Communication and Security in Self-Organizing Networks

by

Dipl.-Ing. Stefan Kraxberger

A PhD Thesis
Presented to the Faculty of Computer Science in Partial Fulfillment of the Requirements for the PhD Degree

Assessors

Prof. PhD Roderick Bloem (TU Graz, Austria)
Prof. Dr. Manuel Diaz (Uni. Malaga, Spain)

November 2011

Institute for Applied Information Processing and Communications (IAIK)
Faculty of Computer Science
Graz University of Technology, Austria

Abstract

Self-organizing networks are a novel communication technology that has gained a lot of importance in the last decade. This thesis focuses on communication and security aspects of self-organizing networking technology. We analyze the problems in such networks with the main focus on security and reliability issues. Using this analysis as basis we present a security concept for self-organizing networks which outlines requirements and recommendations which allow designers and developers of such systems to select appropriate measures to secure their systems based on their requirements and node capabilities. Using this security concept as guidance we evaluate existing mechanisms for self-organizing networks and present solutions for the integral components identity management, secure route establishment and management, secure communication and protection against misbehavior. Within identity management we focus on the issue of providing unique, verifiable and undeniable identities in order to address the issue of Sybil attacks in open systems, which is still a major concern. We present two solutions for this problem based on Trusted Computing principles and mechanisms facilitating the Trusted Platform Module and the PrivacyCA concept to address privacy concerns. We investigate routing algorithms according their suitability as building blocks for secure solutions for self-organizing networks. We present the most important existing secure routing protocol and provide an evaluation of its capabilities and outline its drawbacks. Based on this analysis we present our own solution which incorporates the ideas and recommendations outlined in the security concept. Our secure routing algorithm is designed to incorporate heterogeneous devices ranging from wireless sensor nodes to powerful workstations. The secure routing algorithm is able to adapt quickly to a changing environment and copes well with churn. Another major point in the design and implementation was to provide protection from the most common attacks known in self-organizing networks. We show the design and architecture of a framework we developed in which we have integrated all the previous protocols into a comprehensive solution for secure overlay networking. We evaluate the performance and efficiency of this solution in a scenario as it was used in a European research project as well as in overlay networking simulation. The framework can be used efficiently in closed systems if cryptographic material must be deployed manually and in open systems if Trusted Platform Modules are available.

Acknowledgements

I would like to thank all people who supported me in writing this thesis.

I would like to thank Reinhard Posch for the possibility to do my PhD study at IAIK. As a member of the former Advanced Network, I especially thank Udo Payer who merits special acknowledgements for his support in any work and project related issues. I also would like to thank Karl-Christian Posch and Roderick Bloem who have been great mentors in my work as a University Assistant at TU Graz. I also would like to thank Manuel Diaz as external assessor of the thesis and for making the trip to Graz.

Furthermore, I thank Peter Teufl, Günther Lackner, Mario Lamberger, Martin Pirker and Ronald Tögl for many helpful discussions in different projects. Many thanks go to my colleagues at IAIK. Special thanks go to Stefan Tillich for his valuable inputs not only during our collaboration within the SMEPP project. I also would like to thank Daniel Hein for many fruitful discussions and collaboration time during the SECRICOM project. Furthermore, I thank all who reviewed parts of this work.

Finally, I would like to thank my family and friends who supported me during my studies. Without them, the work would not have been possible. Special thanks go to my parents, my brothers Andreas and Hermann. I also thank my friends and study colleagues Nelson Langkamp, Hannes Schneider and Christoph Gral.

Dipl.-Ing. Stefan Kraxberger
Graz, May 2011

Table of Contents

Abstract iii

Acknowledgements v

List of Tables xi

List of Figures xiii

List of Publications xvii

Acronyms xxi

Notation xxiii

1 Introduction 1
1.1 Self-Organizing Systems 2
1.2 Security in Self-Organizing Systems 4
1.3 Outline 6

2 Background 9
2.1 Overlay topologies 10
2.1.1 Distributed Hash Table (Structured) Overlays 10
2.1.2 Superpeer Overlay 11
2.1.3 Unstructured Overlay 12
2.2 Related Work 13

3 Assumption and Notation 19
3.1 Network assumptions 19
3.2 Node assumptions 20
3.3 Security Assumptions and Bootstrapping 21

4 Security Concept 23
4.1 Introduction 24
4.2 Background 24
4.3 Concept composition 26
4.3.1 Security levels 27
4.4 Applied security concept 30

4.4.1 Admission security 31
4.4.2 Data security 32
4.4.3 Session key protection 32
4.4.4 Global security considerations 33
4.4.5 Dealing with exposed credentials 33
4.4.6 Selection of security levels 34
4.5 Conclusion 34

5 Secure Identities in Dynamic Overlays 35
5.1 Motivation 35
5.2 Background 37
5.2.1 Current Problems and Attacks 37
5.3 Related Work 39
5.4 Trusted Computing Background 41
5.4.1 PrivacyCA 42
5.5 Assumptions 44
5.6 Trusted Identity Management 44
5.6.1 Endorsement Key and EK Certificate 45
5.6.2 Platform Certificate 45
5.7 Endorsement Key Solution 45
5.7.1 EK-based Authentication Protocol 47
5.8 Privacy CA Solution 49
5.8.1 Distributed Privacy CA 51
5.8.2 AIK protocol 52
5.9 Implementation and Evaluation 53
5.9.1 Trusted Authentication using EK solution 53
5.9.2 Trusted Authentication using Privacy CA solution 55
5.10 TPM Time Measures 57
5.10.1 Trusted Authentication using EK solution 57
5.10.2 Trusted Authentication using PrivacyCA solution 58
5.11 Security and Privacy Analysis 59
5.12 Conclusion 60

6 Secure Routing 61
6.1 Introduction 61
6.2 Security Routing Concept 62
6.3 Routing Security 63
6.3.1 Security Threats 64
6.3.2 Dynamic Source Routing 65
6.3.3 TESLA 66
6.3.4 Ariadne 67
6.4 Implementation 68
6.4.1 Sender Setup 68
6.4.2 Initialization Process 69
6.4.3 Key Exchange 71
6.4.4 Routing Processes 72

6.5 Performance Evaluation . . . 75
6.5.1 Bandwidth Requirements . . . 75
6.5.2 Minimal Timeout . . . 76
6.5.3 Route Recovery . . . 77
6.6 Discussion . . . 78
6.7 Scalable Secure Routing . . . 79
6.7.1 Network assumptions . . . 79
6.7.2 Node assumptions . . . 81
6.7.3 Security Assumptions and Bootstrapping . . . 81
6.8 Secure Routing Protocol . . . 81
6.8.1 Architecture . . . 81
6.9 Implementation . . . 83
6.9.1 Route Discovery . . . 83
6.9.2 Route Maintenance . . . 86
6.10 Security analysis . . . 88
6.11 Performance evaluation . . . 88
6.12 Related Work . . . 92
6.13 Conclusion . . . 93

7 Secure P2P Framework 95
7.1 Introduction . . . 95
7.2 Assumptions . . . 96
7.3 Framework Design . . . 96
7.3.1 Architecture . . . 97
7.4 Implementation . . . 101
7.4.1 Initialization . . . 101
7.4.2 Join protocol . . . 102
7.4.3 Routing protocols . . . 103
7.4.4 Routing table . . . 104
7.4.5 Insider Attack / Malicious Peer Detection . . . 104
7.5 Performance evaluation . . . 106
7.5.1 Only the shortest route is used . . . 107
7.5.2 All routes are used . . . 112
7.5.3 Neighbor-Route Distribution . . . 117
7.5.4 Message delivery latency . . . 119
7.6 Related Work . . . 132
7.7 Conclusion . . . 133

8 Secure P2P Applications 135
8.1 Introduction . . . 135
8.2 Motivation . . . 136
8.3 JADE Multi-Agent System . . . 138
8.3.1 JADE Background . . . 138
8.3.2 JADE RMI . . . 139
8.3.3 JADE Messaging . . . 139
8.4 Secure Multi-Agent System . . . 140

8.4.1 Implementation 140
8.4.2 Object-to-object concept 142
8.4.3 Sending semaphore 142
8.4.4 NodeProxy generation 143
8.4.5 A message sending sequence 143
8.5 Security Analysis 144
8.5.1 Analysis of the SePP-Jade solution 145
8.6 Performance evaluation 145
8.7 Related Work 146
8.8 Conclusion 147

9 Conclusion 149

Bibliography 153

List of Tables

4.1 Admission security levels 28
4.2 Data security levels 29
4.3 Protection levels 29
4.4 Measures against security breach 33

5.1 Test Time Measures for TAP without PCA 58
5.2 Test Time Measures for TAP with PCA 59

6.1 Processing time of the different security levels at the participating peers 89

8.1 Processing time of the different security levels at the participating peers 146

List of Figures

2.1 Overlay in general 15
2.2 DHT based ring topology 16
2.3 Superpeer topology 17
2.4 Unstructured topology 18

3.1 Real network topology with Internet and LAN nodes 21

4.1 Levels of security in the group concept 30

5.1 Overview of the remote attestation process including prerequisites such as AIK creation and certification. 43
5.2 Authentication process using Endorsement Key solution. 46
5.3 Basic security related exchange protocol 47
5.4 EK-based join/authentication protocol 48
5.5 Authentication process using PrivacyCA solution. 51
5.6 Trusted Authentication Protocol without PrivacyCA. 54
5.7 Trusted Authentication Protocol with PrivacyCA. 56

6.1 One-way hash chain 67
6.2 Direct time synchronization between peers 69
6.3 Time synchronization request message format 70
6.4 Time synchronization response message format 70
6.5 Time synchronization final message format 71
6.6 Key request message format 72
6.7 Route request message format 73
6.8 Route reply message format 74
6.9 Comparison of the needed bandwidth for DSR and Ariadne . . 75
6.10 Comparison of the needed bandwidth depending of the route length and the length of the used TESLA keys and MACs . . . 76
6.11 Minimal timeout for a Route Discovery with bootstrapping . . . 77
6.12 Minimal needed time out for a Route Discovery without bootstrapping 78
6.13 Real network topology with Internet and LAN nodes 80
6.14 Overlay network topology with groups of different security levels 82
6.15 Scalable-secure routing protocol with security level medium . . 84
6.16 Scalable secure routing protocol in security level high 85

6.17 Secure routing protocol route error message with security level medium . . . 87
6.18 Secure routing protocol route error message with security level high . . . 88
6.19 Route discovery latency if no route is available . . . 90
6.20 Data transmission latency if route is available . . . 91

7.1 Middleware architecture . . . 98
7.2 Route length distribution with 2 neighbors per node and 1000 nodes in total . . . 108
7.3 Route length distribution with 4 neighbors per node and 1000 nodes in total . . . 109
7.4 Route length distribution with 10 neighbors per node and 1000 nodes in total . . . 110
7.5 Route length distribution with 20 neighbors per node and 1000 nodes in total . . . 111
7.6 Route length distribution with 30 neighbors per node and 1000 nodes in total . . . 112
7.7 Amount of established routes for 1000 nodes . . . 113
7.8 Route length distribution with 2 neighbors per node and 1000 nodes in total . . . 114
7.9 Route length distribution with 2 neighbors per node and 1000 nodes in total . . . 115
7.10 Route length distribution with 5 neighbors per node and 1000 nodes in total . . . 116
7.11 Route length distribution with 10 neighbors per node and 1000 nodes in total . . . 117
7.12 Route length distribution with 15 neighbors per node and 1000 nodes in total . . . 118
7.13 Amount of established routes for 1000 nodes . . . 119
7.14 Route distribution in relation to the amount of neighbors per node . . . 120
7.15 Route distribution in relation to the amount of neighbors per node in logarithmic depiction . . . 121
7.16 Maximum message delivery latency with a maximum of 20% down nodes and different amounts of churn. . . . 123
7.17 Maximum message delivery latency with a maximum of 10% down nodes. . . . 124
7.18 Maximum message delivery latency with a maximum of 5% down nodes. . . . 124
7.19 Average message delivery latency with different amounts of churn. 125
7.20 Average message delivery latency with a maximum of 10% down nodes . . . 126
7.21 Average message delivery latency with a maximum of 5% down nodes. . . . 126

7.22 Message queue length with a maximum of 20% down nodes with different amount of churn. 127
7.23 Message queue length with a maximum of 10% down nodes . . 128
7.24 Message queue length with a maximum of 5% down nodes. . . . 128
7.25 Message delivery latency for the overall average and all messages without queuing. 129
7.26 Message delivery latencies for messages which where piggypacked on route errors and for the overall maximum. 131
7.27 Amount of route requests sent in relation to the message delivery latency. 132

8.1 Example of a disaster response scenario with on-site equipment 137
8.2 Secure Jade Architecture 141
8.3 Message sequence 144

List of Publications

In Refereed Journals

1. Daniel M. Hein, Ronald Toegl, and Stefan Kraxberger. An Autonomous Attestation Token to Secure Mobile Agents in Disaster Response. *Security and Communication Networks*, 3(5):421–438, 2010.

2. Sandra Dominikus and Stefan Kraxberger. Secure communication with RFID tags in the Internet of Things. *Security and Communication Networks*, 2011. Accepted but not yet published.

In Refereed Conference Proceedings

1. Udo Payer, Peter Teufl, Stefan Kraxberger, and Mario Lamberger. Massive Data Mining for Polymorphic Code Detection. In Vladimir Gorodetsky, Igor V. Kotenko, and Victor A. Skormin, editors, *MMM-ACNS*, Volume 3685 of *Lecture Notes in Computer Science*, pages 448–453. Springer, 2005.

2. Udo Payer and Stefan Kraxberger. Polymorphic Code Detection with GA Optimized Markov Models. In Jana Dittmann, Stefan Katzenbeisser, and Andreas Uhl, editors, *Communications and Multimedia Security*, Volume 3677 of *Lecture Notes in Computer Science*, pages 210–219. Springer, 2005.

3. Stefan Kraxberger, Udo Payer, and Stefan Tillich. General Security Concept for Embedded P2P Systems. In *Mobiquitous '08: Proceedings of the 5th Annual International Conference on Mobile and Ubiquitous Systems*, pages 1–5. ICST, 2008.

4. Stefan Kraxberger and Udo Payer. Secure Routing Approach for Unstructured P2P Systems. In *SECUREWARE '09: Proceedings of the 2009 Third International Conference on Emerging Security Information, Systems, and Technologies*, pages 210–216. IEEE Computer Society, 2009.

5. Udo Payer, Stefan Kraxberger, and Peter Holzer. IPv6 Label Switching on IEEE 802.15.4. In *SENSORCOMM '09: Proceedings of the 2009 Third International Conference on Sensor Technologies and Applications*, pages 650–656. IEEE Computer Society, 2009.

6. Stefan Kraxberger and Udo Payer. Security Concept for Peer-to-Peer Systems. In *IWCMC '09: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing*, pages 931–936. ACM, 2009.

7. Peter Danner, Daniel Hein, and Stefan Kraxberger. Securing Emergency Response Operations using Distributed Trust Decisions. In *Proceedings of the 2010 Fourth International Conference on Network and System Security*, NSS '10, pages 62–69, Washington, DC, USA, 2010. IEEE Computer Society.

8. Stefan Kraxberger, Peter Danner, and Daniel M. Hein. Secure Multi-Agent System for Multi-Hop Environments. In Igor V. Kotenko and Victor A. Skormin, editors, *MMM-ACNS*, volume 6258 of *Lecture Notes in Computer Science*, pages 270–283. Springer, 2010.

9. Stefan Kraxberger, Günther Lackner, and Udo Payer. WLAN Location Determination without Active Client Collaboration. In Ahmed Helmy, Peter Mueller, and Yan Zhang, editors, *IWCMC*, pages 1188–1192. ACM, 2010.

10. Stefan Kraxberger. A Scalable Secure Overlay Framework for Heterogeneous Embedded Systems. In *2010 IEEE 35th Conference on Local Computer Networks (LCN)*, pages 236 –239, October 2010.

11. Sandra Dominikus, Manfred Aigner, and Stefan Kraxberger. Passive RFID Technology for the Internet of Things. In *Internet Technology and Secured Transactions (ICITST), 2010 International Conference for*, pages 1 –8, nov. 2010.

12. Bernd Bergler, Christopher Preschern, Andreas Reiter, and Stefan Kraxberger. Cost-effective Routing for a Greener Internet. In *Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, GREENCOM-CPSCOM '10, pages 276–283, Washington, DC, USA, 2010. IEEE Computer Society.

13. Stefan Kraxberger. Scalable Secure Routing for Heterogeneous Unstructured P2P Networks. In Yiannis Cotronis, Marco Danelutto, and George Angelos Papadopoulos, editors, *PDP*, pages 619–626. IEEE Computer Society, 2011.

14. Manfred Aigner, Stefan Kraxberger, Sandra Dominikus, and Hannes Gross. Low-cost RFID Tags as IPv6 Nodes in the Internet of Things. In Ping Wang Guilin Wang Tieyan Li, Chao-Hsien Chu, editor, *Radio Frequency Identification System Security - RFIDsec 2011 Asia Workshop Proceedings*, volume 6 of *Cryptology and Information Security Series*, pages 114–128. IOPress, 2011.

15. Clemens Orthacker, Peter Teufl, Stefan Kraxberger, Gnther Lackner, Michael Gissing, Alexander Marsalek, Johannes Leibetseder, and Oliver Prevenhueber. Android Security Permissions - Can we trust them? In *Proceedings of the 3rd International ICST Conference on Security and Privacy in Mobile Information and Communication Systems*, Lecture Notes in Computer Science, Berlin, Heidelberg, 2011. Springer.

16. Peter Teufl, Stefan Kraxberger, Clemens Orthacker, Gnther Lackner, Michael Gissing, Alexander Marsalek, Johannes Leibetseder, and Oliver Prevenhueber. Android Market Analysis with Activation Patterns. In *Proceedings of the 3rd International ICST Conference on Security and Privacy in Mobile Information and Communication Systems*, Lecture Notes in Computer Science, Berlin, Heidelberg, 2011. Springer.

17. Stefan Kraxberger, Christoph Gratl, and Udo Payer. Performance Evaluation of IPv6 Label Switched Paths for Wireless Sensor Networks. In *IWCMC '11: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing*, pages 1459–1463, New York, NY, USA, 2011. IEEE Computer Society.

18. Peter Teufl and Stefan Kraxberger. Extracting Semantic Knowledge from Twitter. In *ePart 2011 - Third International Conference on eParticipation*, 2011. Accepted but not yet published.

Contributions to Refereed Workshops Without Formal Proceedings

1. Stefan Kraxberger and Udo Payer. Markov Model for Polymorphic Shellcode Detection. In *Proceedings of the 2005 Fifth International Networking Conference*, July 2005.

2. Udo Payer and Stefan Kraxberger. SoOM-based Lightweight Policy Verification. TERENA Networking Conference 2006, May 2006.

Glossary

AIK	Attestation Identity Key, 41
AMS	Agent Management Service, 138
AODV	Ad-hoc On-demand Distance Vector, 62
API	Application Programming Interface, 10
BFT	Byzantine Fault Tolerance, 37
DAA	Direct Anonymous Attestation, 36
DHT	Distributed Hash Table, 10
Directory Facility	DF, 138
DNS	Domain Name System, 9
DSDV	Destination Sequenced Distance Vector, 62
DSR	Dynamic Source Routing, 62
EK	Endorsement Key, 41
FIPA	Foundation of Intelligent Physical Agents, 138
HMAC	Hash-based Message Authentication Code, 5
IMTP	Internal Message Transport Protocol, 139
IP	Internet Protocol, 11
JAAS	Java Authentication and Authorization Service, 146
JADE	Java Agent Development Environment, 136
JavaME	Java Micro Edition, 97
JavaSE	Java Standard Edition, 97
JCE	Java Cryptograph Extension, 100
JDK	Java Development Kit, 89
JXTA	Juxtapose, Java P2P protocol, 14
MAC	Message Authentication Code, 73
MAS	Multi Agent Systems, 135
MD5	Message Digest Algorithm, 11
MTM	Mobile Trusted Module, 20

NAT	Network Address Translation, 3
OLSR	Optimized Link State Routing, 62
P2P	Peer-to-peer, 9
PCR	Platform Configuration Register, 41
PDA	Personal Digital Assistant, 20
PE	Platform Endorsement, 45
PKI	Public Key Infrastructure, 22
PrivacyCA	Privacy preserving Certificate Authority, 36
RFID	Radio Frequency Identification, 2
RMI	Remote Method Invocation, 136
RTT	Round trip time, 13
SDM	Secure Docking Module, 100
SePP	Secure P2P, 82
SHA-1	Secure Hash Algorithm, 11
SSL	Secure Socket Layer, 136
Superpeer	Peer on a higher level in the hierarchy, 11
Sybil	Multiple illegal nodes created by the same user or device, 4
TCG	Trusted Computing Group, 6
TCP	Transmission Control Protocol, 86
TCS	Trusted Core Services, 41
TESLA	Timed Efficient Stream Loss-tolerant Authentication, 13
TORA	Temporally Ordered Routing Algorithm, 62
TPM	Trusted Platform Module, 6
TSS	TCG Software Stack, 41
TTL	Time To Live, 73
UDP	User Datagram Protocol, 86
USB	Universal Serial Bus, 36
X.509	Standard for a PKI for single-sign on, 44
XML	eXtended Markup Language, 97
ZRP	Zone Routing Protocol, 62

Notation

We use the following notation to describe security protocols and cryptographic operations in all the following chapters. These are the most commonly used notations. If there are some additional notations required which are more specific to the described issues in the chapter itself we will provide an additional notation section in the specific chapter.

Symbol	Description
$A,\ B$	Principals, such as communicating nodes
S_K	A symmetric key
P_A, P_B	Public (endorsement) keys of principals $A,\ B$
Pr_A, Pr_B	Private (endorsement) keys of principals $A,\ B$
$H(M)$	Cryptographic hash value of message M
$MAC_{S_K}(M)$	Message authentication code (MAC) of message M using the key K
$SIG_{Pr_A}(M)$	Digital signature of message M using the private key Pr_A of principal A
$ENC_K(M)$	Encryption of message M using the key K
$DEC_{S_K}(M)$	Decryption of message M using the key K

For convenience we assume MAC and signature functions that take a variable number of arguments, simply by concatenating them in computing the function.

1 Introduction

Experience without theory is
blind, but theory without
experience is mere intellectual
play

Immanuel Kant

In this thesis we were curious to find out if providing security, trust and connectivity for self-organizing systems, especially for overlay networks, can be solved using state-of-the-art means from the fields of cryptography, trusted computing, network security and distributed computing. In addition we wanted to investigate how existing and new mechanisms and protocols can be integrated into a comprehensive framework which enables secure overlay networking in a simple and intuitive manner. This search was more than simply trying to find some gaps in the current research and fill it. Our approach was guided by the rather philosophical question:

"How should a generic open self-organizing system look like where entities could participate without having to worry that other entities are able to sabotage the system itself, impersonate or create multiple identities, steal sensitive information and threaten their privacy?"

One reason for asking this question in the first place was on the one hand the massive increase in available mass communication technologies such as peer-to-peer clients, grid computing, social networks and content distribution networks. On the other hand we felt that with the desire of a growing number of individuals to express themselves and to distribute their opinions globally, the ability to do that in a secure, trusted and privacy preserving manner would be of paramount importance. Thus, with this question in mind, we started to investigate the problems and available solutions in the broad field of security and self-organizing systems.

1.1 Self-Organizing Systems

The kind of systems which fall into the category of self-organizing systems is rather diverse. From RFID tag swarms over wireless sensor networks to all kinds of overlay networks such as peer-to-peer, grid computing, social networks and mobile ad-hoc networks, all can be considered self-organizing. The issue of how to provide connectivity in self-organizing systems has been studied intensively already. However, the overall system design and the heterogeneity of the involved nodes are of paramount significance and have a tremendous impact on the complexity of any adequate solution. Thus, even if communication characteristics for distinct self-organizing systems have been analyzed already, we also performed experiments in order to find suitable protocols and parameters for a comprehensive solution which takes security into account. Security and trust on the other hand have been studied usually only in a narrow and incomprehensive manner, leaving aside the aspects of integration and compatibility of these parts in a system as a whole.

An overlay network, as outlined in Lua et al. [LCP$^+$05], is a self-organizing network based on the idea of sharing resources among all nodes. The term overlay relates to the fact that they introduce a new communication layer on top of the existing IP infrastructure. Overlays usually provide various functions such as efficient search and distribution of data objects, redundant storage of data objects in the system, selection of neighbor peers for performance reasons, stable large-scale routing architectures, inherent scalability, unique, abstract and hierarchical identifier provision, basic authentication, trust and privacy protection, as well as fault tolerance. In pure overlay systems each node acts as a server and a client. All nodes are equal. No centralized administration is required since the necessary management tasks are performed by every node locally using distributed algorithms. An overlay network can be categorized as *unstructured* or *structured.*

Unstructured overlay networks provide means for node communication without enforcing any specific node organization. The overlay topology is organized in random graphs in flat or hierarchical manners (e.g. Super-Peer layer). They usually provide data discovery mechanisms by means of flooding, with or without limits, or random walks. Conversely, structured overlay networks have been developed to improve the performance of the data discovery process in overlay networks. Structured overlay networks organize nodes and data in an overlay using distributed algorithms, such as distributed hash tables (DHT), which impose constraints on node topology and data placement. Structured overlay networks enable data discovery with an upper bound of $O(logN)$ hops and $O(logN)$ graph neighbors for each node for N overall nodes in the network. These networks have shown to perform better for data localization in homogeneous Internet-scale systems. For instance, it is not only guaranteed that data is found if it exists in the system, but also that the search scales well with the number of peers in the system [LCP$^+$05]. Thus, structured overlay networks are known for their efficiency at locating rare data items in systems with a high number of nodes. However, recent research shows [PSZ08] that also unstructured overlay networks perform very well at locating rare data items in large-scale systems if some improvements are applied.

It has been shown that the constraints imposed on the node topology and data placement cause a high amount of maintenance traffic and also that the routing efficiency decreases significantly [RGRK04] in structured overlays. However, recent work in this area has improved the efficiency of structured overlay networks in order to cope with churn [CCR04b, CCR05] almost as efficient as unstructured networks. Also structured P2P system are believed to be less suitable for heterogeneous networks where nodes have different capabilities in terms of computation power, bandwidth, storage and energy consumption. Previous research was done to increase the ability of structured overlay networks to cope with such requirements [CCR04c, CCR04a]. However, their ability to efficiently implement keyword search[1] as well as their ability to handle unreliable peers has not been proven or tested [LCP$^+$05]. Thus, the current competition between structured versus unstructured overlays will go on and as stated in [LCP$^+$05], it depends on the application and its required functionalities which overlay is best suited.

Despite this controversy and a myriad of research in overlay networks, the issue of security has not been addressed adequately. Research on how peers can interact securely or how the whole distributed system itself can be secured is in its infancy. But even worse, most of the *secure* approaches apply concepts from the client-server model to the self-organizing networks world which is obviously not adequate. This means that at this point no overall security concepts and development frameworks for overlay networks exist which address the relevant topics in an self-organizing manner, thus leaving them vulnerable in real-life scenarios. The results of [SM02, Wal02, CDG$^+$02] have outlined security problems in overlay networks and provided some initial solutions especially for structured overlays. In [Bel01], Bellovin has shown that additional security problems exist in unstructured overlay networks. However, the work of Bellovin [Bel01] also shows that unstructured overlay networks can increase the ability to provide connectivity in the presence of Network Address Translation (NAT) and Firewalls and are thus ideal to be used as general overlays without being restricted to any specific use case such as content distribution.

Since self-organizing systems are most commonly implemented as overlay networks, although native ones such as mobile ad-hoc networks or wireless sensor networks exist, we focused our research or more precisely the research described

[1] Is the ability to search for arbitrary string patterns for instance of the name of the stored object rather than requiring to exactly know the hash key of a specific object in advance.

in this work on overlay networks. For a introduction into the different overlay topologies which are available and an outline of the benefits and caveats see Section 2.1.

1.2 Security in Self-Organizing Systems

We believe that the fundamental guidelines for creating secure structured overlays, as partly outlined by Castro et al. in [CDG$^+$02], are also adequate for all open, dynamic self-organizing systems. In their work they state that before any such system can be even thought of as being secure, the routing infrastructure must fulfill first the following requirements:

- Secure assignment of node identifiers,
- Secure routing table maintenance, and
- Secure message forwarding.

Without addressing those fundamental requirements all the higher layer security means are rendered meaningless. One can provide as much sophisticated cryptographic mechanisms, authentication protocols or reliability methods as wanted, but since there are unsolved problems in the core mechanisms they fail to provide the promised level of security.

The first point of these three requirements is the most important one in our opinion. Because if self-organizing systems can not guarantee the provision and verification of unique and trustworthy identities, there is no way to exert control over the node and system behavior. Without adequate creation and verification of identities it is not possible to detect and condemn malicious behavior in such systems.

Take the simple case where all entities choose their own identity in the system. First, it is intrinsically impossible to guarantee that the chosen identities are unique in the system. This may be tackled in a straight-forward manner by only allowing access to the system to entities whose chosen identifier do not exist already in the system. Now, you already have a very obvious problem: how should we treat the case that an entity leaves the system and another one immediately thereafter joins the system using the same identifier, which in large systems can occur with high probability. There are some intelligent solutions to that problem, for instance see the work of Dinger and Hartenstein [DH06] which also provided a taxonomy and review over available solutions. But even if these solutions are *intelligent* and make use of sophisticated algorithms and complex mathematic schemes, they can only be regarded as remedies for the effects of applying an inadequate solution, namely using self-chosen identifiers in the first place. The more serious problem is outlined by Douceur in his seminal paper [Dou02], about the Sybil attack and its implications, every solution to the identity problem must also prevent entities to create additional node instances in the system, even if their identities are unique, otherwise such nodes could inhibit the overlay networks functionality.

In order to enable detection of misbehavior as well as counteractions an adequate solution to the problem of providing secure, unique and Sybil-resistant identifiers in self-organizing systems must be found.

The second and the third point relate to the issue of confidentiality, authentication and integrity during the route maintenance as well as the message forwarding phase. The issue of secure routing table maintenance includes the secure discovery, management and maintenance of routes. Management refers to the organization of the routing table as well as other protocol specific parameters and with maintenance we mean the process of tearing down routes and disseminating this information throughout the system. In structured overlays, routing is usually based on distributed hash tables (DHT) and nodes only interact with their neighbors for that matter since the topology of the overlay itself is fixed. Thus, if nodes join and leave the network the routing tables must be updated and messages between neighbors are exchanged. Castro et al. introduced in their approach a second routing table which has the constraint that not any appropriate neighbor, who is sharing the required portion of the identifier space, is selected but only the nearest one. Since routing in the general context of self-organizing networks must not rely on DHTs but a variety of other techniques can be used, the establishment, forwarding and selection of routes must be protected in addition to the maintenance since in each phase important information about the network topology and state are exchanged.

Secure message forwarding requires that a specific message is delivered to the destination over the intended path and that the message itself can not be compromised and any interruption and dropping of messages should be detected. When a message is forwarded to the destination, each correct node must guarantee that it delivers the message to the next node in the route if a specific route has been selected from the source. Using hash-based message authentication codes (HMAC) or digital signatures provide authentication and integrity of the message. Encrypting the message using specific keys, which are selected depending on the level of security, can also provide confidentiality.

We must not only protect the system from outside attackers but also from malicious or misbehaving nodes within the system. Thus, different mechanisms must be combined in order to assure these requirements.

In addition, we believe that it is necessary to provide an overall security concept for self-organizing systems. Especially, the heterogenous nature of entities participating in such systems in terms of computational power, resource availability, energy efficiency, mobility, and means for threat mitigation need to be addressed by such a concept.

Since the different entities in self-organizing system can have different capabilities, providing various levels of security is key to establish a useful and secure self-organizing system.

In some scenarios or for specific entities it may only be necessary to provide protection from outsider attacks, without being able to verify the authenticity of specific entities. Thus, in such circumstances it is sufficient to be able to verify if a specific message is authenticated using a distinct key. In some cases, it may also be necessary to specifically identify the sender and all the relying nodes of a message and to guarantee the confidentiality and integrity. In addition, in some scenarios it may be required that the statements of previous sentence are guaranteed and also that the privacy of each entity is protected. Thus, it is necessary to design a versatile security concept which can account for all these requirements but still remains manageable with appropriate effort.

In our work we have addressed the three outlined requirements and provide solutions for each. Since we believe that the highlighted statements are essential for the realization of an adequate solution we have taken them into account during the design and implementation of our solutions. In order to verify and show the potential and adequateness of our solutions we have integrated them into a development framework which can be used to develop secure overlay networking applications. This was driven by our believe that solely providing solutions for particular security problems is not sufficient for realizing a secure system. First, we proposed one general and one specific security concept [KPT08, KP09b] for self-organizing overlays which allows the selection of adequate security measures based on the overall system requirements and the available resources of the participating nodes. Thereafter, we have designed and implemented a secure routing protocol for unstructured heterogeneous overlays based on this security concept. The secure routing protocol provides protection from outsider attacks and enables the detection as well as the prevention of insider attacks. We want to mention that our system was not intended for open Internet-scale systems but rather for controllable environments such as in private or public enterprises. This is mostly due to the fact that the complexity of key management and distribution in open Internet-scale systems can not be addressed adequately using existing mechanisms. However, during our research we have also found a solution which allows us to use our secure overlay networking framework in such environments with the assumption that each node is equipped with Trusted Platform Modules (TPM) specified by the Trusted Computing Group (TCG).

1.3 Outline

We first introduce the history and state-of-the-art of self-organizing systems with the focus on security and trust in Chapter 2. We will outline the most important trends in that area and provide detailed information about the seminal work done in this area. We will introduce our scalable security concept for general self-organizing systems in Chapter 4. The ideas of the security concept has been published in two papers. The first, called *General security concept for embedded P2P systems* [KPT08] has been presented at the Mobiquitous 2008 and was mainly concerned with the security concept and the specifics in embedded P2P systems. The security concept has been applied to embedded P2P systems during the EU project SMEPP which as about a secure middleware for embedded P2P systems. The second paper *Security Concept for Peer-to-Peer Systems* [KP09b] applied the security concept in a more broad manner to P2P systems in general and it was presented at the ACM/IEEE IWCMC 2009. Thereafter, we begin with an investigation into secure and trusted identities and provide a solution based on trusted computing mechanisms and resources in Chapter 5. The ideas presented in this chapter have been submitted to NSS 2011 in a paper titled *Trusted Identity Management for Overlay Networks.* In Chapter 6 we evaluate existing secure routing protocols according their applicability to our problem. We will also introduce a new scalable secure routing algorithm, based on the dynamic source routing protocol, which incorporates the previously defined security concept. Some of the content of this chapter has been published in two papers. The first was about the analysis of existing secure routing protocols and their applicability to P2P networks. The paper has been published in the IEEE conference proceedings of Secureware 2009 under the title *Secure Routing for Unstructured P2P Systems* [KP09a]. The new secure routing algorithm which is based on the security concept and especially designed for unstructured overlays has been published in the 19th International Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP 2011). The paper has been titled *Scalable Secure Routing for Heterogeneous Unstructured P2P Networks* [Kra11]. In the Chapter 7, we describe the design and implementation of our secure overlay networking framework which incorporates the secure identities and secure routing algorithm into a comprehensive solution for secure overlay communication and management. The concept and architecture has been published in the 35th Annual IEEE Conference on Local Computer Networks (LCN 2010). The paper has been titled *A scalable secure overlay framework for heterogeneous embedded systems* [Kra10]. Within this chapter we also evaluate the applicability and performance of our framework. Thereafter we use the developed secure overlay framework as basis for a well-known multi-agent system (MAS) and show how easily it can be integrated with existing applications in order to provide security for an underlying heterogeneous enviroment. This particular application has been published in the proceedings of the 5th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security (MMM-ACNS 2010). The paper has been titled *Secure Multi-Agent System for Multi-Hop Environments* [KDH10]. Some other publications where parts of the theory, concepts and applications of the work presented in this thesis have been published are in the Journal Security and Communication Networks with the title *An autonomous attestation token to secure mobile agents in disaster response* [HTK10] and the 4th International Conference on Network and System Security (NSS 2010) with the title *Securing Emergency Response Operations Using Distributed Trust Decisions* [DHK10].

For a list of my other publications which are not related with the content of this thesis see the references at the end of this thesis. These publications are concerned with intrusion detection, green networking, security and privacy in mobile computing, wireless networks and the Internet of Things.

2
Background

Science is built up of facts, as a house is built of stones; but an accumulation of facts is no more a science than a heap of stones is a house

Henri Poincare

The concept of distributed systems, such as peer-to-peer (P2P) and other overlays or ad-hoc networks, has been around for a long time. Even the *Internet* itself was conceived as a distributed system in the first place [Cro69]. It can be said that the Usenet is the first implementation of a distributed system since it copied files between computers without central control. Another important system, the Domain Name System (DNS) makes use of distributed computing principles and its request/response mechanism is very simple, efficient and scalable so that many routing protocols made use of these mechanism. However, in the late 90's, as the Internet expanded in unprecedented manner, the prevalent computing concept became the client-server model. This was because the use-case patterns of the Internet changed towards more asymmetry and centralization since it better reflected the business models of that time. The client-server model was therefore the straightforward choice at that time. With the beginning of the new century, with linear increasing amount of Internet users [Min08] and exponentially growing network traffic [Odl03], the client-server model was not able anymore to deliver the anticipated network transfer rates and to provide the desired reliability.

This was the time as the first file sharing clients appeared, which again made use of distributed computing concepts, especially overlay concepts. In big parts the trend towards overlay can in big parts be attributed to the fact that it is possible to achieve higher transfer rates and faster downloads in the presence of huge amount of users, then was and still is possible with the *traditional* client-server model. The first file sharing application has been Napster, which became very popular very fast. Shortly after, the ideas from the distributed computing domain have been applied to applications from almost every area, not only for file sharing. Also a huge amount of different approaches and variations within the domain of distributed computing have been developed. Therefore, basic mechanisms for storing data redundantly, efficient data distribution and peer organization and management have been studied. Most of the work has been done in the context of overlay networks, since they attracted the biggest amount of normal internet users and have therefore been the ideal playground to test ideas and evaluate algorithms. Subsequently we provide a short introduction into the different overlay topologies and outline their benefits and caveats.

2.1 Overlay topologies

The common denominator of overlay networks is that they form a virtual interconnection structure independent from the underlying physical communication infrastructure as shown in figure 2.1. In terms of the Internet Protocol Suite this can be seen as a new layer between the transport and the application layer. Whatever kind of underlying technologies are used, the overlay layer introduces new functionalities which can be used by all applications implemented on top of the overlay using its API. The amount of available overlay network implementations has grown tremendously since the first peer-to-peer networks have been very successful in providing high performance data distribution to users all around the world. Thus, content distribution networks, which form the "backbone" of almost all highly frequented content sharing websites such as YouTube, MySpace and similar, make use of such overlay networks.

Figure 2.1 illustrates one of the most prominent facts of overlay networks which is that real physical interconnections of hosts must not be expressed in the overlay network topology. However, in some implementations of overlay networks also parameters from the underlying physical network are taken into account in the formation of the overlay network. For instance, it may be beneficial to select the overlay neighbors from the hosts which are physically nearby in order to realize low latency between neighbors since they are most often contacted in overlay networks. Sometimes also the available bandwidth to other nodes influences the characteristics of a particular node especially in superpeer overlay networks as we will see in the following sections.

2.1.1 Distributed Hash Table (Structured) Overlays

One of the best known or at least most researched kind of overlay network is the structured overlay. Structured overlay networks impose an ordered structured on the whole network and on each node joining the network in order to optimize the search mechanisms. Most commonly, a distributed hash table (DHT) is used to create this kind of structure. Usually they are organized in a logical ring structure, as shown in figure 2.2, since it provides a very intuitive model and can easily be implemented. Some of the best known DHT overlay implementations include CAN [RFH$^+$01], Chord [SMK$^+$01], Tapestry [ZKJ01], Pastry [RD01] and Kademlia [MM02].

Such a DHT ring has a maximum number of nodes n. Each node uses the overlay network specific ID generation function to obtain its overlay ID. The ID generation function is usually a well-known secure hash function such as MD5 [Riv92] or SHA-1 [iosat02]. Depending on the overlay network each node uses specific information as input for the ID generation function such as the IP address or random numbers. Each node then tries to join the overlay by contacting existing nodes in the network and performing a specific algorithm which updates the network topology and provides the joining node with the required neighborhood and overlay information. Another common criteria of DHT based overlay networks is that data objects or object keys in general can be located with $Olog(n)$ requests. This can be explained by the fact that every data object which should be introduced to the overlay network is also hashed and then stored or linked to the node with the nearest matching overlay ID. If a node searches a specific datum it asks the neighbor whose overlay ID matches the hash value of the desired datum best. The neighbors of each node are organized in such a manner that the maximum distance between a datum and a neighbor is half the amount of overall nodes as illustrated in figure 2.2. The neighbor with the closest matching ID again performs the same process. This continues until the datum has been located. For instance in Chord, each node has also $log(n)$ neighbors which can be contacted for that reason. Thus, each datum can be found with at most $log(n)$ lookups with a high probability if it is available. One downside is that it is not so simple to search for arbitrary or partly known data since the whole search string is required to obtain the hash value.

2.1.2 Superpeer Overlay

Another often used overlay topology is the superpeer overlay. A superpeer topology is not like the structured overlay which has tight constraints on its topology but is also not completely unorganized such as the unstructured overlay. In the superpeer topology an distinct amount of peers form a hierachy over the common overlay. These superpeers are selected by parameters specific to the overlay network. For instance random selection, peers with the most amount of data available, peers with highest performance or resources such as bandwidth and computational power. The topology and the selected superpeers can change overtime since overlays are dynamic and in order to achieve the best performance the topology changes according to the current network status.

Usually the topology of the superpeer overlay network looks like as shown in figure 2.3. Common peers form together a normal overlay network based on their neighbor connections. At some point, usually when the network grows beyond some specific size, either some peers from all of these peers are selected to become superpeers based on the criteria described before or they have been chosen to be superpeers from startup. During this phase the neighbor connections of all peers are realigned with the new topology and the superpeer responsible for a specific *sub-overlay network*. Searching in these overlays is managed mostly through the superpeers. Each node who initiates a search first asks his neighbors and also sends the request to the superpeer. Each superpeer usually holds a list of the most requested references stored at the nodes connected to it. Therefore, each superpeer first asks other superpeers if the specific reference in the search request is managed by one of their connected nodes. If yes, the associated node is returned immediately. If not the superpeer forwards the search request to its *child nodes*. In such an overlay topology the performance is improved against unstructured networks but the downside is that if something goes wrong many more nodes are affected by the failure since superpeers are the bottleneck. Thus, some superpeer implementations let normal nodes connect to more than one superpeers which again decreases the performance but enhances the robustness.

2.1.3 Unstructured Overlay

The simplest topology and organization structure of overlay networks is the unstructured overlay. An unstructured overlay is usually formed without any organizational function in place. Thus, peers usually connect to others which can be reached through a local broadcast or a multicast. If for instance one node which has never taken part in the overlay before wants to join the network it usually connects to some well-known long-living nodes which often can be determined through the Internet. After the join process has finished successfully and the node has obtained the current member list of the network, the node selects nodes as it neighbor nodes. This can be done in different ways, for instance neighbors can be chosen at random among the members, nodes with the lowest latency are used or nodes with the highest bandwidth. There a lot of different overlay implementations available and almost as many neighbor selection mechanisms. Since this kind of overlay network forms itself very dynamically, the topology can change very quickly and each node has several connections to other nodes these networks are thought of being amongst the most robust networks available.

Figure 2.4 shows a possible topology as it may be formed by unstructured overlay networks . Since initialization and topology creation is more or less *chaotic* any snapshot of real-world unstructured overlay networks may look very different. Nevertheless, it covers most of the relevant features. The amount of neighbors of one node and the amount of nodes in general are usually not fixed and also usually not limited. The only upper-bound which usually exists is the diameter a search and also route requests travel or more precisely the hops one message can travel. These networks are usually based on flooding or random-walk style routing and search algorithms.

2.2 Related Work

We have investigated different secure ad-hoc routing protocols since no secure routing protocol for unstructured P2P networks existed so far and they have similar requirements. A wide range of research work such as [ZH99, PH02, Zap02, JLR04, WZH05, KGSW05] have applied standard cryptographic mechanisms to well-known routing algorithms or described theoretical *best practices* for establishing routes in a secure manner. They do not think about providing a scalable protocol which can be adjusted to the users security needs or the node requirements. They used digital signatures in a redundant and lavish manner, thus excluding devices with less computational power.

The Ariadne [HPJ05] protocol, which is based on TESLA [PCTS02], is able to provide *cheaper* asymmetric cryptography by means of delayed disclosure of symmetric keys. This allows very constrained devices to participate in the secure route discovery process. However, this protocol lacks in the specification and adequate guidance of selecting the required parameters. Additionally, in the bootstrapping and security update processes this protocol requires common asymmetric cryptography (digital signatures and public key certificates) which has not been taken into account in their performance evaluation. Also they assume use-cases which are not realistic in overlay networks. For instance, they assume that only very few peers will actually communicate with each other and thus would have to perform the more expensive cryptographic functions. All other peers are only required to forward the requests. As shown in [KP09a] the use of Ariadne in P2P networks is not practical because we can not assume that only some specific peers will exchange data. If all peers should be able to communicate with each other than each peer not only requires symmetric keys but also public-private keys and pair-wise shared secrets. Thus, in an environment with arbitrary interactions such a scheme is not applicable.

Another secure routing approach for ad-hoc routing mechanism has been provided by [GZ02, GZA02]. They provide security for the ad-hoc on-demand distance vector protocol. This protocol has several features in common with the dynamic source routing protocol, such as the separation in two phases (discovery and maintenance). The main problem is that it does not provide verifiable secure routes. This is because each peer only maintains its local routing view by only maintaining a forward and backward hop for a specific destination. Thus, it is not possible to give the source and destination control over used routes and therefore prevent insider attacks.

