerla and J. T.-C. Tsai. Multicluster, mobile multimedia radio network. *ACM-Baltzer Journal of Wireless Networks*, 1:255–265, 1995.

[6] T.-C. Hou and T.-J. Tsai. An access-based clustering protocol for multihop wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 19(7):1201–1210, July 2001.

[7] F. Kamoun and L. Kleinrock. Stochastic performance evaluation of hierarchical routing for large networks. *Computer Networks*, 3:337–353, Nov. 1979.

[8] J. M. Kleinberg and S. Lawrence. The structure of the web. *Science*, 294(5548):1849–1850, Nov. 2001.

[9] L. Kleinrock and F. Kamoun. Hierarchical routing for large networks: Performance evaluation and optimization. *Computer Networks*, 1(3):155–174, 1977.

[10] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster-based approach for routing in dynamic networks. ACM SIGCOMM *Computer Communication Review*, 27(2):49–64, Apr. 1997.

[11] G. S. Lauer. Hierarchical routing design for SURAN. In *Proceedings of the* IEEE *International Conference on Communications (*ICC*)*, pages 93–102, Los Alamitos, CA, USA, 1986. IEEE Computer Society Press.

[12] C. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *IEEE Jour. Selected Areas in Communications*, 15(7):1265–1275, Sep 1997.

[13] S. McCanne, S. Floyd, K. Fall, and K. Varadhan. The network simulator `ns-2`. The VINT project, `http://www.isi.edu/nsnam/ns/`.

[14] M. Mihail, C. Gkantsidis, A. Saberi, and E. Zegura. On the semantics of Internet topologies. Technical Report GIT-CC-02-07, College of Computing, Georgia Institute of Technology, Atlanta, GA, USA, 2002.

[15] J. Nuevo. Mobility generator program for NS-2, 2002. `http://externe.inrs-emt.uquebec.ca/users/nuevo/NSmobgenerator.htm`.

[16] T. Ohta, S. Inoue, and Y. Kakuda. An adaptive multihop clustering scheme for highly mobile ad hoc networks. In *The Sixth International Symposium on Autonomous Decentralized Systems (ISADS'03)*, pages 293–300, Apr 2003.

[17] C. A. Santivanez, R. Ramanathan, and I. Stavrakakis. Making link-state routing scale for ad hoc networks. In *Proceedings of the Second* ACM *international symposium on Mobile ad hoc networking & computing*, pages 22–32, Long Beach, CA, USA, 2001. ACM Press.

[18] S. E. Schaeffer. Stochastic local clustering for massive graphs. In T. B. Ho, D. Cheung, and H. Liu, editors, *Proceedings of the Ninth Pacific-Asia Conference on Knowledge Discovery and Data Mining (*PAKDD*-05)*, volume 3518 of *Lecture Notes in Computer Science*, pages 354–360, Berlin/Heidelberg, Germany, 2005. Springer-Verlag GmbH.

[19] J. Sucec and I. Marsic. Clustering overhead for hierarchical routing in mobile ad hoc networks. In *Proceedings of the Twenty-first Annual Joint Conference of the* IEEE *Computer and Communications Societies*, volume 3, pages 1698–1706, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.

[20] S. E. Virtanen and P. Nikander. Local clustering for hierarchical ad hoc networks. In *Proceedings of WiOpt'04: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pages 404–405, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

Requirements for a security architecture for clustered ad-hoc networks

Licentiate's Thesis

M.Sc Maarit Hietalahti

Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory for Theoretical Computer Science
Espoo 2007

Teknillinen korkeakoulu
Tietotekniikan osasto

Tietojenkäsittelyteorian laboratorio

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF LICENTIATE'S THESIS

Author:	M.Sc Maarit Hietalahti	
Name of thesis:	Requirements for a security architecture for clustered ad-hoc networks	
Date:	11. August, 2007	**Pages:** 4 + 69
Department:	Department of Computer Science and Engineering	**Chair:** Tik-79
Supervisor:	Professor Kaisa Nyberg	
Instructor:	Dr. Pekka Nikander	

Mobile wireless ad-hoc networks are generally thought to consist of only tens or at most a few hundreds of nodes, as their management becomes difficult with growing network size. However, real-life networks often have natural hierarchical structures or clusterings. Such structures can be used to make the management of very large ad-hoc networks a feasible task.

In self-organized wireless ad-hoc networks, the nodes participate in the routing operations and there is no pre-existing infrastructure supporting routing or other network operations. This creates security issues that do not exist in conventional fixed networks. It is justified to ask why nodes should spend their energy in relaying other nodes' messages, or how would messages reach their destination reliably and in time.

This licentiate's thesis concentrates on the security requirements pertinent to a clustered ad-hoc network architecture. New problem areas caused by the nature of ad-hoc networks, like routing security and the difficulty of inducing cooperation, are studied together with the traditional issues of authentication of participants and creating keys for secure communication. A large literature survey covers routing attacks and secure routing protocols, cooperation mechanisms, authentication, public key management, group key management, clustering methods, and clustered routing protocols in ad-hoc networks.

In the end, some requirements and suggestions for a secure clustered architecture are drawn on the basis of the existing solutions and their applicability to clustered network.

Keywords:	hierarchical ad-hoc networks, security
Language:	English

TEKNILLINEN KORKEAKOULU LISENSIAATINTYÖN TIIVISTELMÄ

Tekijä:	M.Sc Maarit Hietalahti	
Työn nimi:	Turva-arkkitehtuurin vaatimusmäärittely klusteroituneille ad hoc -verkoille	
Päivämäärä:	11. elokuuta 2007	**Sivuja:** 4 + 69
Osasto:	Tietotekniikan osasto	**Professuuri:** Tik-79
Työn valvoja:	Professori Kaisa Nyberg	
Työn ohjaaja:	TkT Pekka Nikander	

Liikkuvat langattomat ad hoc -verkot hahmotetaan yleensä koostuvaksi vain kymmenistä tai enintään sadoista solmuista, sillä ad hoc -verkkojen hallinta vaikeutuu verkon koon kasvaessa. Kuitenkin käytännössä solmut usein ryhmittyvät luontaisiin hierarkioihin tai klustereihin. Tällaisia rakenteita voidaan hyödyntää verkon hallinnoinnin helpottamiseksi.

Itseorganisoituvissa langattomissa ad hoc -verkoissa solmut osallistuvat reititykseen eikä verkon toimintoja tukevia rakenteita ole aluksi olemassa. Tästä seuraa turvallisuusongelmia, joita ei ole perinteisissä kiinteissä tietoverkoissa. On hyvä kysymys, miksi solmut käyttäisivät rajoitettuja voimavarojaan toisten solmujen viestien välittämiseen, ja miten viestit saataisiin toimitettua perille luotettavasti ja ajallaan.

Tämä lisensiaatintyö keskittyy klusteroituneelle ad hoc -verkolle tyypillisiin turvavaatimuksiin. Tutkitaan uusia ongelma-alueita, joita ad hoc -verkkojen luonne aiheuttaa, kuten reitityksen turvaaminen ja yhteistyön aikaansaaminen, sekä perinteisiä autentikoinnin ja avaintenluonnin ongelmia. Laaja kirjallisuuskatsaus käy läpi reitityshyökkäyksiä, reitityksen turvaamisratkaisuja, yhteistyömekanismeja, autentikointia, julkisen avaimen hallintaa, ryhmä-avaimen hallintaa, klusterointimenetelmiä ja klusteroitumista hyödyntäviä reititysprotokollia ad hoc -verkkojen alueelta.

Lopuksi esitetään turvavaatimuksia ja ratkaisuehdotuksia klusteroituneen ad hoc -verkon turva-arkkitehtuuriin pohjaten olemassaoleviin ratkaisuihin ja niiden soveltuvuuteen tähän ympäristöön.

Avainsanat:	hierarkiset ad hoc -verkot, turvallisuus
Kieli:	englanti

Acknowledgements

This licentiate's thesis was started in project Samoyed at the Laboratory for Theoretical Computer Science, and most of the work was written during that project. The part on group key agreements was written during an EE department project ADHOC. The personnel and financiers of the projects have been very supportive in making this licenciate thesis.

I would like to thank my instructor Dr. Pekka Nikander for his advises, understanding and patience with the changing timetables that a wide topic like this one causes. I thank prof. Pekka Orponen, who was the responsible professor in the Samoyed project and my supervisor prof. Kaisa Nyberg, who made it possible for me to finish the work after the Samoyed project ended.

Thanks go also to my mother for helping with childcare while I was writing this thesis. Most of all I thank the best part in my life, my family. My husband Tommi for his love and support and my son Niklas for making it all worthwhile.

Espoo 16.8.2007

Maarit Hietalahti

Contents

Abstract ii

Tiivistelmä iii

Acknowledgements iv

Contents 1

List of Figures 3

1 Introduction 5

2 Background 7

2.1 Different types of ad-hoc networks 7

2.1.1 Ad-hoc access networks 8

2.1.2 Ad-hoc sensor networks 9

2.2 Ad-hoc network routing 9

2.2.1 On the computational complexity of route selection 10

2.3 Distributed security and ad-hoc network security 11

2.3.1 Public key cryptography 11

2.3.2 Forming and managing security associations with public keys 12

2.3.3 Threshold cryptography 12

2.3.4 TESLA 12

3 Threats and problems in ad-hoc networks 14

3.1 Routing attacks 15

3.2 Problems related to connectivity and cooperation 16

3.3 The problem of managing security associations and trust relations 17

3.3.1 The problem of managing PKI in an ad-hoc network . . . 17

3.4 Securing communications 18

4 Clustered and hierarchical ad-hoc networks 19

4.1 Hierarchical and clustered routing 19

4.1.1 Hierarchy 20

4.1.2 Hierarchical routing protocols 21

4.1.3 Zone routing protocol 22

4.1.4 Fish-eye state routing protocol 22

4.1.5 LANMAR 22

4.1.6 Geometric routing 22

4.1.7 Clustered routing 23

4.2 Clustering algorithms 23

4.2.1 Identifier-based clustering 24

4.2.2 Highest-connectivity clustering 25

4.2.3 LCC 25

4.2.4 Weight-based clustering 25

4.2.5 ABCP 26

4.2.6 MOBIC 26

4.2.7 Other clustering methods 26

4.2.8 Performance of clusterings 27

5 Ad-hoc security solutions, solution space 28

5.1 Secure routing 28

5.1.1 Multi-path routing 29

5.1.2 Securing reactive/on demand routing protocols 30

5.1.3 Other security solutions for routing protocols 31

5.2 Enabling cooperation 32
5.2.1 Reputation system 32
5.2.2 Connection to intrusion detection 35
5.2.3 Payment systems 35
5.2.4 Theoretical analysis of cooperation mechanisms 38
5.3 Managing trust relations, authentication, certificates and public keys 39
5.3.1 Managing public keys and certificates 39
5.3.2 Authentication in ad-hoc networks 41
5.4 Group keys in hierarchical ad-hoc networks 44
5.4.1 Security requirements for group key establishment 44
5.4.2 The challenges of group key establishment in ad-hoc network environment 45
5.4.3 Background: group key agreement protocols 45
5.4.4 Group key establishment schemes for clustered ad-hoc networks 47
5.4.5 Clustered AT-GDH 48

6 Requirements for a security architecture 51
6.1 Securing routing 51
6.2 Requirements for clustering 54
6.3 Enabling network connectivity and stimulating cooperation 55
6.3.1 Cooperation inside a cluster 55
6.3.2 Cooperation between clusters 55
6.4 Managing security associations and trust relations 56
6.4.1 Trust relations inside a cluster 56
6.4.2 Trust relations between clusters 56
6.5 Secure communications 56
6.6 Summary of the most important requirements 57
6.7 Future work 57

Bibliography 69

List of Figures

2.1 An ad-hoc access network 8

4.1 An example of clustering 20
4.2 An example of a two-tier network 21
4.3 An example of clustering:cliques 24

6.1 Secure communications in ad-hoc networks 52

Chapter 1

Introduction

It is often stated that severe scaling problems will face any attempts to build large ad-hoc networks. In many real-life situations, however, networks have some kinds of natural hierarchical structures or clustering, which can be used to support their management. For example, in ad-hoc access networks the mobile nodes are organised to maintain contact to a supporting fixed network, therefore the static access points of the fixed network induce a natural clustering among the mobile nodes. Such real-life hierarchical structures can be used to assist in the operations of an ad-hoc network.

In this work, the results of a large literature survey combined with a requirements analysis are presented, focusing on secure routing, maintenance of operations, and induction of cooperation in the ad-hoc network. Authentication, managing trust relations, and key management (group keys) solutions support meeting these requirements. Privacy or anonymity of nodes is left outside of the scope. Service architectures and trust managing architectures are beyond the scope of this work, too.

The term *ad-hoc network* refers here to self-organising networks where the nodes form connections between themselves, without the help of pre-existing routers or other infrastructure. Self-organising means that the user need not concern herself with network management. The same principle applies for security operations, short of managing the (physical) security of the ad-hoc devices and what the equipments' access control function demands (for example, remembering pass-phrases).

Overlay networks or sensor networks are out of the scope of this work, except when the solutions are also applicable in mobile ad-hoc networks. There are two reasons for this. First, overlay networks as, the name suggests, need the support of an underlying network. Second, sensor networks are usually relatively stable

networks with special low-capacity devices made for measuring and gathering information. Such networks have different demands and uses from most typical mobile ad-hoc networks that consist of portable devices, such as laptops and smart phones.

Chapter 2

Background

2.1 Different types of ad-hoc networks

Networks of communicating nodes where the network offers no explicit or separate supporting infrastructure are called ad-hoc networks. In these sorts of networks, end-to-end connections are formed in a hop-by-hop manner utilising the other nodes, without the help of separate routers. In other words, links are formed directly between participating nodes, each of which can forward packets, potentially towards the destination node. In some applications, connections are limited to a small number of hops or just one hop. Such ad-hoc networks are either very small, with limited purposes, or work only as access networks to a fixed network (ad-hoc access networks are discussed in Section 2.1.1).

Communication in ad-hoc networks is wireless, usually based on radio connections, such as the Wireless Local Area Network (WLAN) [1] or BlueTooth [2]. The nodes may have relatively little memory and computational capacity, limited battery-life, they can be physically small, and are usually not tamper-resistant. This is especially the case with ad-hoc sensor networks (see Section 2.1.2). By convention, the unqualified term *ad-hoc network* usually refers to mobile wireless ad-hoc networks, while in ad-hoc sensor networks the nodes are usually not mobile.

Peer-to-peer (P2P) networks, in turn, are typically not mobile ad-hoc networks - but overlay networks. Overlay networks are virtual networks formed on the top of some other network, such as the Internet, some Local Area Network, an ad-hoc network, etc. While the connections are formed in a hop-by-hop manner between end nodes, as in ad-hoc networks, this happens using an underlying network infrastructure. In spite of that, some security solutions models are applicable in

ad-hoc networks either in the application layer or as abstract models.

