Aus dem Institut für Telematik
der Universität zu Lübeck

Direktor:
Prof. Dr. rer. nat. Stefan Fischer

Managing Security and Dependability in Ubiquitous Computing Environments

Inauguraldissertation
zur
Erlangung der Doktorwürde
der Universität zu Lübeck
– Aus der Technisch-Naturwissenschaftlichen Fakultät –

Vorgelegt von

Herrn Dipl.-Wirt.-Inform. Stefan Ransom

aus Braunschweig

Lübeck, im September 2009

Erster Berichterstatter:	Prof. Dr. rer. nat. Stefan Fischer
Zweiter Berichterstatter:	Prof. Dr.-Ing. Lars Wolf

Tag der mündlichen Prüfung: 15. Januar 2010

Zum Druck genehmigt.
Lübeck, den 15. Januar 2010

To my family, for a lifetime of love and support.

Danksagung

A journey of a thousand miles begins with a single step.
[Lao-tzu]

Die vorliegende Dissertation wurde im Institut für Telematik der Universität zu Lübeck unter der Leitung von Herrn Prof. Dr. Stefan Fischer verfasst. Diese Arbeit wäre ohne die Unterstützung einiger wichtiger Personen nicht möglich gewesen, bei denen ich mich an dieser Stelle ausdrücklich bedanken möchte

In erster Linie möchte ich mich bei meinem Doktorvater Herrn Prof. Dr. Stefan Fischer für die Bereitstellung des Arbeitsplatzes, die Möglichkeit zur Promotion sowie die Betreuung meiner Arbeit bedanken. Des Weiteren möchte ich mich bei Herrn Prof. Dr. Lars Wolf für die Übernahme des Koreferats bedanken.

Mein besonderer Dank geht weiterhin an alle aktuellen und ehemaligen Mitarbeiter am ITM, Claudia Becker, Dr. Carsten Buschmann, Dr. Alexander Carôt, Maick Danckwardt, Nils Glombitza, Prof. Dr. Horst Hellbrück, Daniela Krüger, Martin Lipphardt, Dr. Dennis Pfisterer, Stephan Pöhlsen, Peter Rothenpieler, Dirk Schmidt, Birgit Schneider, Dr. Axel Wegener und Prof. Dr. Christian Werner, insbesondere für die tolle Zusammenarbeit, die wertvollen Diskussionen und Anregungen, und nicht zuletzt auch die schöne Zeit – ich werd's vermissen!

Nicht zuletzt gilt mein Dank meiner Frau und meinen Kindern, die mich während der stressigen Zeit des Niederschreibens ertragen mussten und mich immer wieder aufgebaut haben wenn ich mal wieder den Kopf in den Sand stecken wollte. Ohne die warmen Worte, lieben Gesten und das Lachen in Euren Gesichtern wäre die Zeit deutlich trüber gewesen. Dank Euch von ganzem Herzen — ich liebe Euch!

Kurzfassung

Die Themen Sicherheit und Zuverlässigkeit spielen in Computernetzen und verteilten Systemen seit jeher eine wichtige Rolle. Insbesondere aber im Hinblick auf die fortschreitende Realisierung des Ubiquitous Computing in der Informationsgesellschaft gewinnen diese Themen noch einmal zusätzlich an Bedeutung. Das Ubiquitous Computing, welches sich sinngemäß mit allgegenwärtigem Rechnen übersetzen lässt, stellt eine grundlegende Veränderung der IT-Landschaft dar und zeichnet sich insbesondere durch die Verschmelzung der virtuellen mit der realen Welt aus. Die Computer werden in die Umwelt so integriert, dass sie der Nutzer großteils nicht mehr als solche wahrnimmt. Da diese Computer jederzeit miteinander vernetzt sind und kommunizieren können, lassen sich mit ihrer Hilfe vielfältige hilfreiche Anwendungen realisieren, die den Nutzer in seinen alltäglichen Arbeiten unterstützen. Gleichermaßen wächst jedoch auch das potentielle Gefahrenpotential, beispielsweise der allumfassenden und dauerhaften Überwachung der Nutzer. Weiterhin müssen die Anwendungen, die sich intern die Zusammenarbeit vieler einzelner Computer erfordern, sehr robust und zuverlässig arbeiten, um nutzbringend einsetzbar zu sein und Akzeptanz bei den Nutzern zu erhalten.