Others like [MGLB00, AD01, LS06] have tried to provide security with means of repudiation or some other kind of trust. But all these approaches rely on statistical methods and not on mathematical primitives. Statistical methods can provide means to increase security if cryptographic measures are already taken. But these mechanisms can not provide security if the worst case adversaries are assumed. Because they rely on some kind of historic data or on other heuristics they can be betrayed by playing along for most of the time but deviating from the protocol if it is in the interest of the adversary.

A similar structured overlay network approach with slightly different architecture which also addressed the 3 requirements of [CDG$^+$02] is the one from Wang et al. [WZH05]. Instead of using a ring topology for the P2P system they based their system on the Internet Autonomous System Topology in which the peers are organized by their physical network locality. In this approach the peers are grouped together into teams as their basic unit. In the proposed system only selected peers perform routing activities. Instead of using IP addresses in conjunction with certificates and node IDs they propose to use a network physical characteristic, called net-print. The reliance of the IDs on the round-trip time (RTT) to selected routers as a trustworthy component is disputable. The other requirements are addressed in a similar manner as in [CDG$^+$02].

The well-known P2P framework JXTA also states to be a secure P2P network [YW02]. JXTA makes use of well known mechanisms like SSL/TLS [DR06], X.509v3 certificates and other common security primitives, protocols and standards. Nevertheless, these mechanisms are used only for point-to-point encryption and for peer authentication at the application layer. There are no secure routing primitives or any special authentication mechanisms for group security available directly from the JXTA API. No general security concept exists which addresses the security requirements, as stated in [CDG$^+$02], directly in the JXTA framework. There are no explicit mechanisms to protect the P2P network against attacks from adversaries or threats from misbehaving and selfish peers. The primary goal of JXTA in terms of security is to provide cryptographic primitives (encryption, digital signatures, hashing, ...) or more precisely interfaces for such services such as JAAS to the application layer.

Another framework for secure P2P networking is GNUnet [BGH$^+$02]. GNUnet does not require a centralized service and Bennett et al. also state that they do not use any other trusted service. In their definition, security more or less equals anonymity. Their framework is focused on anonymous censorship-resistant file-sharing. They use a technique which makes it impossible, according to their description, to distinguish between messages originating from one peer from the messages that this peer simply forwards. As in all pure overlay network, each nodes acts as a router and uses encrypted connections to communicate with others. The connections between nodes are designed in a manner so that the bandwidth utilization is stable. All nodes monitor each other in order to implement a game-theoretic model which rewards nodes with higher contribution with better service. As stated, their focus was on anonymity rather than on secure identities, secure routing and trust. Especially in the area of trust and secure identities their framework is not adequate. Also the routing protocol does not use authentication and integrity protection since it would enable identification within their model and thus anonymity would not be possible anymore.

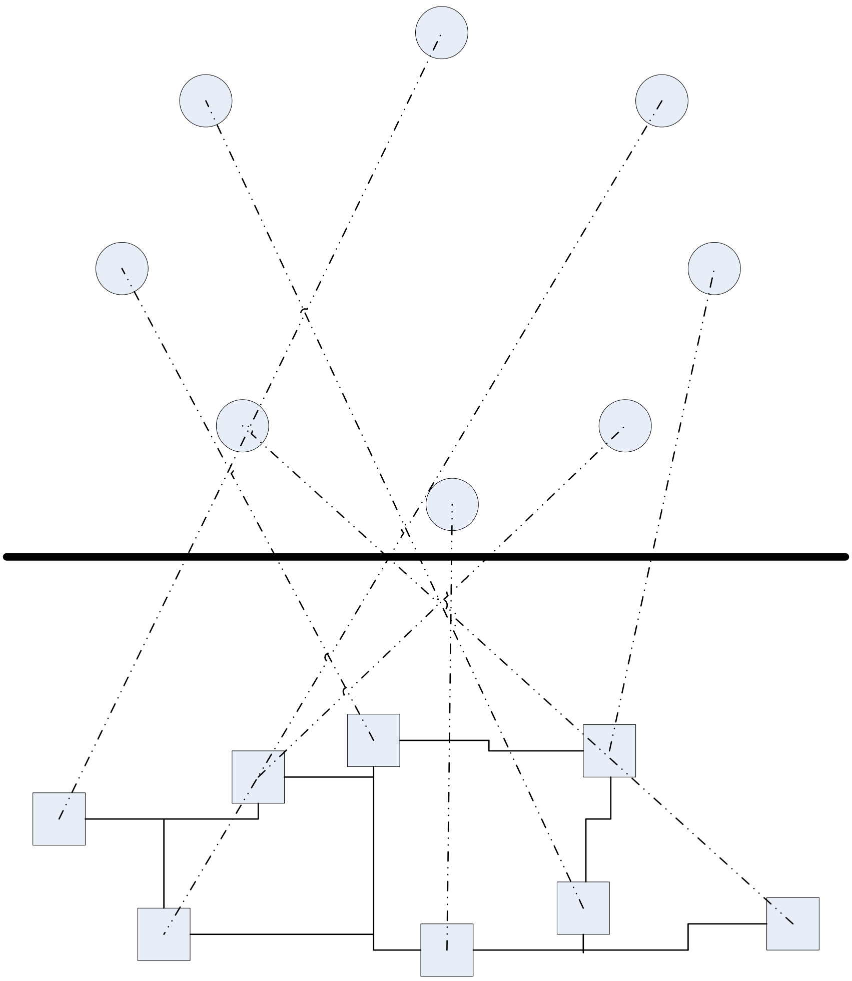


Figure 2.1: Overlay in general

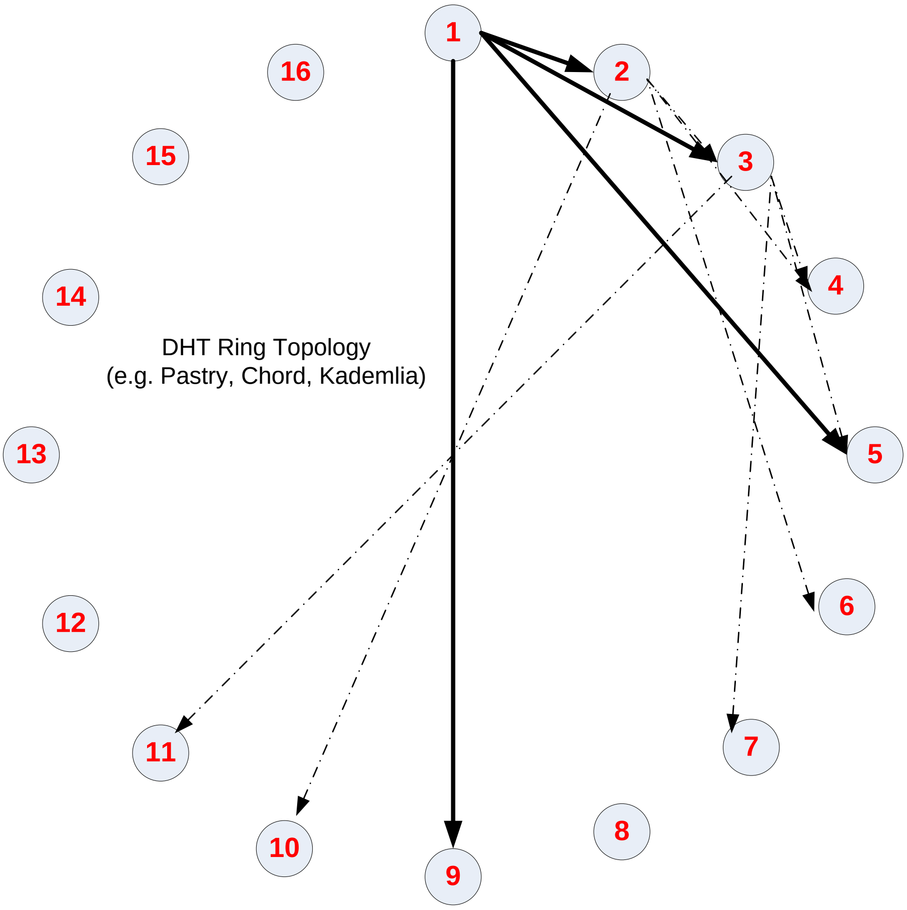


Figure 2.2: DHT based ring topology

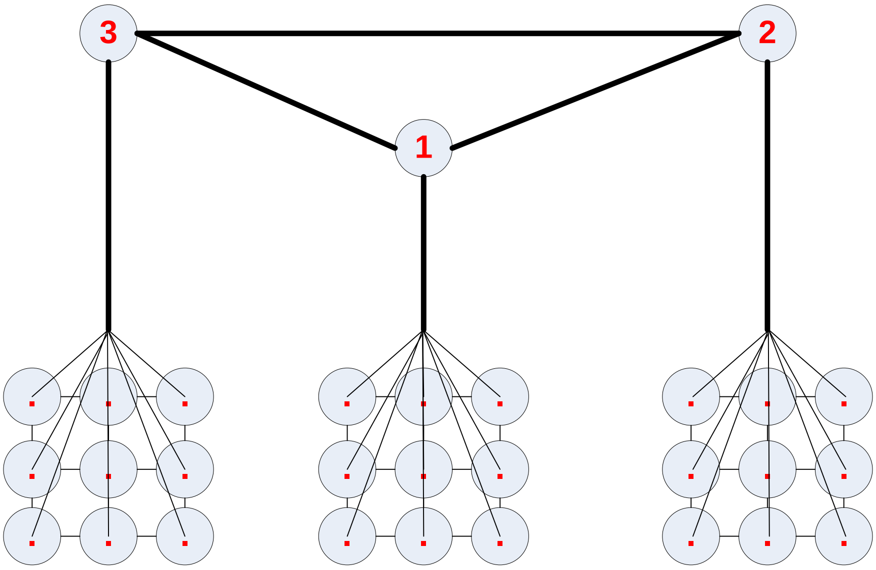


Figure 2.3: Superpeer topology

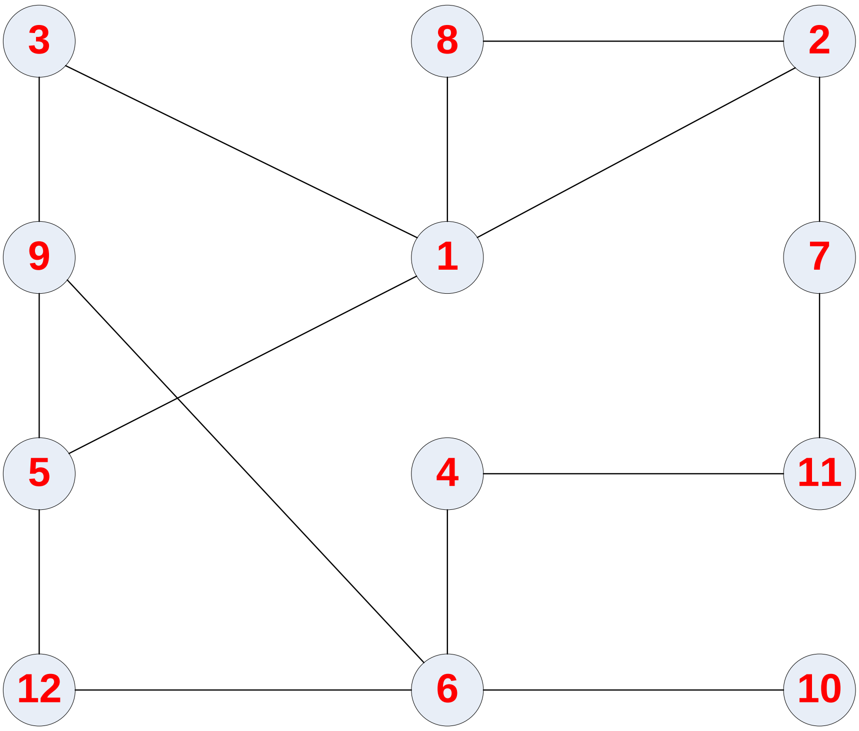


Figure 2.4: Unstructured topology

3
Assumption and Notation

The main character of any living system is openness

Ilya Prigogine

In this work we do not address the issue of data distribution or efficient content management but rather how we can provide secure and trustworthy identifiers, establish secure paths between the nodes in system, provide a scalable framework which allows nodes to communicate and manage the self-organizing system in a secure, reliable, trustworthy and privacy preserving manner. Depending on the desired application, the framework for secure unstructured overlays can be used as either multi-hop communication network or content distribution network.

3.1 Network assumptions

The following assumptions are concerned with the underlaying physical network composition and structure. These assumptions apply to all the different issues and solutions provided in this thesis and also the resulting secure overlay networking framework.

We assume that network links are bidirectional; that is, if node A is able to send messages directly to node B, then B is also able to send messages directly to A. We do this although the routing algorithms used in this thesis would also work in networks with unidirectional connections. But since before the routing algorithms are applied, we use two-way communication for establishing and verifying identities it would not be guaranteed that the nodes involved in this process are able to communicate without bidirectional links.

We also assume that nodes may belong to networks whose addresses require to use NAT to connect to other nodes in the system. Thus, the internal address may be different from the external address seen by other nodes. For instance, consider the scenario depicted in Figure 3.1. The nodes *H,I,J,K* and *L,M,N* are behind a NAT or Firewall. Thus, the secure overlay network may get separated into several parts due to the unavailability of nodes, such as *D*,*H* and *A*, which connect the separated overlays. But with the availability of such nodes the separated overlays must be able to merge into a common overlay. A mechanism must exist which allows overlay network nodes, belonging to the same logical overlay network, to find and identify each other as soon as a "bridge" node gets available. After the separate networks have verified that they actually belong to the same network, the separate networks must merge. We assume that it is best that the networks with less nodes or in the case of equal nodes from the "younger" networks start to join the "older" network. We make this assumption since in terms of security it requires several new mutual authentication requests which guarantees that only legitimate nodes can obtain the relevant overlay information although it also increases the time complexity of the overall system. Since such events will not be commonplace and for one node itself the additional time required for the renewed authentication is marginal but the overall system security benefits greatly we think it is a valid assumption.

3.2 Node assumptions

The resources of different overlay network nodes may vary greatly, from nodes with very little computational resources to powerful resource-rich nodes equivalent in functionality to high-performance workstations or servers. To make our work and results as general as possible, we have designed a security concept and a routing protocol to also support nodes with few resources, such as PDAs and mobile phones.

In addition, we assume that in order to allow trusted and secure identities to be used we require that nodes are equipped with a Trusted Platform Module as specified by the Trusted Computing Group [Tru07d]. Although current resource constrained devices such as PDA's or mobile phones are not already equipped with such a TPM, the TCG has specified and developed a concept called Mobile Trusted Module (MTM) to also provide such tamperproof hardware tokens for these devices and developments such as the ARM TrustZone technology predicts that also such devices will have such hardware in the future. Our security concept and all derived protocols and the framework function also without the existence of TPMs but with the inconvenience that keys and certificates must be distributed to the nodes in advance and a separate Public Key Infrastructure must be maintained.

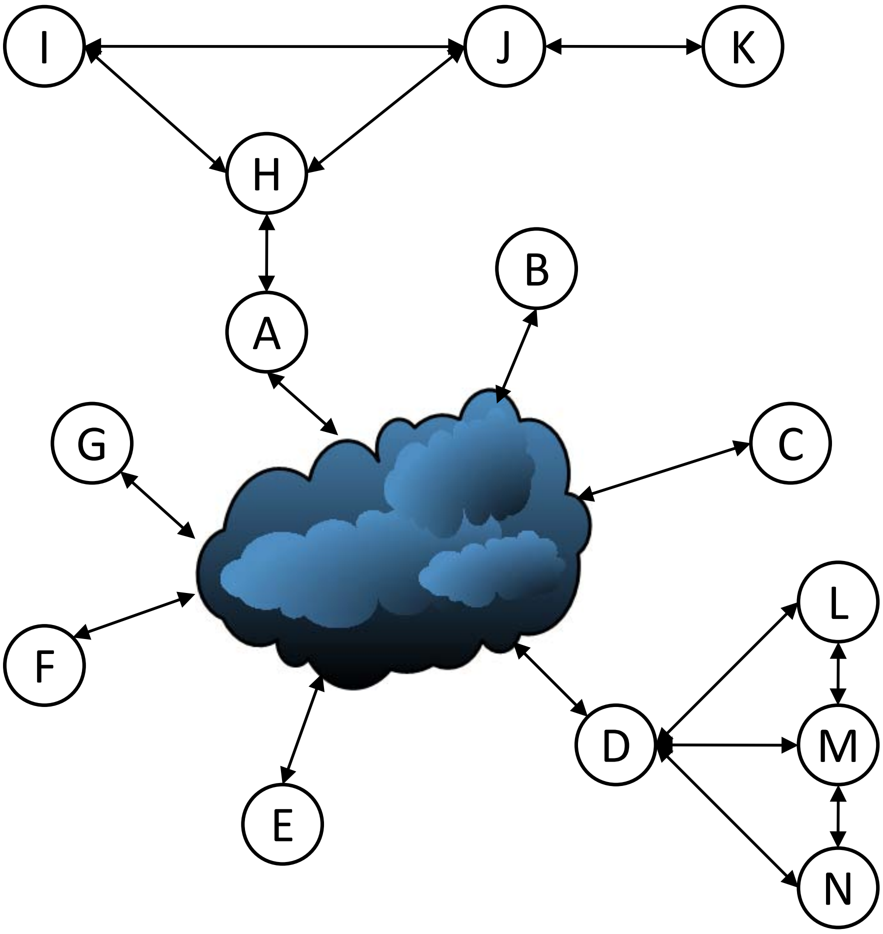


Figure 3.1: Real network topology with Internet and LAN nodes

3.3 Security Assumptions and Bootstrapping

The security of all our protocols and solutions relies on the secrecy and authenticity of keys stored in nodes. We rely on the following keys to be set up, depending on which security level is used by the node. In the case that TPMs are available the key setup and distribution process is automatically performed during the join process. If this is not the case the recommendations and process described below can be used.

- If a ***shared secret key*** is used, we assume a mechanism to set up a secret key for a network with n nodes.

- If ***public-private key pairs*** are used, we assume a mechanism to set up one authentic public-private key pair for each node. In addition, the authentic public keys of legitimate certificate authorities must also be set up for each node.

To set up shared secret keys, most key establishment protocols involve a so-called *trusted third party* or *trusted authority.* It is also possible to use distributed approaches which allow to establish keys between nodes if they already have some trust relationship between each other. If no TPMs are available we require only one shared secret key instead of pair-wise shared keys which can be efficiently handled by pre-deployed keying. Thus, we can either use a single network-wide key shared by all nodes or a set of keys randomly chosen from a key pool such that two nodes will share one key with a certain probability as outlined in [EG02, CPS03].

To set up public-private keys an offline PKI approach can be used if TPMs are not available. Thus, the private and public key as well as the trusted authority's public key are embedded in each node. The public keys of other nodes are authenticated using the trusted authority's public key. Using such a system allows us to provide the required authentication but it is not possible to handle revocation. In order to also enable revocation one can either use a distributed PKI solution such as [ADH05, LMT09] or implement a distributed revocation system [AD01, LZPL05, CCF08].

It is necessary to note that if our system is used with an offline PKI approach it is not possible to execute it as an open system as explained before. The main reason is that we do not allow arbitrary nodes to join and thereafter establish secure communication through key agreement in our system. Although it is possible that arbitrary nodes join our system, security is only provided for nodes which possess authentic credentials. These legitimate nodes can join and communicate in a secure manner establishing a virtual private overlay. Thus, only if TPMs are present (meaning that all nodes are equipped with an TPM) our secure overlay network can be used as open system. Because TPMs have built-in certified public/private keys it is possible to verify the authenticity of these keys without previously distributing keys as described before.

4
Security Concept

> It is necessary to study not only parts and processes in isolation, but also to solve the decisive problems found in organization and order unifying them, resulting from dynamic interaction of parts, and making the behavior of the parts different when studied in isolation or within the whole
>
> Ludwig von Bertalanffy

In this chapter we will focus on the specification of a security concept for heterogenous and dynamic overlay networks. We have designed and specified a new security concept for such networks because the available approaches where either only concerned with a very specific kind of overlay, i.e., structured P2P networks, or they are not intended to incorporate different kinds of devices and capabilities as well as scalability for the security concept in itself. The security concept that we envision takes the broad range of different devices into account which should be able to participate in such an overlay network also considering future developments of the Internet of Things. Therefore, an important aspect of the design is the search for and definition of security primitives and mechanisms which can also effectively be applied to systems with limited resources like mobile devices and sensor nodes (e.g. limited computational power, amount of available memory and energy, etc.).

4.1 Introduction

The design and implementation of all the protocols and mechanisms as well as the secure overlay framework itself are based on the described security concept. The security concept must be designed in a way to cope with the specifics of heterogeneous network environments such as a diverse set of devices, resources and use cases. One important aspect in the design of the security concept was configurability. Thus, giving either each node the freedom to select its desired level of security with respect to its capabilities or to allow system administrators to specify which security levels are available and what their requirements are.

Such a security concept must provide a simple way to select adequate security measures for a wide range of devices. In our case, the capability to support powerful workstations as well as constrained mobile devices is equally important since we deal with heterogeneous networks. With the security concept which is outlined in this chapter it is possible to achieve a specific security level in the presence of heterogeneous nodes with diverse capabilities. In addition we have identified mobility as a requirement of paramount importance. Since mobile peers can participate in our overlay system, the underlying communication mechanisms (especially routing) must be able to cope with a changing environment in a secure and efficient manner.

Another important characteristic of our security concept is transparency. Since nodes with different capabilities can participate in the network, the achievable *security level* of a specific communication session must be determinable in advance. Meaning, that before the actual communication process can be started all nodes participating in this session must be able to see what the capabilities of the other nodes are in terms of security and are able negotiate the security level which should be used.

First we will outline some related work concerned with security concepts in heterogeneous environments and self-organizing networks. Thereafter, we introduce how our security concept is composed theoretically. We have identified two domains where this security concept applies in our work. We then detail what particular steps are required in each domain in order to provide security adequately. Thereafter, we identify the theoretical security levels which are achievable using specific cryptographic credentials and protocols for each particular step. Thereafter, we describe the security concept we actually applied in our prototype implementation and practical work of this thesis and outline some general considerations which came up during the implementation.

4.2 Background

Zhou and Haas [ZH99] have investigated the security requirements of the upcoming area of self-organizing networks with focus on ad hoc networks in 1999. One reason to do that was that these kind of networks do not rely on any fixed infrastructure but rather use other nodes in the network to stay connected which was not common at this time. The fact that other nodes are used for connectivity is also true for overlay networks in general. They studied the effects of possible threats which ad hoc networks face and what security goals can be achieved within these networks. They have also identified several new challenges in terms of security and explored possible solutions for ad hoc networks. In addition they also outlined new opportunities and advantages of such networks due to their inherent redundancy of communication paths. They applied replication and threshold cryptography to build secure and reliable key management services within their framework. The security goals they identified where availability, confidentiality, integrity, authentication, and non-repudiation. In their work they provided an overview of security threats which ad hoc networks face and presented security objectives which need to be achieved in order to consider such a network secure. They mainly concentrated their efforts on secure routing and key management which are the most important issues in their opinion. They give some recommendations about how routing protocols should protect the exchanged information and how key management can be solved in a decentralized environment which is very useful for our approach in overlay networks.

One of the first works which introduces the notion of different levels of security in self-organizing networks, especially in mobile ad-hoc networks, is the one of Yi et al. [YNK01]. Their main focus has been to incorporate trust between and of nodes into a routing framework and how the routing decisions can be grounded on these trust representations. They proposed a protocol matrix based on specific protocol properties which provide security for routing protocols. The matrix is a list of the common properties required in secure systems such as authenticity, authorization, confidentiality, integrity, non-repudiation, ordering and timeliness. They outline all the required properties as well as common security threats such as interruption, interception, modification and fabrication, but they fail to describe how these properties can be used to realize security levels or what specific measures protocol designers must take to end up with a secure protocol. Thus, their approach provides a very theoretical high-level view on the matter containing all the required information but it does not provide details on how such an security level based routing protocol can be implemented or what specific mechanisms must be used to protect protocols from attacks. However, the ideas presented in their work can be used for directions on how to design and implement a scalable security concept for routing protocols.

More recent work on providing security in self-organizing networks has been done by Palomar et al. [PETCR06]. They have done a survey on security issues and analyzed existing solutions and gave recommendations for future research directions in the domain of overlay networks. They provide a good discourse on philosophical issues which have been emphasized and discussed heavily within the overlay network community. In the context of overlay routing they have identified some key topics. These include in the domain of *User Community* such topics as trust and social profiles, identification versus anonymity and node authentication and access control. In the domain of *Content* they have outlined issues such as replication search and retrieval, as well as content integrity and authentication. Their analysis of existing solutions is then based on these properties. Unfortunately, the resulting analysis and the representation lacks clarity and is not very helpful in the security concept designing process. Also, only very few existing solutions have been analyzed. Thus, the recommendations for future research directions are not very comprehensive and provide only very little guidance.

Fujii et al. [FRHS09] present a security analysis of overlay routing protocols with focus on providing a security model which allows to evaluate different security levels of routing protocols. They define a concept called *Regular Path* which they use to indicate the security level of the analyzed protocol. A regular path is simply the routing path which does not contain any malicious node. For their analysis they have also used the three aspects introduced by Castro [CDG+02] namely ID assignment, table maintenance and message forwarding. Their notion of security unfortunately only covers the issue of availability and does not investigate issues such as confidentiality, authentication, non-repudiation and integrity which is most important for us. However, the given results can provide basic guidelines for defining new secure routing protocols in the context of overlay networks with respect to the probability of successful route establishment for the three different types of routing mechanisms (random, chord, diversity) they have analyzed.

4.3 Concept composition

In this section we outline the theoretical foundations on which we grounded the security concept. In our studies we have identified two major areas of communication which are relevant in an overlay network. These two domains are different in terms of the used protocols, importance for the functioning of the overlay network itself, as well as in the relative amount of time communication is performed in these domains. However, both domains are similar in terms of security requirements. Thus, the security concept for heterogeneous overlay networks also features these two domains of security. We will outline the fundamental phases or steps required to provide security in both domains in a combined manner. Meaning that we will make use of the same credentials and security levels in both domains although the protocols which will make use of these credentials can be different.

- Routing security
- Group security

Both domains will be considered with the same basic *group* concept in mind. A group is simply a virtual aggregation of an arbitrary amount of peers which follow the same policies and use the same protocols. Every node must be able to communicate in a secure manner with any other node inside the same group. Depending on the security level of the group, different mechanisms and keys may be used. All nodes belong to a *default group* after they have joined the overlay network. Thereafter, peers can create or join other groups which are composed of a subset of the peers belonging to the default group. Routing security will be regarded only in terms of a *base group* or a *default group.* Every peer belongs to this default group after he has joined the overlay system. All of the routing aspects are managed inside of this group because every peer can be used to find paths to other peers.

As outline before, both domains have the same security requirements and therefore, the following security aspects apply to both domains.

1. Establishing secure communication,
2. Performing secure communication, and
3. Upholding secure communication.

Establishing secure communication relates to the secure joining of peers, where the new peer and an existing group member perform mutual authentication using the required credentials as described in section 4.3.1. When the new peer successfully joins the group, it will be provided with additional cryptographic material such as a shared secret key or pair-wise shared secrets. With this cryptographic material members of the same group can communicate securely since all of them are provided with the same keys (point 2). Usually symmetric keys are used for protecting the communication since they can be used more efficiently. However, in some security levels also asymmetric cryptography is used if a higher level of protection is required and if we also want to detect insider attacks. In the context of routing security, this means that the overlay route messages are protected with this cryptographic material. In the context of *normal* communication, all messages are protected using the specific cryptographic keys. *Upholding secure communication* relates to how to prevent and limit damage from exposed keys. For instance, we either require session keys to be updated regularly or tamper-proof hardware is used to store keys. For information about how to distribute cryptographic material in such environments see [MvOV01, LN03].

4.3.1 Security levels

The overall security in the overlay system can be set individually for all communication processes on three separate axes. These three axes conform to the three security aspects given above.

- Admission security (entity authentication and authorization)
- Data security (message authentication and confidentiality)
- Session key protection (rekeying and local side-channel attack countermeasures)

Admission security refers to *establishing secure communication*, Data security refers to *performing secure communication* and Session key protection refers to

Upholding secure communication. For each security aspect, several levels of security are possible using different kinds of cryptographic material, as stated in the following tables 4.1,4.2,4.3. We want to note here that we do not address the issue of how the credentials are distributed to the nodes. We assume that all peers posses appropriate credentials prior. In the prototype implementation of our secure overlay network we have used two specific methods. In the first one we assumed that we deal with a closed system and that all credentials will be obtained by or distributed to the nodes prior to joining the overlay network and in the case of asymmetric credentials and offline PKI signed all the certificates. In the second approach we utilized the functionalities provided by trusted computing and the TPM in order to realize a open system and also to provide privacy protection.

Level	Admission Security
4	Public-Private Keys (PPK)
3	Pairwise Shared Secret Keys (PSK)
2	Shared Secret Key (SSK)
1	Passphrase, Password, Pin, ...
0	None

Table 4.1: Admission security levels

Admission security relates to the process of *establishing secure communication.* It combines entity authentication and authorization, secure neighborhood management and secure bootstrapping. Most notably the join process for the overlay network itself and groups is addressed with this aspect. Depending on the selected security level for admission each node must already possess specific credentials. As it can be observed from table 4.1 the highest security level can be achieved if public-private keys are used. This is because all other types of keys can be established using public-private keys. Using pairwise shared secrets allows to achieve the second highest security level since each node in the system shares one pair of keys with each other node and therefore if one node is compromised only these keys must be revoked. All other nodes can still perform secure communication using their keys. With a shared secret key it is possible to achieve a higher security level than with just a password since such shared keys (e.g.: AES) are much longer than normal pass-phrases or passwords. During the admission process new credentials may be distributed to the joining node depending on the security level and whether privacy protection is used by the overlay or not. Privacy protection is only available if TPMs are available and the trusted authentication protocol with a PrivacyCA is used as described in Chapter 5. Depending on the security level selected session keys, pair-wise shared secrets or public private key pairs are provided to the joining node. If no TPMs are available and public/private keys are used for authentication the node continues to use these keys. The distributed keys are used for all processes related with data security.

Level	Data Security
8	Signature + Encryption (PPK + PSK)
7	Signature + Encryption (PPK + SSK)
6	MAC + Encryption (PSK + PSK)
5	MAC + Encryption (PSK + SSK)
4	MAC + Encryption (SSK + SSK)
3	Signature (PPK)
2	MAC (PSK)
1	MAC (SSK)
0	None

Table 4.2: Data security levels

Data security refers to the protection of the data payload (i.e. the routing information). For the communication process *symmetric keys* are established to *perform secure communication* in order to increase the performance of *data security* mechanisms. These mechanisms include the protection of routing information as well as the protection of normal communication. The benefits of symmetric keys are that they can be updated or refreshed in order to protect against side-channel attacks. For example, data security level 4 demands that each message within a group needs to be authenticated and encrypted using the current session key of the group (whether for routing or normal communication). Here again using public private keys allows for the highest security level. Thereafter, pair-wise shared keys and globally shared keys provide the other lower security levels. Since, in data security we also can provide confidentiality the highest security level is the combination of public-private keys with pair-wise shared secrets. All other combinations according to the previously outlined security ranking generate the lower security levels.

Level	Admission Security
*	Key revocation, Misbehavior detection, SCA countermeasures, ...
+	Session key refreshing
	Nothing

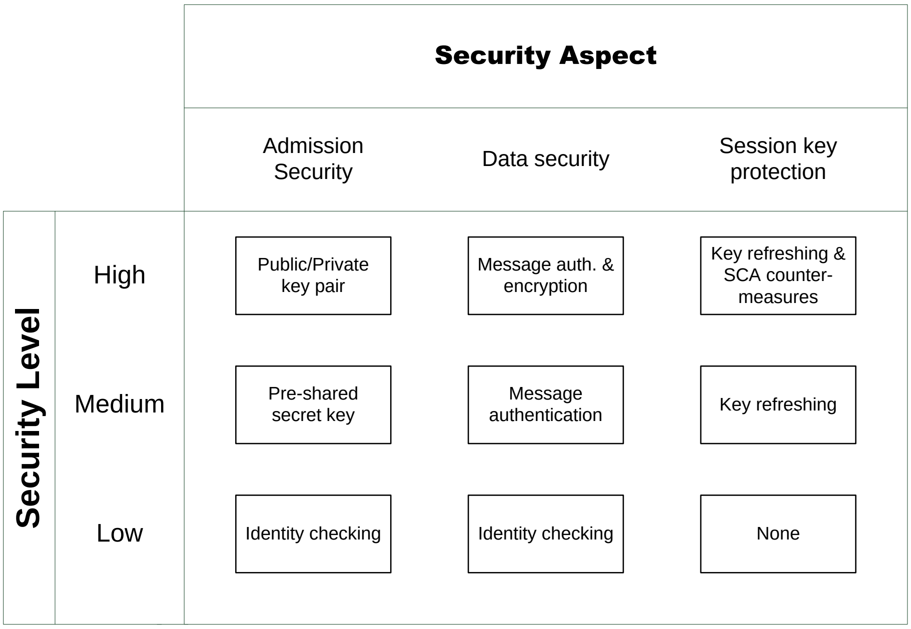
Table 4.3: Protection levels

Secret protection relates to the process of *upholding secure communication.* All mechanisms and services which prevent and limit damage from exposed keys or any other secret material belongs to this aspect. This can be achieved through means such as side-channel attack protection, session key refreshing, malicious peer detection or trust management. Session key protection refers to how the session key of a group is protected from being learned by an attacker (e.g. through means of side-channel attacks) and how to limit the damage in case that an attacker actually gets hold of the session key.

Although a lot of different security levels are possible in the individual areas, in practical applications some combinations of security levels can not be considered as principally sensible.

4.4 Applied security concept

For the implementation and verification of our framework we selected specific combinations of the previously mentioned possibilities. For demonstration and practical reasons we only use three different levels of security. These levels are *low*, *medium*, and *high*. The security level can be set independently on three separate axes. These three axes conform to the three security aspects given above. In figure 4.1 we have outlined the different aspects of our applied security concept.



Security Level	Admission Security	Data security	Session key protection
High	Public/Private key pair	Message auth. & encryption	Key refreshing & SCA counter-measures
Medium	Pre-shared secret key	Message authentication	Key refreshing
Low	Identity checking	Identity checking	None

Figure 4.1: Levels of security in the group concept

Security level *high* requires each peer to posses legitimate public and private keys and applies admission security level *four* (see table 4.1 and data security level *five* (see table 4.2). In security level *medium* each peer must posses the same shared secret key and admission security level *two* and data security level *one* are enforced. The *low* security level only requires that the identity of the peer is unique. Thus in the lowest security level all peers, whose identity does not exist in the system already, can join the network. For admission and data security level *zero* are applied. In the described research we have used session key refreshing in the security levels *medium* and *high*. These are the security levels we used in our prototype implementation but it is also possible to create other overall security levels from the possible combinations outlined in the tables.

4.4.1 Admission security

Credentials for admission security regarding routing and group communication fall into one of the following two basic categories for the security concept we applied in our overlay network:

- Pre-shared secret key (admission security level medium)
- Individual public/private key pair for each peer either pre-distributed or through TPMs. Authenticity of the public key is established by certification with a global public/private key (admission security level high)

For joining the overlay system and for securing each single group, each peer must at least possess one type of credential to be admitted into the overlay if not security level low is selected. Pre-shared secret keys are simpler to set up and can be applied more efficiently to protect messages, but they also bear a higher potential danger to the overall security in the case of their exposure. Individual public/private key pairs require more complex cryptographic primitives and protocols, but they limit the damage of key exposure and can even allow for the exclusion of misbehaving peers from groups and/or the overlay system.

The credentials are usually provided in a static manner to the overlay application in case TPMs are not available. The list of authorized members for a particular group are therefore by default fixed with the following exceptions:

- New devices supplied with the appropriate credentials can enter the overlay network
- A key validation service can provide dynamic authorization for groups and/or the whole overlay system
- Trusted Platform Modules are available and the keys are signed by a recognized CA

With individual public/private key pairs for group security, there are several options how peers are authorized for joining particular groups:

- A single key pair with an attribute certificate of the public key containing the list of groups which can be joined
- A separate key pair per peer for each group, where the public keys for the same group are certified with a different group public/private key pair

Moreover, individual public keys could be authenticated and authorized dynamically by an additional trusted service, therefore allowing for dynamic group management and the exclusion of misbehaving peers. In the authentication process for joining the overlay system or a specific group, the credentials are used in a mutual authentication protocol (e.g. three-way challenge-response) and then used to communicate the session key to the new peer. In order to join a group, a peer must have the required credential. In order to join *normal* groups, a peer must be already a member of the overlay system (i.e. in the possession of the routing credentials). The group then authorizes entry of the peer into the group or denies it. Authentication and authorization should be possible with every member of the group. The default decision for admission will be based statically on the presented credentials. An additional dynamic group authorization service is possible but currently out of the scope of our research.

4.4.2 Data security

Once a device has joined the overlay system (routing security) or a group (group security) through successful authentication and authorization (which is not the case in security level low), it will receive the corresponding session key. The session key is used to protect the data exchanged within the group (overlay routing information or group communication) via the unprotected network infrastructure. If not security level low is used the data must be at least authenticated (as originating from an authentic peer or group member). This corresponds to data security level medium. We use a simple keyed MAC for this process. If security level high is applied, the data must be encrypted with this session key and authenticated via a digital signature using the private key of the node.

4.4.3 Session key protection

Session key protection involves two strategies.

- Hardening the extraction of session keys from devices via side-channel attacks

- Limiting the value of session keys learned by an attacker

As the session keys will be the most frequently ones used, they will also be the most vulnerable against side-channel attacks. Side-channel resistant implementation of the cryptographic primitives can be included in peers which are expected to be subject to side-channel attacks. Periodic refreshing of the session key decreases the chance of successful side-channel attacks and can be used to limit the damage of exposed session keys.

At session key protection level low, the session key is never refreshed. At level medium, the session key is refreshed at certain intervals. In session key protection level high we additionally envisioned the use of side-channel resistant implementations of the cryptographic primitives on the respective peer. This is only possible if such a cryptographic coprocessor is available. In the EU-project SMEPP we have implemented such a side-channel resistant coprocessor and we where thus able to achieve this security level. Usually such an coprocessor will not be available and therefore other mechanisms such as issuing new public/private key pairs using the PrivacyCA concept can be used.

The admission credentials (pre-shared secret key or private key) should be protected by the same side-channel attack resistant implementations as the session key is when security level high is enabled.

4.4.4 Global security considerations

The basic assumption for overlay system security is that most of the authenticated peers will be well behaved. Attackers which are in the possession of some credentials for the overlay application (malicious insiders) will be considered as an exceptional case. They must be detected by the peer or network monitoring modules by identifying suspicious behavior (e.g. dropping of packets). Depending on the type of credential and the available security services, the attacker can be excluded from the overlay system (see Section Dealing with exposed credentials). Attackers without proper credentials (malicious outsiders) cannot participate in the overlay network and are restricted to attacking the raw communication infrastructure.

4.4.5 Dealing with exposed credentials

The following list contains obtainable credentials and the appropriate countermeasures to exclude an attacker which is in possession of such a credential.

Obtained credentials	Exclusion measure
Group session key	Negotiation of new group session key Group entry denied by dynamic key validation service
Routing session key	Negotiation of new routing session key overlay system entry denied by dynamic key validation service
Peer private group key	Negotiation of new group key
Peer private routing key	Negotiation of new routing key
Group pre-shared key	N/A
Routing pre-shared key	N/A
Global private key	N/A

Table 4.4: Measures against security breach

The above list contains the credentials with increasing severity of their exposure. Therefore, more critical credentials must be better protected against disclosure, e.g. by side-channel attack countermeasures. Also notable is that the disclosure of a private key can only be remedied with the help of a dynamic key validation service, which can revoke public keys at runtime. Attackers which possess authentic credentials and exhibit no apparent misbehavior cannot be detected by the overlay system.

4.4.6 Selection of security levels

For a specific overlay application, the selection of security levels can be done on several levels (globally, group-wise, or locally). As secure routing is a basic service underlying the whole overlay system, the routing security level is a global decision. Therefore, the admission security level and data security level for routing must be set globally. Furthermore it must be decided globally, whether the session key protection level for routing is set to 0 or not.

Group security levels are set group-wise. Most of the group security levels are decided by the group creator. This includes the admission security level and data security level for a group. Also the decision for a session key protection level of the group of low or medium/high is done by the group creator. In both cases, for a session key protection level unequal to low, each peer can feature individually level medium or high, as this involves only local differences of the implementation of the cryptographic primitives. The decision should be based on the degree of exposure to side-channel attacks.

4.5 Conclusion

Our work is the first step towards a general security concept for overlay network which are heterogenous in nature. Currently, no security concept is available which incorporates a heterogenous set of devices into one comprehensive solution which also allows to select different security levels for specific nodes. In doing so, we have divided the concept into two parts. A routing security concept and a group security concept. Both concepts are based on the same underlying understanding of a group. The routing concept currently is aware of one *base* group whereas the group security concept can handle an arbitrary amount of groups without breaking the security policies. We have also specified the different security levels which can be achieved using this concept, which are numerous, through the usage of different type of cryptography. Since not only the desired strength of the used cryptographic mechanisms is relevant in context of overlay network but also their resource requirements, in terms of computational power, memory and energy, one can easy determine the best combination of mechanisms and the corresponding security level by using the described security concept. Therefore, our work helps future middle-ware and applications designers and developers to select the appropriate cryptographic mechanisms and are able to specify a security level for their system.

5
Secure Identities in Dynamic Overlays

> Awareness of universals is called conceiving, and a universal of which we are aware is called a concept.
>
> Sir Bertrand A. W. Russell

In the previous sections we discussed what essential parts a solution for secure overlay networking must contain. In addition to these three essential parts we also designed a security concept which allows to incorporate heterogenous devices into a security solution. In this chapter we will outline how the first of the three essential parts, namely the provision and verification of unique identifiers, can be solved. For that reason, we first provide some motivation and background on this topic. Thereafter, we introduce the concepts, mechanisms and resources which we require in order to realize our solution.

5.1 Motivation

One particular characteristic which all overlays have in common, is that they provide abstraction from the underlying physical resources. This includes also that nodes in an overlay are addressed using an artificial identifier rather than a physical address. Using artificial identifiers has some advantages since they can be of arbitrary structure and length and enable multi-homing very easily. If identifiers are used which are derived from a physical address, multi-homing is not possible without significant overhead. Skype [BS06] for instance uses arbitrary character strings as node identifiers which are unique inside the Skype system. A message sent to one node identifier is delivered to all running programm instances of this node identifier simultaneously independent of the underlying physical address. In other systems the arbitrary identifier is used to establish a specific structure in the overlay. For instance Pastry [RD01], CAN [RFH$^+$01], and Tapestry [ZKJ01] use such an approach. However, the fundamental problem with artificial identifiers is how to create and assign unique and verifiable identifiers to each node in the system. This is essential because all mechanisms and protocols for building secure overlay networks are based on the assumption that unique and trustworthy identifiers exist. As already outlined in Chapter 1, Doucuer described the Sybil attack [Dou02] which allows the creation of an arbitrary amount of identifiers if no physical binding between the identifiers and the node exist. Thus, it must be ensured that a binding between one artificial identifier and one particular node exist which can also be verified remotely.

The most common solution for providing unique and verifiable identities currently is to use public key cryptography and a certificate authority. The certificate authority issues a public key certificate for each node identifier by signing the nodes public key. Thus, we can be assured that the node identifier is unique and by using cryptographic protocols it is possible to verify the identity and authenticity of every node. Unfortunately, the process of issuing and distributing certificates is not simple for large scale open overlay networks such as content distribution systems and grid computing. Also having a central point of failure, as introduced through the dependency on a certificate authority, is often not beneficial in overlay networks, since one of the main goals is to provide service reliability and robustness. However, there are concepts available how to make such a certificate authority reliable and also how they can be realized in a distributed manner. Thus, the issue of issuing and distribution is the problem which has to be solved.

To ensure non-repudiable identities in overlay networks in general and peer-to-peer networks in particular, a hardware identity token certified by the manufacture's own infrastructure can be considered a robust solution. Unfortunately, common off-the-shelf computer architectures do not provide such services in a customizable way. While smart cards or USB tokens may be a compelling solution in theory, the secure distribution of them is rather expensive and cumbersome, leading to large administrative overhead.

On the other hand, the TCG-specified TPM has extensive support for identity and anonymity services, such as certified Endorsement Keys, PrivacyCAs and Dynamic Anonymous Authentication (DAA). Hundreds of millions of desktop and server PCs ship with TPMs. Curiously, those are seldom employed in practice, indeed required to be shipped in a "disabled" state, and so most users do not make use of their unique identity.

In this work we provide a solution for deriving and providing unique and verifiable identifiers for large-scale overlay networks using concepts, mechanisms and resources provided by Trusted Computing. First we provide a fully distributed solution solely based on the TPMs Endorsement Key (EK) which allows mutual authentication and verification of all nodes in the system. However, since the TCG has expressed privacy concerns if the EK certificate is used directly it introduced the PrivacyCA concept in order to provide anonymous credentials. Thus, in addition to the first solution we propose a solution which is based on the PrivacyCA concept although it somewhat interferes with the concept of distributed system. However, it is possible to either provide a distributed PrivacyCA or to make a centralized PrivacyCA reliable and robust in order to not greatly affect the distributed systems concept. In our solution we propose a modification of the PrivacyCA concept to provide unique identifiers protected by the tamper resistant TPM to denominate the peers in our network. We use the standard interfaces provided by the TPM and the TCG Software Stack as well as libraries developed by the Trusted Computing Group of the IAIK. We implement a trusted authentication protocol based on these solutions which provides unique and verifiable identities and mutual authentication for joining an overlay network. Note that the protocol could also be implemented on customized smart cards, but with additional costs and the need to establish a new trusted authority.

In the following section we will give a short overview over the application scenario and a motivation. Section 5.3 describes some work already done on this topic. This is followed by a short introduction to Trusted Computing and the Privacy CA [PTHD09]. In Section 5.7 we outline our distributed solution based on the Endorsement Key Certificate. In Section 5.8 we provide an additional approach using either a robust stand-alone or distributed Privacy CA in order to address possible privacy concerns. The paper is concluded with a security analysis, a performance evaluation and a discussion.