This work concentrates on mobile and wireless ad-hoc networks. Sensor networks and peer-to-peer solutions are presented only if they are applicable in this environment as well.

2.1.1 Ad-hoc access networks

One of the most promising applications for ad-hoc networks is their use in connecting mobile nodes to a fixed network, i.e., as ad-hoc access networks. In ad-hoc access networks (see Figure 2.1), the mobile nodes are organised to maintain contact with a supporting fixed network, and the static access points of the fixed network thus induce a natural grouping or clustering (see Section 4) among the mobile nodes. Sometimes, multi-hop connections are limited to few hops, so that the ad-hoc network forms an "extension" network at the border of a fixed network.

For example, a two-tier ad-hoc access network is presented in [3], where the second layer consists of hosts acting as gateways to a cellular or a fixed network.

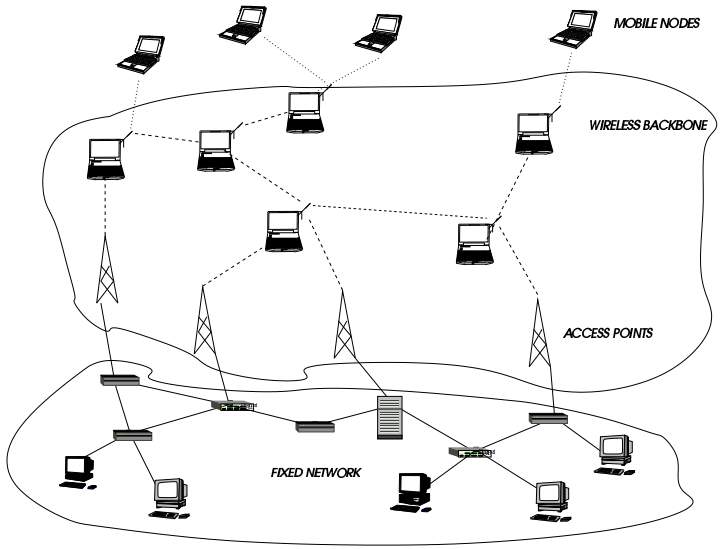


Figure 2.1: An ad-hoc access network

2.1.2 Ad-hoc sensor networks

A sensor is a small device designed for gathering information from its vicinity and either storing or transmitting it to a base station. An ad-hoc sensor network consist of several sensors that form connections in an ad-hoc fashion via other sensors without pre-existing infrastructure. These devices are very small, low-cost with limited capacity, even in comparison with other ad-hoc networks. For example, the Mica mote, a small (several cubic inch) sensor has a 4 MHz 8-bit processor with 128 KB of instruction memory, 4 KB of RAM for data, and 512 KB of flash memory. The radio is a 916 MHz low-power radio, delivering up to 40 kbps bandwidth on a single shared channel and with a range of up to about a few tens of meters. The device is powered by two AA batteries. [4]

Computational power in sensor networks is so limited that public key cryptography is often too expensive. Memory is limited, hence the nodes cannot maintain complicated state information. Radio transmission is costly, which means that message expansion caused by signatures is costly too. Moore's law is not likely to help: sensors are preferred to become cheaper instead of adding performance. Most solutions presented in the mobile ad-hoc network literature require capabilities beyond those of a sensor network but there are some exceptions.

Sensor networks are usually not designed with security in mind, yet security is difficult to add later on. If adversaries can disrupt or interfere with routing, sensor network becomes crippled or useless. Resource limitations are a two or three orders of magnitude worse than in typical mobile ad-hoc networks, which means that sensor network security is a difficult challenge.

2.2 Ad-hoc network routing

The routing protocols for ad-hoc networks can be divided into three categories: proactive, reactive, and hybrid routing. A *proactive routing* protocol maintains topology information of the network, in a manner where the correct routes to all hosts are known - all the time. A *reactive routing* protocol finds routes only on demand, i.e., when there is a packet to be routed to the destination. A *hybrid routing*, uses both systems in combination, usually so that the routes to nearest hosts are maintained in a proactive fashion and routes beyond a certain predefined area are looked up reactively.

A common technique used by many routing protocols in ad-hoc networks is broadcasting *route queries*. The relaying nodes add their own identifiers to the header of the query. When the query reaches its destination, the accumulated route can

be read in the header. This route is sent back as a *route reply*. This part of the protocol is called *route discovery*.

A *source routing* system is one where the sender of a packet determines the complete route to destination first, and lists this route explicitly in the data packet's header. A hop-by-hop routing system is one where each node independently determines the next hop for each data packet. The main drawback of source routing in ad-hoc networks is that the routing information at the source may become obsolete, leading to failed delivery. The main drawback of hop-by-hop routing protocols is that they easily lead to routing cycles.

In a *link-state* routing protocol, each node maintains information of the status of every link in the network. In a *distance-vector* routing protocol, every node maintains a list of distances for each neighbour and each destination. When sending or forwarding a message, a node selects the neighbour with the shortest distance to that destination. The exact route is not known to the sender, hence the route information is not spread throughout the network but the distances need to be kept up to date. One of the most common proactive routing protocols is the Destination-Sequenced Distance-Vector Routing (DSDV) [5].

Reactive routing protocols include the Dynamic Source Routing (DSR) [6] and the Ad-hoc on-demand distance vector routing (AODV) [7]. Both AODV and DSR flood a route query to the network when the route that is needed is not known.

2.2.1 On the computational complexity of route selection

Theoretically, finding an optimal route in a general graph is NP-hard. However, for a graph in Euclidean space with limited strategies for route selection, the problem becomes simpler: In the class of local probabilistic control schemes, route selection takes $O((loglogN)^2)$ time in any graph. In Euclidean space, with probability $1 - O(1/n)$, routing between arbitrary two nodes takes $O(\sqrt{n})$ time. In a rectangular domain space with height H and width n/H, routing between two arbitrary nodes can be done in time $O(n/Hmin[H, \lceil(logn)/H\rceil])$. These results that were presented in [8] are limited to static networks.

In mobile networks, the dynamics makes it practically impossible to find an optimal route. Therefore, it is more important to quickly find a route that is near-optimal, or at least stable enough, so that the packets may reach their destination before expiring.

2.3 Distributed security and ad-hoc network security

In dynamically changing ad-hoc networks, where hosts may be unreachable from time to time or the network can be partitioned, one needs to consider the notions of distributed security. The security solutions used may not assume that a particular node is reachable anytime. Therefore, the need for on-line contacts to central entities has to be minimised. In addition, nodes should be able to perform the computations needed by the most critical network operations by using only the information provided by its neighbourhood (Locally computable solutions). Optimisations requiring information over all nodes or links become unfeasible, as the network topology changes rapidly.

The reliability of communications itself is an especially important security issue, too. In ad-hoc networks where nodes themselves act as routers, any compromised nodes can seriously hinder the operations of the network by not cooperating in or abusing the routing of messages. This issue is discussed further in Sections 3.1 and 3.2.

Following subsections describe some basic security schemes and concepts used later in this thesis.

2.3.1 Public key cryptography

In public key cryptography, each principal has two keys, a public key and the corresponding private key. These keys are linked so that the private key is used to decrypt the data that is encrypted with a public key. In order to send an encrypted message to someone, one needs to know the public key of the recipient. It is distributed openly, and in some cases even used as the recipient's primary identifier (See SPKI [9]). The private key is kept secret, known only to the owner of the key-pair, who can open the message. The asymmetry is a feature that helps key management, for knowing the public key does not help an attacker to decrypt the message.

Public-private key pairs are also used in digital signatures. There, the secret key is used in creating a signature that can be verified by anyone knowing the public key. Public key cryptography was introduced by Diffie et. Al [10] and a common public key algorithm is RSA [11].

2.3.2 Forming and managing security associations with public keys

Public Key Infrastructure A public-key infrastructure (PKI) is used for distributing and managing large quantities of public keys. PKI has one or several *Certification Authorities* (CA) that issue credentials, *Certificates*, as a proof that certain public key belongs to a certain principal. CA signs a certificate with its own private key, whose corresponding public key is expected to be known to everyone.

Certificates can be also used for authorisation, in which case the CA signs a proof that certain principal is granted access to a service. Sometimes a public key may become compromised, or it becomes necessary to cancel the access. A certificate may be *revoked* by the CA that has issued it by having the issuer publish a certificate revocation list, a CRL. Every service provider is expected to check the CRL when verifying a certificate's validity. Usually, certificates have an expiry time, so that they are automatically cancelled after a period of time.

A public key infrastructure is often assumed to exist in ad-hoc networks by many authors of security solutions. However, it is not trivial to build one in ad-hoc environment. More on this in Section 3.3.1.

SPKI [9] is a public key infrastructure where the public keys themselves act as identifiers for principals. There are some other ways to identify nodes and authenticate public keys, for example, self-certified keys [12] and SUCV-identifiers [13].

2.3.3 Threshold cryptography

In threshold cryptography [14], a common secret, like the private key of the whole system, is "cut" into pieces and shared between n principals. The cutting is done so - that none of the principals can use the key alone - but a specified number k of them can collaboratively use the key.

2.3.4 TESLA

TESLA [15] is a broadcast authentication protocol used in many solutions for authentication and secure routing. TESLA employs one-way key chains. A *one-way key chain* is a sequence of keys generated by repeatedly applying a one-way function on a random number. The keys are then used (or disclosed) in backward order, so that the next key cannot be calculated from the previous ones- but the relationship between two consequent keys can be verified with the one-way

function. These kinds of key series are widely used as one time key lists.

In TESLA, the sender discloses a new key at appropriate time intervals, according to a pre-determined key-schedule. Key disclosure is delayed - so that the key is published only after it has been used. The schedule is known by the receivers, so they can estimate the validity time of a specific key.

When TESLA is used for authenticating traffic, keys from the one-way key chain are used as Message Authentication Codes (MAC). Delayed key disclosure means that the receiver can verify a packet only after the key has been revealed. Even when the keys used are symmetric, TESLA has an interesting asymmetric property. In broadcast authentication, the receivers need to able to verify the MAC, but on the other hand, they should not be able to generate a valid MAC themselves. TESLA achieves this asymmetry without using asymmetric cryptographic primitives. TESLA's central security issue is to determine the sending time of each packet, hence, the system must have at least a loose time synchronisation.

This packet authentication mechanism can also tolerate packet losses, because later keys can be verified by repeatedly applying the one-way function until the keys of missing packets are skipped.

Chapter 3

Threats and problems in ad-hoc networks

Security issues in ad-hoc networks usually include the following:

- Confidentiality of communications: between two users or within a group
- Availability of communications: preventing DoS-attacks, ensuring cooperation (forwarding packets)
- Authenticity and integrity of communications
- Sometimes it is important to prevent unauthorised use of network services (access control)
- Privacy (location, identity etc.)

The goal is to enable communications in a secure way, from enabling packet forwarding and secure routing to establishing trust relations and encryption keys. In an ad-hoc network environment, one needs to consider the notions of distributed security (see Section 2.3). No particular node can be expected to be reachable at all times. The network topology varies according to the movements of the nodes. Therefore, the need for on-line contacts to a central entity has to be minimised and credentials of formed trust relations should be portable.

In ensuring cooperation, it is usually important to make difference between intentional and unintentional bad behaviour. The dropping of packets may happen due to an error or be a calculated act to save batteries at the expense of packet forwarding. However, the receiver may not be able to deduce which is which. This problem affects especially reputation-based stimulation of packet forwarding (See 5.2.1).

3.1 Routing attacks

In ad-hoc networks where nodes themselves act as routers, any compromised nodes can seriously hinder the operations of the network by not cooperating in or abusing the routing of messages. If the routing tables are forged, packets will not reach their destination. Same can happen, if the headers of the messages are tampered with. This section describes some of the principal attacks against ad-hoc network routing.

Spoofed, altered, or replayed routing information ad-hoc routing is vulnerable to these types of attacks, as every node acts as a router, and can therefore directly affect routing information.

Selective forwarding A malicious node can selectively drop only certain packets. Especially effective if combined with an attack that gathers much traffic via the node, such as the sinkhole attack or acknowledgement spoofing. The attack can be used to make a denial of service attack targeted at a particular node. If all packets are dropped, the attack is called a *black hole*. If a node takes part in route discovery, but does not forward the data packets, it is called a *grey hole*.

Sinkhole attack In a sinkhole attack [4], a malicious node uses the faults in a routing protocol to attract much traffic from a particular area, thus creating a sinkhole. A *rushing attack* [16] is one type of a sink hole attack gathering most of the traffic directed to a specific target node. It exploits a property of several on demand routing protocols of forwarding only the first route request targeted to a specific node. As a result, any route between request initiator and the target that is further than two hops always contains the attacker.

Sybil attack The Sybil attack [17] is targeted to undermine the distributed solutions that rely on multiple nodes' cooperation or multiple routes. In a Sybil attack, the malicious node gathers several identities for posing as a group of many nodes instead of a one. This attack is relevant not only as a routing attack but it can be used against security schemes. Many crypto-schemes in ad-hoc networks divide the trust between multiple parties. For example, to break a threshold crypto scheme one needs several shares of the shared secret. Misbehaviour detection may also rely on reports from several peer nodes. More information on these schemes in Subsection 5.2.1 and Section 3.3.

Wormhole attack The wormhole attack [18] usually needs two malicious nodes. The idea is to distort routing with the use of a low-latency out-of-bound channel to another part of the network where messages are replayed. These can be used, for example, to create sinkholes and to exploit race conditions.

HELLO flood attack In a HELLO flood attack [4] a malicious node can send, record or replay HELLO-messages with high transmission power. It creates an illusion of being a neighbour to many nodes in the networks and can confuse the network routing badly.

Acknowledgement spoofing If a protocol uses link-layer acknowledgements, these acknowledgements can be forged, so that other nodes believe disabled nodes to be alive.

3.2 Problems related to connectivity and cooperation

Availability of network services is an important part of security in ad-hoc networks. Having one's packets forwarded to their destination does not only depend on securing the routing against malicious attackers but also on the cooperation of other nodes on the route.

A frequently asked question in connection with open ad-hoc networks is what motivates the nodes to forward other nodes' packets when battery power is limited and when the nodes have no common background. However, ensuring packet-forwarding and cooperation is still emerging research. Fixed networks generally do not have this problem, at least not on the node level. There is a separate infrastructure for routing and there is no special need for energy-conservation (i.e., routers do not usually refuse from routing legal traffic).

In open networks, where nodes are individuals acting in their own best interest, the nodes are thought to be selfish, due to their limited battery-life. They are likely to save their limited energy and they may not be willing to forward packets to the benefit of others, unless otherwise motivated. Therefore, packet forwarding has to be stimulated, assuming that the goal of the nodes is to send as many of their own packets as they can, while being sure that the packets will be forwarded. The same problem has consequences also when nodes are not independent but part of a common organisation, as conserving energy is a common isuue in every ad-hoc network that is composed of small devices with limited batteries. Solutions that

aim to divide the packet-forwarding load fairly between the nodes will probably also be useful in saving the total energy of the network.