Diese Dissertation beschäftigt sich mit Fragen rund um die Thematik wie Sicherheit und Verlässlichkeit in Ubiquitous Computing-Umgebungen etabliert und gehandhabt werden können. Hierbei erfolgt einerseits eine Fokussierung auf zwei wichtige Teilbereiche, die Integration von Sicherheitsmaßnahmen in die verbauten Geräte, die die technische Basis des Ubiquitous Computing ausmachen, sowie die Weiterentwicklung der in bestehenden Weitverkehrsnetzen eingesetzt Routing-Verfahren. Andererseits wird in einem dritten Schwerpunkt die Frage erörtert, ob die heutzutage vorwiegend eingesetzten Konzepte und Verfahren zur Erreichung der Sicherheitsziele noch zeitgemäß sind oder ob eine datenzentrierte Sicht auf Sicherheit mehr Vorteile verspricht, insbesondere im Hinblick auf die qualitativen Veränderungen im Ubiquitous Computing.

Bezüglich der Integration von Sicherheitsmaßnahmen in ubiquitär eingesetzte Geräte muss insbesondere darauf geachtet werden, dass diese möglichst klein und günstig sein müssen um überall einsetzbar zu sein. Hieraus ist augenblicklich ersichtlich, dass diese Geräte sehr beschränkt in ihren Möglichkeiten sein werden, beispielsweise nur geringen Speicher, Rechenleistung, Energievorrat oder Übertragungsgeschwindigkeiten unterstützen. Diese limitierenden Eigenschaften gilt es bei der Integration von Sicherheitsmaßnahmen zu beachten. Eine Beispieltechnologie für diesen Bereich bilden drahtlose Sensornetze, die in Ubiquitous Computing-Umgebungen zahlreich zum Einsatz kommen werden. In dieser Arbeit werden in diesem Kontext zwei Lösungsansätze vorgestellt. Zum

Einen wird eine Sicherheitsarchitektur für mobile selbstorganisierende Sensornetze entworfen und deren praktische Umsetzung beschrieben. Da jedoch der spezielle Fokus auf ein Einsatzszenario diverse Nachteile mit sich bringt, wird zum Anderen ein zweites, allgemeiner einsetzbares Verfahren beschrieben, welches die Annotation von Sicherheitsmaßnahmen beim Designprozeß der Anwendung erlaubt. Das Verfahren unterstützt den Entwickler nachhaltig bei der Integration sinnvoller Sicherheitsmaßnahmen. Ausgehend von der Spezifizierung abstrakter Sicherheitsmerkmale durch den Anwendungsentwickler werden automatisch mögliche Sicherheitslösungen generiert und hinsichtlich der erreichten Sicherheit und verbleibenden Risiken evaluiert. Die Ergebnisse werden dem Anwendungsentwickler präsentiert, der hierdurch in die Lage versetzt wird, einerseits die optimal passende Sicherheitslösung für seine Anwendung zu wählen. Andererseits kann er durch die detaillierten Rückmeldungen mögliche Sicherheitsrisiken besser einschätzen und in der weiteren Anwendungsentwicklung berücksichtigen. Durch die Integration des Verfahrens in einen Middelware Generator kann dem Anwendungsentwickler schließlich automatisch der benötigte Quellcode zur Verfügung gestellt werden. Sowohl die theoretischen Grundlagen wie auch die praktische Umsetzung dieses Verfahrens werden detailliert beschrieben und evaluiert.

Im Bereich des Routings begutachtet diese Arbeit natur-inspirierte Routing-Verfahren näher, um die Frage zu klären, ob autonom agierende Verfahren das Potential besitzen die Verlässlichkeit der Weitverkehrsnetze zu steigern. Natur-inspirierte Verfahren orientieren sich hierbei entweder am Schwarmverhalten von Tieren oder an Grundsätzen der Evolution. Verschiedene Verfahren werden zu Beginn der Untersuchung auf Basis der verfügbaren Literatur verglichen, um den viel versprechendsten Ansatz zu bestimmen. Dieser wird dann unter theoretischen Gesichtspunkten wie auch simulativ mit dem State-of-the-Art Routing-Verfahren OSPF hinsichtlich der verschiedenen Aspekte von Verlässlichkeit nähergehend untersucht. Es werden sowohl die einzelnen Untersuchungen detailliert präsentiert, wie auch die Ergebnisse aufbereitet und bewertet.