5.2 Background

In our work we consider a *dynamic heterogeneous overlay network* with N participating nodes. The term *dynamic* refers to the fact that the number of nodes currently joined to the overlay network varies a great deal over time.

5.2.1 Current Problems and Attacks

A multitude of different overlay networks have been developed over the last decade. Issues such as selfishness, reliability and security have been of interest in recent research work. This is related to the fact that it has been observed that in open overlay networks not all users or nodes are providing equal amounts of resources to the system or actively disturb the system's execution. The problems in overlay networks in terms of security and trust can be separated into two domains: *system intrinsic* and *application specific.*

Several solutions have been proposed which address the problems of selfishness, reliability and security in the **application domain** of overlays in the presence of BAR nodes. Among them the most work has been published in the context of establishing trust between nodes on the basis of reputation systems [AD01, KSGM03, JIB07, HZNR09]. Using reputation allows P2P file-sharing applications to tackle selfish behavior of individual nodes by applying statistical obtained metrics on their file-sharing habits. Also others proposed that mechanisms developed in the context of Byzantine Fault Tolerant (BFT) can be used to provide reliability and mitigate selfishness and malicious behavior [CDG$^+$02, AAC$^+$05, BRDP10]. However, although several solutions for particular problems have been proposed, a concise solution which not only focus on **application specific** problems is needed as explained subsequently.

Castro et al. [CDG$^+$02] identified three major **system intrinsic** requirements, which are essential to ensure security and reliability of the overlay and to provide a solid basis to build application specific solutions. These three requirements are:

1. secure identifier assignment mechanism,
2. secure routing table maintenance, and
3. secure message forwarding.

If these requirements are not addressed, any overlay is vulnerable to insider attacks such as route fabrication and disruption, message interception and corruption, Sybil [Dou02] and Eclipse attack [CDG$^+$02] performed by Byzantine or rational nodes. This is because all the application specific solutions assume that all nodes in the system have unique identifiers which can be verified remotely from all other nodes. Thus, the first point in the requirements list is the most important one, since all application specific as well as the other system intrinsic requirements are based on secure identifiers.

Currently, the most common approach which is applied in real-world applications and which guarantees unique and verifiable identifiers is to use certified identifiers issued by a trusted third party such as a well-known certificate authority. We will discuss the proposed solutions based on certified identifiers and trusted third parties in more detail in Section 5.3. However, although this solution enables unique and verifiable identities it also has the following obvious weaknesses.

- Low acceptability by the users due to difficult registration and creation process of the node identifiers
- Complex distribution and status update of the identifier certificates
- Centralized solution which introduces a single point of failure
- Additional costs through certificate maintenance by the CA

The most important restriction or weakness relates to the complex creation, distribution and management of public key certificates which are used to bind identifiers to a specific public-private key pair. This must be attributed to the fact that the identification process is only shifted from the overlay network to the trusted third party. In order to obtain a certificate the user has to provide proof of his identity using traditional exogenous processes such as showing a passport to a registration officer. Such a organizational effort is a very limiting barrier for applying certified identifiers on a large scale.

On the other hand Trusted Computing mechanisms and hardware are already available in common state-of-the art desktop PCs and notebooks. Thus, we propose in our work to use the information and keying material enclosed in the Trusted Computing's TPM. The TPM, which is deployed in an increasing amount on current motherboards, contains a cryptographic key as well as identity assuring certificate.

5.3 Related Work

Schechter et al. [SGS03] provide an overview of content theft and how it has increased with the introduction of overlay networks. They provide an overview of the characteristics of Trusted Computing as well as problems of overlay networks such as ensuring integrity, confidentiality and availability. Although they provide no concrete mechanisms or methods how overlay networks can be secured and trusted, they conclude that if Trusted Computing holds its promises such theft can be limited or mitigated even in overlay networks.

In [GRB03] Garfinkel et al. describe an intuitive model for understanding the capabilities and limitations of the mechanisms provided by Trusted Computing. They describe a flexible OS architecture to support Trusted Computing. In addition, they presented a range of practical applications that illustrate how Trusted Computing can be used to improve security and robustness in distributed systems. They also identified that Trusted Computing can be used to provide secure identities which can be used to prevent identity theft and Sybil attacks in overlay network. Unfortunately, this paper also only provides a general overview without giving any methods or mechanisms which can be used to secure overlay network.

Balfe et al. [BLP05] show how the functionalities provided by Trusted Computing can be used to establish a pseudonymous authentication scheme for peers and extend this scheme to build secure channels between peers for future communications. They propose the use of the Direct Anonymous Attestation mechanism, which has been adopted by the *Trusted Computing Group (TCG)* in the TPM specification [Tru07d], to create pseudonyms for each peer and use them for authentication. Their solution is more concerned with privacy and anonymity than with preventing Sybil and eclipse attacks.

Bazzi and Konjevod [BK05] propose a system to counter multiple identities of one physical node by facilitating the nodes physical location. Within their work it is possible to provide and verify identities for overlay networks even in the presence of malicious nodes. They assume that two nodes do not occupy the same physical location and that the two-way message propagation delay has a specific upper bound. Their approach relies on the concept of beacon nodes which report distances to normal nodes. These distance values are then used to issue geometric certificates. A geometric certificate consists of a set of distance values between different beacons and a node. These values are signed by the beacons and the node. Their approach provides a very elegant solution to the Sybil problem. However, the involved measurements and required message exchanges as well as message analysis requirements are very high.

In [GNC$^+$06] Gu et al. proposed a mechanism, called random visitor scheme, to defend against identity attacks such as Sybil and eclipse. They try to solve the identifier authentication problem in overlay networks using identity-based cryptography and identity ownership proof. A randomly chosen delegate carries some credential that can prove sincerity, ownership, and moderate exclusiveness of a node. They only provide a preliminary discussion relating cost and strength of their mechanism.

[DH06] et al. propose a system to tackle the issue of Sybil attacks which is based on two premises. First, they classify the overlay identifier assignment process and separate network participants from network nodes. Thus, two challenges in the context of Sybil attacks are identified. First stability over time, and second identity differentiation. Thereafter, they propose an identity registration procedure called self-registration that makes use of the inherent distribution mechanisms of a overlay network. Using this self-registration mechanism they assume that it might be possible to effectively regulate the number of nodes per participant. But they also clearly state that their approach is not Sybil-proof, but that it might be Sybil resistant. They also concluded that still several questions remain unsolved within their solution such as how long the time period lasts in which one can safely assume the network to be dominance-free and the also the *Sybil resistance level* of their improved distribution and verification mechanisms for the self-registration is unknown.

Lesueur et al. [LMVTT08, LMT09] describe solutions to establish and maintain distributed certification schemes in structured overlay networks. Since their original solution is still susceptible to the Sybil attack they proposed to couple their solution with SybilGuard [YKGF06]. Instead of using SybilGuard we achieve the protection from sybil attacks through the reliance on the TPM and the fact that the endorsement key certificates are unique and not detachable from the TPM hardware. Their solution, as almost all other proposals for distributed key management, make use of *threshold cryptography* which is based on secret sharing as described by Shamir [Sha79]. With threshold cryptography it is possible to sign or cipher a message using t shares chosen among those issued to n nodes, where each node owns one distinct share. t shares are needed to sign or cipher a message, but $t-1$ shares hold no information on the shared secret key. Thus, an attacker must obtain t shares to be able to recover the full shared secret key.

Jyothi and Janakiram [JD09] propose a solution that enables all honest peers to protect themselves from Sybils with high probability in large structured overlay network. In their proposed Sybil defense system, they associate every peer with another non-Sybil peer known as SyMon. A given peer's SyMon is chosen dynamically such that the chances of both of them being Sybils are very low. The chosen SyMon is entrusted with the responsibility of moderating the transactions involving the given peer and hence makes it almost impossible for Sybils to compromise the system. In contrast to our system, where each node has only one unique identity and thus Sybils attacks are not possible, their system allows Sybils in the system but it is built on the premises that no Sybil nodes are chosen as SyMon.

5.4 Trusted Computing Background

Trusted Computing is a concept of computing which provides a solution to the problem of how different computing platforms, which have no prior relation, can establish a trust relationship. This concept has been specified by the Trusted Computing Group (TCG). It also provides means to authenticate a computing platform's software configuration in a distributed manner. The process of software configuration authentication is called attestation and is performed by measuring the configuration and mapping it to a distinct value. This measured value is signed using a unique private key which is stored and will only be processed on a special hardware component called Trusted Platform Module [Tru07c] (TPM). With attestation it is possible for a remote entity to verify the software identity of a computing platform and enforce policies according to this software configuration.

The TPM is used as hardware anchor of trust as outlined by the TCG. This hardware anchor provides several features such as cryptographic primitives, secure storage units and configuration registers similar to a smart card. But in contrast to a smart card these features are physically bound to a specific computing platform and are enclosed in a tamper-resilient casing. The cryptographic features include operations for key generation, hashing, random number generation, and public key cryptography. The TPM is also able to enforce policies on the stored cryptographic material. For instance on the Endorsement Key (EK) which provides the TPM with a unique identity.

In addition to the TPM hardware the TCG also specifies a software infrastructure which provides Trusted Service Provider (TSP) interfaces to access the Trusted Computing functionality on the TPM. The software infrastructure is called the TCG Software Stack (TSS)[Tru07b]. Within the TSS the main functionalities such as key and key cache management, command generation and synchronization are implemented as system service called the Trusted Core Services (TCS). Through a specific device driver library (TDDL) the TCS can communicate with the TPM. In order to protect the measurements taken from the platform from tampering the TPM has also a set of specific registers called the *Platform Configuration Registers* (PCRs) where they are stored. If requested the TPM performs a operation, referred to as `TPM_Quote`, which signs the state of the PCR with a signing-capable key hosted by the TPM. These signing-capable keys are called *Attestation Identity Keys* (AIKs). It is guaranteed that these Attestation Identity Keys are always protected by a TCG standard compliant TPM and that the private part is never used outside the TPM. In order to verify that the signature made with a TPM protected AIK these keys are certified by a Trusted Computing associated PKI.

Every current business PC has a built-in TPM on the motherboard. But these TPMs are not activated per default. In order to use the TPM it must be initialized and the user must take ownership of the TPM. After the TPM has been initialized the TPM can be used.

5.4.1 PrivacyCA

The TCG remote attestation architecture requires that a claimant sends a very detailed description of its system state including a signature to a verifier. With these detailed descriptions including the signature, created by using the same key, it is possible for an arbitrary verifier to track every possible platform and thus the issue of privacy protection must be taken into account. In addition the detailed description can also be used to find weaknesses and mount attacks on these systems. Thus, a process is required which makes it impossible that a distinct platform can not be identified in the case that different independent services are accessed. This, is an advantage over traditional public key infrastructure since certificates issued by common certificate authorities can be correlated over service boundaries. But in the context of trusted computing it should rather be possible for a verifier to establish that the remote platform has indeed a standard-compliant TPM. Also the verifier must be able to establish that the system state descriptions are signed with a key which belongs to such a platform but the verifier must not know the long-term identity of the platform as long as it interacts with the same identity for the duration of the service access.

Issuing AIKs

For that matter the TCG has specified the PrivacyCA service [Tru] which protects the privacy of the users or more precisely the computing platforms. The PrivacyCA issues certificates for specific attestation identity keys. These certificates ensure that a specific AIK is owned and protected by a TPM enforcing specific policies which are outlined by the TCG. One requirement is that the issued AIK certificates have no relation to the identity of the specific TPM which requested the AIK certificate. Thus, by issuing several AIKs for one distinct TPM it is possible to hide the identity of the computing platform.

Since in our case we want to protect the system from identity attacks we limit the PrivacyCA to only issue one AIK per TPM for a defined period of time. Thus, every user can obtain one AIK from the PrivacyCA at a time. The user can revoke the AIK by leaving the overlay network which triggers an event in the PrivacyCA to release the binding between the AIK and the associated EK. The node can thereafter obtain a new AIK by joining the overlay network again. If the user does not leave the network properly (e.g.: the users computer crashes) the user must wait until the AIK is revoked automatically after the specified expiration date. Thus, it is guaranteed that each host can only obtain and possess one identity in the overlay network and is not able to perform a Sybil attack.

Work-flow PrivacyCA

In Figure 5.1 a schematic overview of the AIK certification process using a Privacy in the context of remote attestation application is given. First the key generation process inside of the TPM is triggered. This process creates an RSA key pair. Thereafter, the platform performs a cryptographically secure protocol with the PrivacyCA where the AIK public key and the platforms endorsement key are exchanged. The PrivacyCA verifies the validity of the exchanged information and responds with an AIK certificate if the information complies with its policy. The response is encoded in such a ways so that only the requesting TPM can decipher the response. The trusted platform then activates the identity associated with the AIK. Thereafter, the TPM measurements will be signed with the activated identities corresponding AIK private key and the TPM is able to provide the public AIK upon request from a remote attestation verifier. The verifier is able to check if the provided signature corresponds to the public key in the certificate and can check the validity of the certificate using the PrivacyCA.

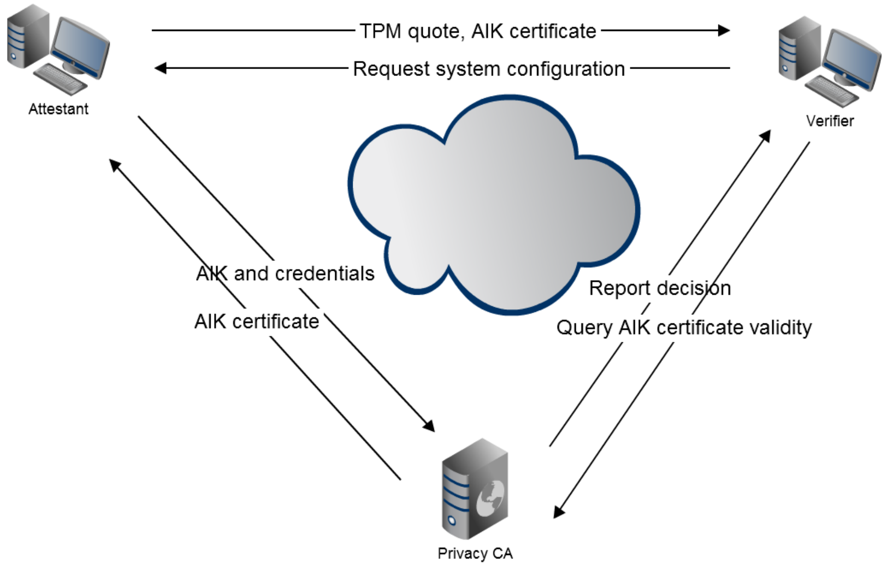

Attestant
TPM quote, AIK certificate
Request system configuration
Verifier
AIK and credentials
AIK certificate
Report decision
Query AIK certificate validity
Privacy CA

Figure 5.1: Overview of the remote attestation process including prerequisites such as AIK creation and certification.

Security and Privacy Policy

The PrivacyCA behavior is governed by the defined policy which usually specifies two properties. First, who is allowed to obtain an AIK certificate and second how much information about the certificate and the AIK request will be stored at the PrivacyCA.

For instance, consider a PrivacyCA which is only allowed to issue certificates for well-known computing platforms. In such a deployment scenario the computing platforms are required to register prior to system execution. Thus, in such an scenario it may be valid to store all the information which has been obtained by the certificate issuing process. In a scenario where the PrivacyCA can issue certificate to all the platforms even if they are not registered beforehand a more sensible policy might be the better approach in order to protect the privacy of the users. Thus, the PrivacyCA can have policies from do not log anything to record everything depending on the use case and the choice should be left to the operator.

5.5 Assumptions

In order to realize our solution all nodes participating in our overlay must fulfill the following requirements.

1. Each node must be equipped with a TCG compatible TPM.

2. Each node must be able to execute TPM specific functions.

3. Each node must be able to perform cryptographic functions such as calculating hash values, encrypting and decrypting blocks of binary data, generating random numbers and creating and verifying digital signatures.

4. In case of the PrivacyCA solution each node must posses the public key certificate of the PrivacyCA.

In addition, each node must also be able to communicate with other nodes using a reliable network protocol. We do not place restrictions on the amount of memory available, the processor power, or the hardware components the nodes are equipped with, but they must at least be sufficient to fulfill the previous requirements.

5.6 Trusted Identity Management

In the following sections we outline the TCG specified components and procedures that facilitate the realization of a trusted identity management service for overlay networks. We will discuss specific keys, certificates and protocols and their intended role. In this discussion, we only consider the elements relevant to the identity creation and management based either solely on the endorsement key or on a PrivacyCA solution. The first solution is the simpler and more elegant one, but since this solution may raise some privacy concerns we also outline an privacy-preserving alternative which is based on the PrivacyCA concept (see section 5.4.1).

Security credentials in general may be represented in different formats. The credential standard document of the TCG [Tru07a] describes credentials in the concrete instantiation of either X.509 certificates [HPFS02] or attribute certificates [FH02]. Additionally, some TPM functions produce binary blocks of data, the internal structures of which do not conform to any standard.

First, we provide some background information about the credentials and keying material which are contained in the TPM and are required for both our solutions. In the sections thereafter we describe the two solutions.

5.6.1 Endorsement Key and EK Certificate

Every TPM hosts a unique Endorsement Key (EK). The asymmetric key pair is stored in non-volatile memory inside the TPM. It is impossible to retrieve the private part of the key. A corresponding TPM Endorsement certificate contains the public part of the key pair. This certificate represents an assertion that a certain TPM conforms with the TCG specifications and that the private Endorsement Key is indeed protected by the TPM. The Endorsement certificate is signed by the entity which created and inserted the EK.

As the Endorsement Key uniquely identifies a TPM and hence a specific platform, the privacy of the platform user or users might be at risk if the EK is employed for remote attestation operations. As a consequence, the TCG rigorously restricted the set of operations that can be performed with the EK. For instance it is used during the taking of ownership of the TPM by the user. Notably it cannot be used to sign PCR values or arbitrary data. This policy is enforced by the TPM.

5.6.2 Platform Certificate

A system manufacturer vouches for the components of a platform (except the TPM) with a Platform Endorsement (PE) credential. It represents an assertion that the specific platform incorporates a properly certified TPM and the necessary support components. A PE states that the platform architecture conforms to TCG specifications. A reference to the specific TPM on the platform is included.

5.7 Endorsement Key Solution

For this specific solution we make use of the $TPM\_MakeIdentity$ and $TPM\_ActivateIdentity$ commands. These commands are usually used for creating and activating the association between a platform identity and an AIK. More generally these commands have the following functions which can be used to establish a distributed identity management and authentication solution.

1. $TPM\_ActivateIdentity$ activates the associated TPM identity which has been created through $TPM\_MakeIdentity$ and is specified through the AIK's public key hash as input parameter.

2. *TPM_ActivateIdentity* verifies the association between the AIK private key and its identity.

3. *TPM_ActivateIdentity* decrypts the input data blob and extracts the session key and verifies the connection between the public and private endorsement keys used.

Thus the *TPM_ActivateIdentity* command allows us to send arbitrary data, encrypted with the public key of the TPM's endorsement key, to the TPM which then decrypts the data and returns it to the user. This function only works when the public key used to encrypt the data corresponds to the TPM's private endorsement key. This enables the same functionality as available with a traditional PKI but additionally guarantees the binding of a specific public AIK certificate and key to a distinct computing platform.

In order to use `TPM_ActivateIdentity` we first need to create a normal RSA Attestation Identity Key (AIK) pair on the TPM. This AIK is created using the `TPM_MakeIdentity` command. Since we do not make use of the AIK as envisaged but rather require it only for enabling the `TPM_ActivateIdentity` command we omit the details about the AIK and the AIK certification process for now. The details are explained in Section 5.8 where the AIK is used as intended. Once, an AIK has been created the `TPM_ActivateIdentity` command can be used an unlimited amount of times.

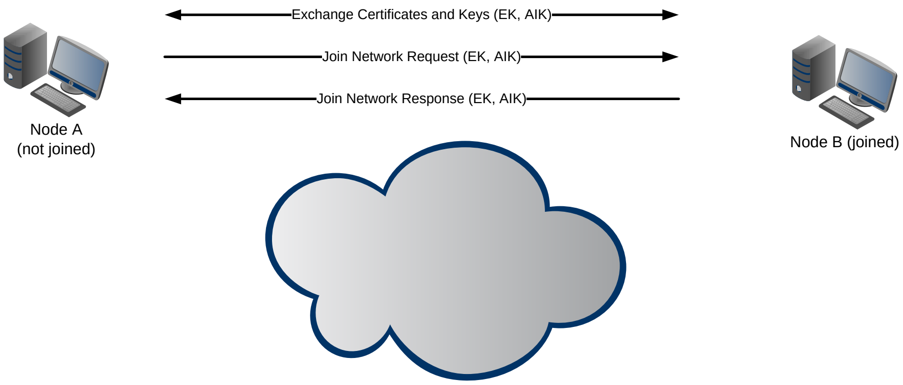

Exchange Certificates and Keys (EK, AIK)
Join Network Request (EK, AIK)
Join Network Response (EK, AIK)
Node A
(not joined)
Node B (joined)

Figure 5.2: Authentication process using Endorsement Key solution.

In Figure 5.2 an overview of the authentication process using the endorsement key is outlined. The basic requirements are that both nodes have each others endorsement key certificate in order to verify the identity of each other. Since the TPM is involved each time the endorsement key is used to encrypt or decrypt data and each node can only have one endorsement key we can guarantee that each node can only have one representation in the overlay network and that it is the one he has provided.

5.7.1 EK-based Authentication Protocol

Every TPM must be equipped with a corresponding TPM EK certificate, as outlined in the TCG specification. This certificate contains the public part of the EK pair, which serves as TPM identity. The private part, called the private Endorsement Key, is stored permanently inside the TPM and can not be retrieved once inserted. The certificate is signed by the TPM manufacturer and vows that the specific TPM conforms with the required specifications and the private Endorsement Key is kept safe by a TPM. The certificate of the TPM manufacturer, which is required to verify the authenticity of the EK certificate, can be downloaded from the manufacturer's homepage. However, to this date the only manufacturer to include a TPM EK certificate on chip is Infineon.

If a TPM is shipped without a manufacturer issued certificate, late construction of an EK certificate may be applicable in selected scenarios, e.g. a limited deployment in a department-wide setup. Since this is not a suitable approach for open Internet-scale overlays we assume that all nodes participating in the overlay network have an Infineon TPM. Other nodes can not take part in our system because self-created EK certificates can not be verified without a reliable registration process.

First we provide the basic exchange protocol description which we are using in the context of TPM backed credentials and keys. The sequence in Figure 5.3 shows the initial process of exchanging the EK public key certificates and thereafter the process of exchanging messages in an authenticated and confidential manner. In the following description of the EK-based exchange protocol we denote one party as the initiator (I) and the second party as the verifier (V). The initiator is the node which wants to be authenticated and join the overlay. The verifier is a node already inside the overlay.

1. $[I \rightarrow V] : EK_{CertI}$
2. $[V \rightarrow I] : EK_{CertV}$
    3. $[I] : Message = (\ldots, t_I, I)$
    4. $[I] : Message* = ENC_{Sym_I}(Message)$
    5. $[I] : Sym_I* = ENC_{EK_{PV}}(Sym_I)$
6. $[I \rightarrow V] : Message*, SIG_{EK_{PrI}}(Message*), Sym_I*, SIG_{EK_{PrI}}(Sym_I*)$

Figure 5.3: Basic security related exchange protocol

First the initiator sends the public key certificate of its endorsement key EK_{CertI} to the verifier. The verifier responds with its own public endorsement key certificate EK_{CertV}. Thereafter, the initiator prepares the message which is sent to the verifier containing the relevant information and encrypts the message using a symmetric key Sym_I. The initiator also encrypts the Sym_I with the public endorsement key of the verifier EK_{PV}. The initiator calculates a signature using its private endorsement key EK_{PrI} for the encrypted message as well as the encrypted symmetric key. At last the initiator sends the encrypted message, signature over the encrypted message, encrypted symmetric key and the signature over the encrypted symmetric key to the verifier. Each message which is exchanged using this protocol contains at least the identity of the sending party and a current timestamp in order to account for replay attacks. Using this protocol the verifier can be sure that the confidentiality of the transmitted content is protected as well as that the content has been created by the initiator. Now that we have defined and outlined the basic authentication and confidential exchange protocol we can provide the description for the authentication and join process in context of the overlay using EKs. In the following description of the EK-based identification (authentication) protocol we denote the joining or claiming entity as initiator (I) and the joined or verifying entity as verifier (V).

$$\begin{aligned}
1.\ [I] &: AuthentRequest = (I, V, EK_{CertI}, r_I, t_I) \\
2.\ [I \rightarrow V] &: AuthentRequestS*, SymS_I* \\
3.\ [V] &: AuthentResponse = (V, I, t_V, r_I, r_V) \\
4.\ [V \rightarrow I] &: AuthentResponseS*, SymS_V* \\
5.\ [I] &: AuthorizRequest = (I, V, r_V, t_I) \\
6.\ [I \rightarrow V] &: AuthorizRequestS* \\
7.\ [V] &: AuthorizResponse = (V, I, t_V, result) \\
8.\ [V \rightarrow I] &: AuthorizResponseS*
\end{aligned}$$

Figure 5.4: EK-based join/authentication protocol

In Figure 5.4 the steps of the join/authentication protocol, which are based on the Needham-Schroeder-Lowe protocol [NS78, Low96], are outlined. The outlined protocol is somewhat modified to the original version, since in our case larger amounts of data are transmitted. Thus, we make use of symmetric cryptography to increase the performance of the overall system. In this version the steps outlined in the basic exchange protocol 5.3 are omitted but are implicitly marked through the $S*$ at the ending of the request or response. For instance, *AuthentRequest* is the created message and *AuthentRequestS∗* indicates that the authentication request is encrypted and signed. $SymS_I*$ indicates that the symmetric key used to encrypt the authentication request is also signed and encrypted as outlined in the basic exchange protocol.

The initiator first creates an authentication request which contains its own identity. The identity of the verifier, its own public key certificate of the endorsement key, a fresh secure random number and a timestamp. The authentication request is encrypted with a newly created symmetric key using AES [DR02]. The symmetric key itself is encrypted with the public endorsement key of the verifier using RSA [RSA78]. The encrypted and signed authentication request and symmetric key are thereafter sent to the verifier. The verifier uses the public key of the initiator to verify the authenticity and integrity of the authentication request and the symmetric key. If the signature is valid and the timestamp t_I is fresh enough the request is accepted. The private endorsement key is then used to decrypt the symmetric key and the symmetric key is used to decrypt the authentication request. Now the verifier creates an authentication response which contains in addition to the identities and the initiator created secure random number also a timestamp and another secure random number generated by the verifier. This authentication response is treated the same way as the request and then sent to the initiator. The initiator verifies the security of the response also in the same manner as the verifier. In addition it checks if the secure random number r_I is the same as the one originally sent and the timestamp t_V is fresh enough. The initiator now creates an authorization request using the received secure random number r_V and a new timestamp t_I in addition to the identifiers. The request is again encrypted and signed and sent to the verifier. The verifier checks validity of the request and the correctness of the secure random number r_V. It then creates an authorization response containing a new timestamp and the result of the authorization process. After this step the identities of both the initiator and verifier are successfully verified and the both nodes are mutually authenticated.

If all these checks succeed the verifier can be assured that the initiator possesses the private endorsement key which matches the specified identity and the that none of the common attacks such as man-in-the-middle or replay have been applied. In addition to that the verifier can also be sure that no identity attacks such as Sybil or Eclipse have been performed. Since the used identity has only a one to one relation to the keys used for authentication it can be to assure that only legitimate nodes with only one identity exist in an overlay network if this solution is used for authentication and joining. The only thing which must be considered is that each node gives away its public endorsement key certificate. Thus, it is possible to track user behavior if the endorsement key certificate is also used by other services in a similar manner. For instance, if other services require nodes to provide them with the certificate they would be traceable and their privacy maybe compromised. How important this issue is can currently not be verified since no real applications of trusted computing are available. But since the endorsement keys are long-term keys this issue could be exploited. Therefore, we also provide a solution that solves this problem by incorporating the PrivacyCA concept, which was originally intended by the TCG for that matter.

5.8 Privacy CA Solution

As an alternative to the unique and privacy sensitive EK, the TCG introduced Attestation Identity Keys (AIKs) and the associated AIK certificates. AIK certificates do not contain any information that links the certificates to the specific platform hosting the AIKs. The AIK certificates assure that the identity keys are indeed TPM hosted. Thus, using this solution would enable us to provide trusted identities without compromising the privacy. Unfortunately, the original intention of the PrivacyCA was to provide an unrestricted number of AIKs for a particular endorsement certificate or TPM. Thus, it would again be possible to mount identity attacks such as the Sybil or Eclipse attack.

Therefore, in our PrivacyCA solution we must ensure that one particular TPM or endorsement key certificate is issued one AIK key-pair at a time. This can easily be solved at the organizational level by modifying the original PrivacyCA configuration. Since in our case we want to protect the system from identity attacks we limit the PrivacyCA to only issue one AIK per TPM for a defined period of time. Thus, every platform can obtain one AIK from the PrivacyCA at a time. The user can revoke the AIK by leaving the overlay network and can thereafter obtain a new AIK by joining the overlay network again. If the user does not leave the network properly (e.g.: the user's computer crashes) the user must wait until the AIK is revoked automatically due to specified expiration date. Thus, it is guaranteed that each host can only obtain and possess one identity in the overlay network and is not able to perform a sybil attack.

Balfe et al. [BLP05] provided a similar concept using DAA instead of the PrivacyCA. In their work the trusted DAA service is realized as a traditional trusted third party and is located outside of the overlay network. Under normal circumstances this assumption is adequate since almost all solutions in the context of PKI rely on the same assumptions. Thus, we think it is appropriate to make use of the PrivacyCA concept for providing trusted identities. However, for a more general solution the PrivacyCA should also be used in a distributed manner in order to prevent a single point of failure. How to make a certificate authority distributed is out of scope for this thesis but several solutions have already been proposed such as [AD01, LZPL05, ADH05, CCF08, LMT09].

In our solution the PrivacyCA is an integral part of the overlay network as well as the authentication process. Each node must obtain an AIK certificate from the PrivacyCA and must also possess the PrivacyCA public key in order to be able to participate in the required security protocols. This could also be achieved if each peer obtains such a certificate prior to joining the overlay network. But since we want to achieve an open and dynamic system which is also reliable it is necessary that the PrivacyCA is also available during the overlay network *runtime* so that new nodes can also join at any time.

In figure 5.5 an overview of the authentication process using the PrivacyCA is given. In contrast to the endorsement key solution, the joining node must first obtain an AIK certificate from the PrivacyCA with which it then authenticates itself inside the network. The PrivacyCA ensures that only one AIK certificate is issued per endorsement key at a given time period. The authentication process with the AIK facilitates a normal authentication process using Needham-Schroeder-Lowe. The nodes inside the network must only verify that the AIK certificate is still valid and if the issuer is the trusted PrivacyCA.

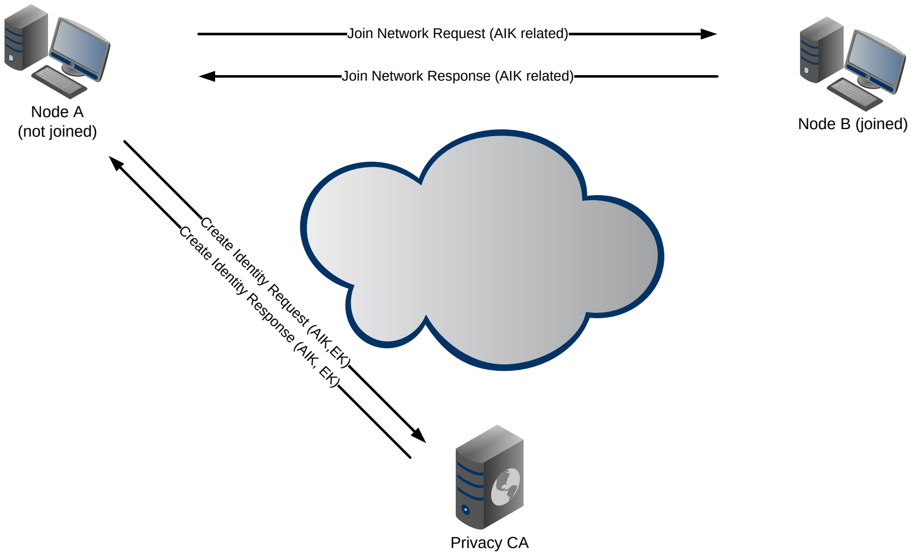

Join Network Request (AIK related)
Join Network Response (AIK related)
Node A
(not joined)
Node B (joined)
Create Identity Request (AIK,EK)
Create Identity Response (AIK, EK)
Privacy CA

Figure 5.5: Authentication process using PrivacyCA solution.

5.8.1 Distributed Privacy CA

An overlay may be formed on-the-fly as open system for a specific purpose. Thus, it is reasonable to assume that some *initial* nodes bootstrap an overlay by admitting *outside* nodes according to a predetermined *policy* and by issuing some cryptographic credentials to them. For example, a policy may specify the criteria for admitting outside nodes such as the possession of a valid endorsement key certificate and whether admitted nodes are allowed to admit other outside nodes themselves.

It is important to note that the outside nodes which are allowed to join the network may not be known in advance. In our solution it is possible for all nodes to join as long as they satisfy the *policy*

In order to realize the before mentioned open system and to give the joined nodes the possibility to protect their privacy despite using the endorsement key and attestation identity key certificates for the registration and authentication process we make use of a distribute PrivacyCA concept. Lesueur et al. [LMVTT08, LMT09] describe a solution to establish and maintain distributed certification schemes in structured overlay networks as outline in Section 5.3. Applying such a solution to the PrivacyCA concept would allow for issuance, management and revocation of anonymous credentials within the overlay network in a distributed manner. Using a distributed PrivacyCA solves the problem of privacy by issuing anonymous credentials for each node possessing an endorsement key certificate and the single point of failure issue is solved by partitioning the PrivacyCA functionality to a distinct amount of nodes. The distributed PrivacyCA itself holds a record of the mappings between issued anonymous credentials and endorsement key certificates in order to only allow one anonymous credential per endorsement key certificate. If a node leaves the overlay the associated mapping is erased and thus if the node wants to join again obtains new anonymous credentials. If a node does not perform the leave procedure for any reason the mapping associated with its endorsement key certificate is erased after a specific timeout. This solution solves the privacy problem of using the endorsement key certificate for authentication with the assumption that initial nodes exist which are trustworthy and perform the distributed PrivacyCA functionality.

5.8.2 AIK protocol

Each client platform with a TPM must perform a cryptographically secure protocol with the PrivacyCA to obtain an AIK certificate. In the TCG specifications the commandos and interfaces for the TPM as well as the PrivacyCA actions are described. However, no particular transport protocol for the exchange of the information between the client's platform TPM and the PrivacyCA is specified. Thus, we subsequently outline the steps required to obtain a AIK certificate from the PrivacyCA:[1]

1. On the client platform the `Tspi_TPM_CollateIdentityRequest` command is executed by the TSS which triggers the creation of a new AIK.

2. Through the TSS the `TPM_MakeIdentity` function is invoked in order to create a new attestation identity key-pair (AIK). The public AIK signed with the private AIK is returned by the TPM to the TSS.

3. The TSS thereafter creates the request which is sent to the PrivacyCA. The request contains the signed public AIK and the platform's endorsement certificates and is encrypted with the PrivacyCA's public key.

4. The PrivacyCA decrypts the request using it private key and verifies the validity of the content. The PrivacyCA also checks that no AIK certificate has been issued to the client or if a prior AIK certificate exists its validity has expired.

5. After the PrivacyCA has verified the request successfully it issues an AIK certificate. This AIK certificate is encrypted with a symmetric key. The symmetric key itself is also encrypted with the clients TPM public EK. Thus, the response can only be decrypted by the client TPM.

6. At the client the `Tspi_TPM_ActivateIdentity` TSS command is executed using the response from the PrivacyCA. Thus, the clients TPM is called

[1] Note that some details are omitted for clarity of presentation. For a complete description please refer to the TCG specifications [Tru], [Tru07b] and [Tru07c].

to decrypt the response. If the associated AIK is available the symmetric key is returned.

7. The symmetric key can then be use to decrypt the AIK certificate contained in the response of the PrivacyCA.

The activated AIK consists of two parts. The identity keys stored in the TPM and an certificated issued by the PrivacyCA which is associated with them. The AIK certificate proves that the AIK key pair belongs to an authentic TPM. The AIK certificate contains information the endorsement key and platform endorsement (PE) certificates. However, the contained information does not to allow to track the client's platform or to identify the specific hardware configuration. An AIK certificate can additionally include an arbitrary string which is chosen by the client. This string can be used for distinction of the certificate among a set of owned certificates. This maybe helpful for instance if the activated AIK is associated with a specific task.

5.9 Implementation and Evaluation

5.9.1 Trusted Authentication using EK solution

In this authentication mode we follow the protocol described in Section 5.7.1. The protocol starts with an initial exchange of keys, the AIK public key and the Endorsement public key. Having different peers with their TPM enabled, the authentication process will make encryptions and decryptions with each TPM. Each machine must be able to encrypt the data so that only the target machine is able to decrypt this information. This is done with the exchange of the AIK public keys and the endorsement public key. Thus, we are able to encrypt data using the public key of the other TPM.

The implementation details of the trusted authentication protocol using the EK solution are illustrated in Figure 5.6 and outlined below.

1. Peer 1 invokes `TMP_CollateIdentityRequest2` to generate a key pair called Attestation Identity Key (AIK). Peer 1 thereafter sends the Endorsement Public Key (EK_{PUB1}) and the Attestation Public Identity Key (AIK_{PUB1}) to Peer 2.

2. Peer 2 invokes `TMP_CollateIdentityRequest2` to generate a key pair called Attestation Identity Key (AIK). Peer 2 thereafter sends the Endorsement Public Key (EK_{PUB2}) and the Attestation Public Identity Key (AIK_{PUB2}) to Peer 1.

3. Peer 1 generates a random number (R1) and a symmetric key. The symmetric key with a hash of AIK_{PUB1} and AIK_{PUB2} is encrypted with EK_{PUB2}, obtained in the initial phase (2). This structure is called asymmetric blob. R1, among other data as outline in Section 5.7.1, is encrypted with the symmetric key creating a symmetric blob. The resulting packet

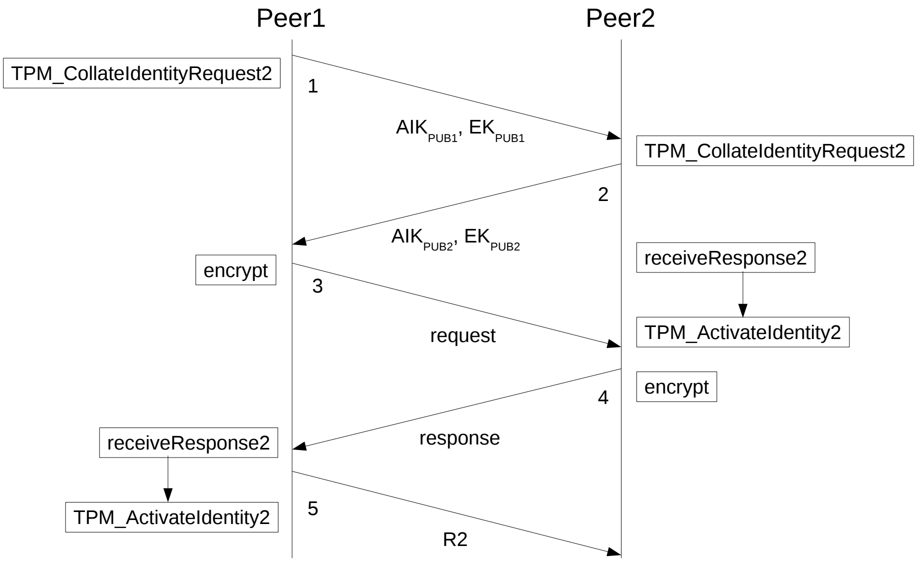


Figure 5.6: Trusted Authentication Protocol without PrivacyCA.

formed by the concatenation of symmetric and asymmetric blobs assures that the request can only be decrypted by the intended recipient TPM, in this case Peer 2.

4. Peer 2 invokes `receiveResponse2` to decrypt the data by calling the `TMP_ActivateIdentity2` function. This way we obtain R1. Peer 2 generates a random number R2 and encrypts R1 and R2 following the same process as peer 1.

5. Peer 1 receives R1 and R2 decrypting in the same way as peer 2 did. Then it checks that the received R1 corresponds with the R1 that was sent before. Finally, peer 1 sends R2 without any encryption to peer 2, who will check if R2 matches.

After this protocol is finished mutual authentication is guaranteed and the already joined peer knows about the identity of the joining peer and has verified if it possesses a TPM and the AIK keys are protected by it. Thus, the joining peer has an activated AIK key pair which can be used further on. Depending on the applications which are based on such a solution several options for key distribution are now available. For instance, if an overlay network which implements our security concept uses this solution the verifier peer (in the examples case peer 2) could provide peer 1 with the current session key, with specific pair-wise shared keys or issue an certificate for either AIK or any other public-private key pair. Thus, the newly joined peer would possess keys which are authorized inside the overlay network. On what basis or policy peers are allowed to join such a network is not the scope of this thesis. For instance, it would be possible that all peers which possess an authentic TPM and EK may join. However, such an approach would not inherently be secure against insider attacks but since the EK is used it is possible to detect misuse using specific detection mechanisms.

5.9.2 Trusted Authentication using Privacy CA solution

We have implemented a `TPMmanager` that handles communication with the TPM hardware. We needed to modify several libraries from the trusted java framework [2] in order to be used within our solution, these changes will be detailed further on.

The process starts when a user wants to connect to a network, this user will be called attestant. This attestant has to obtain an AIK certificate that does not contain any information that could link to the specific platform hosting the AIK from the PrivacyCA. To avoid the creation of possible identity attacks, a PrivacyCA has to check that a certain TPM or endorsement key certificate is only issued one AIK certificate. The flow that we follow is based on the AIK protocol described in Section 5.8.2. The implementation details of the trusted authentication protocol using the EK solution are illustrated in Figure 5.7 and outlined below.

1. The client prepares a request to send to the PrivacyCA :

    - In the first step the client calls the method `createRequest` that invokes the `Tspi_TPM_CollateIdentityRequest` TSS function to initiate a new AIK creation.
    - The TSS invokes the `TPM_MakeIdentity` TPM function to create the new attestation identity key (AIK) that is a RSA key pair.
    - The TPM returns a structure containing the public AIK, signed with the private AIK key. Then, the TSS generates a request where the data is stored in an identityProof structure containing the signed blob returned by the TPM, the first random number $R1$ and the EK and PE certificates of the platform.
    - This data is encrypted with the public key of the PrivacyCA.
    - The client sends to the PrivacyCA this request.

2. The PrivacyCA prepares a response to send to the client :

    - The PrivacyCA calls the method `processRequest` where it decrypts the data, and then it validates its content and the certificates contained in the request. The most important checks are that for the specific EK no AIK certificate has been issued or if the issued AIK

[2] Trusted Computing for the Java Platform `http://trustedjava.sourceforge.net/`

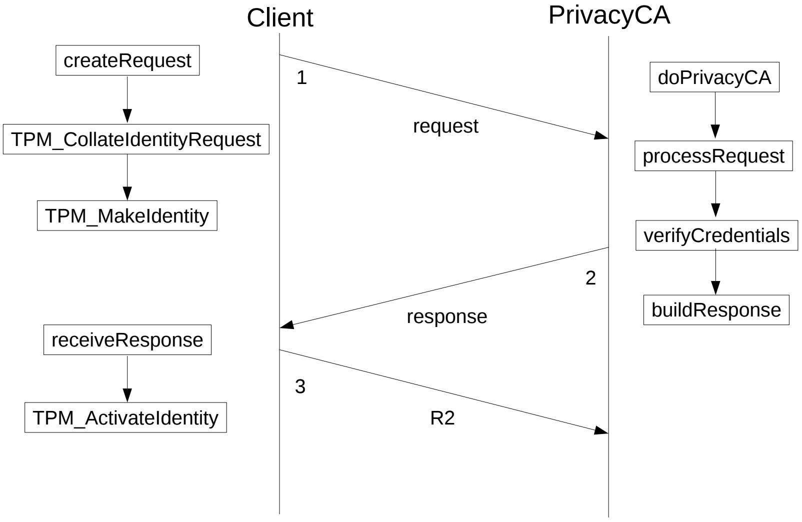


Figure 5.7: Trusted Authentication Protocol with PrivacyCA.

certificate has expired that no AIK certificate has been issued for the contained AIK. All other checks are similar to the ones a ordinary certificate authority performs.

- On successful validation the PrivacyCA issues an AIK certificate, encrypted with a symmetric key.
- This symmetric blob also contains the random number $R1$ that was received before and a new random number generated by the PrivacyCA (R2).
- The symmetric key along with a hash of the public AIK is encrypted with the public key of the EK of the TPM, obtained from the EK certificate of the request. This structure is called asymmetric blob.
- The result package formed by the symmetric and asymmetric blobs assure that the response can only be decrypted by the intended recipient TPM.
- After building the response, it is sent back to the client.

3. The client checks the received data from the PrivacyCA:

   - Then, the client calls the `receiveResponse` method that invokes the `Tspi_TPM_ActivateIdentity` TSS function where it decrypts the response.

- The TSS subsequently requests the platforms TPM to decrypt the response package with the private part of the EK. If the referenced AIK is available on the TPM, the symmetric key is returned and it is used to decrypt the symmetric blob contained in the response.
- The client checks if the random number received from the PrivacyCA is the same number that was generated. Finally, the client sends R2 to the PrivacyCA, in this case without encryption.