3.3 The problem of managing security associations and trust relations

There are many types of trust relations that can be formed in ad-hoc networks. This work concentrates on the relationships between devices, assuming that the trust relations that the users of those devices may have can be translated into trust relations between their devices.

Whether a device trusts another device may be considered as an all-or-nothing relation, or it may depend on the object of trust. Trust relations can be thought of as an abstract structure built on top of the network, i.e., a kind of an overlay network. There is also need for trust relations in the initiation of the network, in the so called bootstrapping phase. In order to build security associations between nodes, it is often required that there is some initial basis of trust in the network, for example, a certificate authority (CA). Another way to bootstrap trust is to directly pre-distribute keys or shared secrets. Whatever the initial trust relationship between the nodes is, there are many constructs in the literature for maintaining old and forming new trust relations.

Sometimes it is important to prevent illegal use of network services. This means that nodes must somehow be authorised before joining the network.

Routing information messages often need to be authenticated. When routinely authenticating routing information messages, the authentication method should be very lightweight.

Sybil attack may also be targeted on trust management systems. An attacker may present itself to be highly trusted by its peers by presenting credentials signed by multiple fake identities.

3.3.1 The problem of managing PKI in an ad-hoc network

The security associations and trust relations that nodes in an ad-hoc network initially have may consist of the relations that devices have through a public key infrastructure (PKI, see 2.3.2). Standard PKI is difficult to implement in ad-hoc networks. Having a previously constructed infrastructure for public keys is a hard requirement for an ad-hoc network but there are networks, such as that of a single

administrative unit, that can have one. A few constructs have been proposed in the literature for bootstrapping a PKI in ad-hoc networks, see Subsection 5.3.1.

PKI has a relatively distributed nature with one exception: for revocation purposes, nodes frequently need to connect to a central entity, the certification authority (CA), for acquiring an updated CRL. Ad-hoc networks do not tolerate well central entities that have to be reachable anytime. There are ways to alleviate this, for example, by assigning several CA:s that each have a certificate from a root CA. Still it is possible for an ad-hoc network to have partition with no CA.

When using certificates, their maintenance is necessary. When a node has been found to be compromised, its certificate has to be revoked. Revocation in ad-hoc networks is not an easy thing to do, due to the aforementioned problem with central entities. It is generally prudent to use implicit revocation, where certificate is valid for only a limited period. This means that only the well-behaving nodes will have their certificates renewed. The network is also changing dynamically, which means that the new nodes need to have a certificate as well.

3.4 Securing communications

Securing communications in ad-hoc networks can be done with or without the help of the user of the device. Again, we concentrate more on the self-organised network, where the user need not concern herself with creating and managing encryption keys.

Encryption of communications between two parties can be implemented with public keys or with public key authenticated symmetric session keys. Public keys are useful in authentication purposes but may also be used for securing communications between two parties. Usually, communications are encrypted with a shared symmetric (session) key that is established with the help of public key cryptography. The main reason is that public key encryption and decryption are considered too expensive for bulks of data, another reason is that some public key crypto-systems are vulnerable to known cipher-text attacks when there is very much cipher-text available.

Chapter 4

Clustered and hierarchical ad-hoc networks

This chapter reviews the literature on clustered and hierarchical ad-hoc networks. This environment is where the solutions and requirements, discussed in Chapters 5 and 6 respectively, are to be employed.

The structure of this chapter is as follows. First, in Section 4.1, different forms of hierarchical and clustered routing is discussed. The problem of cluster formation is handled in Section 4.2. These sections present the part of the literature survey that deals with clustering and hierarchical ad-hoc network routing per se, setting aside the related security issues that are discussed later in Chapter 5. In this chapter, the basic outcome of the performed literature survey is that there is, so far, only few works that attempt to take a combined view to clustering and routing, even though both clearly require similar type of information about the local connectivity and topology.

4.1 Hierarchical and clustered routing

A cluster is a collection of nodes (geometrically) close together (see Figure 4.1). Clusters can be formed for a common cause or as a reaction to a factor that is common to the nodes. A cluster-head is a special node in a cluster that acts as a leader for the cluster, for the purposes of routing or initialising the cluster formation, for example. Cluster-heads are not always necessary, some clustering protocols do not use them at all.

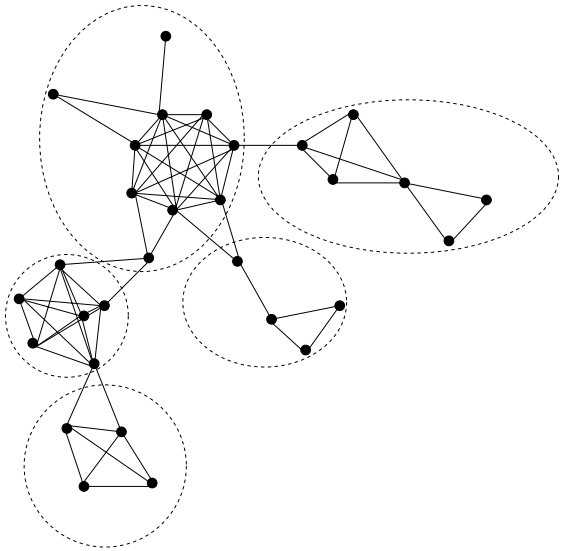


Figure 4.1: An example of clustering

4.1.1 Hierarchy

A hierarchical structure in a network is composed of nested groupings (clusterings) of nodes, forming a tree topology. A route from a leaf node to another is formed via the routes between their respective groups. This sort of routing is called hierarchical routing.

In one of the first papers describing hierarchical routing, Kleinrock et al [19] determine optimal clustering structures so as to minimise the size of the routing tables. The price for this is the increase in the average message path. However, the increase need not be very large: Bounds were found for the maximum increase of path length, so that in the limit of a very large network, enormous table size reduction may be achieved with essentially no increase in network path length. The performance of the proposed hierarchical routing system was evaluated in [20]. It should be noted that as these papers were written in the 1970's, they deal with fixed networks, and the size of a large network was thought in terms of hundreds or thousands of computers.

A *two-tier ad-hoc network* means a hierarchical network consisting of only two

layers. In other words, the nodes are grouped in some way, and the groups can have cluster-heads but there are no nested groupings (see Figure 4.2). Two-tier networks are most common hierarchical structures in the literature, as several layers of hierarchies are expected to waste too many usable routes.

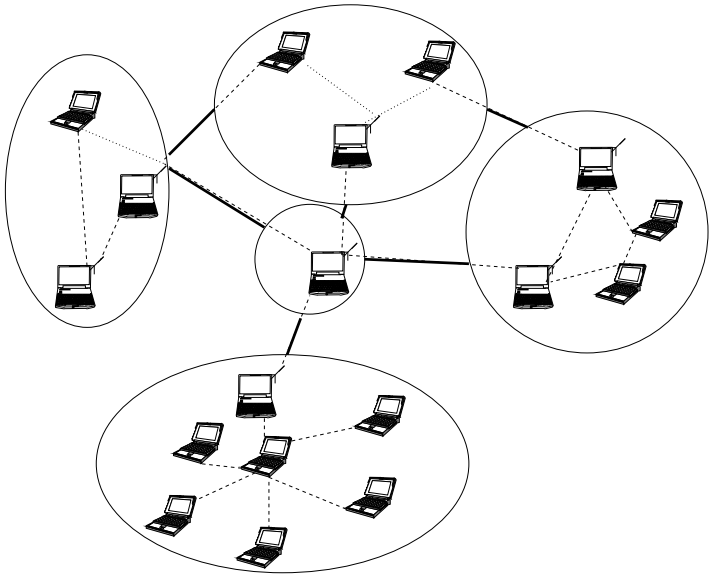


Figure 4.2: An example of a two-tier network

4.1.2 Hierarchical routing protocols

The Hierarchical Source Routing Protocol (HSR) [21] uses link state routing in hierarchically arranged network. Another example of hierarchical routing is [22], the Extended Hierarchical State Routing (EHSR). It is meant for military environments, battlefields. The hierarchical structure is extended also to a physically multilevel environment, by using unmanned airborne vehicles (UAV:s).

4.1.3 Zone routing protocol

The Zone Routing Protocol ZRP [23, 24] is a hybrid protocol. Every node has a routing zone specified by a *zone radius*. The zone radius is expressed in terms of hops. The routing zone is similar to a cluster, except that every node defines its own zone and acts as a cluster-head in its zone. The protocol works pro-actively when forwarding packets inside the zone, and reactively when the destination is beyond the zone radius.

Other zone-based routing protocols include [25] and [26].

4.1.4 Fish-eye state routing protocol

The Fish-eye state routing protocol FSR [27, 28] is also a hybrid protocol. In the fish-eye protocol, the border between intra-zone routing and inter-zone routing is a vague one, so that routes to destinations close-by are known best and the routes to further destinations are known only approximately. The idea is that packets are sent first to the area where the receiver is last known to have been. When the packets are closer to their destination, the hosts in the area will know the exact route.

4.1.5 LANMAR

Landmark routing, as hierarchical routing, was first introduced to fixed networks in [29]. Pei et al., extended it to mobile environments [30]. LANMAR combines FSR and Landmark routing. Exact link information is known only between neighbours, routing between groups uses landmarks. A packet directed to a remote node initially aims at the landmark. When the packets are closer to their destination, routing switches to the accurate route provided by FSR.

4.1.6 Geometric routing

Geometric routing has the basic assumptions that all nodes have information of its own and its neighbours position, and the sender knows the position of the intended destination but not the route. Other papers on regional and scalable routing include [31, 32, 33].

4.1.7 Clustered routing

Hierarchical routing can take advantage of a clustered structure by using a different routing scheme inside a cluster and outside it, as hybrid routing. Ideally, this should simplify the routing information needed by each node by abstracting away the topology of other clusters. There are a number of cluster-based routing algorithms for ad-hoc networks (for example, [19, 34]) and analysis of its reducing effect on routing table sizes [19] and signalling traffic [20, 35].

In [34], routing is based on clusters. The nodes forward messages towards the closest member of the destination cluster. However, in order to route the packets a node must calculate the shortest distances to each member of another cluster and therefore have knowledge of the whole network topology. Also in [36], routing is performed on top of a clustered topology, and [37] presents a cluster-based architecture.

4.2 Clustering algorithms

In order to arrange themselves into clusters, the nodes need to run a clustering algorithm. Many algorithms need the knowledge of the whole network topology, while others perform the computations knowing only the neighbouring nodes and their possible cluster-memberships [38, 39].

Clustering algorithms differ in what types of clusters they produce. Some clusterings permit no overlapping, so that every node belongs to exactly one cluster. Others may have overlapping clusters [34, 40, 41, 42], some require it because they are used with routing protocols that rely on overlapping [34]. Many clustering algorithms choose special nodes, cluster-heads, that take care of the cluster formation and later of the maintenance of the cluster [38, 43, 44, 45, 40, 41]. Some clustering algorithms form cliques (see Figure 4.3), i.e., clusters where every node is at a one hop distance from every other node [34]. Some only require that the distance to the cluster-head is one hop [45, 40, 41].

The role of a cluster-head varies in different protocols. In [35] the cluster-heads are not used as routers but merely for pointing the direction of the cluster. In [39, 37, 41] the cluster-heads are only used in the cluster formation but not in routing. Clustering often contains some selection of gateways. Either one gateway is selected or all possible gateways. Most identifier and connectivity based systems choose only one gateway between two clusters.

Some clustering algorithms are described in more detail in the following subsections.

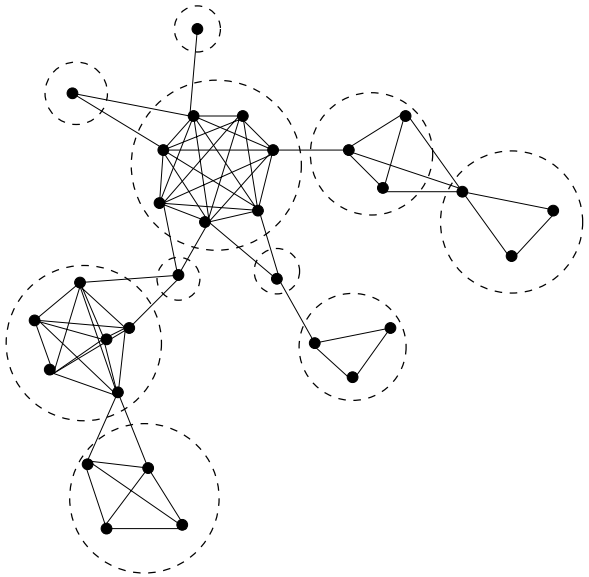


Figure 4.3: An example of clustering:cliques

4.2.1 Identifier-based clustering

Identifier-based clustering refers to algorithms in which the cluster-head selection is based on a numbering of nodes so that the highest or lowest numbered node becomes a cluster-head in its neighbourhood. A node's neighbourhood means all other nodes within one bidirectional link from the node. Identifier-based clustering does not associate mobility and the clustering process in any way. Identifier-based clustering is presented in [46, 47, 40, 41].

For example, in a distributed lowest-id clustering, a node elects itself a cluster-head, if it has the lowest identifier in its neighbourhood. Otherwise it elects the neighbouring node that has the lowest identifier, unless that has relinquished the status to another node.

4.2.2 Highest-connectivity clustering

Highest-connectivity clustering is similar to identifier-based clustering, except that (in the distributed version) the most highly connected node in neighbourhood is elected as a cluster-head, and only the uncovered nodes, i.e., the nodes that have not yet selected a cluster-head are included in the comparison. One such clustering algorithm is presented in [41]. All nodes are within one hop of a cluster-head. The clusters can overlap but cluster-heads cannot be neighbours to each other. A node that can hear at least two cluster-heads is called a gateway. Changes in the network topology can cause a change in the cluster-head of a cluster, even if no new nodes have joined the cluster. [41] also introduces a stability metric for clusters, calculating the number of cluster-head changes per unit time.

4.2.3 LCC

Re-clustering is needed whenever the nodes move so that the current cluster-head no longer has the lowest identifier. A clustering method called LCC, lowest cluster change [48], is an improvement of lowest-id clustering. When a node comes into the area of another cluster, re-clustering is not performed unless the node is a cluster-head itself, or when a node wanders out of the reach of its cluster-head. In former case, one of the cluster-heads relinquishes its role, in the latter, the node becomes a cluster-head itself. The reduction in the number of re-clusterings reduces the overhead and thus LCC performs better than lowest-id.

LCC can also be used in conjunction with the highest connectivity clustering. However, in the same paper, it was also shown that lowest-id clustering performs better than the highest connectivity clustering. This is due to the fact that usually the order of id's changes less frequently than the number of connections per node.

4.2.4 Weight-based clustering

Weight-based clustering is a generalisation of lowest ID clustering. Each node has a unique weight, hence the nodes can be totally ordered. The weights are calculated with mobility- or other metrics.

For example, Basagni [38] uses a weight-based criterion in clustering. Clustering coordinators are selected based on mobility-related parameters. Two clustering methods are presented: DCA (Distributed Clustering Algorithm) and DMAC (Distributed Mobility-Adaptive Clustering). The clustering method works best in quasi-static networks, and its time complexity depends on network topology, not

on network size. It requires the knowledge of one-hop neighbours only.

4.2.5 ABCP

In Access-Based Clustering Protocol (ABCP) [45] a new node originally joins any cluster within its reach. When a cluster-head relieves itself from cluster-head duties, it chooses a successor, the node with most connections within the cluster. This is for making as little changes to the clustering as possible. ABCP uses a separate control channel for control messaging. In ABCP, every node is only one hop away from cluster-head.

4.2.6 MOBIC

[43] presents a mobility metric for forming clusters. The metric is based on the ratio of the power levels of successive transmissions measured at any node from all its neighbouring nodes. A clustering method MOBIC (a lowest relative mobility clustering algorithm), uses this metric for selecting cluster-heads. It is claimed to perform better than lowest ID. MOBIC uses the same stability metric as [41].

The relative mobility $M_Y^{rel}(X)$ is calculated as follows.

$$M_Y^{rel}(X) = 10 \log_{10} \frac{RxPr_{X \to Y}^{new}}{RxPr_{X \to Y}^{old}}$$

and the aggregate local mobility value M_Y as follows

$$M_Y = var_0(M_Y^{rel}(X_1), M_Y^{rel}(X_2), \ldots, M_Y^{rel}(X_m)) = E[(M_Y^{rel})^2]$$

This mobility metric can be calculated in a distributed manner.

4.2.7 Other clustering methods

"*A Clustering Scheme for Hierarchical Control in Multi-Hop Wireless Networks*" (S. Banerjee and S. Khuller) [42] presents a clustering scheme for wireless sensor networks that is meant for the hierarchical control structures of MMVN [36]. MMVN is a modular system of link- and network-layer algorithms enabling support for distributed, real-time multimedia applications. The goals of the clustering are 1. each cluster is connected, 2. All clusters should have a minimum and a maximum size constraint, 3. Any node in a network belongs to a constant number

of clusters, 4. two clusters should have a small overlap and 5. clusters should be stable. A cluster is formed here by first creating a spanning tree and then naming some subtrees clusters. The algorithm is not locally computable but there is a distributed version presented also.

In "*Local clustering for hierarchical ad-hoc networks*" [39] S. E. Virtanen and P. Nikander present a clustering method where the cluster-formation can be computed by the nodes locally. The cluster formation is based on the ratio of the number of links inside the cluster to the number of links going out of the cluster. It does not produce much extra traffic (none when cluster size is at most 256 nodes, as the messages are piggybacked in existing link-layer or IP layer messages) and it produces "intuitively good" clusters i.e., clusters that have many links between its nodes.

More information on different ways to form clusters can be found, for example, in "*Cluster-Based Networks*" by M. Steenstrup [49]. It presents many different ways to form clusters and reasons for using clustered topologies.

4.2.8 Performance of clusterings

In "*Clustering Overhead for Hierarchical Routing in Mobile ad-hoc Networks*" [35] the overhead caused by clusters is compared to a situation, when clusters are not used. The average clustering overhead per node per second is found to be only polylogarithmic to the node count.

According to "*Giant clusters in random ad-hoc networks*" [50], the most significant measure is the ratio of the size of the biggest cluster to the size of the whole graph. The paper contains a simulation that supports the conclusions. The simulation is done using rather strong assumptions, such as a constant node density.

The clustering methods in the literature have been developed for different purposes, hence, some of them form deeper hierarchies and in some, the main operations of routing are left for the cluster-head to handle.

A side-effect of clustering and cluster-heads is that when nodes are no longer equally important in maintaining connectivity, a well-connected node or a cluster-head may become a single point of failure. Very deep hierarchies can reduce the amount of routes, which also leads to weak points in the network.

The performance of routing depends on the size of the cluster, according to "*Cluster-based approach for routing in dynamic networks*" [34]. The paper discusses performance issues in clustering but does not solve all problems presented. The cluster formation and the dynamics of the situation are described well. However, the performance analysis was limited to few runs with a small number of nodes.

Chapter 5

Ad-hoc security solutions, solution space

There are few publications concerning specifically the security of clustered or hierarchical ad-hoc networks. However, more research has been done on the security of general ad-hoc networks. This chapter deals with some of those publications that are applicable and/or have consequences also in the case of clustered/hierarchical ad-hoc networks.

Section 5.1 focuses on secure routing, without any direct consideration to clustering. A number of solution proposals are briefly explained. Section 5.2 continues with the hard problem of cooperation, looking at both reputation and payment systems. Section 5.3 provides a brief look at the basic, ad-hoc networking related problems to key and trust management, and Section 5.4 gives an extensive survey of distributed group key handling methods.

An integrated view to the different solution pieces is deferred until Chapter 6.

5.1 Secure routing

Many routing protocols are designed without considering any authentication or authorisation of the messages and nodes that participate in routing. In a dynamically changing network, it is of course difficult to validate the routes that are aggregated results of many nodes. This section deals with preventing the routing attacks made by the malicious nodes.

5.1.1 Multi-path routing

Malicious nodes attacking the routes can drop packets, or manipulate them in order to stop communication. Instead of using only one path between the source and destination, a multi-path protocol takes advantage of ad-hoc network's route multiplicity: several paths are used simultaneously for communication. Multi-path protocols deal with routing attacks by tolerating, rather than detecting and isolating malicious nodes. For example, the protocol discussed in the next paragraph, SMT, is a multi-path protocol. Other multi-path routing schemes include [51, 52, 53, 54].

SMT The Secure Message Transmission (SMT) [55] detects compromised transmissions. SMT uses an Active Path Set (APS), which is a set of diverse disjoint paths between the two end nodes that are valid at the time. APS can be discovered by any underlying route discovery protocol, so that it is independent of the routing protocol. Both proactive and reactive protocols can be used. The message dispersion is based on the Rabin's algorithm [56] that adds limited redundancy to the data. The message and redundancy are divided into pieces. A partial reception can lead to a successful message reconstruction, if M out of N transmitted pieces are received successfully, when the redundancy factor is N/M. The source of communication manages the APS information: it updates the rating of each path in its APS based on the feedback provided by the destination. Each path is associated with two ratings: Short-term and long-term. The short term rating is decreased by α each time a failed transmission is reported and increased by β for each successful reception. If the short term rating drops below a threshold value, the path is discarded. The long-term rating is a function of the number of the successfully received (and acknowledged) pieces over the total number of pieces transmitted across the route. The long-term rating also has a threshold value, below which the path is discarded. A simulation compared the protocol to two variations of SMT: The Non-secure single-path (NSP) data forwarding protocol that has no or message dispersion or security mechanisms, and the Secure single path (SSP) transmission protocol that has no message dispersion.

In the simulation, 50 nodes moved in a 1000 m $\times$ 1000 m network coverage area according to the random way-point mobility model. There were 15 simulation runs that lasted 300 seconds each. The number of adversaries varied from 0 to 25 nodes. The attack used was black hole, i.e., attackers discard all data packets forwarded across routes they belong to. The route discovery was assumed secure. Network topology was assumed to be such that for any two nodes, it is highly likely that (at least) two node-disjoint paths exist.

The performance of SMT and SSP in this simulation was almost the same. NSP experienced sharp degradation in message delivery and substantial packet loss, when with SMT and SSP the effect of adversaries is smaller. SSP showed better performance than SMT regarding the percentage of dropped packets by the attackers. As the number of adversaries increases, SMT increases the dispersion factor and the number of utilised routes. However, as the number of paths increases, it becomes more likely for the routing operations to be subjected to adversaries.

The simulation results imply that SMT can support quality of service for real-time communications. This is due to the low end-to-end delay when using simultaneously multiple routes. SMT seems more capable of supporting real-time traffic but it introduces more overhead when compared with SSP.

5.1.2 Securing reactive/on demand routing protocols

Ariadne Ariadne [57] is a secure routing protocol based on the on-demand routing protocol DSR. It authenticates routing messages and can use different schemes, shared secrets between each pair of nodes, shared secrets combined with broadcast authentication, or digital signatures. Symmetric cryptography is computationally more efficient than public key cryptography and Ariadne can be used, for example, with Tesla, see Section 2.3.4.

Each node includes a MAC in the message to protect against removing nodes from the route list in the route replies. The intended destination buffers the reply until the relaying nodes release corresponding TESLA keys. The route reply includes a MAC from the intended destination to certify that the request was verified correctly. Sender can authenticate each entry of the accumulated path in the reply message.

Nodes also record the efficiency of each route and prefer more efficient routes. This protects against choosing weak or compromised routes.

Ariadne inherits the same requirements that the authentication scheme has, for example, Tesla's loose clock synchronisation.

S-RIP A Secure Distance Vector Routing Protocol, S-RIP [58], is trying to tackle the problem of validating routes by using a reputation system (See 5.2.1) in conjunction with route discovery. The consistency of an advertised route is confirmed by consulting some of the nodes that have propagated that route. A reputation-based framework determines, how many nodes should be consulted, a trade-off between security and efficiency. In S-RIP, a well-behaved node can uncover inconsistent routing information, assuming that there is no collusion be-

tween the misbehaving nodes.

Self-organised Network-layer Security in Mobile Ad-Hoc Networks "Self-organised Network-layer Security in Mobile Ad-Hoc Networks" [59] considers both secure routing and ensuring packet forwarding. It operates on network-layer, in the context of AODV routing protocol. The security of the routing system is based on a reputation system, so that nodes without a valid token are ignored. The details of the operation of the reputation system is explained in Section 5.2.1.

The system also has a decreasing overhead over time, when the network is in good condition without attacks. A problem is that the neighbourhood is expected to be very stable. This system becomes very inefficient when the nodes have high mobility.

5.1.3 Other security solutions for routing protocols

A paper by Karlof et al., [4] deals with sensor networks but is partly applicable in mobile ad-hoc networks. In this solution, link layer encryption and authentication with a common symmetric key prevents most outsider attacks: adversary cannot join the topology. Replay attacks are prevented by using an increasing counter, as usual. However, an attacker can still forward packets without altering them. Encryption can make selective forwarding difficult but does nothing to a black hole attack.

Insider cannot be prevented to participate in the operations of the network and she can masquerade as any node: This means that identities should be verified but public keys cannot be used as was seen before. A solution was suggested in [4]: nodes share own unique symmetric keys with the base station. Another one presented was limiting the number of neighbours per node: attacker can not form symmetric keys with too many nodes in the network. The HELLO flood attack prevention can be done by verifying the bi-directionality of the link.

With wormhole attacks, geographic routing helps but brings another problem: should you trust the advertised location information? Additional solution given in [4] for the wormhole attack is restricting the structure of the topology. Yet another way to deal with wormholes is packet leashes [60].It defends against wormholes by comparing the time it takes for the packet to be delivered with the sender's geographical location and an exact time stamp attached to the message.

A defence against rushing attacks is described in [16]. It consists of secure neighbour discovery, route delegation and randomised selection of which route request is forwarded.

SRP The Secure Routing Protocol (SRP) [61] for Ad-Hoc Networks can be applied to DSR, IERP [62] and other such routing protocols that use queries in finding routes. SRP tries to ensure a correct route discovery. Security associations are built end-to-end, i.e., in every route discovery, only the source and destination must have a security association between them. When a route query is sent, the receiver responds by sending a message containing a message authentication code (MAC) over the path used. Hence, the sender knows that a path has been found and that the route reply comes from the intended destination. Unlike in Ariadne (Section 5.1.2), the intermediate nodes need not participate in the authentication of the route.

SEAD In SEAD [63], hash functions are chained in the TESLA-style. It uses one-way hash chains in combination with a distance-vector routing protocol DSDV in order to authenticate routing updates. SEAD is made by the same authors as Ariadne (Section 5.1.2) but is based on a proactive routing protocol and thus has a higher overhead. Therefore it is not so efficient in high mobility situations. SEAD is claimed to be secure against non-colluding adversaries.

Surveys and comparisons A good survey on secure routing in ad-hoc networks has been done in "A Survey in Secure Wireless Ad-Hoc Routing" [64]. On the other hand, more specific security surveys can be found, for example, in " On Security Study of Two Distance Vector Routing Protocols for Mobile ad-hoc Networks" [65] and "Cost/Performance Trade-offs of Two-Layers Ad-Hoc Multi-hop Cellular Access Systems" [66].

5.2 Enabling cooperation

In this section, known ways in the literature to stimulate packet forwarding are explained. It is assumed that the goal of the nodes is to send as much own packets as they can, while being sure that the packets will be forwarded. In general, the solutions presented can be divided into two categories: the reputation systems and systems where nodes trade payments against packet forwarding services.

5.2.1 Reputation system

In a reputation system, where other nodes form an opinion on a node's behaviour, reputation is used to affect directly on how high a priority the node's packets will

receive and how much routing will be directed towards it. Behaviour can mean anything from the will to forward other nodes' packets and will to take part in other common activities, to not disrupting communications or to not compromising common secrets.

The reputation method directly affects routing. Another, bigger problem with reputation systems is the fact that it can give an inside attacker means for making *denial of service* attacks. If negative reputation is allowed to spread, a malicious insider can indirectly attack a well-behaving node by giving false testimony on its behaviour. Consequently, the well-behaving node's packets get low priority or the node can even be shut out of the network or face whatever measures the other parties choose to take towards an uncooperative or compromised node. Sometimes it is difficult to sort out the malicious nodes from their victims, and methods have similarities to intrusion detection systems in ad-hoc networks. All and all, reputation systems should be used with care.

"*Self-Organised Network-Layer Security in Mobile Ad-Hoc Networks*" The work [59] by H. Yang, X. Meng and S. Lu considers both secure routing and ensuring packet forwarding. It operates on network-layer, in the context of AODV routing protocol, and uses a reputation system.

Nodes need to have a token. Tokens are not traded but kept simply as a credential of a node's right to have its messages forwarded. Nodes without a valid token are ignored. Tokens are simple, containing only the owners identity, signing time and expiration time. There is also a system key, to which every node has a share. This is achieved with a threshold crypto scheme with a polynomial of degree $k-1$.

The control mechanisms for the token are very localised; the local neighbourhood verifies a token, monitors a node's behaviour and decides whether the token should be renewed. The renewing is also done collaboratively by the neighbours by signing a new token with the system secret. New nodes joining the network will get a token in this way too.