5.10 TPM Time Measures

The implemented application accesses the TPM hardware of the machine multiple times to perform different operations such as credential creation, data encryption and decryption. In this section we analyze the two authentication modes and we measure the time required for the authentication process. How long it takes to execute the main operations of the TPM and the PrivacyCA, such as encrypt and decrypt as well as the whole authentication process itself. These results will outline the possible benefits and disadvantages of the two modes. All the tests have been made on HP running Linux and equipped with Infineon TPMs. The software has been developed for and is integrated in the Secure P2P framework SePP which is available from Sourceforge `http://sourceforge.net/projects/securep2p/`.

5.10.1 Trusted Authentication using EK solution

The following section shows the resulting tests we performed for the Trusted Authentication Process using the EK solution. We have made several consecutive tests to measure how long the different parts of execution take. Subsequently we outline the different parts for this solution.

initial phase This phase calls the `createRequest2` method, which creates the AIK credential using the TPM. In this phase we also obtain the Endorsement public key of the TPM. When this is done, the AIK public key and the Endorsement public key is exchanged between the peers.

encrypt Generates the random number R1, and prepares the encryption with AIK public key and the Endorsement public key of the other TPM. This way the intended TPM will be able to decrypt the data.

receiveResponse2 Decrypts with its TPM the encryption blob received from the other machine. This encryption blob consists in a symmetric and asymmetric blob structure.

The results of this process is shown in the following table, the last row shows the total time of execution that the process requires. All the measurements are in seconds and milliseconds. The first row shows the mean time of all the performed measurements. The second and third row show the maximum negative and positive deviation from the mean measurements.

Table 5.1: Test Time Measures for TAP without PCA

Test	Mean [ss:SSS]	Max - deviation [ss:SSS]	Max + deviation [ss:SSS]
initial phase	03:385	00:045	00:035
encrypt	00:170	00:005	00:010
receiveResponse2	02:420	00:030	00:040
Total	05:990	00:050	00:050

The initial phase takes the most time and has also relative high deviation from its mean time. The most time is consumed by the TPM as it creates the AIK credentials. The encryption of the request itself has the least influence on the total time. The `receiveResponse2` takes again relative long since it also has to verify the received request apart from decrypting it.

5.10.2 Trusted Authentication using PrivacyCA solution

The following section shows the resulting tests we performed for the Trusted Authentication Process using the PrivacyCA solution. We have made several consecutive tests to measure how long the different parts of execution take. Subsequently we outline the different parts for this solution.

createRequest Implements the creation of AIK credentials and builds an encrypted request using the PrivacyCa public key.

doPrivacyCA Decrypts the request received from the client, extracts the first random number (R1) and also the certificates of the client. After this is done, it generates a new random number (R2), and then the PrivayCA creates a response that is composed of two parts. The symmetric part, that contains a PrivacyCA credential, R1 and R2; and the asymmetric part, that contains among other data the symmetric key, which is used to decrypt the symmetric part. The asymmetric part is encrypted with the endorsement public key of the client and using the AIK public key to assure that only the intended client will be able to decrypt the response.

receiveResponse Receives the response from the PCA and decrypts it using the private key that is stored inside the TPM.

The results of this process is shown in the following table, the last row shows the total time of execution that the process takes.

Test	Mean [ss:SSS]	Max - deviation [ss:SSS]	Max + deviation [ss:SSS]
createRequest	03:50	00:04	00:04
doPrivacyCA	20:10	01:30	01:10
receiveResponse	02:40	00:06	00:10
Total	25:90	01:30	01:20

Table 5.2: Test Time Measures for TAP with PCA

In the PrivacyCA solution the most time is consumed in preparing the keys and certificates for encryption to allow a correct decryption in the TPM. Mostly it is because the PrivacyCA has to create a new AIK certificate based on the AIK credentials received from the TPM in order to build a well formed response.

5.11 Security and Privacy Analysis

The security of our solution depends first on the integrity and tamper-resistance of the TPM. Although the TPM has been designed and manufactured with these specific requirements in mind some initial attacks on the integrity of the data stored on the TPM or transmitted during the communication have been outlined such as [KDP05, Kau07, Gra09]. However, none of these attacks enables the disclosure of the private endorsement key, which is the integral part of our solution.

On the other hand our solution depends on the security of the AIK and the authentication protocol used. The AIK protocol itself uses well-known cryptographic procedures for transmitting data in a confidential and integrity ensuring manner. Thus, none of the sensitive information used in the AIK protocol is revealed. The only requirement is that the joining nodes and the PrivacyCA nodes posses the each others public key certificates. If not they have to exchange them in the beginning. Thus, if the EK certificate is not encrypted in this process, which can be simply achieved by using the PrivacyCA public key, it would be possible for an eavesdropper to learn the EK certificate of arbitrary nodes. The authentication protocol of the endorsement key solution itself is based on the Needham-Schroeder-Lowe [NS78, Low96] protocol which is a standard mutual authentication protocol commonly used. The security of this protocol has been verified by several independent sources and it is also the most used authentication protocol in general.

The most important security issue, which was the intent of the whole work, the provision of unique identities which are not subject to Sybil attacks is ensured through the reliance on the uniqueness of the endorsement keys and the simple registration process in the overlay. Since the manufacturer of the TPM provides the endorsement keys for each platform and signs them, their authenticity can be verified by checking the signature. The possession of the corresponding private key as well as the retention of the keys in the TPM is ensured by the used trusted computing functions and specifications. Thus, our solution can guarantee that a node who has joined does possess a unique identity which is bound to the computing platform it is using and that the same computing platform can not create multiple identities as well as denying its identity.

5.12 Conclusion

In this chapter we showed how mechanisms and information available through Trusted Computing and in particular the Trusted Platform Module can be used to enable the provision of unique trustworthy identities which are associated with one distinct computing platform. Previous approaches relied either on the creation of identifiers through hashing some arbitrary information such as IP addresses, network interface identifiers and other built-in identifiers or the assumption that each participant has been supplied with asymmetric cryptographic material in advance. These approaches can not guarantee a tamper-proof binding between the computing platform and the identifier since for instance all existing network identifiers can change or spoofed and public-private keys can be generated in arbitrary numbers. Even if a trusted third party would be used which would enable one person only to obtain one legitimate public-private key-pair this solution would be on the one hand too static for the requirements of dynamic overlay networks, on the other hand it would impose an unnecessary restriction on the association between identifiers and nodes since one physical person would only be able to participate with one platform in the overlay. It is much more advantageous to bind computing platforms to identifiers because this allows a separate level for user bindings in the system as it is possible for instance within Skype and other overlay networks.

As the measurements show the two approaches have a very different runtime. One of the main reasons that the PrivacyCA solution is that much less effective is because we used the original implementation of it, as provided from the IAIK, and only modified the parts required for our protocol. Thus, maybe if some of the functions which we do not require are omitted the performance would be better. The most time was consumed in the PrivacyCA solution by the key management mechanisms. This could be a good place to start optimizing this solution.

In case privacy is required the only solution which is workable is the one which uses the PrivacCA concept. But if simple deployment and performance is more of concern, for instance in a closed system as within a company, the best solution is to perform the authentication process using the endorsement key solution since it is 5 times faster.

6
Secure Routing

Whenever a theory appears to
you as the only possible one, take
this as a sign that you have
neither understood the theory nor
the problem which it was
intended to solve

Sir Karl Raimund Popper

In the previous chapter we have provided a solution for the first essential part for establishing a secure overlay network, namely the provision and verification of unique identifiers. Having such identifiers enables us to provide solutions for the next essential part which is secure route establishment and maintenance. In this chapter we will now focus on the aspect of routing and especially on secure routing algorithms for unstructured overlays. We first will evaluate what kind of routing algorithm is suitable for dynamic overlay networks. Then we will verify the applicability of the most prominent secure routing algorithm to our environment. We will discuss the advantages and shortcomings of this routing protocols and present our own solution to the this problem. In the end we evaluate and present the performance of our secure routing protocol and discuss our solution.

6.1 Introduction

To be able to apply a security concept and implement secure unstructured overlays it is necessary to first identify and define the constraints for the intended overlay network. Because these assumptions and constraints greatly influence the requirements for the security concept and the secure routing algorithms itself. In general, overlays can be described in terms of node duties and interaction rules. We provide the characteristics of overlays as we envision them. For the work described in this chapter we define an overlay system and the participating nodes in the following way, which is only one of many possible ones.

- No single *online* entity exists controlling the execution of the system.
- All nodes are only governed by the overlay networking protocols and mechanisms.
- The system modifies itself to adapt to changes in the composition of the system.
- The system reacts on communication failures and node faults.

Also specific assumption in relation to the physical capabilities and characteristics of the network and the nodes must be specified for a comprehensive solution. We assume a very heterogeneous system in terms of node resources and network technology. Thus, we do not place any restrictions on the nodes in terms of available computing power, memory or energy capacity. Meaning that a node can either be a powerful server but also a constrained embedded device. Nodes can communicate using any available network or communication interface which is able to transport IP packets. The nodes can be connected via fixed wired or wireless networks or use mobile ad-hoc networks to communicate. Nodes are not required to be at fixed locations or always powered on.

Thus, we end up with a dynamic and heterogeneous system. It is necessary to take that into account in the design of any routing protocol able to function in such an environment. We have studied various routing protocols which are used in similar configurations. We have investigated protocols such as DSR, DSDV, AODV, OLSR, TORA and ZRP. Protocols like DSR, DSDV and AODV [PB94, Joh94, JM96, PC97, PBR99] have been developed for mobile ad-hoc networks and were studied intensively. We have come to the conclusion that the dynamic source routing protocol [JM96] is the best choice for the basis of our secure routing protocol. The complete path from the source to the destination is transmitted in each packet. Thus, the route can be verified at each node on the route. For a more elaborate discussion on routing protocols see Section 6.12.

6.2 Security Routing Concept

The design and implementation of the protocols used in our secure overlay solution, not only the routing protocols, use a simple but efficient security concept as their foundation. This security concept has been designed with heterogeneous system environments in mind. Another aspect in the design of the security concept was configurability, giving each node the freedom to select its desired level of security with respect to its capabilities. For more information on the security concept see Chapter 4.

In order to realize the security concept we have deducted some necessary assumptions. These assumptions are made in order to achieve security in overlays which would otherwise not be possible. Unfortunately, some of the liberties which are usually found in *insecure* overlays have therefore been restricted due to the lack of suitable alternatives. The limitations are related to the distribution of credentials and the verification of authenticity.

- We require each node to possess suitable credentials. Credentials in our case can either be some kind of shared secrets, symmetric or public/private keys.
- We require that each node has a unique verifiable identity. Meaning that it must be ensured that no peer with the same identity exists in the system. In systems with more sophisticated security requirements it may also be required that each identity is undeniably linked with the authentication and secret information. This is a very important requirement as outlined in [DFM00, Dou02, SM02, Wal02].
- In order to achieve the previous requirements each node must have the capability to store crucial information like the credentials and other cryptographic material in a secure manner. Such a protected storage can be as simple as a software *keystore* or more preferably a *smartcard* or the like [RKS02, vO03]. There are also some new concepts which try to not only bind such a protected storage to a user using a PIN or password but also to a specific machine or system state using trusted computing mechanisms [HTK10].

For a specific overlay application, the selection of security levels can be done on several levels (globally, group-wise, or peer-wise). As secure routing is a basic service underlying the whole overlay system, the routing security level is a global decision. Therefore, the admission security level and data security level for routing must be set globally. Group security levels are set group-wise. Most of the group security levels are decided by the group creator. This includes the admission security level and data security level for a group. Also the decision for a session key protection level of a group is done by the group creator. For a session key protection level unequal to zero, each peer can feature individually level one or two, as this involves only local differences of the implementation of the cryptographic primitives. The decision should be based on the degree of exposure to side-channel attacks.

6.3 Routing Security

First we have examined different unsecured routing protocols and we came to the conclusion that Dynamic Source Routing (DSR) is the best fit for our purposes since it has low overhead, is ad-hoc in nature and doesn't require permanent communication between nodes. Thereafter, we investigated different secure variants of DSR which have already been proposed. The most famous secure incarnation of DSR is Ariadne [HPJ02, HPJ05]. Ariadne incorporates most of the other secure versions of DSR [JLR04, KGSW05] into its framework. It also allows for an additional source for asymmetric cryptographic security through the use of broadcast authentication based on delayed disclosure of keys. In the following sections we outline the fundamental features of Ariadne. We mostly outline Ariadne used in combination with the broadcast authentication protocol called TESLA [PCTS02], since it is substantially different to the more common variants using either pair-wise shared secrets or public and private keys. Ariadne is not intended to be used as overlay routing protocol but it is a good choice for secure ad-hoc routing, which has several similarities with overlay routing. We give a short introduction into DSR on which Ariadne is based.

6.3.1 Security Threats

In general security threats can be divided into outsider vs. insider attacks. Outsider attacks are performed by unauthorized users or nodes which do not have access to network. Thus, they have to rely on methods such as fabrication and interception in regard to the security attacks outlined by [Sta99]. Insider attacks are performed by legitimate users of the network which are authorized to access relevant information and possess appropriate credentials. For these kind of attacks all attack methods such as interruption, interception, modification and fabrication are possible. In regard to routing security in case a source routing algorithm is used, we can outline the following outsider attacks:

- Denial of service - is an attack where one or more malicious nodes try to block access to a specific service through sending so many service requests that the service can not handle them all.
- Man-in-the-middle attack - is an attack where a node tries to relay requests and impersonates a legitimate node in order to be able to access the transmitted information.
- Replay attack - is an attack where a node replays previously intercepted requests in order to disrupt or force to terminate ongoing communication between legitimate nodes.

In addition to these attacks insider can also perform some additional attacks which are the following:

- Sybil attack - is an attack where a malicious nodes creates a large number of identities in order to control the networks behavior.
- Eclipse attack - is an attack where colluding malicious nodes try to control the communication possibilities of other legitimate nodes.

- Route modification - is an attack where malicious nodes modify the messages sent during the route establishment or maintenance phase. Depending on the goal of the malicious node different modifications are possible. For instance, the malicious node could try to have all routes running through it in order to disrupt or separate the overlay network. Or it could mount denial of service attacks against a specific node by adding that node to all routes.

- Grey- or Blackhole attack - is an attack where nodes selectively forward traffic. If traffic from specific nodes or of a specific type is not forwarded it is called Greyhole. If none of the traffic is forwarded by a specific node it is called Blackhole.

- Wormhole attack - is an attack where two or more nodes collaborate to tunnel traffic between them in order to disrupt the correct execution of network protocols.

A suitable secure routing protocol must provide protection against all or most of these attacks since there is no definitive solution to prevent the Wormhole attack. Some solutions which impede the Wormhole attack such as temporal and geographical leashes [HPJ03], or graph theoretical considerations [LPM$^+$05] exist. Subsequently we will analyze existing secure routing protocols and will only consider such algorithms which provide protection against all of these attacks except the Wormhole attack.

6.3.2 Dynamic Source Routing

DSR is an simple and efficient on-demand ad hoc network routing protocol based on source routing and composed of two parts, route discovery and route maintenance. The basic principle of source routing is that each peer sending a packet to a distinct destination specifies the sequence of hops the packet has to follow on its way to the destination. DSR consists of two different processes, the *Route Discovery* and *Route Maintenance*, which are explained subsequently. For more detailed information see [Joh94], [JM96], [JMB01].

Route Discovery

If a peer wants to send a packet to a distinct destination but does not have a route to this destination in its local *Route Cache*, the peer initiates the *Route Discovery* process to find one. The sending peer is known as the initiator of the Route Discovery, and the destination of the packet is known as the target. The initiator sends a *ROUTE REQUEST* packet as a local broadcast, specifying the target and a unique identifier. If a peer receives a fresh request, it appends its own peer address to a list in the request and broadcasts the request again until the request reaches its target peer. The target sends a *ROUTE REPLY* back to the initiator of the request, containing the discovered route.

Route Maintenance

The *Route Maintenance* process is initiated if forwarding a packet to the next hop specified in the route fails. Such a broken route can occur due to changes in topology or peer failures. If a peer discovers a broken link it returns a *ROUTE ERROR* to the initiator of the packet, containing a reference to the broken link between the current peer and the next hop peer. The initiator then removes this broken link from its *Route Cache*. For subsequent packets to this destination, the initiator may use any other route to that destination in its *Route Cache*, or it attempts to start a new *Route Discovery* process for that target if no alternative route is available.

6.3.3 TESLA

The dynamic source routing mechanism of Ariadne uses flooding for the *Route Discovery* process. Thus, the first major step towards a secure overlay routing protocol is to enable source authentication in our overlay system. A broadcast authentication protocol suitable for a heterogenous pure overlay system should have the following requirements:

- Low computational overhead,
- Low communication overhead,
- Scalability, and
- Applicable to different communication environments

Broadcast authentication protocols covering these requirements have already been proposed. The research of Perrig et al. [PCTS00], [PCST01] led to the specification of TESLA [PCTS02]. TESLA can also be used with wireless sensor nodes as shown in [PST$^+$02]. The protocol has the following special requirements which must be fulfilled by peers:

- The sender and the receivers must be at least loosely time-synchronized
- Either the receiver or the sender must buffer some messages

The main idea of TESLA is that the sender attaches to each packet a MAC computed with a key k known only to itself. The receiver buffers the received packet without being able to authenticate it. With strict adherence to a publicly known schedule the sender discloses the key k and the receiver is able to authenticate the packet. Consequently, a single MAC per packet suffices to provide broadcast authentication, provided that the receiver has synchronized its clock with the sender ahead of time and exchanged authenticated key commitments. Because of the delayed key disclosure and the characteristics of hash functions, asymmetry is achieved. This means that receivers can verify the authenticity of the message, but can not generate valid authentication information themselves. Because before the keys are disclosed the time interval at which the keys are valid has already expired. Thus, if an attacker uses a disclosed key of another node to generate authenticated message on its behalf, other nodes will not accept these messages since the time interval in which the used key is valid has already passed.

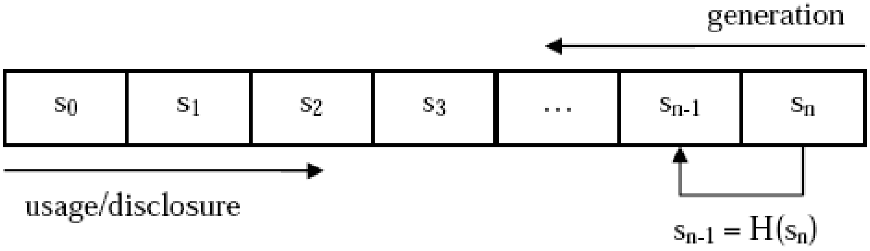


Figure 6.1: One-way hash chain

TESLA repeatedly uses a one-way hash function to generate a one-way key chain. The elements of the one-way hash chain represent the individual symmetric keys which are used for computing the MAC. Figure 6.1 shows the construction of the one-way key chain. The elements are calculated by applying the hash function H repeatedly n times on a randomly selected element s_n. The element calculated last s_0 is used as commitment to the entire one-way chain. Every element s_i in the one-way chain can be verified to be authentic by applying the hash function H repeatedly i times to itself and resulting in the element s_0. The keys s_i are disclosed in the following order $s_1, \ldots, s_n$. Also all receivers must be loosely time synchronized with the sender up to some time synchronization error. For more information about the TESLA protocol please refer to [PCTS02].

6.3.4 Ariadne

Ariadne is a secure on-demand routing protocol for ad-hoc networks based on the dynamic source routing (DSR) protocol. As already mentioned previously, in this analysis we use Ariadne in combination with TESLA. For the two other use cases, pair-wise shared secrets and digital signatures, the security and protocol steps are similar with the exception that the asymmetry relies on the keys. In the other two use cases no keys are disclosed and therefore it is not necessary to wait and buffer messages before they can be authenticated. But since digital signatures are rather expensive in terms of computational power and the key management and storage efforts for pair-wise shared secrets are tremendous, TESLA could be a good alternative.

There are 3 basic mechanisms Ariadne adds to DSR in order to secure it. First, a MAC over the route request is computed with the shared key K_{SD} and added to the REQUEST. This key is only shared by the specific source and destination. Therefore, the destination can easily verify the authenticity and freshness of the request using the shared key K_{SD}).

The second mechanism authenticates every peer participating in the *Route Discovery* process. This is done for the request and the reply. This means that the source of the request must be able to authenticate every peer in the hop list of the reply. Also the destination must be able to authenticate every peer in the hop list of the request in order to identify a legitimate route and create a correct reply. This is achieved by the requirement that each hop authenticates new information in the request. The target buffers the request until all hops can release the corresponding TESLA keys. Then the destination checks this information and creates the reply if all information could be verified.

In order to prevent route modification attacks the third mechanism uses one-way hash functions to verify that the route was not modified. Each node on the route adds its identity to the hop list and signs it with its current hash key. This approach is called per-hop hashing. To be able to modify the route an attacker must be able to invert the used one-way hash function. The other case mentioned in [HPJ02] is not an attack, because if a peer hears a request, without the peer listed he wants to remove, he actually is on a legal route without the other peer.

6.4 Implementation

In this section, we describe how we implemented Ariadne in a overlay fashion which is somewhat different to the original version. First, we use a off-line certificate authority for the bootstrapping process. This CA manages the IDs of all peers in the network and ensures that no ID is used by two peers. It also issues the public/private keys for each peer before start-up. Therefore, all legitimate peers have a public key and a unique peer ID signed by that authority which can be used for secure TESLA key exchange and time synchronization. All the peers are deployed with a copy of the CA's public key to be able to verify the authenticity of other public keys.

6.4.1 Sender Setup

First, the sender defines time intervals of duration T_{int}. Every key of the key chain is then assigned to one interval. Since all keys are only used once, the length of the one-way chain determines the maximum communication interval before a new key chain must be generated. The keys to sign the hop list in a request are chosen so that they can be released at the timeout and the destination does not need to buffer the message until the keys are released. Perrig et al. proposed in [PCTS02] to choose the key disclosure delay according to the formula $d = RTT/T_{int} + 1$, where T_{int} is the interval duration and RTT is the round trip time between sender and receiver.

6.4.2 Initialization Process

In order that each receiver can authenticate messages using TESLA he must be loosely time synchronized with the sender. The receivers must also posses an authenticated key of the one-way key chain used by the sender and be aware of the schedule of key disclosures. The authenticated key of the one-way hash chain is s_0 which is signed with the nodes private key and distributed on request to the receivers.

Time Synchronization

To achieve time synchronization we implemented a simple time synchronization 3-way-handshake protocol. Instead of using the 2-way protocol for time synchronization, as specified in [PCTS02], we use a 3-way protocol. This allows us to obtain information about the propagation delay between sender and receiver at both locations. Figure 6.2 shows the 3-way time synchronization protocol between two peers. The initiator first records its local time t_{iReq} and sends a time synchronization request containing a nonce to the receiver peer. Upon receiving the request, the destination records its local time t_{dReq} and replies with a signed response message containing t_{dReq}, the nonce received from the initiator, and a new nonce. Finally, the initiator completes the process at time t_{iResp} by sending a signed response containing the received nonce to the receiver, which obtains it at t_{iResp}. With the difference $t_{iResp} - t_{iReq}$ of every request-response pair we

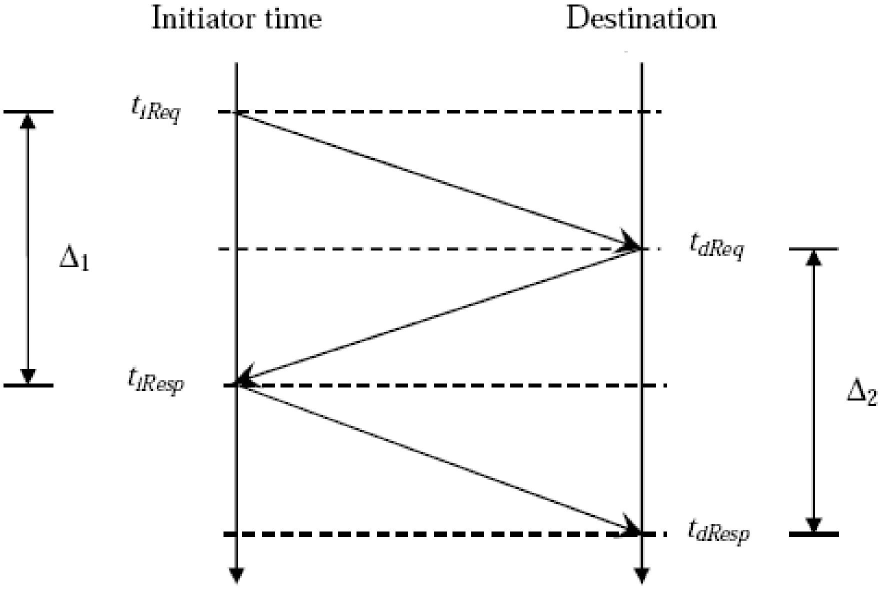
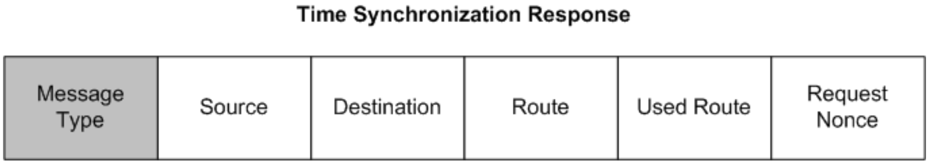


Figure 6.2: Direct time synchronization between peers

measure the round trip time which is later used as maximum time synchronization error. After this process, the initiator can calculate the upper bound on the current destinations's time as $t_d = t_{dReq} + (t_{iResp} - t_{iReq})$ by assuming that the time synchronization error is Δ_1 (the full round-trip time (RTT)). Analogous the receiver can calculate the upper bound on the current initiator's time as $t_i = t_{iResp} + (t_{dResp} - t_{dReq})$.

Time Sync Request



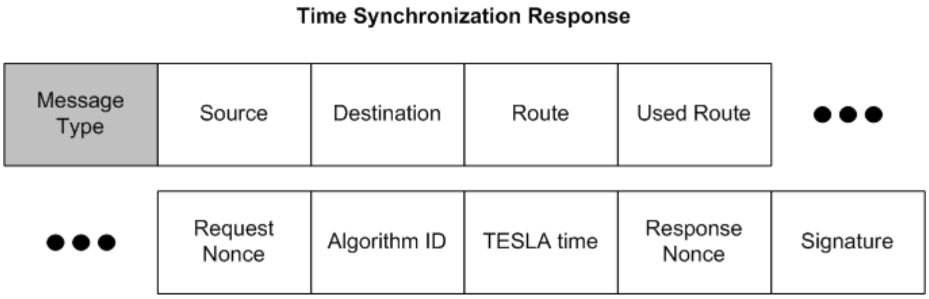
Time Synchronization Response

Message Type	Source	Destination	Route	Used Route	Request Nonce

Figure 6.3: Time synchronization request message format

The time synchronization request message contains six parameters as shown in Figure 6.3.*Source* and *Destination* are set to the IDs of the initiator and receiver peer. *Route* is set to the source route to the receiver peer. To make it possible for the receiver to reply to the request, the source route actually used is stored in *Used Route*. This is needed because time synchronization mostly happens before a route between these peers has been established and thus the route field may be empty when it reaches the receiver. *Request Nonce* is set to a random value to ensure message freshness and avoid replay attacks.

Time Sync Response



Time Synchronization Response

Message Type	Source	Destination	Route	Used Route	●●●
●●●	Request Nonce	Algorithm ID	TESLA time	Response Nonce	Signature

Figure 6.4: Time synchronization response message format

The time synchronization response message from the receiver to the initiator peer consists of ten fields as shown in Figure 6.4. *Source* and *Destination* are set to the IDs of the receiver and initiator peer. *Route* is set to the route over which the message should be sent and can be retrieved by reversing the used route field in the received request message. *Algorithm ID* is a unique identifier of the used algorithm to calculate the signature over the TESLA time and the response nonce. *TESLA Time* specifies the time relative to the time of the start of time interval zero (T_0). The *Response Nonce* is set to the value of the request nonce in the time synchronization message to avoid replay attacks. The *Used Route* and *Request Nonce* fields from the new request are set to the used route and a new nonce.

Time Sync Final

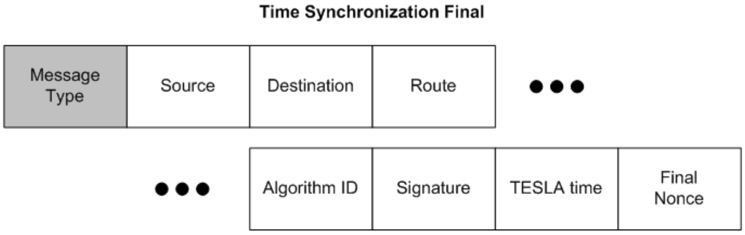


Figure 6.5: Time synchronization final message format

The final time synchronization response message from the initiator to the receiver is equal to the response message. The format of this message is shown in figure 6.5. Asymmetric cryptography is used to sign time synchronization response messages as specified in the original version. Every peer is able to authenticate the time synchronization response with the other's peer public key. After the time synchronization both peers know the round trip time and the TESLA time of the other peer and are therefore loosely time synchronized and can calculate an upper bound of the other's peer TESLA time.

6.4.3 Key Exchange

To be able to authenticate signed messages, TESLA keys and public keys must be known by the communicating peers. If a peer does not have the key to authenticate a received message a key exchange must be performed. The key exchange process uses a simple protocol consisting of two messages, the *Key Request* and *Key Response*.

Key Request

The key request message contains the following fields as shown in Figure 6.6. *Source* and *Destination* are set to the IDs of the initiator and receiver peers and *Route* to the effective route between them. The *TESLA Key Algorithm ID* is the unique identifier of the used algorithm to calculate the *TESLA signature* with the peers private key of *TESLA key* at *TESLA Key Index.* The *TESLA key chain length, TESLA key validity interval* and *TESLA key disclosure delay* are additional information necessary for verifying the signature. The *TESLA Key* is

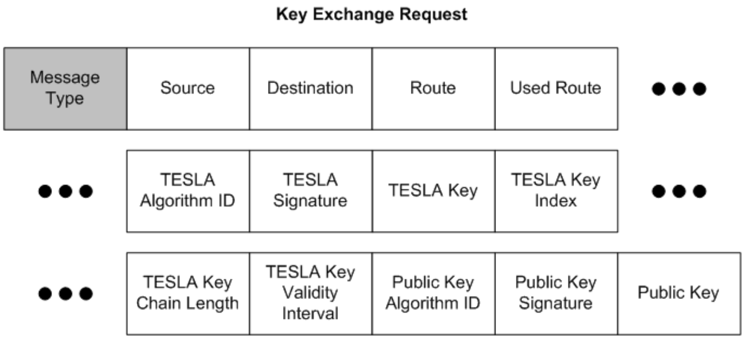


Figure 6.6: Key request message format

an already disclosed key and the *TESLA Key Index* is the corresponding index of that key. The *Public Key Algorithm ID* is the unique identifier of the used algorithm to calculate the *Public Key Signature*. As with time synchronization, key exchange mostly happens before routes have been established. Therefore, the initiator has to send also the used route to the destination so that this peer is able to reply to the request.

Key Response

The destination peer replies to the request with a key response message which has the same fields as the key request message except that the message type is different and that the used route, which is not needed in that case, is omitted. To verify the key exchange messages, first the signature of the contained public key has to be verified with the CA's public key. Afterwards, the contained TESLA key and key schedule can be verified with the previously verified public key.

6.4.4 Routing Processes

Route Discovery

The Route Discovery process works as described in section 6.3.2. Ariadne attempts to authenticate peers and secures only a basic version of DSR without possible optimizations. The Route Discovery process consists of two stages as in the original DSR. First the initiator floods the network with a route request and the target returns a route reply. Each request packet in our implementation contains the following parameters as shown in figure 6.7. *Source* and *destination* are set to the IDs of the initiator and target peers. The initiator sets the *ID* to a unique value not recently used in initiating a route discovery. The *timeout* is the upper bound of the expected arrival time of the request at the target. The initiator sets the *Signature* to the digital signature over the fields *message type, source, destination, ID*, and *timeout*. The hash chain is initialized to the hash over the *Signature* and the peer list, the key indexes and MAC list to empty lists. The TTL is set to a value which specifies how many hops such a request

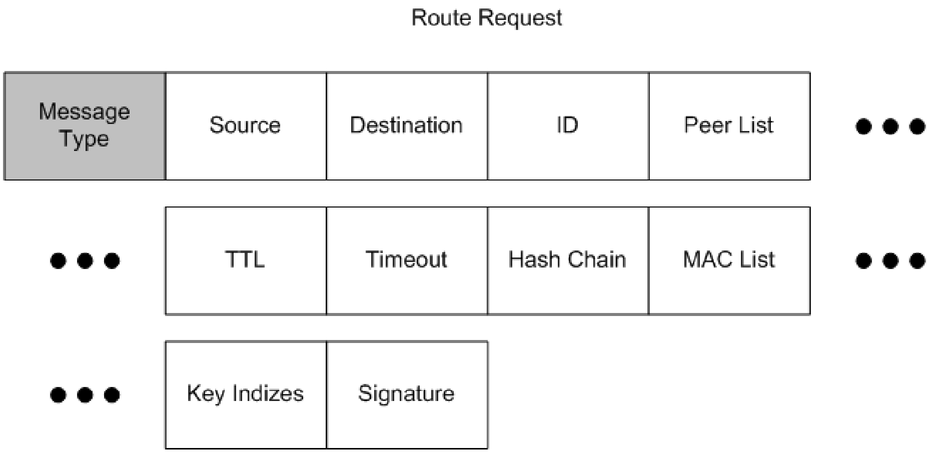


Figure 6.7: Route request message format

may travel at maximum. This value depends on the overlay network size and is adjusted accordingly.

When any peer A receives a request for which it is not the target, the peer performs some checks to verify that the packet is valid, for more information see [HPJ02]. If all checks succeeded the peer modifies the request by appending its own address to the peer list, replacing the hash chain field with $H(A, hashchain)$, appending a MAC over the entire request (except the TTL field because it changes after every hop) to the MAC list, and decrementing the TTL. The peer uses the TESLA key K_{A_i} to compute the MAC, where i is the index of the first key that will be disclosed after the timeout. Finally, the peer broadcasts the modified request. When the target peer receives the request, it checks the validity by verifying that the keys from the time interval specified have not been disclosed yet, that the signature is valid, and the hash chain field is equal to: $H(h_n, H(h_{n-1}, H(h_{n-2}, ..., H(h_1, signature), ...)))$ where h_i is the peer address at position i of the peer list, and where n is the number of peers in the peer list. If the target peer determines that the request is valid, it returns a route reply to the initiator containing the fields shown in Figure 6.8. The *Source, Destination, ID, Timeout, Peer List, MAC List*, and *Key Indexes* fields are set to the corresponding values from the request. The *Signature* is computed over the preceding fields in the reply with the key list initialized to the empty list. The reply is then returned to the initiator of the request along the source route obtained by reversing the sequence of hops in the peer list of the request. A peer forwarding a reply waits until it is able to disclose its key previously used to calculate the MAC. The peer then appends this key to the key list field and forwards the packet according to the source route. When the initiator receives the reply, it verifies that the signature is valid, that each key in the key list is valid, and that each MAC in the MAC list is valid. If all of these tests succeed the peer accepts the reply and the route, otherwise it discards it. For instance,

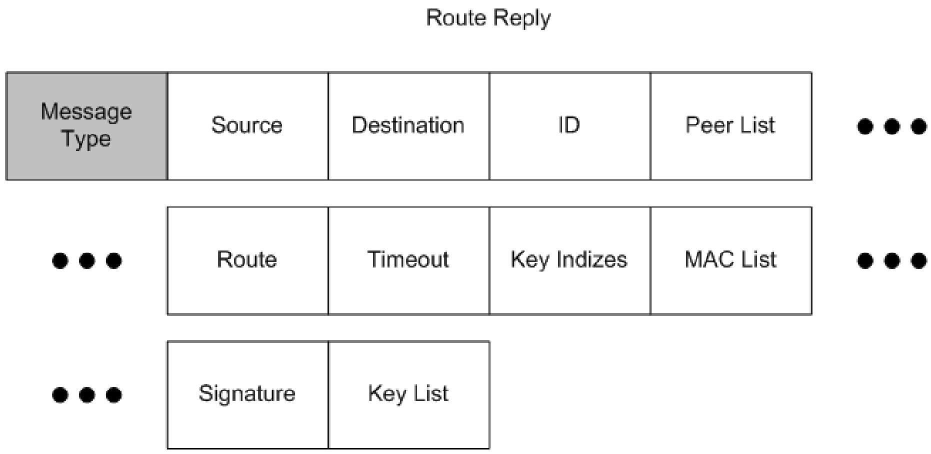


Figure 6.8: Route reply message format

if a peer uses an already disclosed key in order to impersonate another peer this would be identified during the checks. Also if a peer tries to modify the peer list, in order let a route appear shorter, it would be detected since the hash chain field contains the hashes of the removed peers and could not be computed using the modified peer list.

Route Maintenance

Route Maintenance is based on DSR as described in Section 6.3.2. If a peer has to forward a packet and the link to the next hop is erroneous, it has to send a ROUTE ERROR message back to the peer who originally sent the packet. We discuss mechanisms for securing route errors, but we do not consider the case of attackers not sending errors. To prevent unauthorized peers from sending errors, we require that an error is authenticated by the sender. Each peer on the return path to the source forwards the error.

A error packet contains nine fields: (ROUTE ERROR, initiator, destination, route, algorithm ID, signature, TESLA key index, disclosed TESLA key, target). The initiator field is set to the address of the peer reporting the error, and destination is set to the address of the peer which tried to use the erroneous route. The algorithm ID is the unique identifier of the used algorithm to calculate the signature of (ROUTE ERROR, target). The target field is the address of the next hop peer who is not available anymore. TESLA key index is the index of the used TESLA key to sign the message and last disclosed is the last disclosed key. Each peer on the return path searches its *Route Cache* for all routes containing the link indicated by the error. The peer verifies the correctness of the error message. If the message has been verified the peer removes all routes containing the broken link.

6.5 Performance Evaluation

In this section, we present the results of various measurements we have taken. First we discuss the needed bandwidth and show ways how it would be possible to minimize the byte overhead. Then we show the minimal needed timeout for a route discovery depending on the route length and finally we compare the performance of Ariadne with DSR on route recovery. The tests have been performed on personal computers with Intel Core 2 Duo E8600 processors with 3.33 GHz and 4 GB RAM and Windows Vista as operating system. The routing algorithm implementation has been developed and integrated in the SePP framework and was executed on Java JDK 6 Update 10 runtime environments. We established an real-world overlay network with up to 20 nodes connected in series resulting in the maximum path length of 20.

6.5.1 Bandwidth Requirements

Security always comes at the cost of higher overhead. Compared to DSR, Ariadne needs much more bandwidth especially during the initialization when all the bootstrapping has to be done. Without bootstrapping the needed bandwidth for Ariadne is about six times higher and with bootstrapping about 80 times higher. The length of the used TESLA-MACs has some influence, but the most overhead is caused by the asymmetric signatures and the bootstrapping process, as shown in Figure 6.9.

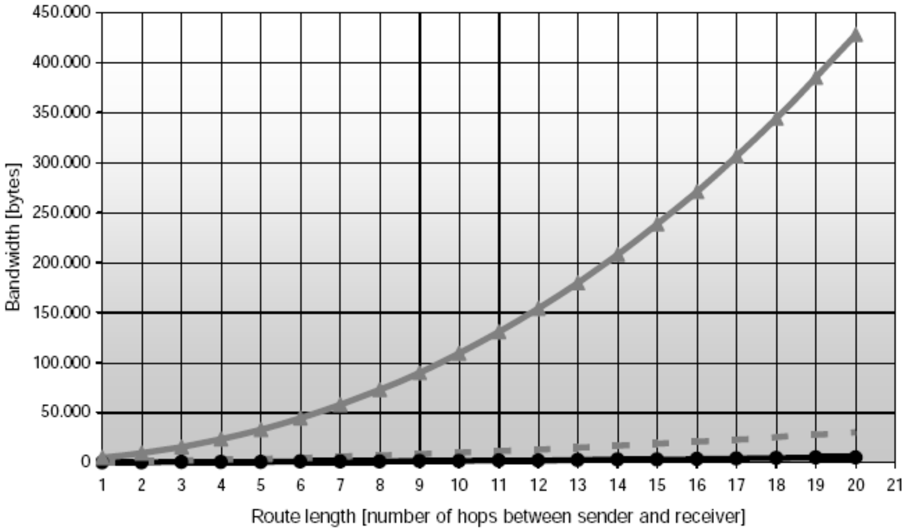


Figure 6.9: Comparison of the needed bandwidth for DSR and Ariadne

During the bootstrapping process the sender setup and initialization as outlined in Section 6.4.2 and 6.4.1 are performed. To avoid or minimize that overhead, smaller keys, MACs and signatures or ID-based systems could be used. Figure 6.10 shows the impact of the route length to the bandwidth needed and compares DSR with two versions of Ariadne. One version uses MACs and TESLA keys with a length of 80 bit and the other version works with 160 bit.

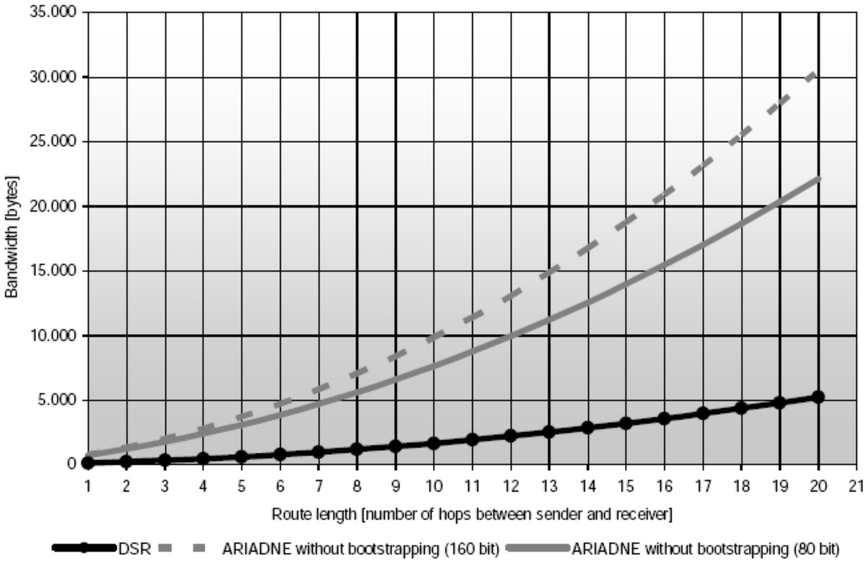


Figure 6.10: Comparison of the needed bandwidth depending of the route length and the length of the used TESLA keys and MACs

6.5.2 Minimal Timeout

The timeout is a very important parameter. All the peers along the route have to choose a key that is secure until the timeout. Therefore, all the peers have to wait until the timeout in order to append their used key to the reply and forward it to the sender. It is desirable to minimize the timeout as much as possible, but if it is too small than no route discovery will arrive in time at the receiver and discovery fails. Therefore, we implemented an exponential back-off algorithm to issue new request messages until a reply is received. Thus, we are able to cope with varying transmission delays in the overlay network and determine the current minimal timeout. However, since there are some variations in delay in the network even if the number of nodes and the topology stays the same an distinct amount of time must be added to the obtained minimal timeout in order to end up with a stable solution.

Figure 6.11 shows the minimal needed timeout as function of the route length with bootstrapping and Figure 6.12 without bootstrapping. From these two

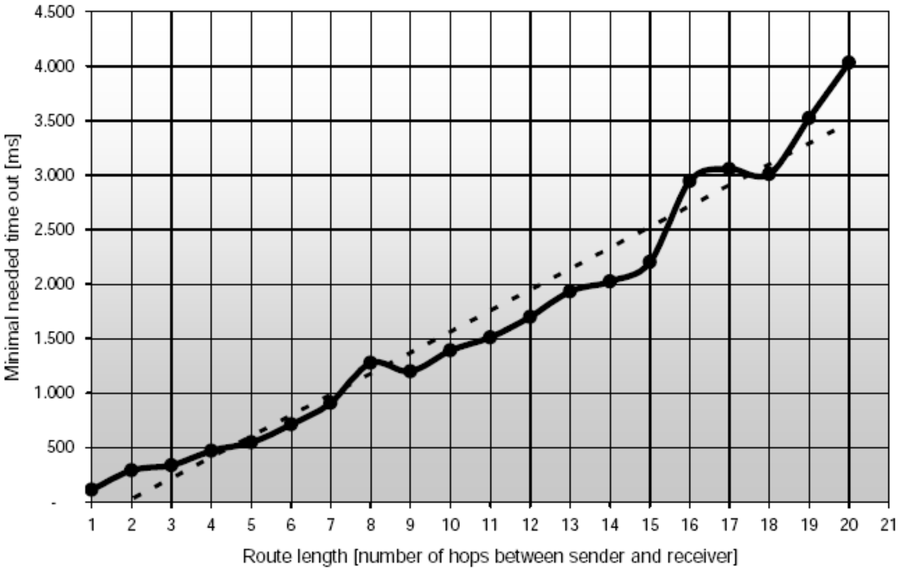


Figure 6.11: Minimal timeout for a Route Discovery with bootstrapping

figures, it is intuitively evident that the bootstrapping process has a high impact on the performance. In both figures the minimal needed timeout for a specific route length is shown. The dotted line shows the mean value of the measurements and the solid line shows the measurements of one consecutive run.

There are quite some timeout variations in one run between the different route lengths. These variations are due to variations in the Java runtime (garbage collection, thread scheduling) and network delays. Also we always measured the timeout for the distinct route lengths separately. For instance we measured the timeout for route length 5 then started the whole route discovery process from scratch and measured the timeout for route length 6. Thus, due to the variations it can happen that the minimal timeout for a route with more hops is lower than for a route with less hops.

6.5.3 Route Recovery

If a route error occurs, another route has to be found. Ariadne needs more time to find a new route than DSR, because much more information has to be sent and timeouts have to be considered. The round trip time, and therefore the time synchronization, is an important factor. The smaller the round trip time, the smaller the time synchronization error and the discovery timeout can be chosen. In that way, peers along the route have to wait less until they can add their key to the reply and forward it to the initiator. DSR needs on average 26 ms to discover a new route if the current one is broken. Ariadne (with a round trip time of 20 ms) needs, an average of 130 ms, which is five times more.