The system also has a decreasing overhead over time, when the network is in good condition without attacks. This is achieved by extending the lifetime of a token every time it is renewed. Obviously, newcomers' tokens have short lifetimes. Due to the threshold crypto, the collaboration among the attackers is assumed to be limited to less than k attackers per neighbourhood, when k is the degree of the polynomial. It also means that this scheme could be vulnerable to the Sybil attack. However, a perhaps more serious problem with this system is that the neighbourhood is expected to be very stable, renewing the same node's token repeatedly. Therefore this system becomes very inefficient when the nodes have high mobility.

CONFIDANT Confidant [67], specifies mechanisms for finding one or several malicious nodes on a route and blacklisting that one. Nodes have a monitor, reputation records, trust records and a path manager. The monitors purpose is to detect malicious nodes. Reputation and trust records are kept for evaluation purposes.

Blacklisting of innocent nodes has not been prevented. It also relies on authentic information in the routing headers and therefore can only be run on top of routing protocols that ensure the integrity and authenticity of routing messages.

Core Core, A COllaborative REputation mechanism to enforce node cooperation in Mobile Ad-Hoc Networks [68] uses a "watchdog" monitor combined with a reputation system. Subjective reputation, indirect reputation and functional reputation are treated separately. The node's combined reputation leads to gradual isolation or cooperation.

URSA URSA [69] uses a reputation system for access control. A node needs to have a valid ticket that identifies well-behaving nodes and grants access to network. The reputation system is similar to that of [59]. The system is very localised. The one-hop neighbourhood jointly monitors a local node and certify/revoke its ticket. When a mobile node moves to a new location, it exchanges tickets with its new neighbours, as the first step to cross verify each other.

Wrong accusations are also taken into consideration: If the node that accuses another one is itself already on the revocation list, accusation is considered to be malicious and dropped. The range of the accusation propagation is important. A large range causes excessive communication overhead, while a small range may not be enough to isolate a mobile misbehaving node. The accusations should be propagated so that before its current ticket expires, the misbehaving node cannot move out of the area where it is convicted by the accusations.

The localised group trust model means that a node is trusted if it is trusted by any k trusted nodes. The value of k is fixed network-wide, tuned according to the network density and desired system robustness.

Initialisation is hierarchical: the authority is only responsible to initialise the very first nodes, selected out of a two-hop local neighbourhood. After that, the initialised nodes collaboratively initialise other nodes, typically their neighbouring nodes.

5.2.2 Connection to intrusion detection

In order to apply a reputation system in a self-organised fashion, a way to automatically detect and evaluate attacks is needed. This is a complicated task, methods may also be found in the literature of intrusion detection. For example, [70] defines a taxonomy of basic and anomalous events in routing, and applies these to AODV with the help of a finite state automaton. The anomalous events include atomic events and combinations of events such as Route Drop and Route Flooding.

More information on intrusion detection in ad-hoc networks can be found, for example, in [71, 72, 73, 74, 75]

5.2.3 Payment systems

This is a system where nodes trade tokens, or essentially any sort of payments, for message forwarding services. All packet forwarding actions are counted, and eventually the forwarding services are payed by the sender to the forwarding nodes.

Usually, tokens are collected for data forwarding only, not for route discovery or control messaging. A node might participate in the route discovery but forward data selectively, and thus create a *grey hole*. This is not explicitly punished, because such behaviour causes the node to lose income. A payment system avoids judgements on node's behaviour. Therefore it also avoids the sometimes complicated management of reputation and trust issues. The sender will pay tokens for the forwarding nodes in the path, hence the route should be previously known, or estimated in some way, so that the trading can take place. This is obviously easier with proactive routing protocols.

A denial of service attack can be very effective when a malicious node forwards nodes into a wrong direction, or a loop, with the help of routing attacks such as a wormhole. This will quickly empty the sender's 'wallet' of tokens.

When a high-capacity node enters a network consisting mainly of low-capacity nodes (like an efficient laptop computer in a sensor network), it may gather tokens away from the market simply by only forwarding packets and not sending any of its own. This kind of behaviour doesn't appear as 'bad' but as a result, the packet forwarding does not function anymore. Therefore, it should not be possible to empty the network of tokens and extra management of the tokens on the market is needed.

Trading tokens is similar to virtual currency systems. There is more discussion

of the advantages and disadvantages of different virtual currency systems in the economics literature.

"*Stimulating Cooperation in Self-Organising Mobile Ad-Hoc Networks*" This solution [76] is meant for civilian ad-hoc networks, where each node is its own authority. Stimulation mechanisms key idea is that every node has a tamper resistant security node with a nuglet counter (tokens). When sending own packets, the number of intermediate nodes N is decreased from nuglet counter (n estimated by security module). The nuglet counter value c must remain positive. If $n > c$ own packets are dropped. When forwarding others' packets, nuglet counter is increased by 1. The nuglets are actually stored in the neighbouring node until a nuglet synchronisation protocol is run, and pending nuglets are released for the use of their owner. The analysis of static aspects of the system gives a result that an optimum (most own packets sent) is reached in the following two cases depending on the ratio r_f/r_o.

a)

$$\text{If } \frac{r_f}{r_o} \geq \frac{NB-C}{B+C}, \text{ maximum } z_o = 1,$$

i.e., no own packets are dropped,

b)

$$\text{If } \frac{r_f}{r_o} \geq \frac{NB-C}{B+C}, \text{ maximum } z_o = \frac{r_f}{r_o}\frac{B+C}{NB-C} < 1,$$

i.e., some own packets are dropped.

How should node behave in order to reach this theoretical optimum? Simulation of single node case with several different behaviour rules shows that the most cooperative rule is the best. Simulations with several nodes had the following parameters. One should note that the battery consumption has not been taken into account here.

Space	500m*500m
Number of nodes	100
Power range	120m
Mobility model	random way-point
Speed	1-3 m/s
Average pause time	60 s
Packet generation rate	0.2 (0.5 0.8) pct/s (Leave Outside)
Choice of destination	random
Routing	geodesic packet forwarding
Initial number of nuglets	100
Nuglet synchronisation interval	5 (10, 15, 20) s
Simulation time	7200 s
Packet transmission (1 link) time	random, average 10ms
transmission error	probability 0.1 => 1s timeout and retry

When comparing the following forwarding rules,

Rule 1' always forward

Rule 2' if $c \leq C$ then forward
else forward with prob. $1 - C/c$

Rule 3' if $c \leq C$ then forward
else drop

90 % of the nodes follow the majority rule, rest 10 % use first 1, then 2, then 3. Focus on z_o of the 10 %. the result was that best performance for the 10 % was achieved when following rule 1, regardless of the majority rule.

The effect of the less cooperative nodes was measured with the following simulation setting: First all nodes use rule 1, then the fraction of less cooperative progressively increases. Simulations were done with 100, 200, 300 and 400 nodes with same node density. Focus was on cumulative throughput (packets delivered/packets sent). The result was that only a mild decrease of throughput appeared when the fraction of less cooperative nodes increased. Variation of the average nuglet level was measured so that the fraction of less cooperative nodes increased.

Results were following: Rule 1 in majority: number of nuglets increased! (problem lied in the estimation of intermediate nodes). When cooperability was decreased: first, the number of nuglets decreased, then they reached equilibrium. When rule 3 was in the majority, the equilibrium level of nuglets was below the initial value 100. With another setting, where the size of the nuglet synchronisation interval grew, the increase slowed. When interval was 20s or higher, nuglets started to decrease.

The protection of the system (tamper resistant security module) included public key infrastructure, security associations and extra headers. Overhead was created by crypto-operations, header and the nuglet synchronisation protocol. All packets are of equal size, and this system is meant only for payload carrying packets. Lots of work was given for the security module but there is still the possibility of dropping others' packets before directing them to security module. This protocol is not very suitable for high mobility situations, due to the pending of nuglets.

Similar solutions have been presented, partly by the same authors, in [77, 78, 79, 80]

5.2.4 Theoretical analysis of cooperation mechanisms

The advantages of having a cooperation method, or how they improve routing, has been estimated by Lamparter et al., [81]. They do not compare actual cooperation methods but estimate the throughput with different forwarding probabilities and simulate packet forwarding with and without cooperation methods

The network is assumed to be sparse, approximately four neighbours per node. Routing is on-demand. Only the forwarding phase is observed, the nodes are assumed to take part in route discovery but maybe not in forwarding. I.e., they may act as *grey holes*. All nodes are assumed to have the same forwarding/dropping ratio. When a node decides to forward a bundle of data for some recipient, it forwards all packets belonging in the data bundle addressed to the recipient, or nothing.

In the limits of this setting, the results work in detecting tendencies. When a cooperation method increases an individual node's participation in the packet forwarding, its effect for the whole network may still be low. Especially so in medium- and large-scaled networks with long routes. However, for small ad-hoc networks with short average route length and certain forwarding probability pairs (with and without a cooperation method), there is improvement in the overall throughput.

Co-operation mechanisms extend the network's overall reachability, in terms of a reasonable number of intermediate hops and an acceptable amount of data received at the final destination. These results of [81] can be applied only in sparse networks with on-demand routing protocol. The authors stipulate that even when nodes have different forwarding probabilities, the tendency is the same.

In A. Urpi, et al., [82], modelling cooperation is viewed from the perspective of non-cooperative game theory. It means that selection of strategies happens from the point of view of a single node. This model takes into account both the available energy and the traffic generated/directed to a node. Analysis of some existing

cooperation mechanisms is done and the paper proposes the use of tit-for-tat [83] as a cooperation strategy.

A game theoretical model is presented in [84] where energetic information is taken into account to describe the conflicting interaction between heterogeneous nodes involved in a forwarding game. Properties of generous tit-for-tat (G-TFT) are studied and it is demonstrated that under an energy constraint G-TFT promotes cooperation if every node of the network conforms to it. High level guidelines towards designing a cooperation enforcement mechanism. It is shown that G-TFT results in a Nash equilibrium and proved that the system converges to the rational and optimal operating point.

[85] proposes a model for the forwarding behaviour of a node. The model requires a specific topology. Defines equilibrium forwarding strategy as a function of topology and routing (path length) information. A punishment mechanism is included.

5.3 Managing trust relations, authentication, certificates and public keys

5.3.1 Managing public keys and certificates

"*Resurrecting duckling*" [86] In the resurrecting duckling model, security is bootstrapped by secret sharing between devices that is called *imprinting* a device, *duckling*, to a *mother*. This means that the device accepts the first key it receives after being activated. After imprinting, encryption of communication is done using the shared key. The mother uploads new authentication methods, access control lists (ACL:s), policies etc. to its ducklings.

There exists several publications in the literature on creating and managing public keys in ad-hoc networks. The public keys are useful in communications between two parties and for authentication purposes. The following solutions are especially interesting in that they are very distributed or have a cluster-like approach.

"*Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks*" Kong et al., [87] present a distributed RSA based PKI authentication and key management scheme that uses threshold cryptography.

All nodes know the system public key. Private key is shared to n parts by a $(n, t+1)$ threshold crypto. Any node can carry a piece of the private key and $t+1$ shares

are enough to sign and create a new share. However, there is a trade-off between availability and robustness.

A central authority gives the initial nodes their personal certificates and the part of CA's private key. This systems may be vulnerable to Sybil attack, where one node gathers many shares using multiple identities in order to reconstruct the system's private key. (See Section 3.1).

This work is updated in [88], by using more efficient algorithms and methods in certificate updates.

In [69], URSA: Ubiquitous and Robust Access Control for Mobile Ad-Hoc Networks: The localised group trust model means that a node is trusted if it is trusted by any k trusted nodes. The value k is fixed network-wide, tuned according to the network density and desired system robustness.

Initialisation is hierarchical: the authority is only responsible to initialise the very first nodes, selected out of a two-hop local neighbourhood. After that, the initialised nodes collaboratively initialise other nodes, typically their neighbouring nodes.

"*Self-organised public-key management*" Čapcun et al., [89] present a distributed public-key management system where every node can issue certificates and send them to each other. A node keeps an updated certificate repository and a non-updated certificate repository. When trying to authenticate another node's public key, the updated repositories are combined in order to find a valid certificate chain. If such is not found, the node combines these to its non-updated repository and tries again. If a valid chain is now found, the node asks for updates for the non-updated certificates. If this does not work either, the node aborts the authentication. Revocation is implicit (by certificate expiration time) or explicit (revocation statement from issuer to all parties that stored the certificate in question.)

The problem of finding a certificate path is thus reduced to finding a path between two nodes in a directed graph, if there are enough certificates. In this, the *small world phenomenon* [90] helps: such graphs formed of certificates given in an ad-hoc network are conjectured to follow the following principles: 1. The graph has small average shortest path length that scales logarithmically with its size 2. clustered vertices. (More information on the small world model in certificate paths can be found in [91].)

Users cross-check their certificates in order to detect frauds. Protocol extension: physical (one-hop) neighbours are considered "helper nodes", whose repositories a node can take advantage of if needed. If needed, a node can also do load bal-

ancing: a node u gives a list of updated nodes that can update u's certificates. It is important to notice that due to the extensive storing of certificates, authentication may still be possible even when the network is partitioned.

5.3.2 Authentication in ad-hoc networks

Authentication is used both in trust management and secure routing. For example, LHAP authenticates the sender. It is thought to prevent attacks, because then the attacker can be caught.

LHAP LHAP (A lightweight hop-by-hop authentication protocol for ad-hoc networks) [92] provides network access control for preventing resource consumption attacks. Hop-by-hop authentication means that every node authenticates the packet before forwarding it. LHAP uses one-way key chains and modified TESLA (Subsection 2.3.4) for packet authentication and PKI for bootstrapping trust. Unlike TESLA, keys are revealed only when they have been used for authenticating packets, not periodically. This way, when there is no traffic, keys do not go to waste. A node commits to a key chain by signing the first used key with the node's public key, and thus the authentication of packets is bootstrapped.

In this modified TESLA, a MAC key used in the previous packet is disclosed in the next packet. This means that the receiver can verify a packet only after the next one has arrived.

Packet authentication mechanism can tolerate packet losses, because later keys can be verified by repeatedly applying the one-way function until the keys of missing packets are skipped. TESLA's central security issue is to determine the sending time of each packet.

LHAP works between the link layer and the network layer. It is independent of the routing protocol used but assumes bidirectional links (like most routing protocols). An assumption which is also needed is that while a packet has been sent but not yet received by immediate neighbour, the packet cannot be hijacked.

All nodes have to belong to the same autonomous system or administrative unit and the system has to have a loose time synchronisation (for TESLA).

"*A Trust-Based Security Architecture for Small and Medium-Sized Mobile Ad-Hoc Networks*" The following is an example on how trust relations can be set up in ad-hoc networks.

[93] presents a security architecture using certificates with trust values. It extends a PKI. Three trust values are assigned to certificates. The solution applies in small/medium-sized network (size of the secure network only). The size limitation is due to bootstrapping in the initialisation phase, which has to be done manually. The system is service-based (Jini-like), having providers and users. Any network node with enough capacity can assume the role of the service directory, if one is lacking. At least one administrator is assumed: some administrative work is acceptable to perform some key actions (initialise, allowing joining, expelling).

Devices can be grouped under same administrative authority (devices of one person, group of persons, or organisation). Devices can be clustered: virtual domains Every device has to run at least one symmetric and one asymmetric cipher, and at least one device has to have enough memory to keep a digital certificate store.

In this architecture, one device may host one or more logical entities: clients or services: lookup (LS), registration (RS) and general service providers. LS keeps a list of available network services in the neighbourhood. RS issues, renews and revokes certificates with embedded trust information and keeps a database of the digital certificates and their trust values. Certificates are not only for identification purposes but also for access control (credentials). Certificate bears the *maximum* trust value that the entity may have. Bootstrapping of the network is done with a self-signed certificate by RS.

The status of an entity may be

1. anonymous guest: Entity is not registered, only public services are available to it.
2. identified guest: Entity is registered but has only short-term rights to services.
3. permanent entity: Entity has long-term privileges that are set up at initialisation.

Network perception is the "network's common intelligence", a reputation based trust system. The behaviour of clients prompts security events (reports of network offences or good behaviour). According to these, RS calculates new trust values and spreads them to other entities. A problem remains that perfect information is not possible in ad-hoc networks (services can be unavailable). Local perception and gossip mechanism come to help, The nodes store information and wait until RS is available. If there is more than one RS, time stamped lists from different RS:s are forwarded, and thus the lists are synchronised. In addition, other entities with enough resources can be demanded to store a local version of a report list. A trade-off is in instantaneous network perception against mobility.

Certificate revocation list (TICRL) is an extension of a regular certificate revocation list (CRL) containing trust information.

- Active entities: Entities that had any change in its trust information
- Suspended entities: Due to sudden loss of trust in a short period of time, these entities have all their rights suspended for a determined period of time
- Blocked entities: Suspended for undetermined period of time. Only network owner or admin can unblock an entity
- Revoked entities: Certificates have been revoked. Full distrust

In the initialisation phase, RS auto-signs a certificate and creates a long integer for domain identifier. The nodes are pre-registered to domains: entity status, alias, initial authentication method. The RS may also add new RS:s for the domain.

In the joining phase, a joining node is asked for an alias and authentication data. RS then signs the node's public certificate and, along the domain identifier, sends it encrypted to the node. A symmetric encryption key is derived from the authentication protocol, or alternatively an auxiliary channel is used. Initial authentication method and entity status define the initial trust value. Permanent entities have higher trust values than guest entities.

There are three trusts values: trust level, distrust level and unknown factor. The security events are classified in six categories:

- three regarding network offences (critical to light)
- three regarding nice behaviour (absence of faults to extreme security awareness)

Each RS builds only one TICRL regarding only certificates issued by it.

Application prototypes were made: digital signer and secure slide-show. Clients were forced to commit faults and the following network perception was observed. The resulting fluctuation of trust information:

- initial trust value = 0.8, initial distrust value = 0.05 and unknown factor = 0.15
- after six network offences, trust = distrust = 0.5
- after ten offences, trust = 0.25, distrust = 0.75

- (suspended and blocked states were artificially suppressed for this example)

All and all, the network size-restriction to small and medium was due to the partly manual initialisation and applied only for the secure part of the network. Mobility impacts report synchronisation among RS. Local perception and other entities taking part in relaying, reports help fixing this problem. The system requires at least one administrator. The handling of negative trust information by ordinary nodes opens possibilities for insider DoS attacks.

5.4 Group keys in hierarchical ad-hoc networks

The purpose of key establishment is to create a common key for a group of two or more participants to be used for encryption and authentication of their communications. For two participants, the Diffie-Hellman key exchange [10] is often the most convenient choice. The multi-party case requires a generalisation of a two-way key exchange.

There are *distributory* and *contributory* group key protocols. A contributory protocol means that all participants take part in the key generation and guarantee for their part that the resulting key is fresh. Key distribution, on the other hand, means that the key is generated by one party and distributed to the other participants. This cannot be done without the help of a previously agreed-on secret that is used in encrypting the new session key. There is also a method called key pre-distribution, whereby the key is completely determined by the previously agreed-on initial key material.

In ad-hoc networks, every pair of nodes cannot reach each other within one hop. This issue of restricted topology, what H. Shi and M. He [94] call *the neighbours communication problem* can be alleviated by a careful choice for the graph structure that can be found in an arbitrary topology. A key agreement protocol using a spanning tree was presented in [95] and Di Crescenzo, et al., [96]. A clustered hybrid protocol using the protocol in [95] in connection with the Burmester-Desmedt key agreement (BD) [97] is presented in [98].

5.4.1 Security requirements for group key establishment

In the context of group key exchange, implicit key authentication means that a participant can be sure that no-one outside the group can learn the key without the help of a dishonest participant. Key confirmation means that after the key has been established, the participants are assured that all legitimate participants

do share the same key. As this would require many all-to-all messages, which may not even be possible in a sparse connection ad-hoc network, achieving key confirmation is not practical. Explicit Key Authentication means that both implicit key authentication and key confirmation hold, i.e., all participants are assured that all legitimate participants know the key and no outsiders do.

An active adversary should not be able to mislead honest participants as to the final outcome. A compromise of past session keys should not allow a passive adversary to find out future session keys and should not allow impersonation by an active adversary in the future. Independence of long term and short term secrets is important when there is an additional long term secret present, for example, private keys of a public-key algorithm or passwords used in authentication.

5.4.2 The challenges of group key establishment in ad-hoc network environment

The limitations of ad-hoc network environment pose some drastic demands on the group key establishment protocols. First, a global broadcast is most probably out of the question, that is, it is not probable that an arbitrary node will have direct connections to all other participant nodes. But on some occasions, a local broadcast from a node to its neighbours is feasible. Also, no fixed topology, such as a ring or a star can be assumed. Consequently, protocols requiring a specific topology either cannot be used at all or become inefficient.

In other words, every pair of nodes cannot reach each other within one hop. The issue, what H. Shi and M. He call *the neighbours communication problem* can be solved with the help of graph theory. This paper makes use of a spanning-tree algorithm, see [95] and [96]. There are also algorithms for generating more balanced spanning trees. See, for example, a survey by Gärtner [99].

The lack of infrastructure means that there are initially no third parties that can be trusted to calculate a random key safely and to distribute it. A lack of common history implies the lack of previously agreed shared secrets.

5.4.3 Background: group key agreement protocols

For the definitions of trees and other graph theoretic notions, see, for example, [100].

The BD broadcast protocol The broadcast protocol [97] assumes that every node is at a one hop distance from another. The protocol is accomplished with only two broadcasts per node.

BD protocol steps:

G is a finite cyclic group and g is a generator of G.

1. Each node m_i selects a random exponent r_i and broadcasts $z_i = g^{r_i}$
2. Each node m_i computes and broadcasts $x_i = (z_{i+1}/z_{i-1}1)^{r_i}$
3. Each node computes the session key $k_i = z_{i-1}^{nr_i} x_i^{n-1} x_{i+1}^{n-2} \cdots x_{i+n-2}$.

TGDH Tree-based Group Diffie-Hellman (TGDH) [101] employs Diffie-Hellman key exchanges in binary key trees. The described key structure results from the dynamic group key operations such as join, leave, merge and partition. There is no initial key agreement protocol.

The key structure in TGDH is very general, it can be used to describe the key structure of any bipartite group Diffie-Hellman key agreement where the resulting keys are used recursively as the new exponents. For example, the key structure of the protocol Hypercube [102] is the same as that of TGDH with perfect binary tree where all leaves are at the bottom level. Paper [103] extends TGDH protocol to improve the computational efficiency by utilising pairing-based cryptography. They use bilinear pairings in a ternary key tree which applies to any two-party and three-party key agreement protocol.

AT-GDH AT-GDH (Arbitrary Topology Generalisation of Diffie-Hellman) [95] employs a spanning tree. A spanning tree contains only the (one hop) links used in initial key agreement. This avoids the neighbours communication problem, as the Diffie-Hellman key exchanges are done only with one-hop neighbours. The operations propagate over the network along the spanning tree. AT-GDH can be used in any connected network topology with bidirectional links, because a spanning tree can always be constructed in such a network.

All leaf nodes (nodes with no children) start by selecting a random secret exponent and blind it and send the result to their respective parents. After a node has received the blinded keys from all its children, they select their exponents and form Diffie-Hellman-type keys with their children repeatedly using the resulting key as the new exponent. When the root has received all the blinded keys of its children, it repeats the same kind of computation as all the other parent nodes. The secret key formed thus between the root and its last child (and all other nodes) will be

the shared session key material for the entire network. In the last phase of the protocol, the blinded keys needed for extracting the group key are propagated up the tree from the parents to their children starting from the root.

AT-GDH does not contain group key management mechanisms, or authenticate the resulting key explicitly. The number of synchronous rounds AT-GDH needs to gather and distribute the blinded keys is twice the height of the tree. The height of the tree is assumed to be logarithmic to the number of nodes in the network, depending on the spanning tree algorithm and the topology of the underlying network links.

Other protocols Di Crescenzo, et al., [96] approach the problem of arbitrary topology from another angle than AT-GDH. They rigorously analyse the effect of physical topology on the actual performance of some key agreement protocols. These include GDH.2 and the BD broadcast protocol. In their analysis, they apply a topology-driven simulation of the logical network over any arbitrary ad-hoc network graph. In connection with the key agreement protocols, they use auxiliary protocols in order to generate efficient embeddings of logical networks over arbitrary ad-hoc networks. The auxiliary protocol suggested for generating a spanning tree is the same as in [95].

5.4.4 Group key establishment schemes for clustered ad-hoc networks

A generic model for key establishment in clustered ad-hoc networks works along these lines: First, nodes form clusters with some clustering method. Then a backbone or a key-tree is formed from the clusters, sometimes the tree extends inside clusters, sometimes the clusters are considered as single vertex in the tree. After this, the initial key agreement begins. Usually keys are established in subgraphs first, and then combined for a whole group wide key. The Diffie-Hellman key exchange (bipartite or tripartite) is typically used recursively as a basis for the group keying. A group key is constructed so that every node can calculate it using its own secret and the blinded secrets of others, or combinations of them. In some scenarios the messages are signed and key confirmation messages are sent for authentication purposes.

Rhee, et al., [104] present an architecture for key management in hierarchical mobile ad-hoc networks. They use implicitly certified public keys (ICPK) [105], an ID-based public key scheme where the public key of each participant is derived from its identity. It provides computationally efficient implicit authentication. A

key confirmation message added to the key agreement protocol makes the protocol explicitly authenticated. A two layered hierarchy is prompted by a physically two-layered network, ground nodes and unmanned aerial vehicles. The layers use different key management methods, the clusters of nodes below use a centralised system, while the aerial vehicles use TGDH. The centralised system inside clusters is not contributory.

Another hierarchical key agreement is proposed in [106]. This is a multilevel hierarchy, where a node can have several cluster keys according to the cluster and its super-clusters it belongs to. However, it is not completely contributory. Keys are agreed among cluster-heads on the same level and then distributed to their respective clusters.

Hybrid key management [107] propose a clustered key establishment, where each cluster selects a cluster-head that makes a key agreement with other cluster-heads. After that, the cluster-head distributes the key to the cluster. Thus, other nodes in the cluster do not contribute to the key. Clustering is made according to the geometric locations of the nodes. The key agreement used can be any group key agreement protocol, for example GDH [108].

ACEKA A cluster-tree-based group key agreement ACEKA is presented in [94]. ACEKA uses ternary trees with the Joux tripartite Diffie-Hellman key agreement [109]. There is a virtual backbone and virtual nodes in addition to the real nodes. ACEKA uses cluster-heads and "sponsors" for management. Authentication is done by signing every message using ID-based cryptography, with a variant of the ElGamal signature scheme.

5.4.5 Clustered AT-GDH

First, the network is divided into clusters with a clustering mechanism that creates very stable clusters. Nodes in a cluster are at a one hop distance from each other, i.e., cliques. In this kind of a cluster, the most efficient group key agreement protocol is the BD broadcast protocol explained before. It takes only two rounds of broadcasts, after which each node can calculate the common group key from its own secret exponent and the blinded shares of others.

When every cluster has a common secret key, the clusters agree a group key by AT-GDH protocol. Cluster-head can represent its cluster and use the cluster key as its secret exponent, instead of selecting a random $k_x \in \mathbb{Z}_q$ in Round 1. (see the box in Subsection 5.4.3). After the AT-GDH protocol run, cluster-heads distribute

(broadcast) the last received message in their cluster, so that other nodes can also calculate the network wide group key.

Now that cluster-heads are not necessarily at a one hop distance from each other, the messages need to be relayed. The gateways relaying the messages are members of a cluster, and know the cluster secret already. However, it affects the communication complexity by adding extra links to the path.

Complexity theoretic analysis of performance This clustered group key agreement is efficient when cliques are large. Radio connections may create relatively large cliques. The complexity was evaluated in respect to synchronous rounds and number of exponentiations. A *synchronous round* means that every participant can send arbitrarily many packages concurrently within a single time tick (round) or receive arbitrarily many at the beginning of a round. The *number of exponentiations* means the total number (the sum) of exponentiations performed by all participants.

Every clique forms a group key in two communication rounds, i.e., constant amount of rounds. In the end, cluster-heads broadcast the key parts in one communication round. The complexity of this clustered key agreement is the complexity of AT-GDH in the number of clusters plus a constant. The resulting communication complexity is logarithmic to the number of clusters c.

The above analysis is done without considering the cost of embedding the protocol in an arbitrary ad-hoc network topology. Thus, each protocol is evaluated in its optimal network topology. However, AT-GDH requires as its embedding only a protocol for generating a spanning tree, which can be done rather efficiently. The protocol described in [95] adds only h communication rounds, h equalling the height of the tree, i.e., the eccentricity of the initiating node.

Adding authentication Previously group key agreements, like the authenticated GDH, A-GDH, relied much on the implicit key authentication property: The group key can not be constructed without the secret share of one of the participants. However, Pereira and Quisquater [110] showed that it is impossible to design a scalable authenticated group key agreement protocol on the same building blocks as A-GDH. Hence other authentication methods are needed. Authentication with ID-based crypto, such as the ICPK [105] public keys, with key confirmation messages, could be used here, as it is independent of the group key establishment method used.

Literature surveys on different solutions for group key management, see, for example, [111, 112]. "*A survey of key management for secure group communication*" [111]

Chapter 6

Requirements for a security architecture

This chapter concentrates on the requirements that can be concluded from the previous chapters. Especially, what is important in securing clustered ad-hoc networks and also what is needed from the clustering algorithm itself.

6.1 Securing routing

In order to make communications secure in ad-hoc networks one needs to confront the following issues:

1. The originating node should have reason to trust that other nodes will not unnecessarily drop or delay its packets (nodes have reason to cooperate).
2. The forwarding nodes should send packets to the right direction (authenticity and integrity of the routing tables).
3. The forwarding nodes should not be able to alter the sender and receiver information of the packets (integrity of headers).
4. The forwarding nodes should not be able to alter the contents of the messages (integrity of message).
5. The forwarding nodes should not be able to read the contents of the message (confidentiality of the message).