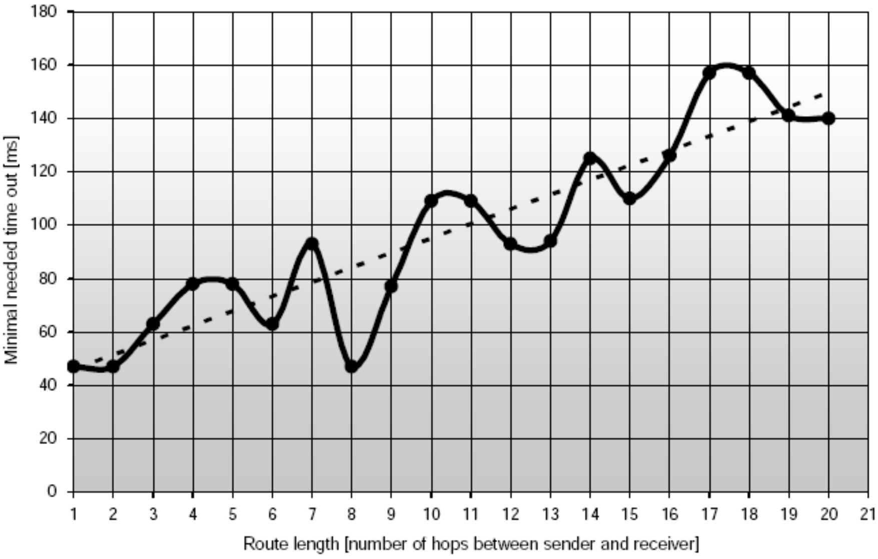


Figure 6.12: Minimal needed time out for a Route Discovery without bootstrapping

6.6 Discussion

Ariadne can provide secure routing using different mechanisms, including a lightweight asymmetric algorithm called TESLA. Although the efficiency of applying asymmetric cryptography to the messages at the forwarding or routing nodes using TESLA is high the source and destination must exchange several initialization messages which have to be protected by common asymmetric cryptography such as RSA or ECDSA. In addition pair-wise shared secret keys must be established between the source and the destination in the initialization phase. Thereafter, Ariadne with TESLA can be used with little performance overhead to protect the communication between the source and the sender. However, the delayed key disclosure introduces additional latency which depends on the network diameter, the physical communication latency and on the synchronization process. This additional latency can be kept relatively low if Ariadne is applied to very small networks where the nodes are physically nearby. We found out that Ariadne works well under the assumption that only few nodes communicate with each other, meaning that only some nodes are actively exchanging data whereas the others only function as forwarding and routing nodes. Unfortunately, these assumptions do not hold in the case of overlay networks which consist of several hundred to millions of nodes which are distributed all over the world and where usually all nodes will take part in the communication process. Thus, we decided to look for an alternative secure routing algorithm which lends some ideas applicable to overlay networks from Ariadne and integrates it to an secure routing algorithm which is based on DSR and provides scalable and dynamic routing for such networks.

6.7 Scalable Secure Routing

In the section on Ariadne we have discovered that although it provides reasonable security and incorporated unique ideas on how to provide asymmetric cryptography it is still not adequate in our scenario. Foremost, the reliance on one specific security measure either TESLA, digital signatures or pair-wise shared secrets is to inflexible for heterogenous environments since they can not be changed during the runtime and are not appropriate for a wide range of devices. On the other hand, the assumptions and requirements that many required security measures must be at place before the systems can be used, is also not applicable for dynamic overlay networks.

To address the lack of solutions for security problems in unstructured overlays, we have designed and implemented a comprehensive framework for secure unstructured overlays. We believe that solely providing solutions for particular security problems is not sufficient for realizing a secure system. In this section we outline the design and implementation of a scalable secure routing protocol for unstructured heterogeneous overlays based on this security concept. The secure routing protocol provides protection from outsider attacks and enables the detection as well as the prevention of insider attacks.

The main focus in this section will be on an adequate secure routing protocol which lends some ideas from Ariadne and integrates our security concept in order to allow for a scalable secure routing. But since authentication is a precondition for our secure routing protocol we will also mention the parts which are related to the join process in order get a better understanding of our solution as a whole. The join protocol and all other protocols which are part of the secure overlay framework are also based on that security concept. The secure overlay framework integrates all secure protocols into a comprehensive solution for secure overlay networking. The secure overlay framework is open source and available from sourceforge[1]. We will also discuss the advantages and differences of our solution to existing routing protocols for overlay networks. In the end we evaluate and present the performance of our secure routing protocol and discuss our solution.

In this chapter we do not address the issue of data distribution or efficient content management but rather how routes between nodes can be secured. Depending on the application, the routing protocol and the framework for secure unstructured overlays can be used as either a multi-hop communication network or content distribution network.

6.7.1 Network assumptions

We assume that network links are bidirectional; that is, if node A is able to send messages directly to node B, then B is also able to send messages directly to A. We also assume that nodes may belong to networks whose addresses require to

[1] `http://sourceforge.net/projects/secureoverlay/`

use NAT to connect to other nodes in the system. Thus, the internal address may be different from the external address seen by other nodes. For instance, consider the scenario depicted in Figure 6.13. The nodes *H,I,J,K* and *L,M,N* could be behind a NAT or Firewall. They have established connections to nodes *D* and *A*. In case of a NAT or Firewall nodes *D* and *A* can communicate to the nodes behind the NAT or Firewall using the outgoing connection from these nodes but may not be able to initiate a separate connection to them. Thus, all the other nodes from the overlay network can communicate with the nodes behind the NAT or Firewall using the relay nodes *D* and *A*. Thus, the secure overlay network may get separated into several parts due to the unavailability of nodes, such as *D*, *A* and even *H*, which connect the separated overlays. But if these nodes are available again the separated overlays must be able to merge into a common overlay.

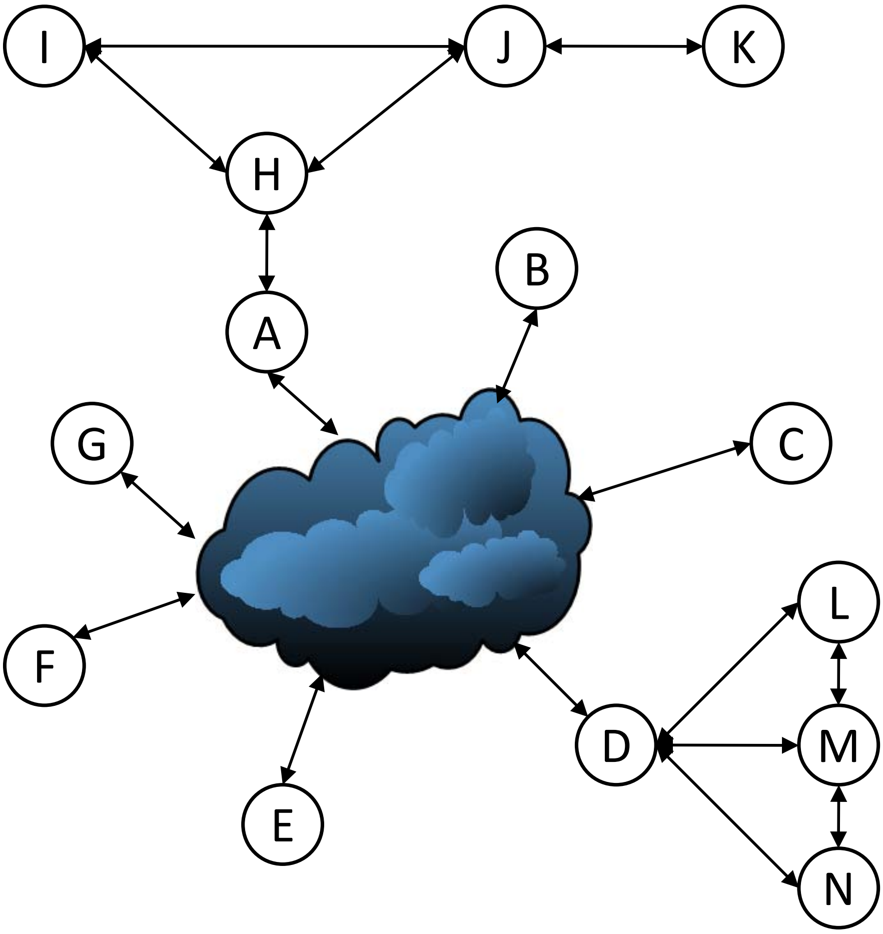


Figure 6.13: Real network topology with Internet and LAN nodes

6.7.2 Node assumptions

The resources of different overlay network nodes may vary greatly, from nodes with very little computational resources to powerful resource-rich nodes equivalent in functionality to high-performance workstations. To make our work and results as general as possible, we have designed a security concept and a routing protocol to support nodes with few resources, such as PDAs and mobile phones.

6.7.3 Security Assumptions and Bootstrapping

The security of our routing protocol relies on the secrecy and authenticity of keys stored in nodes. Depending on which security level is used by the node we rely that the specific keys are available for the execution of the secure routing protocol and we therefore refer to Chapter 3 for more detailed information about the security assumptions and the bootstrapping process.

6.8 Secure Routing Protocol

In this section we introduce our secure routing protocol for heterogeneous unstructured networks. We assume a very heterogeneous environment in terms of node resources and networking technology. Thus, the routing protocol must also be designed to function in such an environment. We have studied various routing protocols which are used in similar environments. Since, we work with unstructured overlay networks we concentrated our search on flooding-style routing protocols. We have chosen the Dynamic Source Routing (DSR) protocol [JM96] as basis for our secure routing protocol. This was due to the fact that the complete path from the source to the destination is contained in each message the secure route can be verified at each node. DSR is also a lightweight protocol in terms of network utilization and resource requirements. For a more elaborate discussion about related routing protocols see 6.12.

6.8.1 Architecture

Our routing protocol can also be divided into the discovery and maintenance phases like the DSR protocol [JM96] we used as foundation.

- *Route discovery* - This phase is triggered if a node S sends a message with a destination information D and the node S does not have a route already available for the destination information D.

- *Route maintenance* - This process is activated if a node S sends a message with destination information D using an already established route and during the forwarding process a link on that path is not available anymore. The source of the broken route is informed of this problem and uses another known route for destination information D or invokes the discovery process.

Both processes operate in an on-demand manner. Meaning that routes are established as needed and removed if they are failing. Thus, no periodic messages are required to guarantee the functionality of this protocol.

The routing protocol also incorporates the ideas developed in the security concept. Thus, its security can be adjusted through the selection of different security levels. For the sake of simplicity we use three security levels *low*, *medium* and *high*, as described before, in our demonstration.

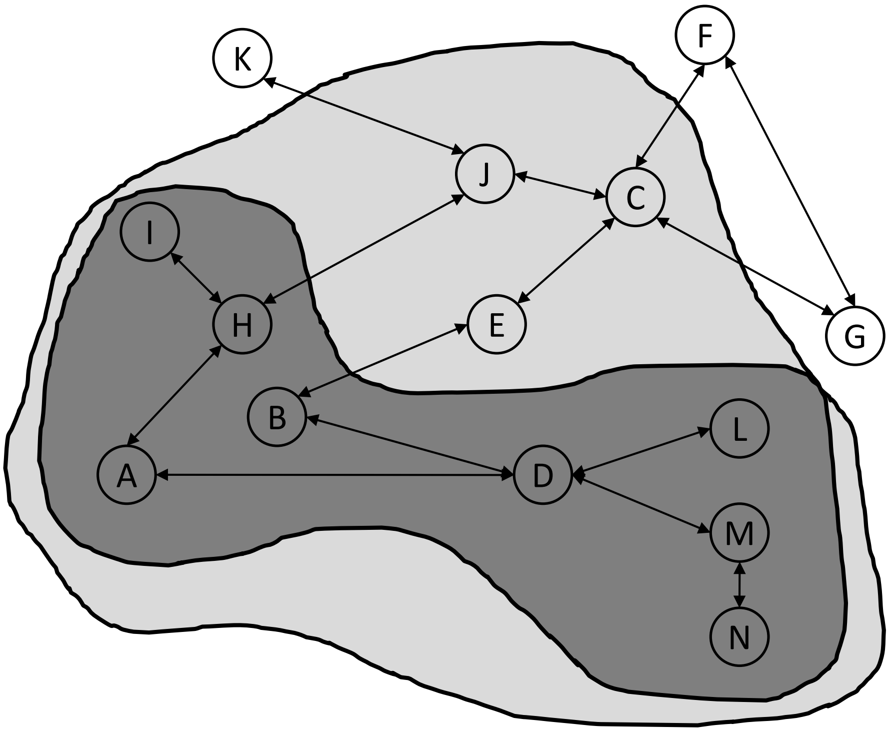


Figure 6.14: Overlay network topology with groups of different security levels

In figure 6.14 the nodes from the real network topology 6.13 are grouped into different security levels. The peers in the light grey area belong to security level *medium*. The peers in the dark grey area belong to security level *high*. All peers outside these areas belong to security level *low*. Our system has been designed in such a way that peers with higher security levels can also support lower security levels if they are provided with the required credentials for their operation. This figure illustrates the fact that nodes can be grouped into arbitrary groups with the same security requirements independent from real network structure and location.

6.9 Implementation

We have implemented the secure routing protocol in an open source secure peer-to-peer framework called SePP. The framework provides all the basic capabilities for overlay networking in heterogenous multi-hop environments. The framework has built in methods for joining a overlay network with different available authentication mechanisms. Furthermore, it provides for communication confidentiality and integrity using symmetric or asymmetric cryptography. It has a modular architecture which allows developers to implement their own protocols for the various task in overlay networking including routing, message transport, authentication, neighborhood management.

Since it already provided the means for joining a overlay network securely, key establishment between the peers inside the network as well as using separate routing protocols it was the perfect platform for implementing and testing our secure routing protocol. We assume that the different keys used inside the secure routing protocol are either already available through the security service module of SePP or can be established using existing functionality. The security service module can obtain the keys from either a keystore, smartcards or available trusted computing hardware.

6.9.1 Route Discovery

Depending on the security level different functions of the secure routing protocol are used or enabled. In security level *low*, where no cryptographic protection is applied, the secure routing protocol provides the same functionality as the dynamic source routing (DSR) protocol. For a detailed description of the DSR protocol please refer to [JM96]. In this section we describe the operations of the secure routing protocol in the security levels *medium* and *high* which are different to the operations of plain DSR. We will sketch an exemplary route discovery process using the network configuration of figure 6.14 with the nodes H, A, and D providing the shortest path from node I to node L.

Route Request (Medium)

In figure 6.15 the message sequence and actions of the scalable secure routing protocol for security level *medium* are outlined. In security level medium the security of the protocol relies on a pre-shared secret key and a session key SK shared between all the authenticated nodes of the overlay network. This session key is obtained by each node after it has joined the overlay network successfully.

The source generates a route request containing source address I and destination information L, message id and a time-stamp t_a. The source peer calculates a message authentication code h_0 over the request using the current network session key SK and broadcasts the request with the appended MAC to its neighbors.

Each neighbor checks if it has seen this request already and processes it if unseen. Otherwise, it is discarded. Each peer also verifies that the request

$$\begin{aligned}
&I : h_0 = MAC_{SK}(I, L, id, t_a)\\
&I \to *\ \ (I, L, id, t_a, h_0)\\
&H : h_1 = MAC_{SK}(I, L, id, t_a, H)\\
&H \to *\ \ (I, L, id, t_a, H, h_1)\\
&A : h_2 = MAC_{SK}(I, L, id, t_a, H, A)\\
&A \to *\ \ (I, L, id, t_a, H, A, h_2)\\
&D : h_3 = MAC_{SK}(I, L, id, t_a, H, A, D)\\
&D \to *\ \ (I, L, id, t_a, S_{PS}, H, A, D, h_3)\\
&L : h_r = MAC_{SK}(L, I, id, t_a, H, A, D)\\
&L \to D\ \ (L, I, id, t_a, H, A, D, h_r)\\
&\ldots\\
&A \to S\ \ (L, I, id, t_a, H, A, D, h_r)
\end{aligned}$$

Figure 6.15: Scalable-secure routing protocol with security level medium

is fresh and is not replayed. This is done by verifying that the current time lies inside the interval of the time of t_a plus the request buffer timeout. The request buffer timeout should be larger than it takes for the longest route to establish. The timeout ensures that only the shortest (or which is almost always the same fastest) routes are used. For instance, if a request reaches a node which has already forwarded a request the node should discard it since the already forwarded request is shorter or faster. After the timeout has elapsed the expired entries in the request buffer are deleted. Thereafter, the node verifies the correctness of the message and appends its address to the hop list if the request is valid. Otherwise the request is discarded. The node calculates the MAC using the session key over the extended request and again broadcasts the request with the appended MAC to its neighbors.

As the request reaches the destination it is again verified and if bidirectional connections are available the reverse route is added to the route cache. If bidirectional connections are not available the destination itself performs the same route discovery process but appends the reply to the request. Therefore, each peer has also to check if a route request contains a reply and performs the actions as described in the next section for that reply also.

Route Reply (Medium)

The destination creates a reply using the hop list contained in the request as route. The destination returns the reply with the appended MAC on the reverse path it has traveled to the destination. Each peer on the reverse path verifies the reply and the route itself. Also the freshness of the route reply is verified at each peer. The route is verified by checking if the own peer ID is contained in the specified route and the reply has been received from the preceding peer in the route. If these requirements are met the reply is forwarded to the next hop (actually the previous hop in the real route) otherwise it is discarded. In order to gain some additional performance all peers along the route can add the established route and the reverse route (if bidirectional connections are assumed) to their local route cache.

Route Request (High)

In figure 6.16 the message sequence and actions of the secure routing protocol for security level *high* are outlined. There are only some marginal differences in the operation of the protocol between the two security levels. Thus, we only describe the part which is different from the one with security level *medium.* With security level medium it is still possible that a legitimate peer inside the network modifies the route request for its own benefit (malicious or not). Using security level high protects from insider attacks and also enables the identification of the malicious peer.

$$\begin{aligned}
&I : S_{PS} = SIG_{PrI}(I, L, id, t_a)\\
&I \rightarrow * \ \ (I, L, id, t_a, S_{PI})\\
&H : h_1 = MAC_{SK}(I, L, id, t_a, S_{PI}, H)\\
&H \rightarrow * \ \ (I, L, id, t_a, S_{PI}, H, h_1)\\
&\ldots\\
&A \rightarrow * \ \ (I, L, id, t_a, S_{PI}, H, A, D, h_3)\\
&L : S_{PD} = SIG_{PrL}(L, I, id, t_a, H, A, D)\\
&L \rightarrow D \ \ (L, I, id, t_a, H, A, D, S_{PL})\\
&\ldots\\
&H \rightarrow I \ \ (L, I, id, t_a, H, A, D, S_{PL})
\end{aligned}$$

Figure 6.16: Scalable secure routing protocol in security level high

The source S creates the digital signature SIG_{PrI} of the complete route request instead of only a MAC and sends it along with the actual route request. The signature is used to prevent forgery of the source which could be used to introduce malicious routes at the destination, if the destination uses the reverse route. Because each peer possess the session key everyone can create a request for other peers on their behalf and send them to the destination. No peer on the path would recognize such a forgery since it can only verified that the last hop in the hop list is the forwarding peer. Therefore, a malicious peer could create requests for different sources and fill the route caches of the hops along the path and the destination with fake routes.

With the use of a digital signature every peer can verify if the route request has indeed been created by the specified source. The signature also provides message integrity. The ID and the timestamp again provide freshness for the request and ensure that the same request is processed only once. As the request reaches the destination peer it is again verified and if bidirectional connections are available the reverse route is added to the route cache.

Route Reply (High)

The destination L creates a reply using the hop list contained in the route request as route if the request is valid. The route selected by the destination L is signed using the digital signature SIG_{PrD}. The destination also calculates the MAC using the session key SK over the complete reply including the signed route. The peer then returns the reply with the signed route and the MAC using the reverse path.

Each peer verifies the reply, the signature of the route and the route itself. The route is verified by checking if the own node ID is contained in the signed route and the reply has been received from the preceding node in the route and the signature of the route is valid. This is done using the public key of the route destination. Thus, it is verified that the route has been selected by the destination peer and it has not been modified. If these requirements are met the reply is forwarded otherwise it is discarded. The source verifies the security of the route reply the same way as the intermediate hops and adds it to its route cache if it is correct. Thus, we can prevent forgery of the route request by some malicious node since this would be detected during these checks. For instance if some node removed some hops from the route the destination would have signed this forged route but it would not be returned to the source since the nodes which have been removed would fail to verify the route.

Thereafter, the message which triggered the route discovery process can be sent using the established route. Each message sent to a non-neighbor peer carries the route over which this message should be sent. Each peer on the route now verifies at reception of such a message if the contained route is already stored in the route cache and has therefore been established in a secure manner. If the route is contained the messages is forwarded to the next hop in the route until it reaches the destination.

6.9.2 Route Maintenance

The route maintenance mechanism is triggered when forwarding a message with a source route where the next hop is not available anymore. Each peer on the route is responsible for ensuring that the message has been received by the next hop. A peer tries to retransmit such a packet for a specific number of attempts. In our case we did not use retransmission because we rely on the TCP protocol which provides this behavior inherently. If we would use UDP we would need to include such a mechanism. If a message can not be forwarded the responsible peer creates a route error and returns it using the reverse path to the source of the message. Again, if bidirectional connections are not available the peer also performs a route discovery to the source of the message and every peer has to check every route request if it contains a route error and must perform the actions described in the next sections.

Route Error (Medium)

In figure 6.17 we outline the sequence of messages triggered by a route error. The responsible peer (initiator) creates a route error message which contains the failing hop H , its own identity I, a message ID and a time-stamp. The initiator peer calculates a message authentication code h_0 over the request using the current network session key S_K and forwards the route error with the appended MAC to the previous hop in the route to the source of the message.

$$\begin{aligned}
&I : h_0 = MAC_{SK}(Error, I, S, id, t_a)\\
&I \rightarrow I_{-1} \ \ (Error, I, S, id, t_a, h_0)\\
&\dots\\
&I_{-n} \rightarrow S \ \ (Error, I, S, id, t_a, h_0)
\end{aligned}$$

Figure 6.17: Secure routing protocol route error message with security level medium

Each peer verifies the authenticity of the route error and updates its route cache if valid. The peer then again forwards the route error towards the source of the original message.

As the route error reaches the source of the original message it is verified again. If the source has not another route to the destination of the original message available in its route cache, a new route discovery process for that destination is started.

Route Error (High)

Figure 6.18 shows the messages which are sent if a route error occurs in security level *high*. The responsible peer (initiator) creates a route error message which contains the failing hop H , its own identity I, a message ID and a time-stamp. The initiator peer calculates a message authentication code h_0 over the request using the current network session key S_K and forwards the route error with the appended MAC to the previous hop in the route to the source of the message.

Each peer verifies the authenticity of the route error and updates its route cache if valid. The peer then again forwards the route error towards the source of the original message.

As the route error reaches the source of the original message it is verified again. If the source has not another route to the destination of the original message available in its route cache, a new route discovery process for that destination is started.

$$
\begin{aligned}
&I : S_{PI} = SIG_{PrI}(Error, I, S, id, t_a)\\
&I \rightarrow I_{-1} \ \ (Error, I, S, id, t_a, S_{PI})\\
&\ldots\\
&I_{-n} \rightarrow S \ \ (Error, I, S, id, t_a, S_{PI})
\end{aligned}
$$

Figure 6.18: Secure routing protocol route error message with security level high

6.10 Security analysis

The security levels and the secure routing protocol are designed in such a manner that for *low* everybody can participate and no cryptographic protection is applied. Thus, no guarantees on the security of the system can be given. There is no protection against the attacks mentioned in Section 6.3.1.

In security level *medium* shared secret keys are used for node authentication. Therefore, the secure routing protocol which enforces authentication based on these secret keys provides protection from outsider attacks. In this security level we are protected from all the outsider attacks but insider attacks are still possible.

In security level *high* digital signatures are used to secure the established routes. Thus, also insider attacks from non-collaborating peers can be prevented. Non-collaboration means that the peer does not control any other peer on a path from the source to the destination. If peers collaborate they can always perform a Wormhole attack by tunneling the route request from one peer to the other and effectively shorten the route length. This attack only works if the controlled path is also the fastest. Otherwise the request would arrive to late to be selected from the destination as new route to the source. For instance, peer D is the source and peer C is the peer who meets the destination information requirements. Now if peer B and peer J of figure 6.14 collaborate and the tunneled request would arrive earlier than the other one traveling over A and E, this attack still succeeds. Such attacks can currently only be mitigated if location information and synchronized clocks are used [HPJ03, PL07].

6.11 Performance evaluation

The following results have been obtained from the implementation of our protocol in the open source overlay framework SePP. Several peer instances have been created which have then formed the overlay system. For the join process of the overlay network we used the SePP framework in the three secure routing protocol security levels *low*, *medium*, and *high*. After the join process every peer possess a session key which is shared amongst all peers in the network. The session key is then also used inside the routing protocol in security level *medium* and *high*.

We have created and tested our implementation in several network scenarios with a varying number of peers. We ensured that different path lengths between peers which send messages exist. Peers or links are also removed randomly between different peers to force the routing protocol to perform route maintenance and establish new routes if necessary. Each peer has a pre-configured list of possible overlay peers which it loads at startup. These peers are contacted and the peer joins the existing overlay network. The overall member list is learned thereafter and messaging between the different members is then initiated. We have measured the time it takes a peer to discover routes to other members in that network depending on the specified security level.

The tests have been performed on HP personal computers with Intel Core Duo E8600 processors with 3.33 GHZ and 4 GB RAM and Windows 7 as operating system. The SePP framework and the secure routing protocol implementation have been executed on Java JDK 6 Update 17 runtime environments. The measurements were taken using the Java inherent `System.nanoTime()` method. This method provides nano-second accurate timings if performed on SE runtime environments. For each security level we have performed 100 consecutive runs where we established routes of lengths from 1 to 8. We therefore created a overlay network with 10 nodes which are connected together in series. Meaning, peer 1 is connected with peer 2, peer 2 is connected with peer 3 and so on. From these measurements we calculated the mean time it took to establish a route. We measured the times for the route request and reply separately in order see if there is a difference.

	Source [ms]	Hop(s) [ms]	Destination [ms]
Low Request	0.5239	0.7543	2.4096
Low Reply	0.8527	0.7238	0.3213
Medium Request	0.7157	0.9204	2.7937
Medium Reply	1.0670	0.8790	0.4032
High Request	42.2055	2.1750	3.6752
High Reply	2.3503	2.2261	39.3460

Table 6.1: Processing time of the different security levels at the participating peers

In Table 6.1 the processing times of the routing protocol with different security levels are given. We have measured the time it took the specific peers - the route initiator *Source*, the peers along the path *Intermediate Hop* and the route endpoint *Destination* - to process a route message. At the source we measured the time when the `getRoute` method was called and the route request was sent to the neighbors. At the intermediate hops we measured the time from the arrival to the forwarding of the route request to its neighbors or to the next hop in case of the reply. At the destination we measured the time from arrival of the request to the point before the reply is created in case of the request. And for the reply from the end of the request measurement until the reply was sent.

At the initiator we measured again the time from arrival of the route response until the route has been added to the route cache.

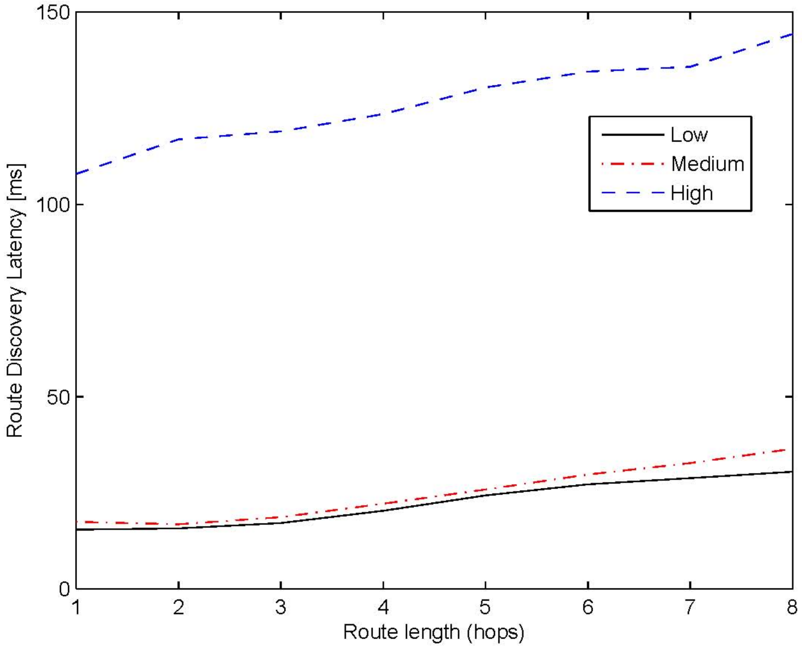


Figure 6.19: Route discovery latency if no route is available

In Figure 6.19 the route discovery latency over different route lengths is shown if no route is available. The discovery latency in security level *low* and *medium* are only slightly different. The difference relates only to the usage of symmetric cryptography, in particular the keyed message authentication function using the shared session key. It shows that the use of symmetric cryptography only introduces a low slightly growing overhead to the route discovery process. Although the additional processing time is almost negligible, it becomes relevant only with higher route lengths which occur seldom, it provides protection from outsider attacks since a the correct key is required to produce and verify the route requests and replies. In case of security level *high* the discovery latency has increased more significantly, about 5 times more than in the unprotected case. This increase is almost completely related with the signature creation process at the source and the destination.

In this experiments we have only measured the processing latencies which are introduced through the application of cryptographic calculations. We did not take into account network latencies since they are not relevant for the protocol performance.

The verification of the signature at the intermediate hops only requires 1/20 of the signature creation time. But nevertheless the use of these signatures protects the discovery process from malicious behavior of non-collaborating peers. This means that if any single peer decides to not perform the protocol as intended they will not gain any advantage over peers which perform the protocol as intended.

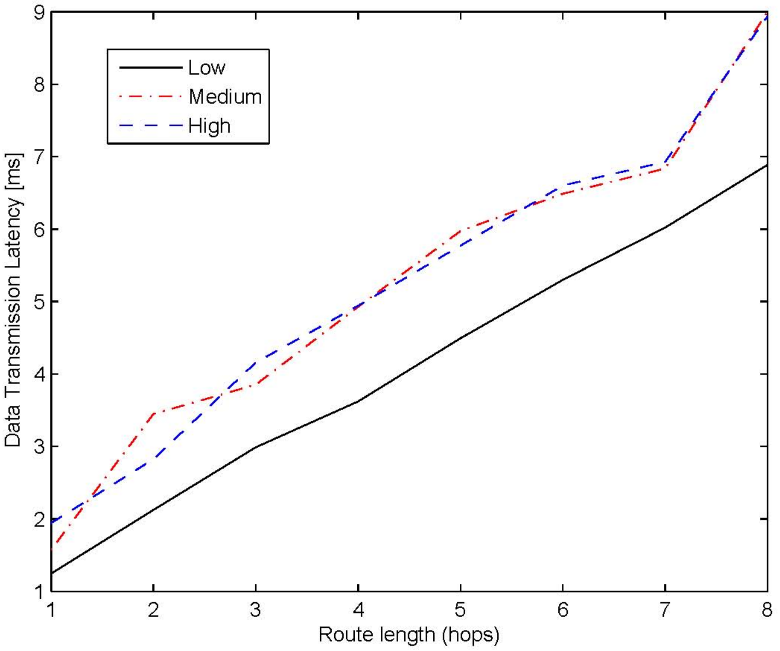


Figure 6.20: Data transmission latency if route is available

In figure 6.20 the data transmission latency over different route lengths is shown if the route has already been established. If a route has already been established the data transmission latency is only influenced by the processing time of the message at each peer. In case of security level *low* only the route contained in the message is checked that it has already been established. In security level *medium* and *high* the route itself is verified but also the message authentication code of the messages is validated. If all checks have been performed successfully the message is forwarded to the destination via the next hop from the route. Therefore, the latencies vary only between security level *low* and *medium* respectively *high* because the later two use the same data security level.

6.12 Related Work

We have investigated different secure ad-hoc routing protocols since no secure routing protocol for unstructured overlay networks existed so far and they have similar requirements. Studies such as [ZH99, PH02, Zap02, JLR04, WZH05, KGSW05] have applied standard cryptographic mechanisms to well-known routing algorithms or provide theoretical *best practices* for establishing routes in a secure manner. Unfortunately, they have not integrated features such as scalability in order to allow users to adjust the security level of the resulting protocol to fit their needs or to integrate heterogenous devices. Most of the protocols solely rely on digital signatures without taking into account resource constrained devices so that they use the cryptographic mechanisms in a redundant and lavish manner. For instance, they apply digital signatures to every request at every hop which requires verification and creation of signatures at each hop in the whole network. In our solution if we use security *high* only the source and the destination have to create a digital signature and only the nodes on the return path have to verify the signature. We also provide a less resource intensive security level for the case that very constrained devices, such as wireless sensor nodes, are present in the network and require some protection.

Another protocol, called Ariadne [HPJ05], which is based on TESLA [PCTS02], is able to provide *cheaper* asymmetric cryptography by means of delayed disclosure of symmetric keys. This enables also very constrained devices to participate in the secure route discovery process. However, this protocol lacks in the specification and adequate guidance of selecting the required parameters. Additionally, in the bootstrapping and security update processes this protocol requires common asymmetric cryptography (digital signatures and public key certificates) which has not been taken into account in their performance evaluation. Also they assume use-cases which are not realistic in overlays. For instance, they assume that only very few peers will actually communicate with each other and thus would have to perform the more expensive cryptographic functions. All other peers are only required to forward the requests. As shown in [KP09a] the use of Ariadne in overlay networks is not practical because we can not assume that only some specific peers will exchange data. If all peers should be able to communicate with each other than each peer not only requires symmetric keys also public-private keys and pair-wise shared secrets. Thus, in an environment with random unbound interactions such a scheme is not applicable.

Another secure routing approach for ad-hoc routing mechanism has been provided by [GZ02, GZA02]. They provide security for the ad-hoc on-demand distance vector protocol. This protocol has several features in common with the dynamic source routing protocol, such as the separation in two phases (discovery and maintenance). The main problem is that it does not provide verifiable secure routes. This is because each peer only maintains its local routing view by only maintaining a forward and backward hop for a specific destination. Thus, it is not possible to give the source and destination control over used route and therefore prevent insider attacks.

Others like [MGLB00, AD01, LS06] have tried to provide security with means of repudiation or some other kind of trust. But all these approaches rely on statistical methods and not on mathematical primitives. Statistical methods can provide means to increase security if cryptographic measures are already taken. But these mechanisms can not provide security if the worst case adversaries are assumed. Because they rely on some kind of historic data or on other heuristics they can be betrayed by playing along for most of the time but deviating from the protocol if it is in the interest of the adversary.

6.13 Conclusion

In this chapter we have investigated different routing protocols and evaluated the applicability of one specific secure routing protocol called Ariadne. Since the existing protocols are not efficient and flexible enough for our scenario, we presented a secure routing algorithm for pure overlays which provides a scalable support for security. This scalable security is based on a security concept which separates between the admission and the communication process in the overlay system or more precisely in a overlay group. We have used the same group notion for the actual routing part and the subsequent data transmission part. In addition different protection mechanisms can be used to increase the selected security levels. Since in our protocol the security can be adjusted to the requirements of the system we achieve a very good security/performance ratio as shown by the results. We have shown that the performance of the secure routing protocol is adequate even if more expensive cryptographic operations are used. The experimental results have been obtained using an open source secure overlay networking framework where we have implemented our secure routing protocol.

This work is part of a comprehensive solution for heterogenous environments with multi-hop characteristic. None of the existing approaches provides a solution which allows for adjustable security. By using this approach we can either require that all devices be at least powerful enough to apply the highest possible security level or we find the least common denominator. We have also tried to find a middle way between these extremes by allowing devices with different security levels to take part in the same overlay network. Therefore, the routes can only be established between nodes with the same security level. But since the requirements for the lower security levels are much less demanding than for the higher security levels we assume that nodes posses also the credentials for the lower security levels. This means that if a node can perform security level high it can also perform security level medium and low since it requires only to store some additional keys which are much less complex in terms of computational requirements and management overhead. This is necessary to allow more restricted devices to route over more powerful nodes also.

7

Secure P2P Framework

A cloud is made of billows upon billows upon billows that look like clouds. As you come closer to a cloud you do not get something smooth, but irregularities at a smaller scale.

Benoit Mandelbrot

7.1 Introduction

To address the lack of solutions for security problems in unstructured overlays, we have designed and implemented a comprehensive framework for secure unstructured overlays, which we called SePP. We believe that solely providing solutions for particular security problems is not sufficient for realizing a secure system. In Chapter 4, we proposed a security concept [KPT08] which allows for selecting adequate security measures based on the overall system requirements and the available resources of the participating nodes. We have designed and implemented a secure routing protocol for unstructured heterogeneous overlays based on this security concept. The secure routing protocol provides protection from outsider attacks and enables the detection as well as the prevention of insider attacks. We have integrated this secure routing protocol into a comprehensive framework for secure unstructured overlay networking. We want to mention that our system is not intended for open Internet-scale systems but rather for controllable environments such as in private or public enterprises. To our knowledge this is the first framework which provides security for unstructured overlay networks for all parts from joining over routing to communication.

In this chapter we provide a functional design and architecture for a framework for secure unstructured overlay networking. Therefore, we first establish some requirements inherent to overlay networks we want to secure with our framework. We deduce some assumptions in accordance to the requirements and the specifics of cryptographic primitives. A security concept based on the requirements and assumptions which provides security for different scenarios is specified further on. After having laid down the theoretical foundation for secure overlay networks we propose a possible framework architecture. We also provide some information about protocols and mechanisms which can provide the required functions and security features. The framework is freely available from sourceforge.

7.2 Assumptions

In this section we outline some of the assumptions we made for the design of our secure overlay framework. The assumptions as outlined in Chapter 3 are also appropriate for the secure overlay network framework itself. These assumptions where:

- Network - We assumed bidirectional links and allowed for nodes to be behind network translation or a firewall.
- Node - We assumed that we have a network with heterogeneous devices in terms of resources.
- Security - We assumed that either keys are setup before the system operation or we can use cryptographic material from TPMs to establish the required keys in each node.

In addition, we want to note that the secure overlay network framework can be used in various communication areas such as multi-hop communication networks or content distribution networks. Usually the differences are only at the application layer, meaning that after the network has formed itself and nodes are able to communicate the kind of operations performed on top of that are only subject to the developers imagination.

7.3 Framework Design

We will first introduce the characteristics of overlay networks in general and derive from them specific requirements for the design of a framework which provides security for such networks. These requirements relate mainly to the duties of each node participating in the network and to the rules of interaction between them. Thereafter, we will make some assumptions which are necessary to realize such a framework. The assumptions stem mainly from the security requirements and the means of their implementation.

The security concept provides a simple way to select adequate security measures to achieve a specific security level for the whole network consisting of heterogeneous nodes with different capabilities. The requirements, assumptions and the security concept determine a specific architecture of a framework for secure overlay networks. The architecture again imposes some constraints on the network and security protocols which actually implement the secure overlay network. The network and security protocols are the cornerstone of every framework providing building blocks for secure peer-to-peer networks and will therefore be subject to intensive analysis and evaluation which we started with this work.

7.3.1 Architecture

We have developed this middleware to provide means to implement, analyze and verify different techniques and mechanisms for providing security to overlay networks. Thus, it was a strict requirement to design the framework architecture in a modular manner in order to switch between the different security methods without requiring the other components to be modified.

An overview of the components of the framework is shown in Figure 7.1. All the components in the figure which are on the left and the right of the Secure Self-Organizing Network Management component are designed as modules and can be replaced by other implementations simply through the specification in the node configuration. This framework and its components run on every node. The components of which the middleware is composed are described subsequently.

Secure peer-to-peer Network Management

This component is the heart of the framework, coordinates the flow of execution and manages the interaction between all the other components. Thus, the system functionality of one node and the behavior of the whole network is controlled through this component. This component implements only very few specific mechanisms but it manages the interfaces to the specific implementations and calls the required methods on demand. Since also the behavior of the whole network is controlled by this component most of the distributed algorithms and mechanisms are implemented here or at least the scheduling of the distributed algorithms is managed. One of the most important features is the processing of messages. Since this component receives messages in two directions we must take care of the message order and also that no deadlocks or starvation issues arise due to that fact.

IO Subsystem

The IO Subsystem consists of several components, the IOHandler, Message Processor, Message Receiver(s) and Message Sender. These components are imple-

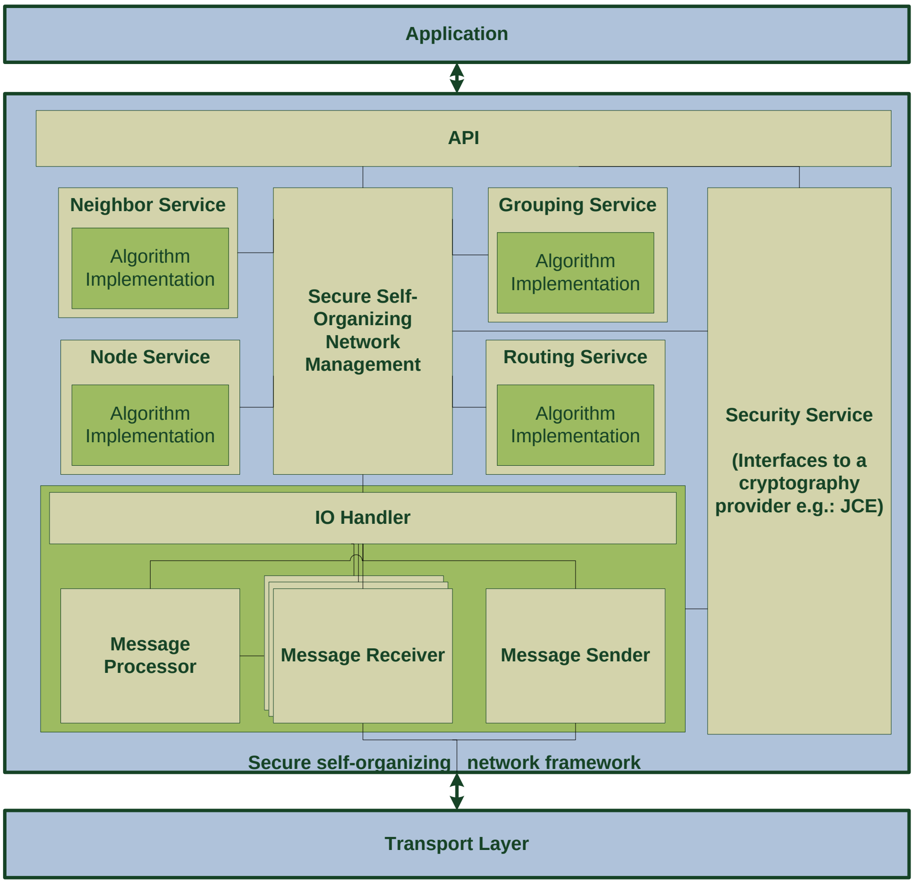


Figure 7.1: Middleware architecture

mented according to the requirements of the devices on which the nodes are hosted. For instance, we have implementations for JavaSE for standard IO and non-blocking IO, JavaME for standard IO. These components handle the reception of data from the network interfaces. Since messages usually are expressed in XML and are often very big, it would decrease the efficiency if we would send them as they are using the socket API, as we have experienced in earlier versions of our framework. Thus, we have implemented a scheduling and buffering mechanism for overlay network messages. Although, we also have a fragmentation mechanism in place in the secure overlay network management, which splits big chunks of application data or reads chunks of data from application data streams into overlay network messages, it is still beneficial to further fragment these messages into data frames on the network layer. This mechanism splits the overlay network messages into data chunks which are then sent with a small trailer to the destination.

Routing Service

The Routing Service performs the task of obtaining and maintaining paths to other nodes. Several interfaces between the secure peer-to-peer network management and this component exist since every time a message must be sent this component is consulted. Also if the sending or receiving process failed for some reason this component is called to establish another route with the remote node. The actual algorithm which is used for establishing and maintaining routes can be changed through the node configuration.

Neighbor Service

This component is responsible for finding and maintaining neighbors for the local node. Finding neighbors depends on the underlying network structure(wired or wireless) this component generates an abstract view of the network topology and selects nodes from the whole network to become virtual neighbors of the local node. This component also interacts with the routing service since it can be used to reduce the length of routing paths. For that it selects members of the network and tries if they can be contacted directly. If they can be reached they are added to the neighbor list and thus reduce the maximum routing path length. Depending on the members which are selected the overall mean routing path length can be reduced tremendously as initial indicated by the seminal work of Stanley Milgram [Mil67] and applied to networks by Watts and Strogatz [WS98].

Grouping Service

The Grouping Service creates, maintains and controls groups, group members and their interaction. This grouping service is used on top of an existing default group which is established during the process of joining the network itself. It provides authenticated and confidential interaction between the members of specific groups. This specific implementation of the group service can be specified through the configuration and can therefore be changed. The group service must also protect the privacy of group members since this information could be misused. For instance, if the operations for finding and joining groups would also allow all nodes to obtain the current members of the group this could be used for subsequent attacks on specific nodes and thus trying to get unauthorized access to such an group. Therefore, we have designed and implemented mechanisms which prevent such an attack. If a particular node searches for specific groups it obtains only information which can be made public without any risk. This information consists of the required security level of the group, the name of the group, a public description of the group and one particular contact point for this group. It would also be possible to implement it without any specific contact point but then the group join message would need to be broadcasted in the whole overlay network. Only after having successfully joined this group the nodes obtain the group member list, which is protected by the group session key.