Secure communications in ad hoc networks

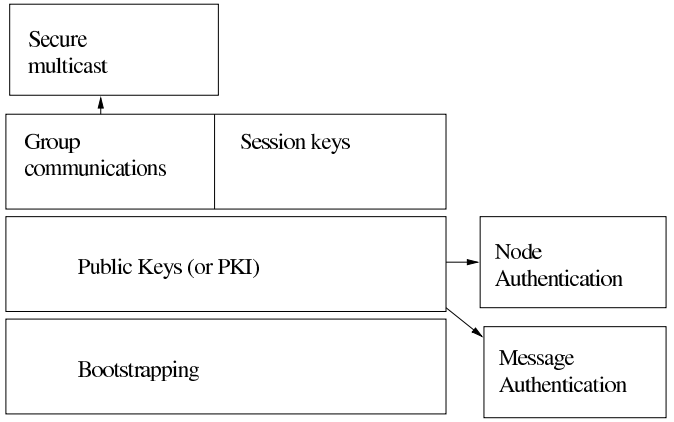


Figure 6.1: Secure communications in ad-hoc networks

6. The receiving node should be able to verify the identity of the sender (authenticity of the sender).
7. Duplicated packets should be detected (packet freshness).
8. Resource consumption attacks should be prevented (revocation, possibly access control).
9. The nodes should be able to perform the computations needed by the most critical network operations by using only the information provided by its neighbourhood (Locally computable solutions).

The desired functionality of the routing protocol could be described, for example, with the following requirements.

- Packets travel from their source to intended destination
- The route used should not differ much from the optimal one

- There is no unnecessary delays, or dropping of packets.

Requirements for secure routing could mean, at least, defence against the (known) routing attacks. Routing should be secure against specified active attacks (all is not possible) and against all passive attacks.

When planning the requirements, one should take into account the following.

- What kinds of states will nodes have or how much memory will be needed?
- What is the resource consumption
- Will the security measures make new attacks easier? Dos attacks, for example?
- How much overhead will security bring?

Routing in clustered network could, for example, operate in the following way.

- Have functionality in the Internet-working level, be connectionless.
- Hybrid system: proactive inside cluster
- Each node belongs to one and only one cluster
- "Distance-vector-like" routing tables: Tables have entries for each node in the same cluster and for each other cluster
- Basic idea of the routing protocol in two-tier ad-hoc networks:
  1. Hello messages broadcasted
  2. Cluster formation
  3. Finding routes (backbone and intra-cluster)
  4. Routing tables ready
- Hello -messages are unauthenticated, or use cryptographically generated identifiers. Messages can be time-stamped.
- There is no cluster-head but some nodes route messages to their neighbouring clusters (there can be many of these).

Routing should be able to withstand the attacks defined in Subsection 3.1. This means that the routing tables should maintain their integrity and authenticity. The routing protocol should find routes in a way that creates unfalsified routing tables. Possible forgeries in the routing tables should be detected.

6.2 Requirements for clustering

An ad-hoc network with clusters has additional properties: The clusters are assumed to stay together longer than the nodes do in average. Clusters are supposed to have more stable internal connections due to the greater amount of links between nodes in a same cluster. When clusters form as a result of some common background, they are likely to have a lot of internal communications as well. However, when nodes are no longer equally important in maintaining connectivity, a well-connected node (cluster-head) may become a single point of failure, a target for attacks aimed at cutting down the cluster's connections to other clusters. Very deep hierarchies can reduce the amount of routes, which also leads to weak points in the network. Routing attacks can also be targeted to cluster formation, by using a wormhole to make unpractical cluster formations. We have the following additional requirements for clustered ad-hoc network security:

- The cluster should have means for fast confidential communications between members of the cluster (common encryption key, group key)
- The members of a cluster should be able to maintain strong trust relations.
- When naming a cluster-head is not avoidable, the role of a cluster-head should be easily transferable to another node in a cluster.

Two-tier networks without specific cluster-heads have less single points of failure. Clustering should be locally computable. Clustering methods are preferred in which the nodes in a cluster stay together much longer than the nodes do in average. This is obviously difficult to predict but we assume that clusters are formed so that when links fail between two nodes inside a cluster, there are still other nodes in the cluster that can route between the nodes without the help of outsider. If not, the cluster breaks, and the nodes outside the cluster connections join a new cluster. Intuitively, the likelihood of long-lived clusters grows when the ratio of links inside a cluster versus links going out of it grows.

This means that the internal connections of a cluster are more stable. When clusters form a group, they are likely to have a lot of internal communications. The stability of a cluster is important when using a reputation system for cooperation management, in order to keep track of the behaviour of neighbouring nodes and form stronger trust relations.

6.3 Enabling network connectivity and stimulating cooperation

If negative reputation is allowed, a malicious node can indirectly attack a well-behaving node by spreading false information on its behaviour. Consequently, the well-behaving node's packets get low priority or the node can even be shut out of the network or face whatever measures the other parties choose to take towards a uncooperative or compromised node. Therefore, a reputation system should be used with care, for example, by banning the spreading of negative information, or using it only in context with threshold systems, were only a coalition of multiple nodes can give negative information on a node.

Trading tokens has other problems: what if someone just takes the tokens and moves from reach before forwarding packets? What if there are a finite number of tokens and a compromised node has more energy than other nodes? Then the node can make a DoS attack by not sending own packets and forwarding others' packets until the system runs out of tokens.

A clustered solution for cooperation is sketched in [113]. The idea in this framework is to use a reputation system within clusters and between clusters a mechanism similar to token-trading. Assuming the clusters are of moderate sizes and long-lived, this approach will make the best of both worlds. Long-lived clusters together with the trust relations inside clusters make reputation systems feasible. Having token trading between clusters only can reduce the management of the currency system.

6.3.1 Cooperation inside a cluster

Clusters form communities, where reputation is the main method for stimulating cooperation. The nodes observe the behaviour of the other members of a cluster, specifically, packet forwarding and reporting. Cluster uses a shared secret (for example a threshold secret) to manage the reputation status of each node. If a node drops too many packets, its priority in packet forwarding is reduced and if it gives false report of another node, its reports can be ignored or other actions taken towards it.

6.3.2 Cooperation between clusters

A cluster has a common secret or a public-private key pair, which is used for signing in transactions between clusters. There can be a virtual currency system

or the clusters can "loan" packet-forwarding services from each other.

6.4 Managing security associations and trust relations

6.4.1 Trust relations inside a cluster

The nodes inside a cluster are likely to be more connected to each other, thus protocols requiring on-line contacts *inside* a cluster may still be realistic. Therefore, a system with public keys and a local central entity is possible. However, as a cluster can still be split, the role of central entity should be divided by more than one node

6.4.2 Trust relations between clusters

As we have a two-tier hierarchy, cluster-heads do not have a central entity between themselves that could work as a trusted party. Trust relations between clusters could be arranged by having cluster-heads or clusters certify each other with the help of certificate repositories, as has been done in [89].

6.5 Secure communications

Clusters may have needs for common session keys for fast encrypted communication inside the cluster. Group key systems are useful for these: a common symmetric key can be used in encrypting a message meant for all nodes inside a cluster, and the same message is forwarded between cluster nodes, instead of sending a separate message for each member of the cluster, encrypted with the respective public key of each node. These keys can be created with, for example, a group key agreement, using shares from every node in cluster, as has been done in [95].

6.6 Summary of the most important requirements

routing security	authentication, time stamps
cooperation method	locally countable, reciprocal
managing trust relations	distributed
communication	not too heavy, distributed, secure
clustering	loc. countable, stable clusters, no single points of failure

6.7 Future work

Bootstrapping the trust relations is one of the topics of future research. How can a node be sure that it has been given the right public key of a CA, or other such entity? In other words, where does the first key come from? This problem does not seem to have a general solution, it depends on the application environment.

Bibliography

[1] ANSI/IEEE. Std 802.11, 1999, information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. Technical report, ISO/IEC 8802-11 IEEE Std 802.11, September 1999.

[2] J. Bray and C. F. Sturman. *Bluetooth: Connect Without Cables*. Prentice Hall, 2001.

[3] C.-F. Huang, H.-W. Lee, and Y.-C. Tseng. A two–tier heterogeneous mobile ad hoc network architecture and its load–balance routing problem. *Mobile Networks and Applications MONET*, 9(4):379–391, August 2004.

[4] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad Hoc Networks, Special Issue on Sensor Network Applications and Protocols*, 1(2-3):293–315, September 2003.

[5] C. E. Perkins and P. Bhagwat. Highly dynamic destination–sequenced distance–vector routing (DSDV) for mobile computers. In *Proceedings of the (SIGCOMM) Conference on Communications Architectures, Protocols and Applications*, pages 234–244, London, UK, August 1994. ACM.

[6] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, 353:153–181, 1996.

[7] C. E. Perkins and E. M. Royer. Ad hoc on–demand distance vector routing. In *Proceedings of the Workshop on Mobile Computing Systems and Applications (WMCSA)*, New Orleans, Louisiana, USA, 1999. IEEE.

[8] M. Adler and C. Scheideler. Efficient communication strategies for ad hoc wireless networks. *Theory of Computing Systems*, 33(5-6):337–391, September 2000.

[9] C. Ellison. SPKI requirements. RFC 2692, IETF Network Working Group, September 1999.

[10] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT–22(6):644–654, November 1976.

[11] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[12] H. Petersen and P. Horster. Self–certified keys – concepts and applications. In *Proceedings of the Conference on Communications and Multimedia Security (CMS)*, volume 97, pages 102–116, Athens, Greece, September 1997.

[13] G. Montenegro and C. Castelluccia. Statistically unique and cryptographically verifiable (SUCV) identifiers and addresses. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, February 2002.

[14] Y. G. Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4):449–457, July–August 1994.

[15] A. Perrig, R. Canetti, D. Song, and J. Tygar. Efficient and secure source authentication for multicast. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, volume 1, pages 35–46, 2001.

[16] Y.-C. Hu, A. Perrig, and D. Johnson. Rushing attacks and defence in wireless ad hoc network routing protocols. In *Proceedings of the Workshop on Wireless Security (WiSe)*, San Diego, CA, USA, September 2003. ACM Press.

[17] J. Douceur. The Sybil attack. In *Proceedings of the International Workshop on (P)eer–to–(P)eer Systems (IPTPS)*, Cambridge, MA, USA, March 2002.

[18] Y.-C. Hu, A. Perrig, and D. B. Johnson. Wormhole detection in wireless ad hoc networks. Technical Report TR01-384, Department of Computer Science, Rice University, December 2001.

[19] L. Kleinrock and F. Kamoun. Hierarchical routing for large networks; performance evaluation and optimization. *Computer Networks*, 1:155–174, 1977.

[20] F. Kamoun and L. Kleinrock. Stochastic performance evaluation of hierarchical routing for large networks. *Computer Networks*, 3:337–353, November 1979.

[21] G. Pei, M. Gerla, X. Hong, and C.-C. Chiang. A wireless hierarchical routing protocol with group mobility. In *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, pages 1538–1542, New Orleans, USA, September 1999. IEEE.

[22] D. Gu, G. Pei, H. Ly, M. Gerla, and X. Hong. Hierarchical routing for multilayer ad hoc wireless networks with UAVs. In *Proceedings of the Military Communication Conference (MILCOM)*, October 2000.

[23] Z. Haas. A new routing protocol for the reconfigurable wireless networks. In *Proceedings of the International Conference on Universal Personal Communications (ICUPC)*, pages 562–565, San Diego, CA, USA, October 1997. IEEE.

[24] Z. Haas and M. Pearlman. Providing ad hoc connectivity with the reconfigurable wireless networks. In *Proceedings of the (SIGCOMM) Conference on Communications Architectures, Protocols and Applications*, Vancouver, Canada, September 1998. ACM.

[25] D. Kim, S. Ha, and Y. Choi. K–hop cluster–based dynamic source routing in wireless ad hoc packet radio network. In *Proceedings of the Vehicular Technology Conference (VTC)*, pages 224–228, Ottawa, Ont Canada, 1998. IEEE.

[26] Q. Li, J. Aslam, and D. Rus. Online power–aware routing in wireless ad hoc networks. In *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom)*, pages 97–107, Rome, Italy, July 2001. ACM Press.

[27] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen. Scalable routing strategies for ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1369–1379, August 1999.

[28] G. Pei, M. Gerla, and T.-W. Chen. Fisheye state routing: A routing scheme for ad hoc wireless networks. In *Proceedings of the International Conference on Communications (ICC)*, volume 1, pages 70–74. IEEE, 2000.

[29] E. E. Tsuchiya. The landmark hierarchy: a new hierarchy for routing in very large networks. *Computer Communication Review*, 18(4):35–42, August 1988.

[30] G. Pei, M. Gerla, and X. Hong. LANMAR: Landmark routing for large scale wireless ad hoc networks with group mobility. In *Proceedings of the International Symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 11–18, Boston, MA, USA, August 2000. IEEE Press.

[31] X. Hong, K. Xu, and M. Gerla. Scalable routing protocols for mobile ad hoc networks. *IEEE Network*, July/August 2002.

[32] X.-Y. Li, K. Moaveninejad, and O. Frieder. Regional gossip routing for wireless ad hoc networks. *Mobile Networks and Applications MONET*, 10(1 – 2):61–77, February 2005.

[33] A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom)*, pages 96–108, San Diego, CA, USA, September 2003.

[34] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster–based approach for routing in dynamic networks. *ACM SIGCOMM Computer Communication Review*, pages 49–65, April 1997.

[35] J. Sucec and I. Marsic. Clustering overhead for hierarchical routing in mobile ad hoc networks. In *Proceedings of the IEEE Conference of the Computer and Communications Societies INFOCOM*, volume 3, New York, USA, June 2002. IEEE Press.

[36] S. Ramanathan and M. Steenstrup. Hierarchically–organized, multihop mobile wireless networks for Quality–of–Service support. *Mobile Networks and Applications MONET*, 3(1):101–119, June 1998.

[37] C. R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, 15(7):1265–1275, 1997.

[38] S. Basagni. Distributed clustering for ad hoc networks. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms, and Networks (ISPAN)*, pages 310–315, Perth, Australia, June 1999. IEEE.

[39] S. E. Virtanen and P. Nikander. Local clustering for hierarchical ad hoc networks. In *Proceedings of WiOpt: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pages 404–405, Cambridge, UK, March 2004. IEEE.

[40] A. Ephremides, J. E. Wieselthier, and D. J. Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. volume 75, pages 56–73. IEEE, IEEE Press, January 1987.

[41] M. Gerla and J. T.-C. Tsai. Multicluster, mobile multimedia radio network. *Wireless Networks*, 1:255–265, 1995.