Node Service

The Node Service performs tasks which are related to the local node execution. It handles the node configuration and other relevant local tasks which are not strictly required for the execution of secure peer-to-peer networks. For instance, this service records statistical data about the data transfer rates, bandwidth, message losses and also performs some kind of node misbehavior detection. The possible applications are manifold but we have not specified or separated them more precisely and therefore use currently this service for all the tasks which are not related with any other service. For instance, this service tries to find misbehaving nodes using the techniques as described by Marti et al. [MGLB00]. Although, our system provides security from outsiders and allows in the highest security level to detect also insider attacks if the malicious nodes do not collaborate and form a wormhole, it is still possible to downgrade the performance of our system through selfish behavior, such as selectively forwarding messages or dropping requests which require too much resources. Thus, with the Pathrater and Watchdog mechanisms, which are explained in Section 7.4.5, it is also possible to detect potential selfish nodes which do not adhere to the general overlay principles such as fully participating in all overlay functions.

Security Service

This component provides the required cryptographic primitives and security functions like encryption, decryption, hashing, secure random number generator, message authentication codes, digital signatures, certificate management, keystore manipulation and other related mechanisms. This service can be accessed on several levels. The different services access it through the secure peer-to-peer network management component. Application designers can use it through the API, although not all of the functions are available. It can also be used by the IO subsystem if necessary although usually the IO subsystem will also access it through the secure peer-to-peer network management. All the functions can either be used through the standard Java JCE implementation or also through the IAIK JCE or IAIK JCE-ME versions, depending on how much resources the device has available on which the overlay framework is executed. We have also implemented special processing mechanisms for different keystore mechanisms, not only all the different keystore formats supported by SUN and IAIK are implemented but also an additional concept called secure docking module (SDM). The SDM has been developed as part of the Secricom EU-project which allows access to keys in the keystore not only through passwords or other common authentication means but also combines it with trusted computing mechanisms. For instance, it is possible to create these hardware keystore and configure it so that specific keys for particular platforms are only released if the local platform, to which the SDM is attached, is in a certain trusted state. Thus, we can ensure that the operating system and the secure overlay network framework have not been tampered with and it is secure to release the key to the platform or the framework, respectively. For more information about this process and possible application scenarios please see [HTK10, DHK10].

API

The API provides interfaces for application designers as the name already suggests. The relevant functions like *init, join, leave, send, receive, broadcast, getMembers, getNeighbors, addNeighbor, findNode, getGroups, createGroup, joinGroup, leaveGroup*, and so on are either realized as common methods or must be implemented as callback functions as part of the application. The API provides different methods for sending and broadcasting data depending on the size and type of data to be transmitted. If only simple messages, which do not exceed some specific amount of bytes, should be sent than it is possible to provide them through ordinary byte arrays. If big amounts of data or streaming data should be sent it is best to use the send/receive/broadcast functions which allow to use ByteStreams. Otherwise the system would have not been able to handle the amount of data and provide it to the user through the API since memory would be the bottleneck.

7.4 Implementation

The behavior of most of the components described in the previous section can be changed because of the modular design and through the implementation of different algorithms. In this section we will provide details about the current implementation and discuss three different protocols used inside the framework.

First we outline the initialization process which is performed for each node at startup. Thereafter we discuss three different types of protocols currently implemented in the framework. One protocol is realized directly inside the secure peer-to-peer network management, one in the routing service and one in the neighbor service. The first protocol could also be realized in the grouping service since it actually performs mutual entity authentication and is used for joining the network, which is the same process as joining a group.

7.4.1 Initialization

During the startup process of each node, distinct actions are performed which can not be counted clearly to any other service or protocol and are therefore outlined in this section.

1. Loading the specific node configuration
2. Searching for available nodes
3. Checking for available networks
4. Creating a network if no suitable is available
5. Joining of suitable network

Each node is created with an unique ID and a set of configuration details specifying the behavior of the peer-to-peer network and the characteristics of the participating nodes. This unique ID can either be obtained from the node configuration in the case of preestablished security parameters or derived from any source which guarantees uniqueness otherwise, for more information see Chapter 5. Since the ID constitutes a security property, it must be guaranteed that an ID is only assigned to one node. In case of dynamic ID assignment during the execution of the network it is also necessary to assign an ID always to the same node. Other node specific configuration details include the security level, activity intervals, some physical network parameters, secure storage location and the type of credentials. Especially the credentials are important details which are actually not stored in the configuration but must be provided during the node join process (for more information see 7.4.2). The configuration details that determine the behavior of the peer-to-peer network include the routing algorithm, cryptographic algorithms, neighbor algorithm and physical network parameters.

After a node has been created and its inherent characteristics have been established, the node searches for a suitable overlay network which it can join. In order to find a network and be able to join it the node must first find other nodes. There are several steps in this process which differ between the different peer-to-peer network implementation. For instance, in a decentralized pure P2P system a node would usually first try to find nodes in its reachable local area through a LAN broadcast. Other nodes will usually also be looked up through known addresses of nodes stored in some kind of repository. In addition, sending messages to a specific multicast group could also be part of the discovery process.

If other nodes and suitable networks have been found, the node performs the actions described in section 7.4.2. If no network could be found the node must create its own network. Therefore, the node uses the set of configuration details which describe the network behavior to create a new peer-to-peer network. The node adds himself to the list of members. The actual join process is only performed if another node joins this network.

7.4.2 Join protocol

After a node has discovered an existing peer-to-peer network and wants to join this network, a protocol must be executed which guarantees the following properties:

- mutual entity authentication,
- key agreement or key transport, and
- key authentication

There exist several algorithms and protocols which provide one or several parts of the described functionality. The join protocol is described in the context of the proposed security concept in order to guarantee the correctness of authentication process, the identity of the participating entities and the established session key.

We decided to use the Needham-Schroeder-Lowe authentication protocol as basis for our join protocol. We added an additional step to the original protocol which provides authenticated transport of an existing key between the joined and the joining node. For detailed information about this protocol see [NS78],[Low95],[Low96]. This step is added because the two nodes performing the join protocol do not derive or agree on keys but rather every joining node is provided with the session key currently used inside the network at the appropriate security level after he is authenticated.

It is not necessary to specify one specific node to be responsible for the join process without violating the security of the network since after authentication all nodes are equal and possess the current session key. Furthermore, it also improves the reliability of the network since otherwise a singe point of failure would be introduced to the system.

7.4.3 Routing protocols

Sending messages to other nodes is the most important function which a framework for peer-to-peer networks must provide. Depending on the intended overlay network type different mechanisms are needed to provide the communication functionality between the nodes. Usually nodes in a overlay network can directly send messages to nodes that are in their neighborhood. Such a neighborhood can be physically or virtually restricted. For instance nodes in the internet may only communicate with nodes with a public address or nodes in an wireless ad-hoc network can only communicate with their physically reachable neighbors. In order to be able to send messages to nodes which can not be reached directly a protocol is required which establishes routes to nodes not in their neighborhood.

The middleware is designed in such a way that different routing protocols can be implemented and used. They routing protocols can be exchanged easily through the use of the factory method pattern. We have implemented proactive and reactive routing algorithms such as DSR, AODV, OLSR, Ariadne and our own scalable secure routing algorithm. We have mainly used algorithms which were designed for mobile ad-hoc networks since they also have peer-to-peer behavior and therefore fit best for overlay networks.

Our main focus was on routing algorithms based on the dynamic source routing protocol (DSR)[JM96] and the ad-hoc on-demand distance vector routing protocol (AODV)[PBR99]. We have also implemented secure variants. The most famous one among these is called Ariadne[HPJ02] and is based on DSR. These protocols have the ability to adapt very quickly to changes in the topology. DSR also does not require periodic exchange of routing messages which further reduces complexity and increases efficiency in dynamic environments. Another property of is that they are also well suited for devices which are constrained in terms of computational power, memory and energy consumption.

We have also designed and implemented a secure routing protocol based on DSR which we call scalable secure routing (SSR). This routing protocol incorporates the ideas developed in the security concept and its security can be adjusted through the selection of different security levels. In Chapter 6 we outlined the message sequence and actions of the scalable secure routing protocol for security level high in Figure 6.16.

Depending on the security level the routing protocol achieves protection from only outsider to also insider attacks. There exist also other sequences of action for the scalable secure routing algorithm depending on the selected security level, for more information see Chapter 4.

7.4.4 Routing table

Somewhat independent from the routing protocol itself different styles of route tables are available. The most simple one is the path cache. In a path cache all routes established at a given node are stored as a whole. This means that for each destination and each route to the destination one entry exists in the path cache. Such a path cache is simple to maintain and since the whole entry is stored it allows to verify the security of the route each time the route is obtained from the table. The other possibility would be to use a link cache which has one benefit over the path cache. With a link cache it would be possible to calculate new routes to destinations even if the previously used route is broken. Instead of storing each route separately in the cache the route is analyzed and only the link or more precisely the neighborhood information is used to generate a comprehensive map of the network using the Dijkstra algorithm. Thus, if a particular link is broken in the network it is possible using the Dijkstra algorithm to find an alternative path to the same destination without initiating a new route discovery process. This algorithm can be used to obtain routes to destinations after a link has been broken and it is even possible to calculate routes to destinations without any prior route discovery process for this destination but then the security assumptions would not be fulfilled anymore. Because it would be possible to extract routes from the link cache which have not been established using the secure routing and the associated verification mechanisms and can thus not be trusted. It may be possible to obtain secure verifiable routes from the link cache but this was not the focus of our work.

7.4.5 Insider Attack / Malicious Peer Detection

The design and implementation of the secure overlay network allows only for legitimate peers to take part in the overlay network execution. Depending on the selected security level, specific requirements must be met by the nodes in order to join the network and participate in the routing and communication process. However, even if all nodes which participate in these processes are authentic and authorized, it is not guaranteed that these nodes behave correctly. Correct, non-malicious and altruistic behavior may be specified differently depending on the overlay network application. But usually this involves such behavior as:

- Forwarding of all messages with equal measures,
- Participation in all distributed overlay functions,

- Provision of network information equally to all nodes, and
- Provision of correct current node status information.

In order to realize such functionality and guarantee the desired behavior of all participating nodes, an overlay network must be able to measure the behavior of its nodes. Since overlay networks are decentralized, these functionalities and countermeasures must be implemented by each node. In addition, the resulting behavior information must be protected from modification and fabrication. Several mechanisms which are usually based on reputation systems have been proposed and implemented in different scenarios, most often in mobile ad hoc networks since they possess a unique feature, namely physical link layer acknowledgments, which allows for simpler implementation of such systems.

We have experimented with reputation-based detection and countermeasure mechanisms called Watchdog and Pathrater [MGLB00], proposed by Marti et al., since they are based on the broadcast nature of mobile ad hoc networks which is somewhat similar to the nature of unstructured overlay networks with broadcast based routing protocols. Using the Watchdog functionality it is possible to detect misbehaving nodes and identify Blackholes, Greyholes, message manipulation, and to some extend also Wormholes. Blackholes are simply data drains where the misbehaving node drops all information which is not generated by itself. A greyhole is similar but is more fine-grained in what kind of messages it drops and what it relays. Message manipulation is not a problem in our system since we use cryptographic means to detect modifications of messages. Each node verifies that the messages which sent to its neighbors are also forwarded by them. Thus, each node watches the behavior of its neighbors and depending on the behavior of each neighbor assigns or modifies the rating of it. In addition, it is also possible to rate an entire route or nodes on the route depending on the reliability of this route. This information can be incorporated into future route discovery processes and enables the source and destination to dismiss routes depending on previously obtained knowledge about the reliability of specific nodes or routes. The reliability of a specific node is based on its reputation and in order to assign ratings to routes the Pathrater mechanism requires a source routing algorithm to be used since all the nodes of a specific route must be known in order to calculate a rating for the whole route. In order that Watchdog can verify if the transmitted message has been received and forwarded correctly passive acknowledgements are used. Instead of relying solely on active acknowledgements the network interface is in promiscuous or monitor mode and checks if a specific message is forwarded again by the neighbor. One of the difficulties in overlay networks is that it is usually independent from the underlying physical network. Thus, in contrast to the research of Marti et al. the underlying physical network must not be a wireless broadcast network which allows to capture messages sent by its physical neighbors. In P2P overlay networks, the neighbors of a node my not be located physically in the same network segment and thus it is not possible to overhear the their communication. Therefore, the overlay network must provide a mechanism to obtain such information for each neighbor of a particular node. This mechanism can be a very simple acknowledgment mechanism provided by the neighbor's neighbor. If node S sends information on a specific route to destination D, each node on the route returns to its two-hop neighbor an acknowledgement about the received message. Thus, with a route as S,A,B,C,D node B sends an acknowledgement for a received message from node A to node S, node C sends an acknowledgement of a received message from B to A and so on and so forth. Thus, each node obtains information about its neighbors if there is a two-hop neighbor which also forwards or receives the same message. This mechanism can not only be used if routes are already established but also if routes are discovered. Over time enough information about each neighbor can be acquired and the Pathrater mechanism can thereafter assign ratings to each neighbor.

Since it only makes sense to gather information about nodes in the overlay network if they can be identified correctly and each node can also verify its identity, the system must be used at a specific level according to our security concept. In the examples given in this thesis using the security level *medium* we can protected from outsider attacks but are not able to identify and distinguish securely between different nodes if they choose to lie about their identity. In security level *high* each node possess a public / private key pair which allows us to identify each node securely and also allows us to verify the identity at any given time. We therefore have the possibility at this security level to even protect against insider attacks using countermeasures such as Watchdog and Pathrater. We have implemented both approaches in the framework and done some experiments using simple networks configurations. We have implemented the same approach also in the PeerSim simulation environment which allowed us to do more extensive experiments with reputation mechanisms.

7.5 Performance evaluation

We have obtained the results from our Java implementation for the secure overlay framework as well as from the same Java implementation used in the simulator called PeerSim. We have created and tested our implementation in several network scenarios with varying amounts of nodes. We ensured that different path lengths between nodes which send message exist. Nodes or links are also removed randomly between different nodes to force the routing protocol to perform route maintenance and establish new routes if necessary. Each node has a randomly generated pre-configured list of nodes which it knows of at startup. These nodes are contacted and the node joins the existing P2P network. The overall member list is learned thereafter and messaging between the different members is then initiated. We have measured the time it takes a node to discover routes to other members in that network depending on the specified security level as presented in Chapter 6. In addition we have taken measurements about the more general characteristics of our secure overlay network using simulation. Subsequently, we will show the results we obtained from the simulation.

7.5.1 Only the shortest route is used

In this section we have evaluated the influence of the amount of neighbors on the route discovery process and the resulting route lengths. All the simulations have been performed with 1000 nodes in the overlay network. Each of the nodes has been initialized with a random set of initial neighbors. The number of initial neighbors (1,2,5,10,15) has been specified in the configuration file. The actual amount of neighbors varies since connections are bidirectional and for that reason the actual amount is almost the double of the initialized size. This statement is correct until some saturation point is reached which is around 20% of nodes as neighbors. At this point already 10% less than double of the initial neighbors are actual neighbors. Each node sends a neighbor request to the initial set of neighbors and adds nodes to its set of neighbors if it receives a neighbor request from other nodes. In this experiment we have not introduced any churn into the system, thus the amount of available nodes always stays the same and no route error is triggered. We have triggered the route discovery process through sending 10 messages from each node to randomly selected destinations. The simulation itself is based on cycles and status information is only generated as long as messages are still transmitted, thus after the 10000 messages have been sent no further status data is logged. We send 100 messages every 50 cycles. In all of these experiments we have limited the maximum route length to 15. Thus, after 14 hops the route request is dropped if the destination has not been reached yet. We also only used the shortest route which was recorded at the destination in this experiment. Usually several route requests reach the destination depending on the amount of neighbors, amount of nodes in the system and the maximum allowed route length.

For all of the following experiments in this section we take into account the shortest and the longest route which is contained in the nodes routing table and create for both an average over the whole network. We also calculate the average of all routes in the whole overlay network. For some cycles at the start of the simulation the initialization functions of the overlay network are performed. For this each node sends out neighbor requests and triggers the join process after it receives a neighbor response from one neighbor. During the join phase several messages are sent which search for neighbor peers, the overlay network itself, perform the Needham-Schroeder-Lowe authentication protocol, and the distribution of the keying material.

In Figure 7.2 the overlay network is initialized with 1000 nodes with each node having 1 initial neighbor specified. Since each node has one neighbor and the connections are bidirectional each node ends up having more neighbors than initially specified. In this case we see that the first routes which are established in the system have a length of less than 9 but over time the average route lengths stabilize around 9.2. Since the shortest routes are established first and we check every 2 cycle it is normal that in the beginning only these are showing up. The shortest routes decline over time to 7.7 and the longest routes almost reach 11. What is not shown in this figure is the fact that with having only 2 neighbors in a 1000 node network only 13% of the messages arrive at the destination because

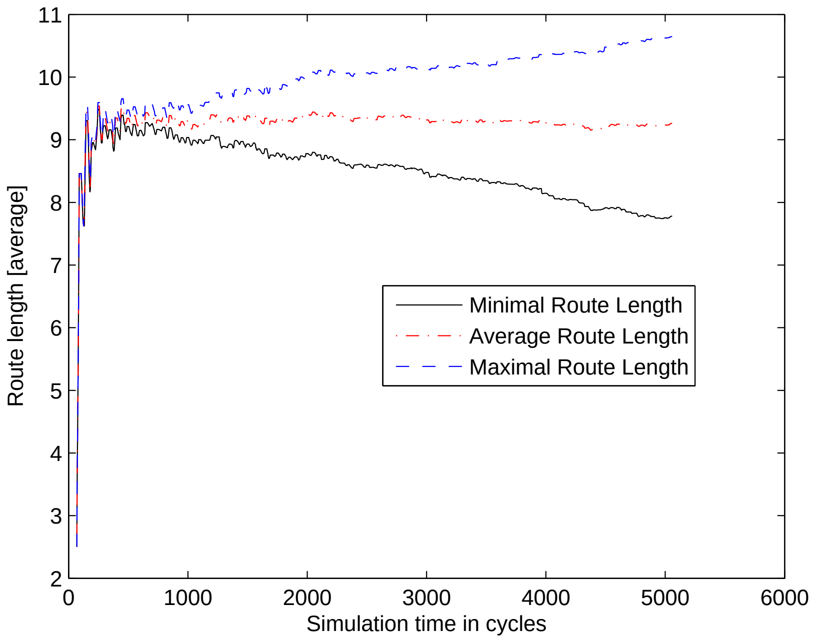


Figure 7.2: Route length distribution with 2 neighbors per node and 1000 nodes in total

for the other destinations no route could be found in the overlay.

In Figure 7.3 we see the route length distribution during the experiment with an initial set of 2 neighbors which corresponds to an average amount of 3.998 neighbors for the whole network. For this experiment again all three curves start out below 4.35 at which the average route length thereafter stabilizes. The shortest routes thereafter start to decline and ends at 2.6 after all messages are sent. But from the curve shape it is clear that this is not the end. If each node would establish routes to other or all other nodes this value would be lower. This is because for the calculation of this value only the shortest route of each node is used. The longest routes increases from 4.35 to around 5.7. The curve shape of the longest routes indicates that this is almost the end point for that value.

In Figure 7.4 we see the route length distribution during the experiment with an initial set of 5 neighbors which corresponds to an average amount of 9.978 neighbors for the whole network. For this experiment all three curves really start at 2.2 hops. In the beginning there is much variation due to the fact that only few messages have been sent and routes with very different route lengths have been established. The shortest routes curve immediately starts to decline and its value is 1.4 at the end of this simulation. Here again is no sign that this is the lowest value for this parameter since the curve is almost a straight line. The

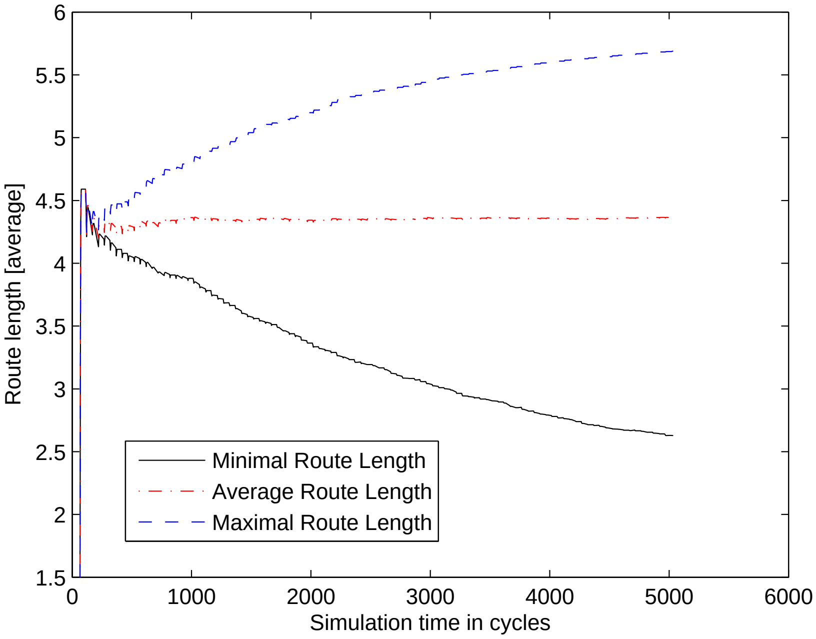


Figure 7.3: Route length distribution with 4 neighbors per node and 1000 nodes in total

average route length again does not change much and stays flat after the first 100 messages at 2.3. The longest routes curve starts immediately to increase and it looks like it finds its asymptote at approximately 3 hops peer route.

In Figure 7.5 we see the route length distribution during the experiment with an initial set of 10 neighbors which corresponds to an average amount of 19.890 neighbors for the whole network. In the beginning all three route length curves start at 1.7. For all the figures the first few cycles are not very meaningful since the few messages sent and routes established have too much influence on the overall result and since the destinations are selected by chance this introduces a lot of variance. However, already after some cycles the general trend or behavior of the route lengths in this setting becomes visible. The general theme is similar to the one in the previous figure 7.4. The mean of all three curves is lower than in the previous one. The overall average route length is after the initial phase a little below 1.7 and decreases very gently until the end of the simulation to little over 1.65 hops per route. The shortest route length starts the same as the overall average but decreases more aggressively to 1.06 hops per route until the end of the simulation. Again this looks not like it will be the end but it also seems not to reach only 1 hop. The longest route length curve again starts out slightly below 1.7 and grows to 2.01 until the end of the simulation. Here we

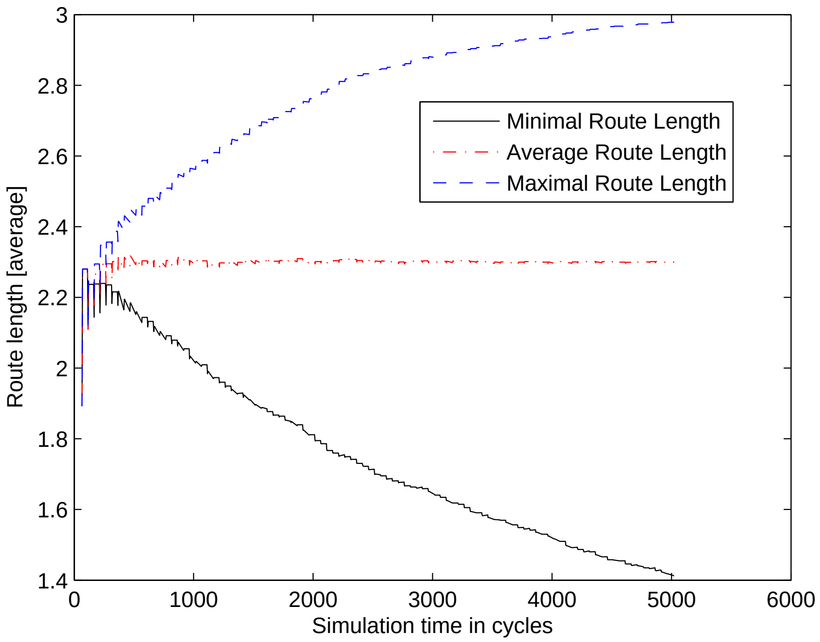


Figure 7.4: Route length distribution with 10 neighbors per node and 1000 nodes in total

will have reached the end point since it does not change much after 3500 cycles anymore.

In Figure 7.6 we see the route length distribution during the experiment with an initial set of 15 neighbors which corresponds to an average amount of 29.76 neighbors per node for the whole network. For this experiment all three curves start at 1.47 hops. The average route length declines a bit and stabilizes at 1.43 hops thereafter. The main characteristics of the figure is the same as the two before with the difference that the overall route length is smaller on average. The longest route length curve immediately starts to increase and reaches almost 2 hops per route. The shortest route length starts immediately to decrease and touches almost 1 which also will be the end in this scenario. In the end of the simulation all messages have been delivered, as it is also the case for the simulation with 2, 5, and 10 neighbors and about 9680 routes have been established which is also almost the same for the mentioned simulations. There are less routes than messages sent since by chance some of the sources are neighbors of the destination and must therefore not establish a route. We also initiate the route discovery process even if one of the existing routes at the source would contain the destination in its path, meaning the destination for a new message is a hop in another route.

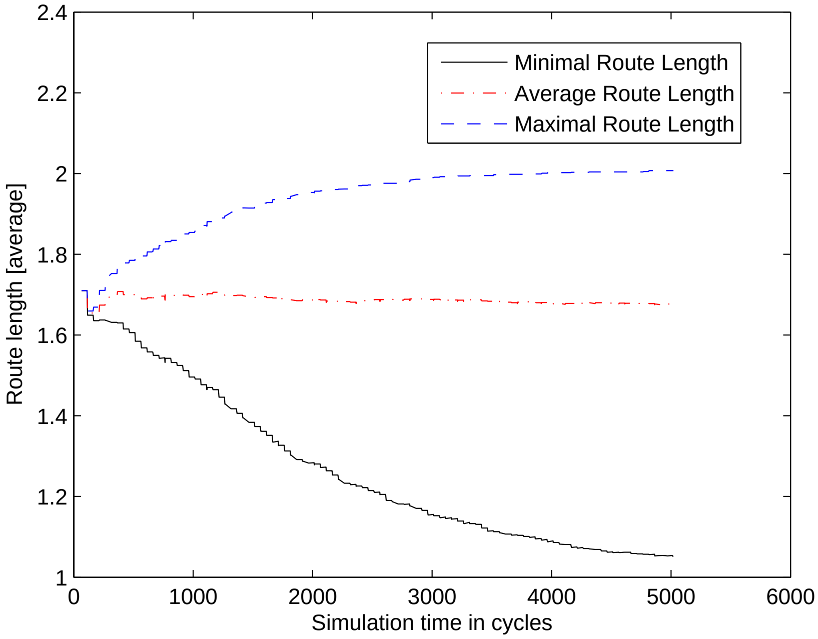


Figure 7.5: Route length distribution with 20 neighbors per node and 1000 nodes in total

It would be possible to use the existing route and send the message using the already established path, but we want to allow the destination also to choose what route should be established in order to only accept trusted nodes as hops.

The last Figure 7.7 for this experiment shows the amount of established routes depending on the amount of neighbors. Since we use always only the shortest route the amount of established routes is almost the same. The slight differences in the amount of routes for the different amount of neighbors comes from the possibility that the source and the destination are neighbors and therefore need not to establish a route.

Another reason for different amounts of routes, which is not applicable for these simulations, is the route request timeout. For each route request there exists a timeout after it is deleted from the request cache in order to allow new route requests for that destination in case of an route error or any other reason why no route has been established to a particular destination already. If the route request timeout is too long it takes too much time for the whole network to deliver messages if it is too short more routes are established as it may be necessary. Subsequently we provide some results with different route request timeouts and show how many routes are established.

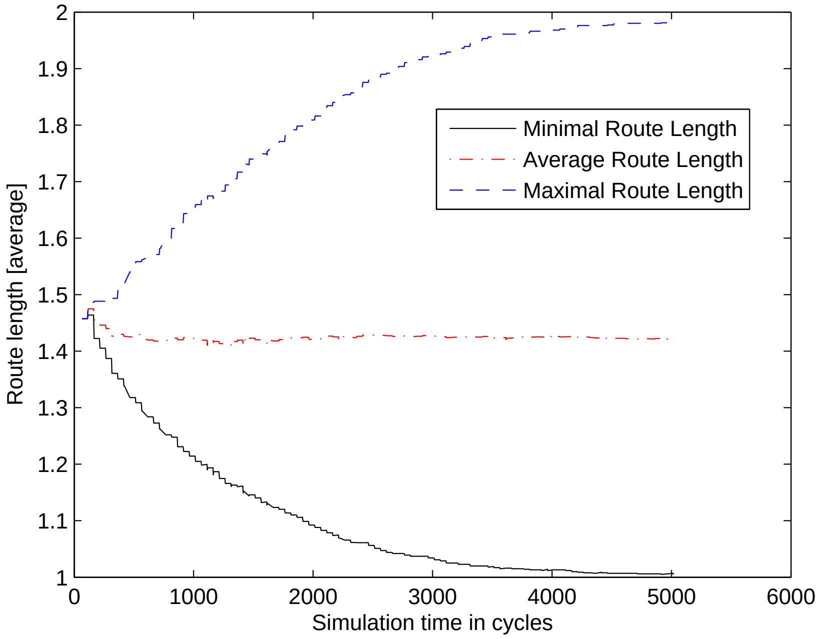


Figure 7.6: Route length distribution with 30 neighbors per node and 1000 nodes in total

7.5.2 All routes are used

In contrast to the previous section, we now establish all routes which are possible within the network triggered by the 10000 messages sent from random sources to random destinations. In this case we use a *path cache* as commonly used by dynamic source routing implementations.

In figure 7.8 the route length distribution of the overlay network is shown which has been initialized with 1000 nodes with each node having 1 initial neighbor specified as before. The figure is almost identical to the one where only the shortest routes are used. But since in this case usually there does not exist another route to the destination the figures are the same. We have the same message delivery success rate of only 13%. This is logical since only some additional routes can be established to the existing ones since all other parameters such as nodes, neighbors and maximum routing length stayed the same.

In figure 7.9 we see the route length distribution during the experiment with an initial set of 2 neighbors which corresponds to an average amount of 3.998 neighbors for the whole network. The length of the shortest routes in the whole overlay network on average start at around 4.5 and decline thereafter to 2.75 at the end of the simulation. This curve is identical to the one where only the shortest route is used. Since this curve is created from the sum over all shortest

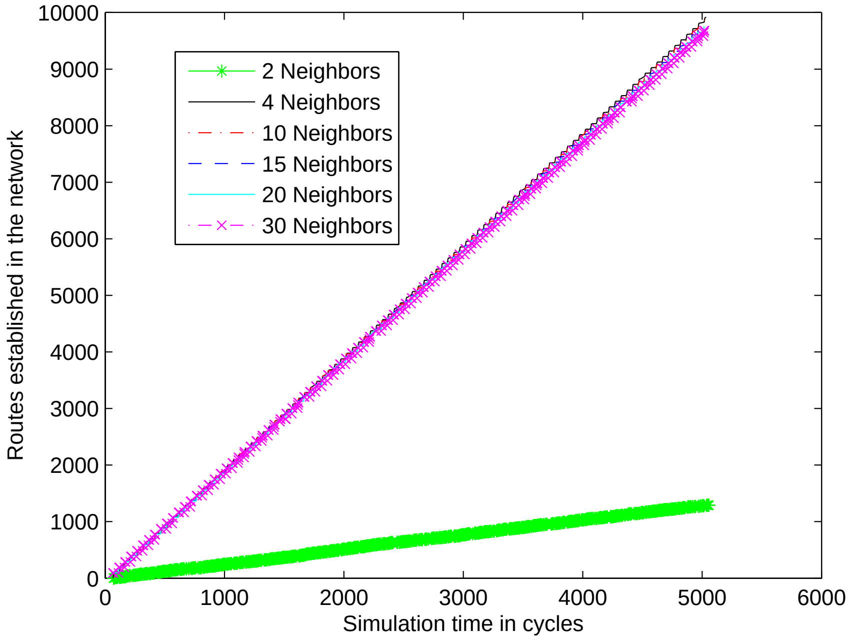


Figure 7.7: Amount of established routes for 1000 nodes

routes of each node divided by the amount of nodes this curve will always be the same all the following simulations as in the case if only the shortest route used. The average length of all routes in the whole overlay network starts out around 5.5 and stays the same for the whole time after some fluctuations in the beginning. Again these fluctuations are cause by the fact that only a few routes have been established and thus provide not a very good average in the beginning. The longest routes on average in the overlay network begin with 6.5 hops and grow slowly until the end of the simulation where it reaches 8.5 hops.

In figure 7.10 we see the route length distribution during the experiment with an initial set of 5 neighbors which corresponds to an average amount of 9.978 neighbors for the whole network. The figure looks similar to the previous one with the exception that the maximum allowed route length stays more flat or more precisely increases linearly at a very slow rate. It starts out at 4 and grows to 4.2 until the end of the simulation. The overall average route length is at around 3.3 the whole time. The shortest route lengths in this overlay network are significantly smaller. In this setting 97769 routes have been established after all of the 10000 messages have been delivered. The established routes have been created by sending over 97 million route requests whereas 87 million requests have been dropped on the way due to reaching the hop limit or returning to a host which has already seen this request. Route requests are also dropped if

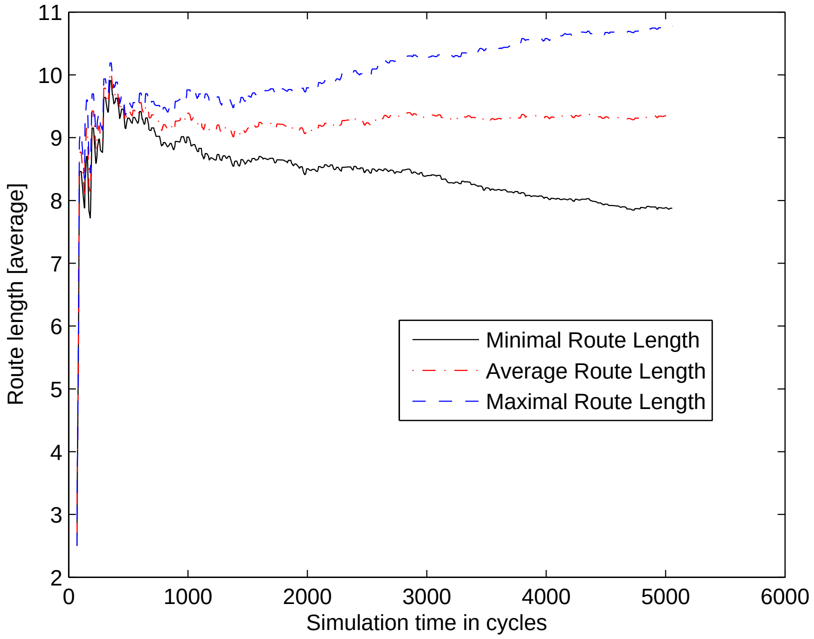


Figure 7.8: Route length distribution with 2 neighbors per node and 1000 nodes in total

the contained route is not correct or if the request has been corrupted. One of the downsides of source routing in general is that it does not scale very well after the networks get too big since all route requests are broadcasted. Where these limits are is not established but as one can see from these experiments is that even with 1000 nodes the amount of messages sent for establishing routes is tremendous although we have not taken churn into account in this simulation. One way to reduce the amount of requests sent is to use very accurate hop limits or use variable hop limits. Meaning that the algorithm should start with very low hop limits and increases this limit if no route could be established. On the other hand source routing allows for very simple and efficient incorporation of security mechanisms with the routing algorithm and verification of the security features at every hop in the route.

In figure 7.11 the route length distribution during the experiment is illustrated with an initial set of 10 neighbors which corresponds to an average amount of 19.890 neighbors for the whole network. Without surprise we see again a similar tendency in the figure with the overall average route length at around 2.7 hops per route. The maximum route length starts out at 3.05 and grows to 3.3 until the end of the simulation again in a rather straight line. The shortest average route length is less then in the previous experiments. It starts at

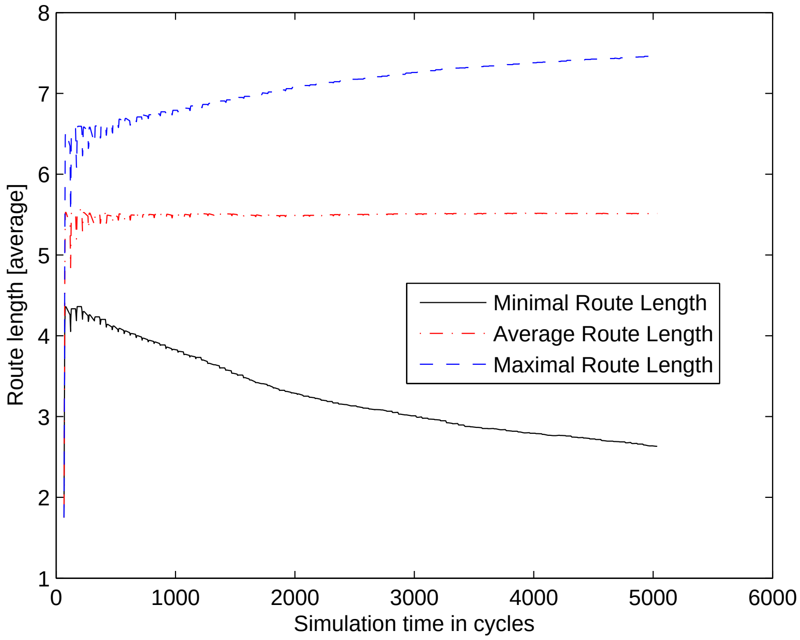


Figure 7.9: Route length distribution with 2 neighbors per node and 1000 nodes in total

around 1.7 hops and drops to 1.07 until the end of the simulation. In this setting over 193000 routes have been established for the 10000 sent messages. The route discovery process triggered almost 189 million route requests to be sent in the 1000 node big overlay network. All the messages sent until the end of the simulation including all the join, authentication, key exchange, routing and message delivery itself accounted for almost 20 GB of data transmitted over the network. The 20 GB of data are made up of all the messages sent by each node. Thus, for instance if a route discovery process has been initiated first the source broadcasts the request to all its neighbors. Thus, each neighbor again broadcasts this message to all of its neighbors and therefore the amount of messages and thereby data sent increases exponentially. Since usually the neighbors in an overlay network are not nodes within the same broadcast domain the message must be sent directly to each neighbor. If either a real physical broadcast or multicasting to neighbors could be used the amount of data transmitted would be limited significantly.

In figure 7.12 we see the route length distribution during the experiment with an initial set of 15 neighbors which corresponds to an average amount of 29.756 neighbors for the whole network. The longest average route length curve starts with 3 or less hops per route and stays there until the end of the

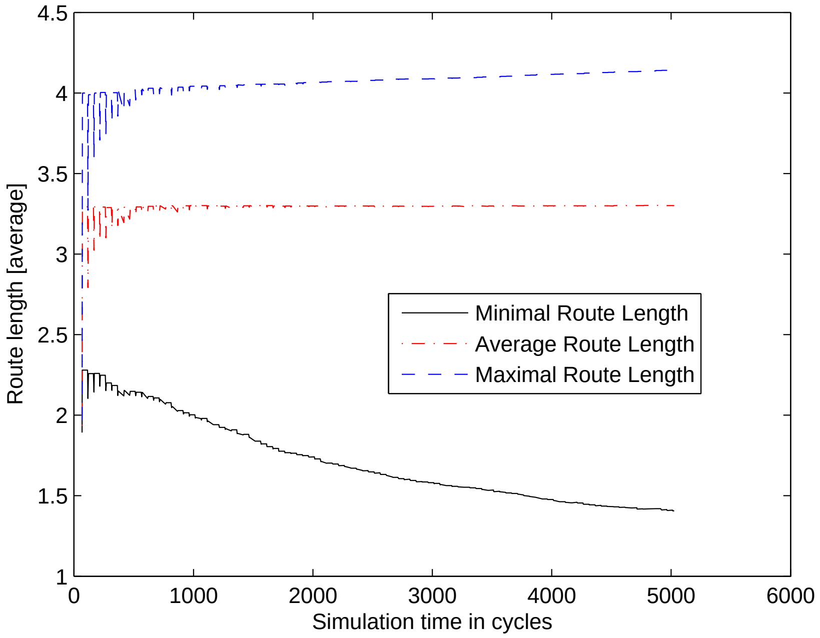


Figure 7.10: Route length distribution with 5 neighbors per node and 1000 nodes in total

simulation. In this figure we can see a lot of jitter for the longest and the average route length, again the reason is the relatively low amount of routes established in the beginning. The overall average route length is little below 2.4 and also stays flat until the end. Only the shortest routes curve varies and shows the same shape and values as if only the shortest route would be used. In this experiment the 10000 messages trigger the establishment of about 286000 routes in the overlay network with 1000 nodes and the maximum of 15 hops per route. The amount of data transmitted from the start until the end of the simulation amount to 34 GB. If all transmissions to the neighbor nodes would be performed by physical broadcasts or multicasts the amount of data sent would be less than 1/20 of the previous amount. Thus, this seems to be a very good motivation to use broadcasting and multicasting as often as possible. Unfortunately, in the current Internet architecture it is not guaranteed that multicast is supported physically by all routers. But since the reduction is very significant, more efficient communication mechanisms should be investigated.

The last figure 7.13 for this experiment shows the amount of established routes depending on the amount of neighbors. In this experiment we use all routes which can be established with the selected request timeout. Thus, the amount of established routes varies much between the 6 different settings since

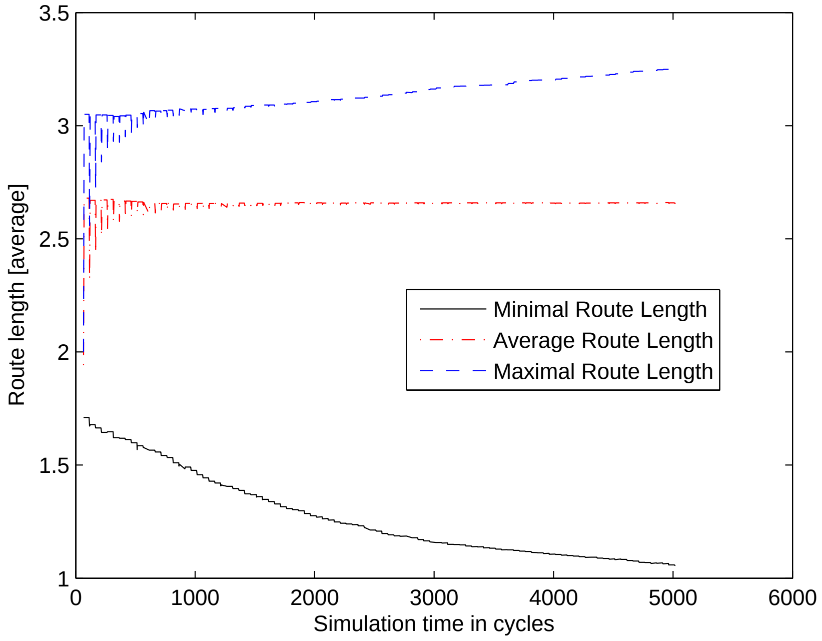


Figure 7.11: Route length distribution with 10 neighbors per node and 1000 nodes in total

the possible routes are related with the amount of neighbors in the network. From the curves for 10, 20 and 30 neighbors one can deduct that for every 10 additional initial neighbors about 90000 more routes are established in an overlay network with 1000 nodes and 10000 messages sent. In the last section of this chapter we verify if there is any regularity for this behavior. Another interesting fact in these simulations is to obtain with what amount of neighbors it is possible to end up with a network where every node is reachable. In this simulation we see that it is possible with 4 neighbors. Since, we have only the opportunity to go from 2 to 4 neighbors we can not verify if it would also be possible already with 3 neighbors. This is because we assume bidirectional connections we usually get double the initial amount of neighbors up to some point where the neighbor set gets saturated which starts at 20% of the amount of overall nodes. This means that if more than 20% of the total nodes are assigned as neighbors we do not end up anymore with effectively the double amount of neighbors.

7.5.3 Neighbor-Route Distribution

Since the amount of routes established usually should be related to the amounts of neighbors each node has in the network we tried to find out the correlation for that. We have experimented with different amounts of nodes in the overlay

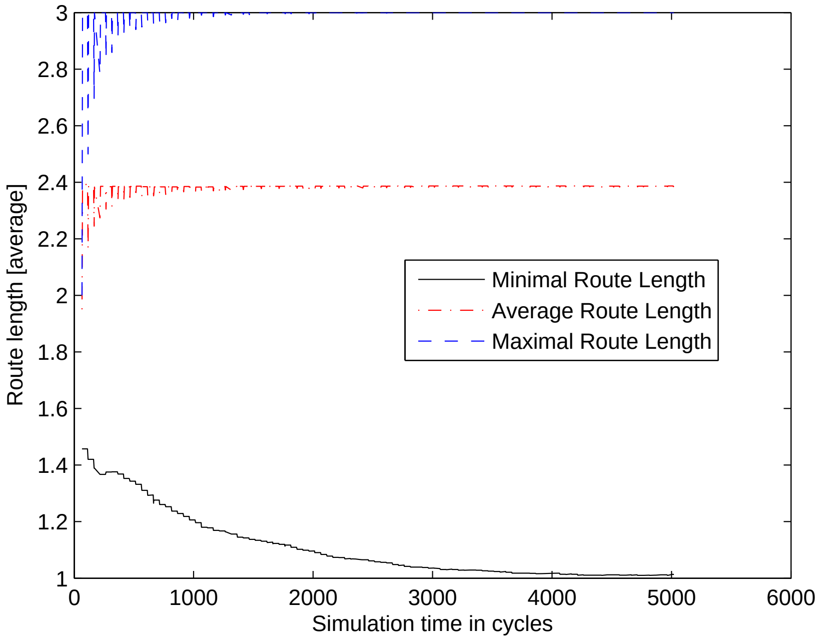


Figure 7.12: Route length distribution with 15 neighbors per node and 1000 nodes in total

network and with the amount of neighbors ranging from 2 to 99% of the overall nodes. We specify the amount of neighbors in percentage since we use three different amounts of overall nodes in the network. We started with 2, 4, 6, 10, 20, 30 percent of neighbors. Then we used 35, 45, 50, 55, 65, 70, 75. The last data points starting with 80 percent lay more together since many initialized neighbor nodes are required to get to 99% of neighbors.

In figure 7.14 the amount of routes in relation to the amount of neighbors per node is illustrated in non-logarithmic fashion. Thus, it is not easy to see the distribution for the network with 100 nodes. But for all neighbor-route distribution it is easy to see that there is a peak for the established routes at around 35% of overall nodes used as neighbors. This means that with that amount of neighbors the most different routes for one destination can be found. This also means that with that specific amount of neighbors the overall network itself is most robust to changes in the topology and node fluctuations.