[42] S. Banerjee and S. Khuller. A clustering scheme for hierarchical control in multi–hop wireless networks. In *Proceedings of the IEEE Conference of the Computer and Communications Societies INFOCOM*, volume 3, pages 1028–1037, Anchorage, Alaska, USA, April 2001.

[43] P. Basu, N. Khan, and T. D. C. Little. Mobility based metric for clustering in mobile ad hoc networks. In *Proceedings of the International Workshop on Wireless Networks and Mobile Computing (WNMC)*, pages 16–19, 2001.

[44] M. Bechler, H.-J. Hof, D. Kraft, F. Phälke, and L. Wolf. A cluster–based security architecture for ad hoc networks. In *Proceedings of the IEEE Conference of the Computer and Communications Societies INFOCOM*, Hong Kong, China, March 2004. IEEE.

[45] T.-C. Hou and T.-J. Tsai. An access–based clustering protocol for multihop wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 19(49):1201–1210, July 2001.

[46] D. J. Baker and A. Ephremides. A distributed algorithm for organizing mobile radio telecommunication networks. In *Proceedings of the International Conference on Distributed Computer Systems (ICDCS)*, pages 476–483, April 1981.

[47] D. J. Baker and A. Ephremides. The architectural organization of a mobile radio network via a distributed algorithm. *IEEE Transactions on Communications*, 11:1694–1791, November 1981.

[48] C.-C. Chiang, H.-K. Wu, W. Liu, and M. Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. In *Proceedings of the Singapore International Conference on Networks (SICON)*, pages 197–211, Singapore, April 1997. IEEE.

[49] M. Steenstrup. Cluster–based networks. In C. E. Perkins, editor, *"Ad Hoc Networks"*, pages 75–138. Addison Wesley, 2001.

[50] G. Nemeth and G. Vattay. Giant clusters in random ad hoc networks. *Physical Review E67*, March 2003.

[51] S. Bouam and J. Ben-Othman. Data security in ad hoc networks using multipath routing. In *Proceedings of the International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Beijing, China, September 2003.

[52] S. Bouam and J. Ben-Othman. SDMP: Securing Data Based MultiPath Routing Protocol. In *Proceedings of the Mediterranean Ad Hod Networking Conference (MedHoc Net)*, Mahdia, Tunisia, June 2003.

[53] J. Wang, H. Zhai, W. Liu, and Y. Fang. Reliable and efficient packet forwarding by utilizing path diversity in wireless ad hoc networks. In *Proceedings of the Military Communication Conference (MILCOM)*, Monterey, CA, USA, Oct–Nov 2004.

[54] T. B. Reddy, S. Sriram, B. S. Manoj, and C. S. R. Murthy. MuSeQoR: Multi–path failure–tolerant security–aware QoS routing in ad hoc wireless networks. *Computer Networks*, 50:1349–1383, June 2006.

[55] P. Papadimitratos and Z. J. Haas. Secure message transmission in mobile ad hoc networks. *Ad Hoc Networks*, 1(1), July 2003.

[56] M.O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM (JACM)*, 36(2):335–348, 1989.

[57] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A secure on–demand routing protocol for ad hoc networks. In *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom)*, Atlanta, Georgia, USA, September 2002. ACM.

[58] T. Wan, E. Kranakis, and P. C. van Oorschot. S–RIP: A secure distance vector routing protocol. In J. Zhou M. Jakobsson, M. Yung, editor, *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS)*, volume 3089 of *LNCS*, pages 103–119, Yellow Mountain, China, June 2004. Springer.

[59] H. Yang, X. Meng, and S. Lu. Self–organized network layer security in mobile ad hoc networks. In *Proceedings of the Workshop on Wireless Security (WiSe)*, Atlanta, Georgia, USA, September 2002.

[60] Y.-C. Hu, A. Perrig, and D. B. Johnson. Packet leashes: A defence against wormhole attacks in wireless ad hoc networks. In *Proceedings of the IEEE Conference of the Computer and Communications Societies INFOCOM*, San Francisco, CA, USA, April 2003.

[61] P. Papadimitratos and Z. J. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, San Antonio, TX, USA, January 2002. SCS.

[62] Z.J. Haas, M. Perlman, and P. Samar. The interzone routing protocol (ierp) for ad hoc networks. draft-ietf-manetzone-ierp-01.txt, IETF MANET Working Group, June 2001.

[63] Y.-C. Hu, D. B. Johnson, and A. Perrig. SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *Proceedings of the Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 3–13, Calicoon, NY, USA, June 2002. IEEE.

[64] Y.-C. Hu and A. Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy*, pages 28–39, May/June 2004.

[65] W. Wang, Y. Lu, and B. Bhargava. On security study of two distance vector routing protocols for mobile ad hoc networks. In *Proceedings of the International Conference on Pervasive Computing and Communications (PerCom)*, Dallas–Fort Worth, TX, USA, March 2003. IEEE.

[66] P. Lungaro. Cost/performance trade–offs in "two–layer" ad hoc multihop cellular access systems. Stockholm, Sweden, May 2004.

[67] S. Buchegger and J.-Y. Le Boudec. Performance analysis of the CONFIDANT protocol: Cooperation Of Nodes – Fairness In Distributed Ad hoc NeTworks. In *Proceedings of the International Symposium on Mobile ad hoc networking and computing (MobiHoc)*, Lausanne, Switzerland, June 2002.

[68] P. Michiardi and R. Molva. Core: A COllaborative REputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Proceedings of the Conference on Communication and Multimedia Security (CMS)*, 2002.

[69] H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang. URSA: Ubiquitous and robust access control for mobile ad hoc networks. *IEEE/ACM Transactions on Networking*, 12(6):1049–1063, December 2004.

[70] Y. Huang and W. Lee. A cooperative intrusion detection system for ad hoc networks. In *Proceedings of the Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, Fairfax, VA, USA, October 2003. ACM, ACM Press.

[71] Y. Zhang, W. Lee, and Y. Huang. Intrusion detection techniques for mobile wireless networks. *Wireless Networks*, 9(5):545–556, September 2003.

[72] F. Kargl, A. Klenk, S. Schlott, and M. Weber. Advanced detection of selfish or malicious nodes in ad hoc networks. In *Proceedings of the European Workshop on Security in Ad–Hoc and Sensor Networks (ESAS)*. Springer, 2004.

[73] R. Puttini, R. de Sousa, and L. Mé. Combining certification–based authentication and intrusion detection to secure manet routing protocols. In *Proceedings of the European Wireless Conference Mobile and Wireless Systems Beyond 3G (EW)*, Barcelona, Spain, February 2004.

[74] W. Zhang, R. Rao, G. Cao, and G. Kesidis. Secure routing in ad hoc networks and a related intrusion detection problem. In *Proceedings of the Military Communication Conference (MILCOM)*, Boston, USA, October 2003.

[75] P. Albers, O. Camp, J.-M. Percher, B. Jouga, L. M, and R. Puttini. Security in ad hoc networks: a general intrusion detection architecture enhancing trust based approaches. In *Proceedings of the International Workshop (WL) Information Systems, International Conference on Enterprise Information Systems*, 2002.

[76] L. Buttyán and J.-P. Hubaux. Stimulating cooperation in self–organizing mobile ad hoc networks. *Mobile Networks and Applications MONET*, 8:579–592, 2003.

[77] L. Buttyán and J.-P. Hubaux. Nuglets: a virtual currency to stimulate cooperation in self–organized mobile ad hoc networks. Technical Report DSC/2001/001, 2001.

[78] L. Buttyán and J.-P. Hubaux. Enforcing service availability in mobile ad hoc WANs. In *Proceedings of the International Symposium on Mobile ad hoc networking and computing (MobiHoc)*, Boston, MA, USA, August 2000. IEEE Press.

[79] L. Blazevic, L. Buttyán, S. Čapcun, S. Giordano, J.-P. Hubaux, and J.-Y. Le Boudec. Self–organization in mobile ad hoc networks: the approach of TermiNodes. *IEEE Communications Magazine*, June 2001.

[80] M. Jakobsson, J.-P. Hubaux, and L. Buttyán. A micro–payment scheme encouraging collaboration in multi–hop cellular networks. In *Proceedings*

of the International Conference on Financial Cryptography (FC), volume 2742 of *LNCS*, pages 15–33. Springer, August 2003.

[81] B. Lamparter, M. Plaggemeier, and D. Westhoff. Estimating the value of co–operation approaches for multi–hop ad hoc networks. *Ad Hoc Networks*, 3(1):17–26, January 2005.

[82] A. Urpi, M. A. Bonuccelli, and S. Giordano. Modeling cooperation in mobile ad hoc networks: a formal description of selfishness. In *Proceedings of WiOpt: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, Sophia–Antipolis, France, March 2003.

[83] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 1985.

[84] V. Srinivasan, P. Nuggehalli, C. Chiasserini, and R. Rao. Cooperation in wireless ad hoc networks. In *Proceedings of the IEEE Conference of the Computer and Communications Societies INFOCOM*, San Francisco, CA, USA, April 2003.

[85] Altman, Kherani, P. Michiardi, and R. Molva. Non–cooperative forwarding in ad hoc networks. In *Proceedings of the International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Barcelona, Spain, September 2004.

[86] F. Stajano and R. Anderson. The resurrecting duckling: Security issues in ad hoc wireless networks. In B. Christianson, B. Crispo, and M. Roe, editors, *Proceedings of the International Workshop on Security Protocols*, volume 1796 of *LNCS*, pages 172–182, Cambridge, UK, April 1999. Springer Verlag.

[87] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for mobile ad hoc networks. pages 251–260, Riverside, CA, USA, November 2001. IEEE.

[88] H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang. Self–securing ad hoc wireless networks. In *Proceedings of the Symposium on Computers and Communications (ISCC)*, pages 567–574. IEEE, 2002.

[89] S. Čapkun, L. Buttyán, and J.-P. Hubaux. Self–organized public–key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):52–64, Jan–Mar 2003.

[90] S. Jin and A. Bestavros. Small–world–internet–topologies possible causes and implications. Technical Report BUCS-2002-004, Boston University, January 2002.

[91] S. Čapkun, L. Buttyán, and J.-P. Hubaux. Small worlds in security systems: an analysis of the PGP certificate graph. In *Proceedings of the New Security Paradigms Workshop*. ACM, 2002.

[92] S. Zhu, S. Xu, S. Setia, and S. Jajodia. LHAP: A lightweight hop–by–hop authentication protocol for ad hoc networks. In *Proceedings of the International Conference on Distributed Computing Systems Workshops (ICDCSW)*, Providence, RI, May 2003. IEEE.

[93] L. A. Martucci, C. M. Schweitzer, Y. R. Venturini, T. C. M. B. Carvalho, and W. V. Ruggiero. A trust–based security architecture for small and medium–sized mobile ad hoc networks. In *Proceedings of the Mediterranean Ad Hod Networking Conference (MedHoc Net)*, pages 278–290, Bodrum, Turkey, June 2004.

[94] H. Shi and M. He. Authenticated and communication efficient group key agreement for ad hoc networks. In *The 5th International Conference on Cryptology and Network Security CANS06*, Suzhou, China, December 2006.

[95] M. Hietalahti. Efficient key agreement for ad hoc networks. Master's thesis, Helsinki University of Technology, Department of Computer Science and Engineering, Espoo, Finland, May 2001.

[96] G. Di Crescenzo, M. Striki, and J. S. Baras. Modeling key agreement in multi-hop ad hoc networks. In *Proceeding of the 2006 International Conference on Communications and Mobile Computing*, Vancouver, British Columbia, Canada, July 2006.

[97] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology – Proceedings of EUROCRYPT*, volume 950 of *LNCS*, pages 275–286, Perugia, Italy, May 1994. Springer.

[98] M. Hietalahti. A clustering-based group key agreement protocol for ad-hoc networks. In *WCAN 2007*, Wroclaw, Poland, July 2007.

[99] F. Gärtner. A survey of self-stabilizing spanning-tree construction algorithms. Technical report, EPFL, 2003.

[100] R. Diestel. *Graph Theory*. Springer, New York, USA, 1997.

[101] Y. Kim, A. Perrig, and G. Tsudik. Simple and fault–tolerant key agreement for dynamic collaborative groups. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 235–244, Athens, Greece, November 2000. ACM.

[102] K. Becker and U. Wille. Communication complexity of group key distribution. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 1–6, San Francisco, CA, USA, November 1998. ACM Press.

[103] S. Lee, Y. Kim, K. Kim, and D. H. Ruy. An efficient tree–based group key agreement using bilinear map. In *Applied Cryptography and Network Security ACNS*, volume 2486 of *LNCS*, pages 357–371. Springer, 2003.

[104] K. H. Rhee, Y. H. Park, and G. Tsudik. A group key management architecture for mobile ad-hoc wireless networks. *Journal of Information Science and Engineering*, 21:415–428, 2005.

[105] C. Günther. An identity–based key exchange protocol. In *Advances in Cryptology – Proceedings of EUROCRYPT*, volume 434 of *LNCS*, pages 29–37, 1989.

[106] G. Yao, K. Ren, F. Bao, R. H. Deng, and D. Feng. Making the key agreement protocol in mobile ad hoc network more efficient. In *Applied Cryptography and Network Security ACNS*, volume 2846 of *LNCS*, pages 343–356. Springer, 2003.

[107] X.-Y. Li, Y. Wang, and O. Frieder. Efficient hybrid key agreement protocol for wireless ad hoc networks. In *Computer Communications and Networks, 2002. Proceedings. Eleventh International Conference on*, pages 404– 409, October 2002.

[108] M. Steiner, G. Tsudik, and M. Waidner. Diffie–hellman key distribution extended to group communication. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 31–37, New Delhi, India, March 1996. ACM, ACM Press.

[109] A. Joux. A one round protocol for tripartite Diffie–Hellman. In *Proceedings of Algorithmic Number Theory Symposium IV*, volume 1838 of *LNCS*, pages 385–394. Springer, 2000.

[110] O. Pereira and J.-J. Quisquater. Generic insecurity of cliques-type authenticated group key agreement protocols. In *Proceedings of the IEEE Computer Security Foundations Workshop (CSFW)*. IEEE Computer Society Press, June 2004.

[111] S. Rafaeli and D. Hutchison. A survey of key management for secure group communication. *ACM Computing Surveys*, 35(3):309–329, September 2003.

[112] M. Hietalahti. Key establishment in ad hoc networks. In *Proceedings of the Seminar on Network Security (NetSec). Telecommunications Software and Multimedia Laboratory, Helsinki University of Technology*. Telecommunications Software and Multimedia Laboratory, Helsinki University of Technology, 2000.

[113] M. Hietalahti. Cooperation in clustered ad hoc networks. In *Proceedings of the 5th Scandinavian Workshop on Wireless Ad-hoc Networks Adhoc'05*, Stockholm, Sweden, May 2005. poster.