In figure 7.15 the same route distribution as before is depicted but using a logarithmic scale for the amount of established routes in order to better accommodate networks with less overall nodes. The curves for the different amounts of neighbors looks almost the same. The four curves show the distribution for 100, 250, 500 and 1000 nodes in the overall network. Another fact which can

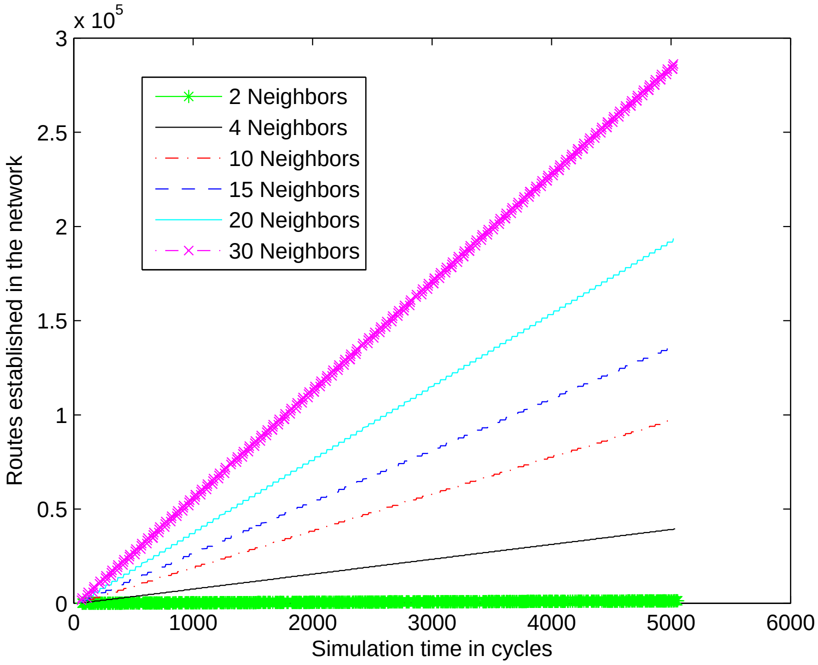


Figure 7.13: Amount of established routes for 1000 nodes

be observed in the logarithmic representation is that starting with 30-35% of nodes as neighbors the amount of established routes does not increase but is decreasing.

7.5.4 Message delivery latency

An important factor in an overlay network is the message delivery latency and especially the latency of the initial message delivery. This means that the time it takes from sending the message from the source until it reaches the destination varies greatly depending on the routing algorithm used. If a routing algorithm is used which works pro-actively, messages are exchanged permanently in order to have the complete routing table available in case a message has to be sent. This allows for lower initial message delivery latency but it also adds a lot of complexity to the overlay network since these messages are exchanged between the nodes constantly. Depending on the algorithm used, either distance vectors or neighbor lists are exchanged or broadcasted in the network. Thus, as overlay networks grow beyond a certain amount of members or if the amount of churn is high such routing algorithms are not efficient. Thus, other algorithms which can incorporate the specific features of overlay networks into their mechanisms are used. We used a on-demand routing algorithm which first checks if the route

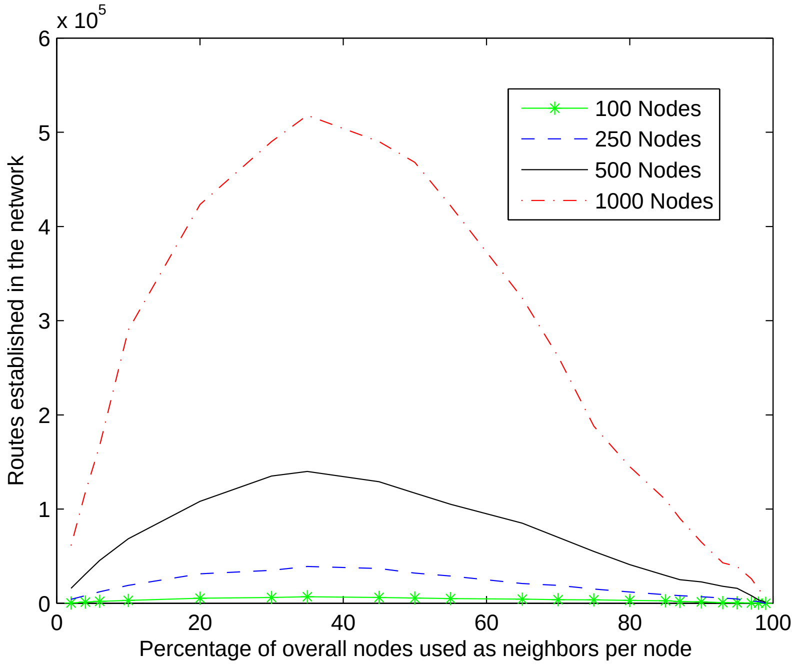


Figure 7.14: Route distribution in relation to the amount of neighbors per node

is already available in the route cache and if not then triggers the discovery process. Depending on the specific implementation of the on-demand algorithm it is possible that only the source and/or the destination obtain a correct route or also all intermediate nodes which are on the route path also store this route in their cache. There exist several improvements for these kind of mechanisms in order to obtain routes faster, learn routes from the discovery process and use the obtained information more efficiently. One of the important features of the route discovery process is the route request timeout. This timeout specifies how long a route request for a specific destination is stored in the request cache. As long as this request is contained in the cache the node does not accept any new request for this destination. Therefore, this timeout has an effect in very dynamic networks with an high amount of churn. Since usually in dynamic networks the routes break very often and new routes to a specific destination must be discovered. Thus, if the timeout is too long it takes too much time to establish new routes which again increases the message delivery latency. Another effect is that if the route request timeout is to short too many route requests are initiated because it takes too much time for the response to arrive at the destination.

Subsequently we will go more into the details of churn and its influence on the message delivery latency. Usually the message delivery latency and churn

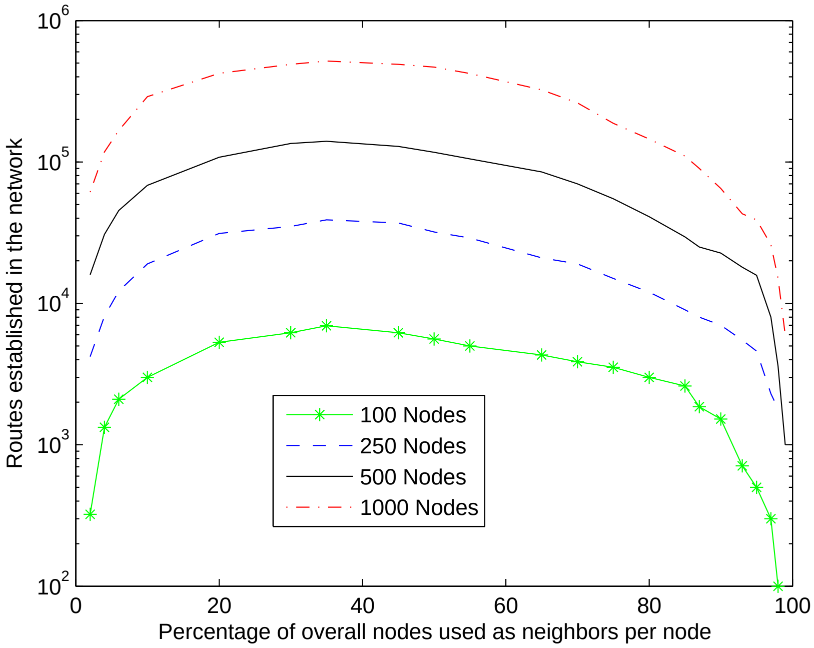


Figure 7.15: Route distribution in relation to the amount of neighbors per node in logarithmic depiction

can not be analyzed without taking the routing algorithm and in our case the route request timeout into account. Thus, we will also investigate the influence of the route request timeout in combination with churn on the message delivery latency. We will do this with different simulation settings in order to get a clear picture what the best parameters for specific networks are so that they can be operated efficiently.

Influence of churn

The efficiency and performance of overlay networks, especially the used routing algorithm, are strongly influenced by the amount and specifics of churn. Churn specifies the fluctuations in overlay nodes through joining and leaving nodes in the overlay network. Hereby also nodes which do leave the network without informing the overlay have the most influence because the neighbors of the leaving node can not take appropriate action as well as the nodes which require the disappearing node to communicate with others. Since overlay networks are very dynamic and have no specified times at which nodes join or leave the network the topology of the overlay changes constantly. Thus, established routes and the destinations of a particular communication session may not be valid anymore during the runtime since intermediate or destination nodes leave the network or fail to function. A robust routing algorithm should be able to update existing routes and perform a new route discovery process if necessary. The routing algorithm used in our overlay network has the ability to adapt to such changes due to routing errors triggered if a node tries to send a message over a route which does not exist anymore. Thus, the node detecting the failure returns a route error (piggy-backing the original message) to the source of the message. The source deletes all the routes in its route cache which contain the failing node. Thereafter, the source checks if another route to the specific destination is available. If not the route discovery process is initiated again. If an alternative route is established the message is sent to the destination thereafter.

It is also worth noting that it makes a big difference what kind of network model will be simulated.

For instance in the previous setting we have sent a few messages to randomly chosen nodes which may be realistic for future Internet-of-Things subnetworks where everything is connected and different entities want to obtain or distribute data between each other. It may also be realistic for environmental monitoring situation where the different sensors simply try to distribute small amounts of data to different nodes for reasons of redundancy and reliability. But for instance in common file-sharing or content distribution networks it more often happens that the nodes exchange big amounts of data with the some nodes without contacting other nodes at all or at least only for status information. Thus, it is also very important to simulate such network settings, since our overlay system should also be applicable in these situations. Thus, we will provide results for both scenarios.

In addition, it is often advisable in situations where constant churn is observed, no matter what kind of network or routing algorithm is used, to establish new neighbor connections in order to keep the network highly connected and to counter problems such as growing route lengths and even partition of the network due to missing links. We will now investigate the behavior of our network under different churn settings where we modify the amount of nodes joining and leaving, the maximum amount of additional or missing nodes, the amount of neighbors and also the possibility of *re-neighboring*. We will do this by obtaining specific values which represent the behavior of our system under such circumstance such as the amount of routes established over time, the amount of routes currently used, the amount of requests sent and dropped, the message queue size, different message delivery latencies and how often it happens that no route to a destination is available due to churn.

For the experiments in this section we used the following overlay network settings. We again used 1000 nodes with a maximum hop distance of 15 for each route request. The route request timeout was fixed to 500 cycles and each node had 2 initial neighbors which again resulted in 3.98 neighbors on average with 2 neighbors effective minimum and 9 neighbors effective maximum. Each node sent 1000 messages to the same other node during the simulation. 100 messages have been sent every 10 cycles in the whole network. Thus, in sum 1 million

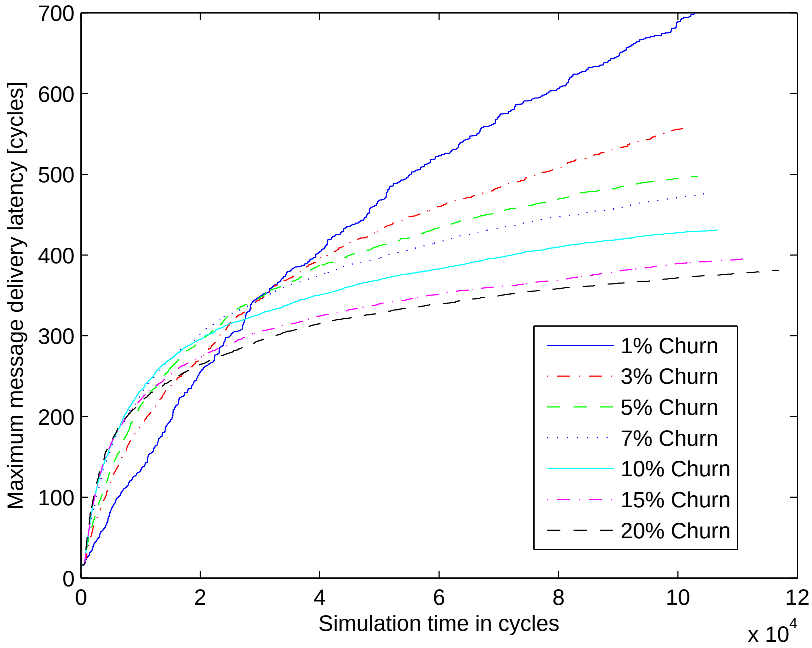


Figure 7.16: Maximum message delivery latency with a maximum of 20% down nodes and different amounts of churn.

messages have been sent and delivered in the whole overlay network after the simulation has finished. Regarding churn we have simulated the same network settings with different amount of churning nodes and maximum down nodes. The churn interval was always the same with a value of 10 cycles. Meaning, that every 10 cycles a specific amount of nodes either were available again or were not reachable anymore. We used amounts of churn of 1,3,5,7,10,15 and 20% of total nodes. Thus, every 10 cycles 10 - 200 nodes were selected to either become active or inactive again, depending on their pervious status. We simulated this setting with 3 different amounts of maximum down nodes of 5, 10, and 20% which resulted in either 950, 900 or 800 active nodes at minimum.

In figure 7.16 the maximum message delivery latency in relation to different amounts of churn with a maximum of 20% down nodes is depicted. This simulation shows a very interesting fact about the influence of churn on the overall network performance. In the beginning the maximum message delivery latency is lower for the simulations which had less amount of churn and vice versa. For the curve with 1% churn the maximum message delivery latency is around 100 cycles after 5000 cycles of the simulation, which is half of the latency of the curve with 20% churn. But after 10000 cycles the whole image changes. The curves for the highest amounts of churn start to asymptotically approach their maximum value much earlier than the ones with lower amounts. Thus, after a little more than 20000 cycles the curve with the lowest amount of churn has crossed the curve with the highest amount of churn and keeps increasing with a much higher rate than all others thereafter. From this figure it is not visible where the maximum message delivery latency value for this amount of churn lies. All the other curves, with a possible exception for the curve with the second lowest churn amount, seem to already approach a value which is lower than 700 cycles asymptotically. There are several reasons for the observed results. First 1% of churn which results in only 10 nodes selected to either become active after being down or vice versa is relatively low. It takes longer that the maximum amount of nodes which are allowed to be inactive is reached and it takes more time in general for a down node to get active again if the amount of churn is lower. Second the amount of down nodes with 20% is relatively high. Thus, 200 out of 1000 nodes are down after the initial phase and this amount of down nodes is almost constant, with sometimes 198 or 199 nodes down, until the end of simulation. Since so many nodes are down there is an increased chance that not many routes to a specific destination can be found. Thus, if a node is crucial for establishing a route to a specific destination the messages can only be delivered if the node is active. These are the reasons for that particular outcome. Thus, the higher the amount of maximum down nodes, the lower the amount of churning nodes and the longer the churn periods the higher the maximum message delivery latency.

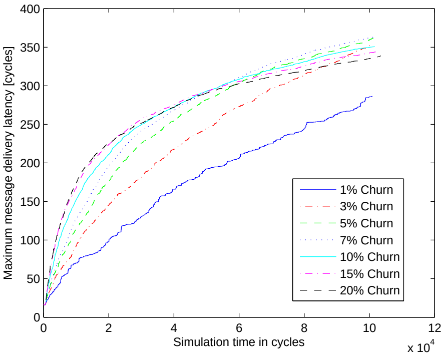


Figure 7.17: Maximum message delivery latency with a maximum of 10% down nodes.

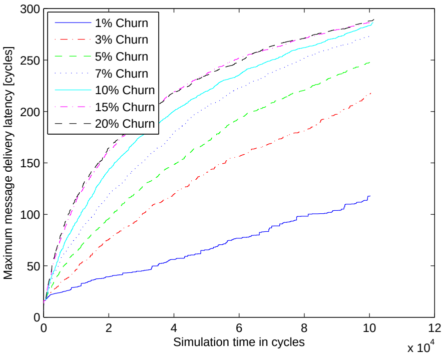


Figure 7.18: Maximum message delivery latency with a maximum of 5% down nodes.

In figure 7.17 and figure 7.18 the same network settings and amount of churn are outlined but the maximum allowed down nodes was different with 10% and 5%. These figures show a slightly different picture but the results are in line with the conclusion we draw from the figure 7.16. In figure 7.17 the curve with the lowest churn rate does not cross the one with the highest churn rate anymore, or more precisely until the end of the simulation. If the simulation is prolonged, for instance by sending 3 times the amount of messages, the lowest churn rate curve still crosses the others but only slightly. Thus, if only 10% of the overall nodes are down, the influence of low churn rates is not that drastic anymore. This can also be observed when figure 7.18 is analyzed. Here the amount of maximum down nodes is low enough so that the amount of churn is positively correlated with the maximum message delivery latency. If the amount of down nodes gets below some point it is usually possible to establish alternative routes and then only the frequency of route errors influences the maximum message delivery latency.

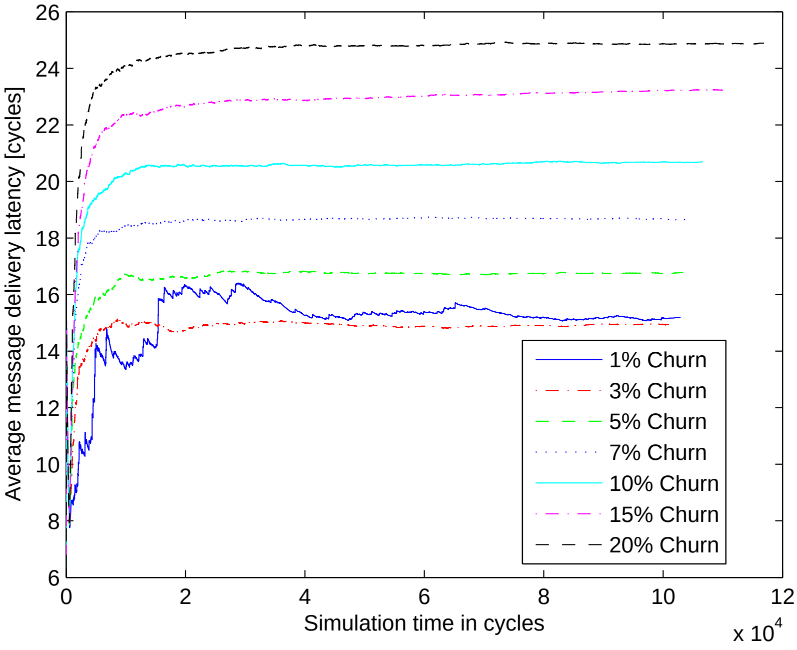


Figure 7.19: Average message delivery latency with different amounts of churn.

In contrast to the maximum latency, as shown previously, the average latency is much more influenced by the average route length and the frequency of route errors. Thus, the figures 7.19,7.20 and 7.21 all show a direct correlation between the amount of churn and the average message delivery latency. The only exception is the curve of 1% churn, which not only shows much more variation then the others but it also has a higher average latency then the 3% churn curve in the setting with a maximum of 20% down nodes. The fluctuations in the beginning can be attributed to the same fact as mentioned for the maximum latency. In the beginning, as less messages have been delivered the extreme outliers, which have been created by the fact that nodes will stay down much longer than in other scenarios, have much more influence on the overall result. As the simulation progresses the curve smoothens out but it still stays above the 3% churn curve.

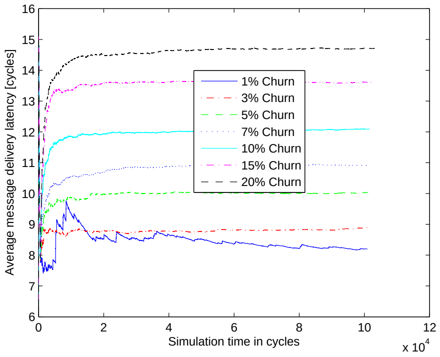


Figure 7.20: Average message delivery latency with a maximum of 10% down nodes

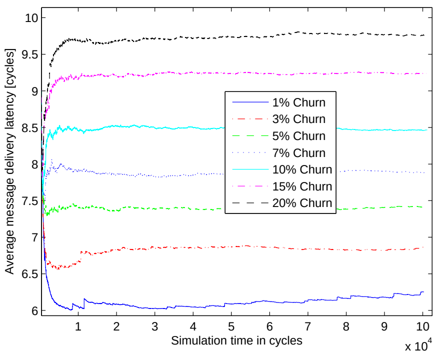


Figure 7.21: Average message delivery latency with a maximum of 5% down nodes.

For figures 7.20 and 7.21 the same is true but now the positive correlation is true for all curves. Although, for the figures with 20% and 10% maximum down nodes it seems that the lowest churn rate curve would decrease with additional simulation cycles and for 5% to increase until it reaches some ultimate value which we have not determined.

Figures 7.22,7.23 and 7.24 show the development of the message queue in relation to different amounts of churn and maximum inactive nodes. All the curves have been calculated using an moving average. The message queue gives us a clue about how many messages can currently not be delivered due to not available routes. This can happen if routes could not be established or routes have been broken during the delivery of messages and thus the message has returned to the source and is stored in the queue. One obvious observation is that the higher the amount of churn the more messages are stored in the queue. For 20% churn the average amount of messages stored is between 220 and 230. For the 3% and 5% of churn it is between 100 and 120. All the curves, with exception of the curve for 1% churn, are almost completely separated and have relative low variation. The average for the curve with 1% churn has its average similar to 3% and 5% but on the contrary its variation is high. The message queue length varies from 50 to 220 messages for this simulation. The reason for this variation is again the same as mentioned for the maximum and average message delivery latency. Since too little nodes are getting active every churn cycle and a lot of nodes are down several destinations can not be reached and the messages must therefore be queued. The specific values show when the most routes are not available. The specific times are random since the nodes which become inactive are also selected by chance.

The overall tendency of the figures 7.23 and 7.24 is the same as for the previous one with 20% of maximum inactive nodes. However, the overall average

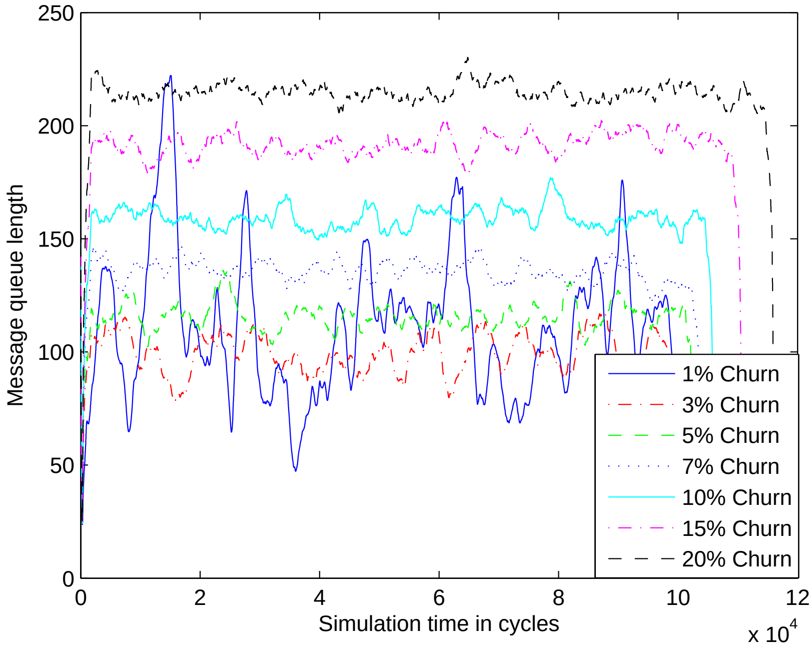


Figure 7.22: Message queue length with a maximum of 20% down nodes with different amount of churn.

of all curves is lower. For 10% of maximum inactive nodes it is between 20 and 100 and for 5% of maximum inactive nodes it is between 5 and 45. Thus, it can be deducted that the lower the amount of maximum inactive nodes and amount of churn is in an overlay network the lower the average message queue length will be. The only exception would be if the amount of churn is very low so that it takes too long that inactive nodes which are important in the system because they have many neighbors are getting active again. Also in the figure with the lowest amount of maximum inactive nodes the simulation with only 1% churn shows much more variation in the queue length then the others. Why the variance starts to increase to the end of the simulation is not obvious and can only be attributed to the random selection process of nodes to become inactive.

Influence of request timeout

In the previous section we have discovered and outlined the general aspects of churn and its effect on overlay networks. In this section we continue to investigate the effects of churn combined with the influence of the route request timeout. Churn in the form of nodes joining requires the overlay network to create new routes if messages are sent to these nodes and leaving or disappearing nodes introduce route errors into the system. Route errors require nodes to return

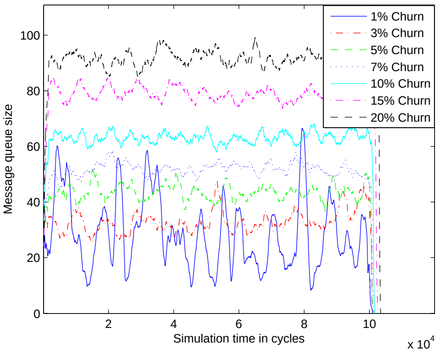


Figure 7.23: Message queue length with a maximum of 10% down nodes

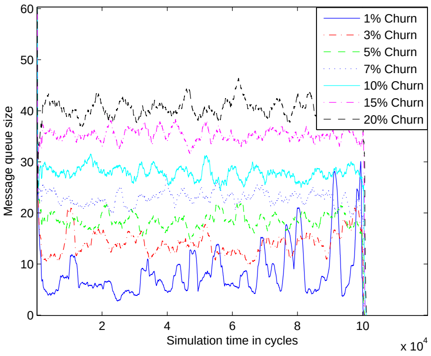


Figure 7.24: Message queue length with a maximum of 5% down nodes.

route error messages and the original message which elicited the error back to the source on the same path as the original message traveled. On the path back to the source each node also checks if the failing node is required in some of its own routes and removes them from its routing table. The source performs the same check and removes all routes with the failing hop and extracts the original message which is piggybacked to the route error. Thereafter, the original messages are sent again and most probably triggers a new route discovery process depending on the overlay network settings. For instance if only the shortest route is returned by any route discovery process the source will not have any alternate route for the destination and thus must start a new discovery process. On the other hand, if all routes are returned which can provide unique and correct paths to the destination it is most likely that an alternative route is available and the source can resend the original message immediately. If it is wise to use the overlay network in the *useAllRoutes* setting depends of how much churn is expected, how much and how often normal traffic is exchanged and also how well each node is connected in the network and how many hops route requests are allowed to travel. If all routes are used many more route response messages are sent which travel a lot of different paths and thus inflicting much additional traffic. But there is as always a tradeoff because route requests produce even more traffic since they are broadcasted through the network and are forwarded until the hop limit is reached or the receiving node has seen the route request already.

Thus, the message delivery latency as well as the amount of traffic created in the network depends very much on the specifics of churn, the routing algorithm used, and the amount of neighbors for each node. For the routing algorithm we use one of the most important parameters is the route request timeout which specifies how long it takes before a new route discovery process for the same destination can be initiated.

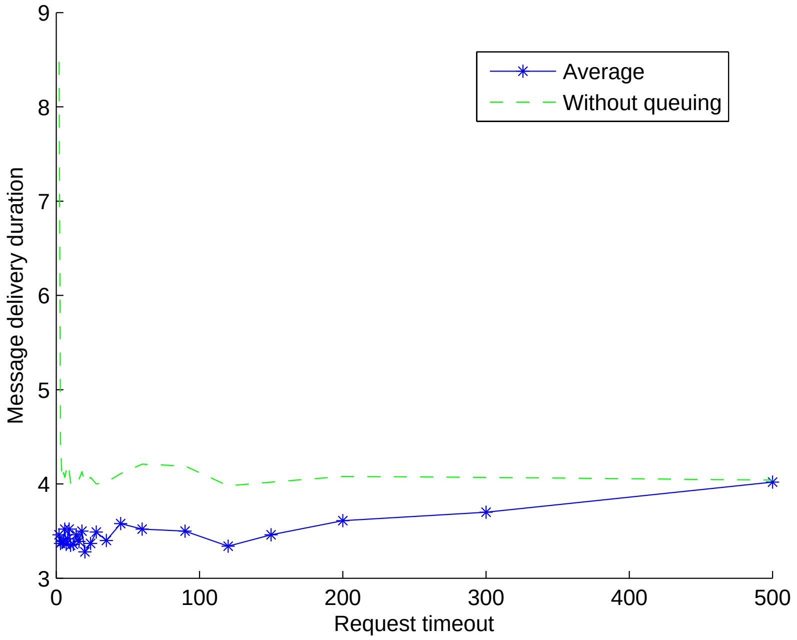


Figure 7.25: Message delivery latency for the overall average and all messages without queuing.

This is not only true for the source but also for each node in the network since each route request for the same destination received from the same source is hold in cache until the timeout is exceeded. For the experiments depicted in Figures 7.25,7.26, and 7.27 we simulated an overlay network with 100 nodes and 4 initial neighbors for each node which resulted in 7.82 neighbors for each node on average. The maximum amount of hops a route request is allowed to travel is 10. Each node has sent 1000 messages to randomly selected destination during the simulation but only 100 messages have been sent during each simulation cycle. In the end 100000 messages have been sent which resulted in 12 million route requests which where received in the overlay network. The churn in the overlay network was 1% of the overall nodes with an interval of 3 cycles. Meaning that every 3 cycles one percent of the nodes are randomly selected and either disappear or participate again in the network. We do not use and new joining peers or peers which leave the network correctly for this simulation since the do not have that impact in the overlay execution. The maximum amount of nodes which are allowed to be down is 20%.

In figure 7.25 the influence of churn in relation to the route request timeout for the overall average message delivery latency as well as the overall average message delivery latency without queued messages is shown. The average message delivery latency from the whole overlay network depicts how long it takes on average for a message to be delivered to the destination with this simulation configuration. It starts at 3.5 cycles with an request timeouts starting at 1 cycle and grows to 4 cycles with a request timeout of 500 cycles. The longest message delivery time without queueing outlines the longest time it takes to travel from the source to destination if the message can be sent immediately. This time starts at around 8.5 with a request timeout of 1 cycle. With a request timeout of 4 this time shrinks to a little more than 4 cycles. For all other request timeout it stays at this level since 4 is the maximum amount of hops which are required to connect two nodes.

In this overlay network configuration the average route length is 1.78 and thus it takes around 3 cycles for a message to be delivered to the destination on average which corresponds to the measured value of 3.5 (if all the queued messages and piggypacked messages would not be counted). The reason why for very short request timeouts is that since on average it takes more than one cycle to reach every destination. Actually within one cycle only all one-hop neighbors are reached. Thus, every new route request is accepted and also route requests which traveled a different path are accepted by all nodes. The only limiting factors are now the route request hop limit, the route response cache at the destination and the routing table at the source. Since it usually makes sense to use the same timeout for the route response cache as for the request cache the destination accepts and responds too much more route requests as necessary if the timeout is too short. Thus, it happens that routes up to the maximum allowed hops will be accepted and route responses will be sent. If the timeout is only 1 cycle long each cycle one route will be accepted at the destination and since longer routes take longer to reach the destination. Although longer routes will be accepted and route responses with this routes will be sent to the source these routes will be added to the routing table only under certain circumstances. For instance only if the routing table at the source does not contain any route for this particular destination already. This means that in order that add longer routes are added to the table the shortest route which has been added before must have been used already and has triggered an route error. Due to this route error the shorter route and has been deleted in the meantime and thus route responses with longer routes which arrive a little bit later make it into the table. This results in longer message delivery latencies and longer average and maximum route lengths.

In figure 7.26 the influence of churn in relation to the route request timeout for the longest message delivery latencies without piggybacked messages as well as the overall longest message delivery latency including also messages which triggered route errors and where returned to the source. Both values are calculated by obtaining the specific value from each node and thereafter dividing it through the amount of nodes in the network. The longest message delivery latency without piggybacked messages starts at about 14 cycles for a request timeout of 1 cycle. The latency grows relatively constant to 170 cycles with a request timeout of 500 cycles. For the overall longest latency it looks a bit

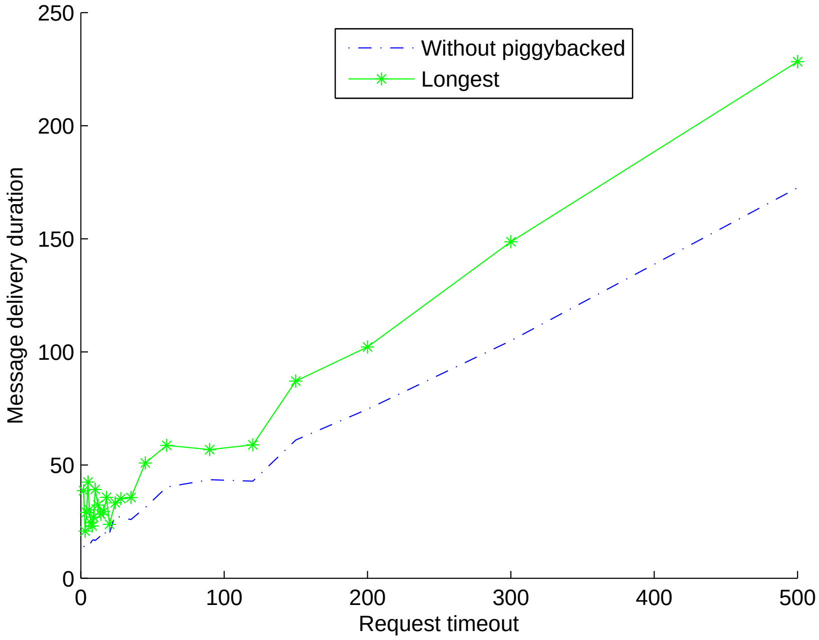


Figure 7.26: Message delivery latencies for messages which where piggypacked on route errors and for the overall maximum.

different in the beginning with a lot of jitter until the request timeout reaches 25 cycles. Most of the jitter can be explained by the randomness of nodes which are joining and leaving. If the route request timeout is relatively short the time until a new route becomes available for a specific destination has more influence on the longest message delivery times than the request timeout itself. Thereafter, the time it takes until new routes get available gets less important and the request timeouts influence grows. At a request timeout of 500 cycles the overall longest message delivery latency is around 230 cycles. If a route error occurs, at most the amount of cycles specified by the request timeout will elapse until a new route discovery process can be started. At the very worst it could happen that after a route error occurred the message is returned to the source and that the route which is obtained in the next discovery process again has a route error during message delivery. This case can happen all over again but with drastically decreasing probability for each subsequent discovery process.

In figure 7.27 the influence of churn in relation to the route request and the amount of route requests delivered is depicted. The figure shows the same longest message delivery as in the previous figure but it also outlines the amount of route requests delivered in the whole network for specific route request timeouts. This allows us to identify the optimal point where both the message delivery latency

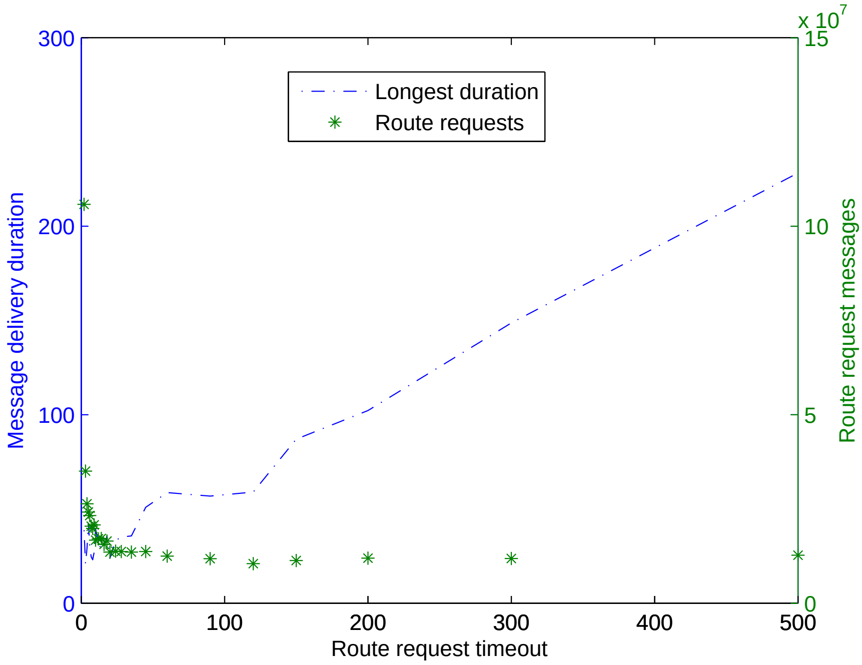


Figure 7.27: Amount of route requests sent in relation to the message delivery latency.

and the amount of route requests delivered are still low. It happens to be at between 20 and 30 cycles for this network configuration. The average route length with a route request timeout of 20 is 3.10 and the longest routes which are established are 4 hops. The amount of route requests sent is about 13.5 million. The overall longest message delivery latency is 23.78 cycles and the average message delivery latency is 3.28. The average route length with a route request timeout of 28 cycles is 3.03 and the amount of route requests sent is 11.3 million. The overall longest message delivery latency is 35.65 cycles and the average latency is 3.4.

7.6 Related Work

One of the first research work which addressed security in overlay network was the one from Castro et al. [CDG$^+$02]. It deals with structured overlay network and routing security based on distributed hash tables (DHT). Some general ideas about how to achieve a secure P2P system are stated. They identified three requirements for secure routing which are secure assignment of node IDs, secure maintenance of the routing table and secure message forwarding. They also investigated the impact of different attacks on structured overlay network and the efficiency of their solution. But since they are only dealing with structured overlay network and have not addressed anything else besides secure routing their work can only provide some advice on secure node ID generation and routing for overlay networks in general.

The well-known P2P middleware JXTA also states to be a secure P2P system [YW02]. JXTA makes use of well known mechanisms and protocols. These mechanisms are used only for point-to-point encryption and node authentication. There are no secure routing primitives or any special authentication mechanisms designed for overlay network. No general security concept exists which addresses the security requirements, as stated in [CDG$^+$02], directly in the JXTA middleware. There are no explicit mechanisms to protect the P2P system against attacks from adversaries or threats from misbehaving and selfish nodes. The primary goal of JXTA in terms of security is to provide cryptographic primitives as service the to the application layer.

7.7 Conclusion

With this work we have provided a general framework for secure overlay networks. The framework is based on a security concept which allows for an straightforward specification of the security level depending on the requirements of the application and the capabilities of the participating devices. The framework has a modular design providing the possibility to change the behavior and the functionality of the overlay network very easily through the configuration. This framework allows developers to create secure distributed applications without taking care of the underlying physical network topology and having in-depth knowledge of security mechanisms and cryptographic protocols. The framework has been developed for Java SE and ME and can be downloaded for testing purposes from our sourceforge project site.

8
Secure P2P Applications

> Interesting phenomena occur
> when two or more rhythmic
> patterns are combined, and these
> phenomena illustrate very aptly
> the enrichment of information
> that occurs when one description
> is combined with another
>
> Gregory Bateson

8.1 Introduction

Multi-agent systems (MAS) have been used to solve problems for which standalone or monolithic systems are not well suited. Examples of problems to which multi-agent systems have been applied include control systems [VDPBS02], [MSH05], timetable coordination [PBG05], and disaster response [GGGR03], [SMTS05]. MAS are especially promising for disaster response scenarios. Since the specific tasks of such scenarios like information gathering, on-demand computation, information distribution, and team coordination are well-suited for MAS.

In close relation to the organization and operation of MAS we find the overlay network concept since both function as a distributed system. Structured overlay networks are very prominent since they are well suited for data storage and distribution. Their internal organization and function is optimized for addressing data in a distributed environment. Conversely, they are ill suited for the purpose of a general overlay network that provides general communication and resource sharing functions as outlined in Chapter 1 and 2. Unstructured overlay networks are better suited for establishing a general overlay since they only provide the means of organizing the overlay topology and providing connectivity between the separate nodes.

Besides all the functions which have been enabled by multi-agent and overlay systems, new kinds of network security threats have been introduced [Jan00, Wal02, Cam04, MK06] as well. These new threats are much more difficult to address because of the composition and distributed nature of these systems. Malicious or selfish nodes can distract, disturb, obstruct and impede the correct execution of overlay network often with little effort [Lun00, Dou02, HPJ03]. In an unstructured overlay network, every entity is equal. Every entity provides the same set of functions. In a centralized system, this is restricted to a selected few. This fact necessitates global protection of all entities and their interactions.

The main contribution of this chapter is to first enable multi-agent systems to work in multi-hop environments and second to provide means to do that in a secure manner. Our work provides a comprehensive solution for building a Java-based multi-agent-system with a secure overlay communication layer. We used the two existing systems, Java Agent Development Environment (JADE) and the Secure overlay networking framework (SePP), and integrated them using the JADE communication interface. Out of the box the communication in JADE is based on Remote Method Invocation (RMI), which only guarantees end-to-end security via SSL encryption. Our approach relies on a overlay system to guarantee not only authentication, integrity and confidentiality in direct-connected networks but also in multi-hop environments. SePP as described in Chapter 7 is based on a scalable security concept that allows to adjust the security measures according to the needs and capabilities of participating devices. In addition, it also provides secure routing algorithms which are designed to counter outsider attacks and allow for the detection of misbehaving legitimate nodes.

In the following sections, we present how the JADE agent middleware and the SePP framework can be combined to develop a secure multi-agent-system for multi-hop environments. We briefly outline the design and implementation of SePP. Thereafter, we present the messaging in JADE and the default network implementation. Our implementation is concisely described with a focus on the specific solutions such as the message dispatching or the transparent proxy generation. At last we provide results and a short benchmark which compares vanilla JADE and our implementation.

8.2 Motivation

MAS technology relies heavily on the existence of a network infrastructure. Unfortunately, in case of an emergency it can not be assumed that an infrastructure bound network is fully working. Emergencies can occur in isolated regions which lack the necessary infrastructure such as comprehensive wired or wireless network coverage. Also, a disaster which caused the emergency could have destroyed or disrupted the required infrastructure. The absence of a functioning static network infrastructure necessitates that crucial information for on-site emergency response must be made available through mobile ad-hoc networks. Ideally, this mechanism is backed by fall-back communication facilities such as GSM, UMTS, satellite communication, and TETRA. The information required to respond properly in case of emergencies usually consists of confidential data including electric grid maps, floor plans or even privacy sensitive personal information that should only be made available to authorized personnel. Thus, the provision of network connectivity, as well as managing access to confidential data during the emergency response operation is a substantial network security challenge.

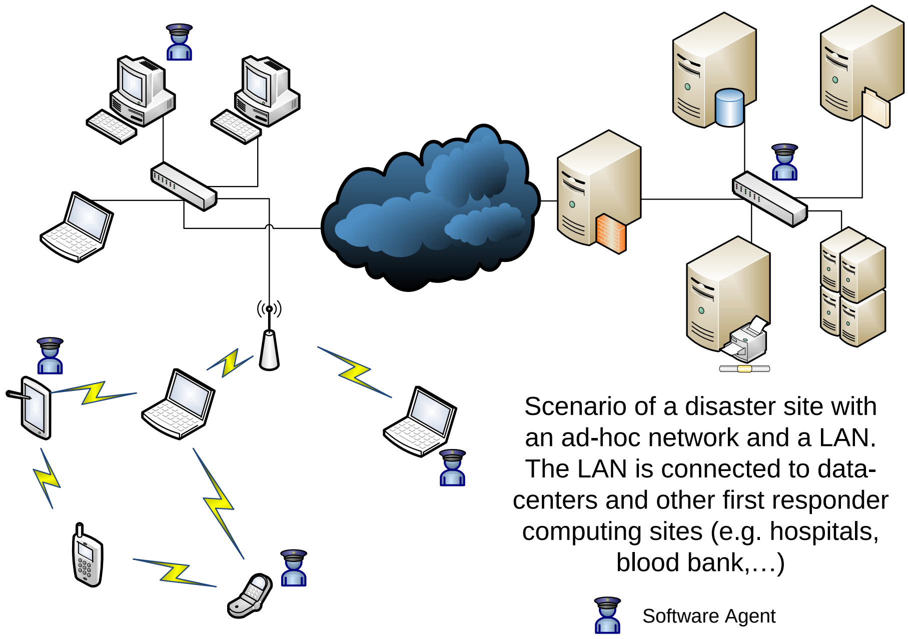

Scenario of a disaster site with an ad-hoc network and a LAN. The LAN is connected to data-centers and other first responder computing sites (e.g. hospitals, blood bank,...)
Software Agent

Figure 8.1: Example of a disaster response scenario with on-site equipment

To illustrate the applicability of the secure overlay network based agent system introduced in this paper, we describe a possible scenario. The object of our scenario is a mine complex with several stakeholders. For crisis operations it is essential that all relevant documentation is made available to the emergency services and on-site personal.

One conceivable emergency situation in a mine complex is the collapse of several tunnels. A solution that provides access to all relevant information pertaining to the affected mine(s) such as emergency plans, legal documents, and reports of mining activities, as well as topographical and cartographic material is necessary. By applying the multi-agent system concept using a secure overlay networking framework as communication layer, it becomes possible to provide a secure decentralized solution to that problem.

During an emergency different organizations have to cooperate. This includes fire and rescue, medical, and police, as well as other emergency services such as mine rescue, or utility services. Each service has its own information infrastructure including hard- and software, and employs different security mechanisms. As a common characteristic, we assume that if security measures exist, they rely on cryptographic keys and functions.

8.3 JADE Multi-Agent System

The Java Agent Development Framework (JADE) is a middleware which simplifies the development of FIPA-compliant agents. The Foundation of Intelligent Physical Agents (FIPA) is an IEEE Computer Society standards organization that promotes agent-based technology and the compatibility of its standards with other technologies. FIPA specifications represent a collection of standards which are intended to promote the compatibility of heterogeneous agents and the services that they can represent. FIPA compliant agents even if they are running on different Agent Management Systems are able to communicate with each other. JADE provides support for server and desktop computers and constrained devices. It has been designed under consideration of scalability and supports it throughout the complete development cycle. In the following overview of the JADE MAS the terms `node` and `peer` refer to specific objects of an MAS which provides communication capabilities and should not be confused with a node or peer of an overlay network.

8.3.1 JADE Background

The Agent Management Service (AMS) of JADE is responsible for creating and terminating Agents and is itself implemented as an Agent. This Agent exists once per Agent Platform, it is running on the so called main agent container. An agent container is a multi-threaded execution environment consisting of one thread per agent including some system threads. To manage the services and slices[1] JADE uses the Directory Facility (DF). Similarly to the AMS the DF is implemented as a single Agent running on the main container. An Agent platform describes a set of one or more Agent Containers, each running multiple Agents. Every platform has one main container, where the AMS and DF of the platform are running. All Agents in JADE are running in so called containers. A container is the runtime environment of one or more Agents. It is responsible for scheduling its Agents and provides them interfaces for communication. Usually one container is running on one machine. The first container of an Agent platform is its main container and has the AMS as well as the DF running on it. JADE uses nodes to establish the communication between Agents of different containers. Every container has its own node, which is an object holding all information needed to send messages to this container. If an containers Agent wants to send a message to an other Agent on a different container, it first has

[1] Is that part of a service that is deployed at a given network node

to get those containers node. This obtained node is a proxy object to the node on the receiving container. For communication between Agents on the same container, its own node is used. Agents can use services to distribute computing. Services are listed at the DF. There Agents can request a service and use it. A service depends of one or more slices. Slices are particular jobs, which get processed by single Agents. So if an Agent uses a service, the computation is split into slices and gets executed by multiple other Agents. JADE uses the Inner Message Transportation Protocol for communication. IMTP describes the communication within JADE and the related classes. It is an heading for the whole inter-container communication.

8.3.2 JADE RMI

The default implementation of JADE is based on Remote Method Invocation (RMI) for the purpose of inter-agent communication. RMI is Java's own kind of remote procedure call in order to provide method invocation over a TCP/IP network. The main properties of RMI are that there is central entity,called the RMI-Registry which binds methods in order to make them callable over the network. RMI has a server/client architecture and many clients which access the server methods. The main-goal while designing RMI was to easily provide an architecture where code can dynamically be loaded from a server (e.g. for updating purposes). This is achieved with this client/server- model and serialization. The RMI mechanism itself provides a "black-box" behavior to the programmer; he does not have to care about marshalling, un-marshalling and transferring over the network. Furthermore from a programmers view it is not required to distinguish between a local and a remote object. The RMI registry is created in the *RMIIMTPManager* class in the `exportPlatformManager()` method and the *PlatformManager* is exposed via the `bind()` call. On the client side the `createPlatformProxy()` method calls `lookup()` and establishes the connection to the main peer. To summarize, we can say that there is a well defined Internal Message Transport Protocol (IMTP) interface for implementing our own network layer, but this interface is heavily RMI-orientated. To provide an own IMTP implementation it is necessary to rebuild the messaging system in a RMI fashion. This results in heavy usage of *Proxy* objects for communication.

8.3.3 JADE Messaging

The process how messages are sent is very important to understand how to implement the IMTP-Layer. This section describes the general sequence of how messages are transmitted inside the JADE Platform. In the following we have to distinguish between the *main peer*, which is responsible for managing the platform with its nodes and services and the *remote peer(s)* which uses the main peer's interfaces through *Proxies*.

The first step to create an Agent-Platform is to create a main peer. This is done by executing the `Boot` class. Afterwards a new *IMTPManager* (RMI or SecureP2P) is created and the PlatformManager object is passed to the `exportPlatformManager()` method to expose the JADE Platform. To connect to the Platform we start another instance of JADE. The address of the main peer is passed to the `createPlatformProxy()` method and the remote proxy to the PlatformManager is established. With this proxy the remote peers are able to access all the functions available in the Platform, e.g. creating Nodes, Slices, etc.

Now, since we have a connection between the remote and the main peer, we create a Node, which is the communication mechanism inside the Agent-Platform between the agents. The node is created via the `addNode()` method of the *PlatformProxy* during the creation of the Agent-Container. From now on we have two local nodes, each on one peer. Each node is associated with an array of services running on the node. The services are managed by the PlatformManager on the main peer with the `findSlice()` and `findAllSlices()` methods. To send a message from the remote peer to the main peer the remote peer asks the PlatformManager via the PlatformProxy which services run on the local node of the main peer. During this request the main peer serializes his local node and the remote peer receives a proxy to the local node running on the main peer. With this proxy the remote peer is able to access the remote node and from now on the two Agent-Containers (e.g. the Agents running on the Container) have the ability to communicate. The main part of the communication is done by the `accept()` method of the nodes, which receives a command as parameter. For an example of the JADE messaging process see figure 8.3.

8.4 Secure Multi-Agent System

Based on the general overview in the previous section, we now present our implementation of a secure messaging mechanism. For an architectural overview of our implementation see figure 8.2. The first step was to emulate the RMI behavior on top of SePP using different message classes. A message class is a generalization of a method call. The knowledge about which method should be called and how it is parameterized is encapsuled within the concrete message classes. We require two message classes per method. First, a request message class, whose payload is the parameters of the method. Second, a response message class, which contains the return value of the corresponding method. We chose this design, because it allows convenient message dispatching and because this architecture is extendible.

8.4.1 Implementation

The *SecureP2PIMTPManager* is the general entity that advertises the agent platform and it provides the remote peers with access to the main peer. The functionality of the *SecureP2PIMTPManager* is comparable to the RMI registry as mentioned above, with the difference that we do not have a special storage facility, where the remote methods are registered. The main peer listens for incoming requests from the remote peers.

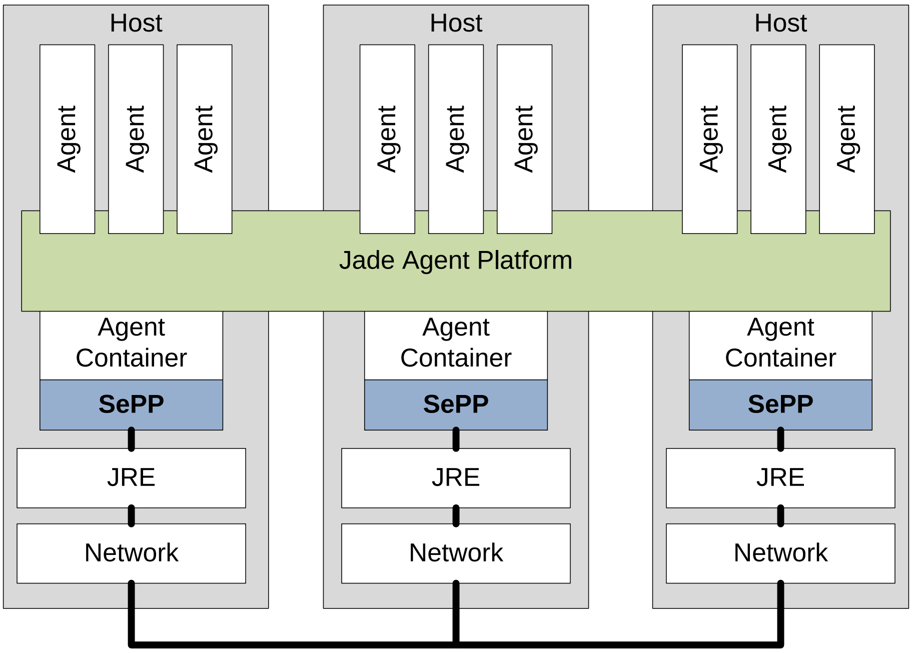


Figure 8.2: Secure Jade Architecture

The Agent-Platform is exposed via `exportPlatformManager()`, called on the main peer. After calling this method, the peer listens to incoming requests from remote peers. The remote peers connect to the main peer through the `createPlatformManagerProxy()` method which receives the peer ID of the main peer as an argument. `createPlatformManager()` is a factory-method that creates a *GetPlatformManagerRequestMessage* which is sent to the main peer. The main peer receives the message and creates a response using the *GetPlatformManagerResponseMessage* object. In the response message the object ID and the name of the *PlatformManager* is stored. With this information the *PlatformProxy* on the remote peer can be created. From now on every communication made between the two peers is done via the PlatformProxy.

As an example we can look at the behavior of the `addNode()` method which is called when a new Agent container is added. When called on from a PlatformProxy the `addNode()` method creates a new *AddNodeRequestMessage* with the object ID of the remote PlatformManager as a destination. This message is sent to the main peer. On the main peer the message is dispatched to the local PlatformManager where the message gets executed with the given parameters and a new container is created. The return-value of the method is put into an AddNodeRequestMessage and sent to the remote peer. When received the return value is passed to the JADE middleware on the remote peer. When the PlatformProxy is a proxy for a local object the same message is build but not sent into the network but is dispatched locally. The decision whether the message is sent into the network or dispatched locally is done at the `send()` method of the *SecureP2PPeer*. The second sending concept inside the JADE middleware, sending from *NodeProxy* to a *LocalNode* is a bit different, because of the different proxy generation. NodeProxies and LocalNodes are held by a *SecureP2PNode* and when the `writeObject()` method during serialization is called the member object is changed from a normal LocalNode to a NodeProxy. The sending mechanism itself is identical in sending via the PlatformProxy. The corresponding message to the method is created, sent and invoked and the return value is sent via the response message.

The class *SecureP2PPeer* is the interface to the SePP network through the SePP framework API. This API allows classes which implement the *Component* interface to register themselves to received messages with a specified message type.

If a message is received at the peer the `receivedMessage()` method is called. In this message the received byte-stream is unmarshalled and the *JadeMessage* is created. Afterwards a new thread is created. In the thread the `dispatchMessage()` method is called where the received and unmarshalled message is passed to the receive method of the registered handler. The thread is required because the receiving messages can send new messages before the method returns. The dispatching functionality waits on the sending semaphore 8.4.3 and the dispatcher cannot dispatch more messages, the system is deadlocked. On the other hand all receiving calls do not depend on a given state; the order of the incoming messages has to be ensured in the Agent code and not in the dispatching mechanism. The advantage in this design approach is that the dispatch mechanism is completely independent from different message types and different handlers. New messages or handlers can register themselves within SecureP2PPeer and the dispatching functionality does not need to be changed.

8.4.2 Object-to-object concept

Each object, that sends and receives JADE-messages, has a unique object ID. Furthermore every Proxy class (SecureP2PNode after Serialization and PlatformProxy) has the remote object ID of its corresponding local entity. Every message passed from a proxy to a local entity has a defined destination address (stored in the proxy) and a defined destination object. We need this information because it is possible to create multiple containers on a given peer. Without this object-to-object concept we cannot distinguish which node is the receiver of the message. With the object-to-object concept we can dispatch the message directly to the object which handles the request associated with the given message.

8.4.3 Sending semaphore

In this section we present how the asynchronous sending and receiving sequences are mapped to the synchronous, RMI-like method invocations. To turn an asynchronous method call in a synchronous method call we use 2 maps. The key of the maps is always the message ID concatenated with the receiving peer id, which is an unique identifier for the message. In the first map the return value is stored when the asynchronous callback method is called (receive in case of JadeComponents). In the second map the semaphores on which the thread of execution waits, is stored. First the waiting Semaphore with initial value 0; means that the first `acquire` forces the thread of execution to wait; is created and stored in the corresponding maps. Then the message is sent via the asynchronous `sendMessage` method of the SecureP2PPeer class. To transfer the synchronous to the asynchronous call we call the asynchronous `sendMessage` method of the SecureP2PPeer and afterwards the acquire method of the Semaphore is called; now the calling thread waits for the Semaphore to be released. The Semaphore is released after the return value of the message is stored in the return-value map. Now the original thread of execution resumes and returns the value from the map. We have to use Semaphores for this functionality because when the message is delivered to the local peer, the `sendMessage` method is blocking and directly calls the dispatching method of the corresponding JadeComponent. As a result the return value is already stored before we call acquire on the Semaphore. If we use a mutual exclusion algorithm the wakeup call would be lost in this case, with Semaphores the thread resumes its execution.

8.4.4 NodeProxy generation

The difference between NodeProxy and a PlatformProxy is that the PlatformProxy is generated explicit whereas the NodeProxy is created implicit through serialization during the messaging mechanism of JADE. This means that we have to transparently switch between LocalNode and NodeProxy. To switch between the nodes we created a class SecureP2PNode which holds the Nodes as an aggregation. When the SecureP2PNodes `writeObject()` method is called during serialization the node is changed from a LocalNode to a ProxyNode. Afterwards the default `writeObject()` method is called and the current state is serialized. Thereafter the LocalNode is restored.

8.4.5 A message sending sequence

This section combines the concepts discussed so far and illustrates a sample communication sequence of the JADE platform. We chose he method *accept* for an illustration example of a message sending sequence. *Accept* is used by the agents for communicating within the middleware. We presume that the *PlatformProxy* was already created and the two nodes are ready to send.

The sequence diagram depicted in figure 8.3 illustrates the whole messaging process, starting from creating the *NodeProxy* until the return of the method *accept*. In the first step the *NodeProxy* is created after serializing the SecureP2PNode from higher layers of the JADE middleware using a mechanism called *transparent proxy generation*. Afterwards the *accept* call on the SecureP2PNode is delegated to its Proxy. This proxy creates a request message

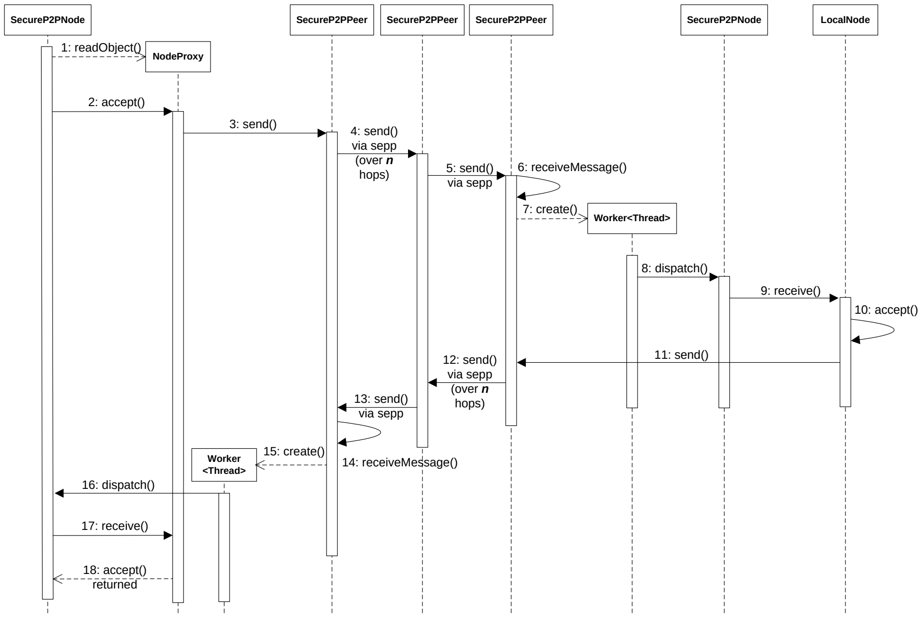


Figure 8.3: Message sequence

and sends it via the SecureP2PPeer class. On the remote container the message is received and a new worker thread is created. This new worker thread forwards the message to its corresponding *JadeComponent*, in this case we have a *LocalNode*. The next step is to invoke the method related to the message and forward the return value via the SecureP2PPeer. Back on the main peer the message is dispatched and the original emphaccept call returns.

8.5 Security Analysis

The main contribution of our work is to first enable multi-agent systems to work in multi-hop environments and second to provide means to do that in a secure manner. To establish the exact boundaries of our security analysis we first define our assumptions on the environment.

1. Each peer or user that is a part of the multi-hop enabled communication subsystem must be authenticated. This means, each user joining the system must provide a prove of it's identity depending on the security requirements of the overall system.

2. A multi-hop enabled MAS must provide means to communicate even in the absence of direct connections between the different parts of the MAS. Thus, it must be possible for agents to communicate with each other in a hop-by-hop manner in addition to direct communication. This fact requires secure

routing algorithms which guarantee that only legitimated and trusted hops are used to forward data.

3. The data communication of the MAS must be protected. Depending on the system security requirements this can include protection from fabrication, modification, interruption and interception. This translates to data authentication and data confidentiality in addition to reply protection.

To conclude the list of assumptions it has to be remarked that it is always important to scale the security mechanisms according to the potential damage. Every user should use the best available security measures given his resources.

8.5.1 Analysis of the SePP-Jade solution

The join process in SePP is secured and provides authentication based on the Needham-Schroeder-Lowe protocol. The different secure approaches used are shared secrets and public/private keys (pre-configured or TPM hosted). This guarantees that only legitimated peers can join and participate in the overlay network.

After peers have joined the SePP network, a secure routing algorithm can be used to guarantee the integrity of the network. With this secure routing algorithm it is possible to protect against outsider and insider attacks as mentioned in Section 6.3.1, depending on what security level is used. The secure overlay network also provides means for data authentication and confidentiality using well-known state-of-the art cryptographic primitives.

8.6 Performance evaluation

To benchmark our implementation we compared it with JADE's *out-of-the-box* RMI implementation. We used the *PartyAgent* application from the JADE examples. Within this application we created 500 *PartyGuest* agents which send several messages to each other. We measured the time from the start of the application until all messages are sent, and the party has officially ended. During this time about 7000 messages have been sent and received from the agents. The party host agent is responsible for about 99% of the messages. We have tested our implementation with the main and remote peer connected directly and also with one intermediate hop between them. The intermediate peer show how much delay is introduced per hop on the route to the remote peer.

The tests have been performed on HP personal computers with Intel Core 2 Duo E8600 processors with 3.33 GHz and 4 GB RAM and Windows 7 as operating system. The SePP framework implementation has been executed on Java JDK 6 Update 17 runtime environments.

The values in Table 8.1 have been obtained from different runs of the *PartyAgent* application. These values show how long one specific run of the application took. In the first column the values from the vanilla JADE version using RMI

	RMI [s]	SePP (direct) [s]	SePP (1 hop) [s]
1	4.40	15.40	16.20
2	3.70	14.80	16.90
3	3.40	14.30	15.90
4	3.60	16.10	15.80
5	3.50	14.00	15.40
6	3.90	14.60	16.10
7	3.40	13.40	15.40
8	3.70	15.00	15.80
9	3.30	16.00	15.30
10	3.40	14.30	16.00
Mean	3.63	14.75	15.88

Table 8.1: Processing time of the different security levels at the participating peers

without any security features. The next two columns show the amount of time it took for the same application to finish using SePP with security level *medium* as communication layer. The first one has been obtained for the case that the two peers have a direct connection. The second one depicts the case that there is one intermediate hop between the main peer and the remote peer. The run time of the JADE version using SePP is about four to five times slower than the RMI version. This fact can be attributed to increased processing time for cryptography and the overlay network management and communication overhead. The usual processing time in SePP without transmission latency is about 500μs. Thus, sending and receiving 7000 messages alone would account for 3.5 seconds, which already is the mean run time of the RMI version.

8.7 Related Work

Multi-agent systems have been used in disaster response scenarios previously. For instance disaster response [GGGR03], [SMTS05] have shown that MAS can be quite helpful under such circumstances. But these approaches have not addressed security or multi-hop communication requirements in anyway. They where only concerned with showing the features MAS can provide in disaster response.

The JADE developers itself have proposed a security extension for its framework [JAD05]. Anyhow, this security framework is only intended as add-on for JADE and therefore does not address all requirements for security in such challenging environments. Also their extension only provided interfaces for JAAS (Java Authentication and Authorization Service) and they did not implement any security itself or give instructions on how to use it. There exist also some other works which have addressed security in JADE. But they are all theoretical and only outline the requirements and discuss the necessary security features formally. One such effort is [VSR07].

Several other works have been concerned with the security of multi-agent systems. Almost all of them have been only of theoretical nature. They have outlined the requirements in terms of security and shown what attacks and threats are possible with in the domain of MAS. The most prominent such work is [Jan00].

8.8 Conclusion

In conclusion, the proposed multi-agent system with SePP as underlying communication infrastructure enables the use of agent technology in multi-hop environments in a secure way. We provided simple means of integrating and enhancing existing MASs with secure communication mechanisms without the need for re-design or re-implementation of the MAS itself. We introduced an RMI-style interaction layer which mediates between the MAS on top of a secure overlay networking framework. The security management is separated from the MAS application and can be adjusted according to the needs of the participating entities. With our solution it is possible to comply with various security requirements in a fine grained manner since it is possible to select security levels from a global to a group scale. The introduced security guarantees increased robustness and the added multi-hop functionalities justify the marginal negative impact on communication performance compared to JADE's RMI solution. Furthermore, we believe that our solution has the potential to increase the efficiency of emergency response operations for scenarios where an existing network infrastructure has been destroyed or disrupted and the different parties had to rely on proprietary or fall-back communication facilities.

9
Conclusion

Interesting phenomena occur
when two or more rhythmic
patterns are combined, and these
phenomena illustrate very aptly
the enrichment of information
that occurs when one description
is combined with another

Gregory Bateson

In this thesis, we examined overlay networks in terms of their security and communication capabilities and by designing our own secure overlay networking framework. The first objective was to study existing security solutions and frameworks for different kinds of networking and to identify their general requirements, concepts and architectures in order develop a security concept suitable for heterogeneous overlay networks. Next we have evaluated existing solutions for overlay networking regarding their routing, communication and security capabilities and suitability to be used as building blocks for developing secure overlay communication protocols and finally a secure overlay networking framework.

During this part of the work, we had to identify an area of networking which was similar enough to overlay networking and already has an large established base of routing protocols and ideally also secure ones. It turned out that this area would be ad-hoc networks. All the properties of mobile ad-hoc networks can also be found in overlay networks but overlay networks incorporate also other types of networking not found in mobile ad-hoc networks. However, the mobile ad-hoc networks characteristics such as mobility, heterogeneity, self-organization, and peer-to-peer are also the dominant ones in general overlay networks. Thereafter, we evaluated several of the most common routing protocols which incorporated different routing philosophies in order to find the most suitable candidate to be used as building block. During this search we identified the Dynamic Source Routing (DSR) protocol as our basis due to its simple architecture, on-demand nature, easy administration and suitability for security adaption. Another objective, which clearly crystalized during the investigation of existing solutions for overlay networking, was the inherent problems of identity management in overlay networks and the lack of adequate solutions. Thus, we also increased our efforts to find and develop a solution which both was compatible with our security concept and could be realized with current technologies whether they were in software or hardware. After these steps we were in the possession of the basic building blocks to design and develop secure protocols for routing and communication in overlay networks. We first evaluated one existing secure routing protocol which was also built upon DSR and thereafter developed our own protocol which incorporates the ideas expressed in the security concept. The last objectives were to incorporate these findings and protocols into a comprehensive solution which provides security for overlay networking and to evaluate and test this solution.

In Chapter 4 we presented a security concept for these overlay network. The security concept provides recommendations and requirements for three different aspects namely for establishing, performing and upholding secure communication in overlay networks. We outlined possible security levels within each of these aspects and specify what kind of information and secrets must be available to realize it. In the end we presented the security concept and the specific levels we applied in the secure overlay networking framework we developed. Chapter 5 outlined the problems of identity management in self-organizing networks and presents two particular solutions which are based on Trusted Computing principles and mechanisms. We presented a trusted authentication protocol for overlay networks which makes use of the public key certificate and the private key, called endorsement key, which are stored in and protected by the Trusted Platform Module. With the first variant of the trusted authentication protocol the Trusted Platform Module and the contained credentials allow us to establish unique and verifiable identities which can not be forged and guarantee that each node can only posses one identity. However, since in this solution the unique and long-term endorsement key is used this solution may not be appropriate in open large-scale solutions with a public or mixed user base since this key could be used to exploit the privacy of the users. Therefore, we present a second solution which facilitates the PrivacyCA concept in order to generate anonymous independent short-term certificates and keys which are bound to the endorsement key. These anonymous credentials do not contain any information which would allow others than the PrivacyCA to associate them with specific endorsement keys. The nodes can obtain these anonymous credentials from the PrivacyCA one at a time and therefore do also guarantee that each node possesses only one identity which is unique. These anonymous credentials are again protected by the Trusted Platform Modules and can be verified by its inherent mechanisms.

We provide a performance evaluation of both versions of the trusted authentication protocol and compare it with a traditional approach which is based on a common authentication protocol using asymmetric cryptography. In Chapter 6 we present an analysis of the best known secure variant of the DSR protocol called Ariadne. We outline the strengths and the shortcomings of this protocol if used in overlay networks. Thereafter, we introduce our solution which is also based on DSR and incorporates the ideas of the security concept. Thus, our secure routing protocol provides not only security in the routing process but also allows to select specific levels of security as outlined in the concept. We evaluated the performance and security of our protocol in a real world scenario as part of our secure overlay networking framework which is used in three European research projects. In Chapter 7 we present the architecture and implementation of the before mentioned secure overlay networking framework. We provide a comprehensive evaluation of its communication capabilities in different networking scenarios using a overlay networking simulator. Chapter 8 presents a specific application of our secure overlay networking framework as it is used in one of the research projects. In this application our framework functions as secure communication infrastructure on top of which an Agent Platform is operating. In this specific case we not only show the applicability of our framework in real world scenarios but also how simple our framework can be used with existing applications, or in this case even an entire agent development platform (JADE).

The main conclusion of this thesis is that security can be provided for overlay networks using already available or eminent technologies without too much negative impact on the efficiency. The major problem of previous solutions was that security was applied only to particular protocols or mechanisms of overlay networks. Thus, either leaving any comprehensive solution, such as an overlay networking framework, still vulnerable to attacks and failures or providing solutions which are only applicable under very narrow assumptions and therefore are not applied in real world scenarios. We showed that with careful design of an overall security concept and later development of the integral components of an overlay network - identity management, secure route establishment and management, and secure communication and protection against misbehavior - based upon this concept it is possible to provide an applicable solution. We want also mention that most of the problems of overlay networks are related to its distributed self-organizing manner but could be solve much more easily if more efficient operational processes, especially in the area of identity management and key distribution, would be available. We evaluated the performance and applicability of our solution as stand-alone application in real world projects and also presented an example where our framework can be used as secure communication basis for existing applications. Our solution should be used in closed environments if the keys and secrets must be distributed manually and can be used in open systems if TPMs are available for providing identities and cryptographic material. Due to the use of cryptographic protocols and its capability to adjust to very dynamic environments however it is not recommended to use our framework in scenarios where the goal is to provide content distribution for a very large user base.

Bibliography

[AAC$^+$05] Amitanand S. Aiyer, Lorenzo Alvisi, Allen Clement, Mike Dahlin, Jean-Philippe Martin, and Carl Porth. BAR fault tolerance for cooperative services. *SIGOPS Oper. Syst. Rev.*, 39(5):45–58, 2005.

[AD01] Karl Aberer and Zoran Despotovic. Managing trust in a peer-2-peer information system. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 310–317. ACM, 2001.

[ADH05] Karl Aberer, Anwitaman Datta, and Manfred Hauswirth. A decentralized public key infrastructure for customer-to-customer e-commerce. *International Journal of Business Process Integration and Management*, 1(1):26–33, 2005.

[Bel01] Steven M. Bellovin. Security aspects of napster and gnutella. In *Proceedings of the 2001 Usenix Annual Technical Conference.* USENIX, 2001.

[BGH$^+$02] Krista Bennett, Christian Grothoff, Tzvetan Horozov, Ioana Patrascu, and Tiberiu Stef. Gnunet - a truly anonymous networking infrastructure. Technical report, In: Proc. Privacy Enhancing Technologies Workshop (PET, 2002.

[BK05] Rida A. Bazzi and Goran Konjevod. On the establishment of distinct identities in overlay networks. In *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, PODC '05, pages 312–320, New York, NY, USA, 2005. ACM.

[BLP05] Shane Balfe, Amit D. Lakhani, and Kenneth G. Paterson. Trusted computing: Providing security for peer-to-peer networks. In Germano Caronni, Nathalie Weiler, Marcel Waldvogel, and Nahid Shahmehri, editors, *Peer-to-Peer Computing*, pages 117–124. IEEE Computer Society, 2005.

[BRDP10] Danny Bickson, Tzachy Reinman, Danny Dolev, and Benny Pinkas. Peer-to-peer secure multi-party numerical computation facing malicious adversaries. *Peer-to-Peer Networking and Applications*, 3(2):129–144, 2010.

[BS06] S. A. Baset and H. G. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–11, april 2006.

[Cam04] Stefano Campadello. Peer-to-peer security in mobile devices: A user perspective. In *P2P '04: Proceedings of the Fourth International Conference on Peer-to-Peer Computing*, pages 252–257. IEEE Computer Society, 2004.

[CCF08] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. A distributed and adaptive revocation mechanism for P2P networks. In *ICN '08: Proceedings of the Seventh International Conference on Networking*, pages 290–295. IEEE Computer Society, 2008.

[CCR04a] Miguel Castro, Manuel Costa, and Antony Rowstron. Peer-to-peer overlays: structured, unstructured, or both. Technical report, Microsoft Research, Cambridge, 2004.

[CCR04b] Miguel Castro, Manuel Costa, and Antony Rowstron. Performance and dependability of structured peer-to-peer overlays. In *DSN '04: Proceedings of the 2004 International Conference on Dependable Systems and Networks*, page 9. IEEE Computer Society, 2004.

[CCR04c] Miguel Castro, Manuel Costa, and Antony Rowstron. Should we build gnutella on a structured overlay? *SIGCOMM Comput. Commun. Rev.*, 34(1):131–136, 2004.

[CCR05] Miguel Castro, Manuel Costa, and Antony Rowstron. Debunking some myths about structured and unstructured overlays. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 85–98. USENIX Association, 2005.

[CDG+02] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):299–314, 2002.

[CPS03] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, page 197. IEEE Computer Society, 2003.

[Cro69] S. Crocker. Host software. RFC 1, Internet Engineering Task Force, April 1969.

[DFM00] Roger Dingledine, Michael J. Freedman, and David Molnar. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, chapter

Accountability measures for peer-to-peer systems. O'Reilly and Associates, November 2000.

[DH06] Jochen Dinger and Hannes Hartenstein. Defending the sybil attack in p2p networks: Taxonomy, challenges, and a proposal for self-registration. In *Proceedings of the First International Conference on Availability, Reliability and Security*, pages 756–763, Washington, DC, USA, 2006. IEEE Computer Society.

[DHK10] Peter Danner, Daniel Hein, and Stefan Kraxberger. Securing emergency response operations using distributed trust decisions. In *Proceedings of the 2010 Fourth International Conference on Network and System Security*, NSS '10, pages 62–69, Washington, DC, USA, 2010. IEEE Computer Society.

[Dou02] John R. Douceur. The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer-Verlag, 2002.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[DR06] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.1. Technical Report 4346, Internet Engineering Task Force, April 2006. Updated by RFCs 4366, 4680, 4681.

[EG02] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47, New York, NY, USA, 2002. ACM.

[FH02] S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization. `http://www.ietf.org/rfc/rfc3281.txt`, April 2002.

[FRHS09] T. Fujii, Y. Ren, Y. Hori, and K. Sakurai. Security analysis for p2p routing protocols. In *Availability, Reliability and Security, 2009. ARES '09. International Conference on*, pages 899 –904, march 2009.

[GGGR03] J.-P. George, M.-P. Gleizes, P. Glize, and C. Regis. Real-time simulation for flood forecast: an adaptive multi-agent system staff. In *AISB'03: Proceedings of the 3rd Symposium on Adaptive Agents and Multi-Agent Systems*, pages 7–11, 2003.

[GNC$^+$06] Jabeom Gu, Jaehoon Nah, Cheoljoo Chae, Jaekwang Lee, and Jongsoo Jang. Random visitor: A defense against identity attacks in P2P overlay networks. In Jae-Kwang Lee, Okyeon Yi, and Moti Yung, editors, *WISA*, volume 4298 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2006.

[Gra09] David Grawrock. *Dynamics of a Trusted Platform: A Building Block Approach.* Intel Press, Intel Press, Intel Corporation, 2111 NE 25th Avenue, JF3-330, Hillsboro, OR 97124-5961, 1 edition, February 2009. ISBN 978-1934053171.

[GRB03] Tal Garfinkel, Mendel Rosenblum, and Dan Boneh. Flexible OS support and applications for trusted computing. In *In 9th Hot Topics in Operating Systems (HOTOS-IX*, pages 145–150. Usenix, 2003.

[GZ02] Manel Guerrero Zapata. Secure Ad hoc On-Demand Distance Vector Routing. *ACM Mobile Computing and Communications Review (MC2R)*, 6(3):106–107, July 2002.

[GZA02] Manel Guerrero Zapata and N. Asokan. Securing Ad hoc Routing Protocols. In *Proceedings of the 2002 ACM Workshop on Wireless Security (WiSe 2002)*, pages 1–10, September 2002.

[HPFS02] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate and CRL Profile. `http://www.ietf.org/rfc/rfc3280.txt`, April 2002.

[HPJ02] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: a secure on-demand routing protocol for ad hoc networks. In Ian F. Akyildiz, Jason Yi-Bing Lin, Ravi Jain, Vaduvur Bharghavan, and Andrew T. Campbell, editors, *MOBICOM*, pages 12–23. ACM, 2002.

[HPJ03] Yih-Chun. Hu, Adrian. Perrig, and David B. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 3:1976–1986, 2003.

[HPJ05] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: a secure on-demand routing protocol for ad hoc networks. *Wirel. Netw.*, 11(1-2):21–38, 2005.

[HTK10] Daniel M. Hein, Ronald Toegl, and Stefan Kraxberger. An autonomous attestation token to secure mobile agents in disaster response. *Security and Communication Networks*, 3(5):421–438, 2010.

[HZNR09] Kevin Hoffman, David Zage, and Cristina Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Comput. Surv.*, 42(1):1–31, 2009.

[iosat02] National institute˜of˜standards˜and technology. FIPS 180-2, Secure Hash Standard, Federal Information Processing Standard

(FIPS), Publication 180-2. Technical report, DEPARTMENT OF COMMERCE, August 2002.

[JAD05] JADE Board. Jade security add-on guide. Technical report, TILAB S.p.A, 2005.

[Jan00] W. A. Jansen. Countermeasures for mobile agent security. *Computer Communications*, 23(17):1667 – 1676, 2000.

[JD09] B.S. Jyothi and J. Dharanipragada. Symon: Defending large structured p2p systems against sybil attack. In *Peer-to-Peer Computing, 2009. P2P '09. IEEE Ninth International Conference on*, pages 21–30, September 2009.

[JIB07] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

[JLR04] Tingyao Jiang, Qinghua Li, and Youlin Ruan. Secure dynamic source routing protocol. *Computer and Information Technology, International Conference on*, 0:528–533, 2004.

[JM96] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, volume 353 of *The International Series in Engineering and Computer Science*, pages 153–181. Springer US, 1996.

[JMB01] David B. Johnson, David A. Maltz, and Josh Broch. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In *In Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5*, pages 139–172. Addison-Wesley, 2001.

[Joh94] D.B. Johnson. Routing in ad hoc networks of mobile hosts. In *Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on*, pages 158–163, Dec 1994.

[Kau07] Bernhard Kauer. Oslo: Improving the security of trusted computing. In *SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 1–9, Berkeley, CA, USA, 2007. USENIX Association.

[KDH10] Stefan Kraxberger, Peter Danner, and Daniel M. Hein. Secure multi-agent system for multi-hop environments. In Igor V. Kotenko and Victor A. Skormin, editors, *MMM-ACNS*, volume 6258 of *Lecture Notes in Computer Science*, pages 270–283. Springer, 2010.

[KDP05] Klaus Kursawe, Schellekens Dries, and Bart Preneel. Analyzing trusted platform communication. In *ECRYPT Workshop 2005, CRASH - CRyptographic Advances in Secure Hardware*, September 2005.

[KGSW05] Frank Kargl, Alfred Geiss, Stefan Schlott, and Michael Weber. Secure dynamic source routing. *Hawaii International Conference on System Sciences*, 9:320c, 2005.

[KP09a] Stefan Kraxberger and Udo Payer. Secure routing approach for unstructured P2P systems. In *SECUREWARE '09: Proceedings of the 2009 Third International Conference on Emerging Security Information, Systems, and Technologies*, pages 210–216. IEEE Computer Society, 2009.

[KP09b] Stefan Kraxberger and Udo Payer. Security concept for peer-to-peer systems. In *IWCMC '09: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing*, pages 931–936. ACM, 2009.

[KPT08] Stefan Kraxberger, Udo Payer, and Stefan Tillich. General security concept for embedded P2P systems. In *Mobiquitous '08: Proceedings of the 5th Annual International Conference on Mobile and Ubiquitous Systems*, pages 1–5. ICST, 2008.

[Kra10] Stefan Kraxberger. A scalable secure overlay framework for heterogeneous embedded systems. In *LCN*, pages 236–239. IEEE, 2010.

[Kra11] Stefan Kraxberger. Scalable secure routing for heterogeneous unstructured p2p networks. In Yiannis Cotronis, Marco Danelutto, and George Angelos Papadopoulos, editors, *PDP*, pages 619–626. IEEE Computer Society, 2011.

[KSGM03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, New York, NY, USA, 2003. ACM.

[LCP$^+$05] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7:72–93, 2005.

[LMT09] F. Lesueur, L. Me, and V.V.T. Tong. An efficient distributed PKI for structured P2P networks. In *P2P '09. IEEE Ninth International Conference on Peer-to-Peer Computing*, pages 1–10, Sept. 2009.

[LMVTT08] François Lesueur, Ludovic Mé, and Valérie Viet Triem Tong. A distributed certification system for structured P2P networks. In *AIMS '08: Proceedings of the 2nd international conference on Autonomous Infrastructure, Management and Security*, pages 40–52. Springer-Verlag, 2008.

[LN03] Donggang Liu and Peng Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM conference on Computer and communications security*, CCS '03, pages 52–61, New York, NY, USA, 2003. ACM.

[Low95] Gavin Lowe. An attack on the needham-schroeder public-key authentication protocol. *Inf. Process. Lett.*, 56(3):131–133, 1995.

[Low96] Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *TACAs '96: Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, pages 147–166. Springer-Verlag, 1996.

[LPM$^+$05] L. Lazos, R. Poovendran, C. Meadows, P. Syverson, and L.W. Chang. Preventing wormhole attacks on wireless ad hoc networks: a graph theoretic approach. In *Wireless Communications and Networking Conference, 2005 IEEE*, volume 2, pages 1193 – 1199 Vol. 2, march 2005.

[LS06] Huaizhi Li and Mukesh Singhal. A secure routing protocol for wireless ad hoc networks. In *HICSS '06: Proceedings of the 39th Annual Hawaii International Conference on System Sciences*, page 225.1. IEEE Computer Society, 2006.

[Lun00] Janne Lundberg. Routing security in ad hoc networks. Technical Report 501, Helsinki University of Technology, 2000.

[LZPL05] Jingfeng Li, Yuefei Zhu, Heng Pan, and Shengli Liu. A distributed certificate revocation scheme based on one-way hash chain for wireless ad hoc networks. In *Mobile Technology, Applications and Systems, 2005 2nd International Conference on*, pages 5 pp.–5, Nov. 2005.

[MGLB00] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom '00, pages 255–265, New York, NY, USA, 2000. ACM.

[Mil67] Stanley Milgram. The Small World Problem. *Psychology Today*, 2:60–67, 1967.

[Min08] Miniwatts Marketing Group. Internet growth statistic. Internet, December 2008.

[MK06] Anirban Mondal and Masaru Kitsuregawa. Privacy, security and trust in P2P environments: A perspective. In *DEXA '06: Proceedings of the 17th International Conference on Database and Expert Systems Applications*, pages 682–686. IEEE Computer Society, 2006.

[MM02] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 53–65, London, UK, 2002. Springer-Verlag.

[MSH05] Francisco P. Maturana, Raymond J. Staron, and Kenwood H. Hall. Methodologies and tools for intelligent agents in distributed control. *IEEE Intelligent Systems*, 20(1):42–49, 2005.

[MvOV01] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 2001.

[NS78] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

[Odl03] A. Odlyzko. Internet traffic growth: Sources and implications, 2003.

[PB94] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 24(4):234–244, 1994.

[PBG05] Gauthier Picard, Carole Bernon, and Marie Pierre Gleizes. Etto: Emergent timetabling by cooperative self-organization. In Sven Brueckner, Giovanna Di Marzo Serugendo, David Hales, and Franco Zambonelli, editors, *Engineering Self-Organising Systems*, volume 3910 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 2005.

[PBR99] Charles E. Perkins and Elizabeth M. Belding-Royer. Ad-hoc ondemand distance vector routing. In *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*, pages 90–100, Feb 1999.

[PC97] Vincent D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *INFOCOM '97: Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*, page 1405. IEEE Computer Society, 1997.

[PCST01] Adrian Perrig, Ran Canetti, Dawn Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In *In Network and Distributed System Security Symposium*, pages 35–46, 2001.

[PCTS00] Adrian Perrig, Ran Canetti, J. D. Tygar, and Dawn Xiaodong Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56–73, 2000.

[PCTS02] Adrian Perrig, Ran Canetti, J. D. Tygar, and Dawn Song. The tesla broadcast authentication protocol. *RSA CryptoBytes*, 5:2002, 2002.

[PETCR06] Esther Palomar, Juan M. Estévez-Tapiador, Julio César Hernández Castro, and Arturo Ribagorda. Security in p2p networks: Survey and research directions. In *EUC Workshops*, pages 183–192, 2006.

[PH02] Panos Papadimitratos and Zygmunt Haas. Secure Routing for Mobile Ad hoc Networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, pages 27–31, 2002.

[PL07] Radha Poovendran and Loukas Lazos. A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks. *Wirel. Netw.*, 13(1):27–59, 2007.

[PST$^+$02] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. Spins: security protocols for sensor networks. *Wirel. Netw.*, 8(5):521–534, 2002.

[PSZ08] Krishna Puttaswamy, Alessandra Sala, and Ben Y. Zhao. Searching for rare objects using index replication. In *INFOCOM 2008. 27th IEEE International Conference on Computer Communications*, pages 1723–1731, April 2008.

[PTHD09] Martin Pirker, Ronald Toegl, Daniel Hein, and Peter Danner. A privacyca for anonymity and trust. In *Proceedings of the 2nd International Conference on Trusted Computing*, Trust '09, pages 101–119, Berlin, Heidelberg, 2009. Springer-Verlag.

[RD01] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Lecture Notes in Computer Science*, pages 329–350, 2001.

[RFH$^+$01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172. ACM, 2001.

[RGRK04] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling churn in a DHT. In *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 10–10. USENIX Association, 2004.

[Riv92] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321 (Informational), April 1992. Updated by RFC 6151.

[RKS02] Erik Riedel, Mahesh Kallahalla, and Ram Swaminathan. A framework for evaluating storage system security. In *FAST '02: Proceedings of the 1st USENIX Conference on File and Storage Technologies*, page 2. USENIX Association, 2002.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[SGS03] Stuart E. Schechter, Rachel A. Greenstadt, and Michael D. Smith. Trusted computing, peer-to-peer distribution, and the economics of pirated entertainment. In *In The Second Workshop on Economics and Information Security*, pages 29–30, 2003.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22:612–613, November 1979.

[SM02] Emil Sit and Robert Morris. Security considerations for peer-to-peer distributed hash tables. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 261–269. Springer-Verlag, 2002.

[SMK$^+$01] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM, 2001.

[SMTS05] Nathan Schurr, Janusz Marecki, Milind Tambe, and Paul Scerri. The future of disaster response: Humans working with multiagent teams using defacto. In *In AAAI Spring Symposium on AI Technologies for Homeland Security*, 2005.

[Sta99] William Stallings. *Network Security Essentials: Applications and Standards.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1999.

[Tru] Trusted Computing Group. TCG infrastructure specifications.

[Tru07a] Trusted Computing Group. TCG Credential Profiles Specifications (Version 1.1, rev 1.014). `https://www.trustedcomputinggroup.org/specs/IWG`, May 2007.

[Tru07b] Trusted Computing Group. TCG software stack specification, version 1.2 errata a. `https://www.trustedcomputinggroup.org/specs/TSS/`, 2007.

[Tru07c] Trusted Computing Group. TCG TPM specification version 1.2 revision 103, 2007.

[Tru07d] Trusted Computing Group. TPM main specification level 2 version 1.2, revision 103. `https://www.trustedcomputinggroup.org/specs/TPM/`, 2007.

[VDPBS02] H. Van Dyke Parunak, Sven Brueckner, and John Sauter. Digital pheromone mechanisms for coordination of unmanned vehicles. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 449–450. ACM, 2002.

[vO03] P. van Oorschot. Revisiting software protection. In *ISC 2003: Proceedings of the 6th International Information Security Conference*, pages 1–13. Springer, 2003.

[VSR07] X. Vila, A. Schuster, and A. Riera. Security for a multi-agent system based on jade. *Computers and Security*, 26(5):391 – 400, 2007.

[Wal02] Dan S. Wallach. A survey of peer-to-peer security issues. In *In International Symposium on Software Security*, pages 42–57, 2002.

[WS98] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

[WZH05] Honghao Wang, Yingwu Zhu, and Yiming Hu. An efficient and secure peer-to-peer overlay network. In *LCN '05: Proceedings of the The IEEE Conference on Local Computer Networks 30th Anniversary*, pages 764–771. IEEE Computer Society, 2005.

[YKGF06] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. Sybilguard: defending against sybil attacks via social networks. *SIGCOMM Comput. Commun. Rev.*, 36:267–278, August 2006.

[YNK01] Seung Yi, Prasad Naldurg, and Robin Kravets. Security-aware ad hoc routing for wireless networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 299–302, New York, NY, USA, 2001. ACM.

[YW02] William Yeager and Joseph Williams. Secure peer-to-peer networking: The jxta example. *IT Professional*, 4(2):53–57, 2002.

[Zap02] Manel Guerrero Zapata. Secure ad hoc on-demand distance vector routing. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(3):106–107, 2002.

[ZH99] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE Network Magazine*, 13:24–30, 1999.

[ZKJ01] Ben Y. Zhao, John Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, UC Berkeley, 2001.

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am ……………………………	…………………………………………………..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

……………………………	…………………………………………………..
date	(signature)