Schließlich widmet sich diese Arbeit der Frage, ob die traditionell eingesetzten Konzepte um Sicherheit in Netzen zu erreichen auch in Ubiquitous Computing-Umgebungen effizient Sicherheit bereitstellen können, oder ob eine datenzentrierte Sicht auf Sicherheit nicht viel versprechender ist. In diesem Zusammenhang werden einerseits die zugrundeliegenden Konzepte datenzentrierter Sicherheit näher erläutert. Andererseits werden die Verfahren, die heute zur Erreichung dieser Konzepte verwendet werden, hinsichtlich ihrer Eignung für den Einsatz in Ubiquitous Computing-Umgebungen untersucht und bewertet.

Abstract

Security and dependability represent important topics in the context of computer networks and distributed systems. Especially with the ongoing trend towards realizing the paradigm of ubiquitous computing, they become even more decisive. Ubiquitous computing is characterized by a fundamental revolution in the information technology as it is known and in use today. In essence, the computing devices are integrated into the real world in such a way that they disappear from view and are no longer perceived as computer systems by the users. Thus, the virtual and the real world effectively merge. This allows on the one hand for many fascinating new applications to support users in their everyday tasks. However, on the other hand this also involves a great potential for massive security risks, such as the complete and permanent surveillance of the users. Furthermore, these systems, that span internally over multiple devices, must be extremely robust and dependable in order to provide usable solutions and gain the users' acceptance.

This dissertation concentrates on questions regarding the management of security and dependability in ubiquitous computing environments. At this, we focus on the one hand on two important aspects, the integration of security into the devices that build the foundation of these environments as well as the advancement of the routing technology used in wide-area networks today to support the qualitative changes that accompany ubiquitous computing. On the other hand, we discuss the question if the traditionally prevalent security concepts still represent efficient and usable solutions in ubiquitous computing environments or whether a data-centric view on security is more beneficial and feasible to implement.

Regarding the integration of security measures into the ubiquitously deployed devices it is important to keep their distinct characteristics in mind. By design, these devices must be extremely small and cheap in order to be deployed everywhere and disappear from view. However, this results in severely resource-constrained devices, regarding CPU, memory, energy, and communication bandwidth. These limitations must be considered during the design of appropriate security solutions. An example of such a technology represent wireless sensor networks which will be massively deployed in future ubiquitous computing environments. We present in this work two approaches to provide security for wireless sensor networks. On the one hand we design and implement a security architecture for a mobile self-organizing sensor network. However, due to several drawbacks that result from the general approach of providing a specialized security architecture for one application scenario, we design and implement a second, more generic framework that allows for annotating security during the

design process of the application. It effecively supports the application by devising and evaluating feasible security solutions. Based on the specification of abstract security aspects, the framework generates possible security solutions and evaluates them regarding the provided security and remaining risks. This evaluation is given as qualitative feedback to the application developer who in turn may choose the most applicable solution for his application. Furthermore, the detailed information fosters his awareness of remaining open risks and enables him to include safeguards in his application to prevent these. Finally, this framework is integrated into a middleware generation tool, in order to automatically provide the application developer with the required source code. We present in detail the theoretical foundations as well the implementation of the framework and evaluate its usefulness.

In the context of routing in wide-area networks we assess the potential of nature-inspired approaches to autonomically increase the dependability of these networks. Nature-inspired approaches to routing either mimic the behaviour of animal swarms or are based on the concepts of the evolutionary process. Based on the available literature we analyse several nature-inspired approaches to determine the most promising approach. Following this, we evaluate this approach theoretically and by means of simulation with regard to several aspects of dependability. Furthermore, the whole evaluation is done in comparison to the state-of-the-art routing approach OSPF. We present the individual evaluations as well as the results and provide a first answer regaring the initial research question.

Finally, we assess to what extend the traditionally prevalent security concepts still provide usable and efficient security solutions in the context of ubiquitous computing environments. In this regard we also introduce a data-centric view on security and evaluate its potential benefits, underlying concepts, and the technologies that are required to implement the concepts. Furthermore, we evaluate whether these technologies are usable and operate efficiently in ubiquitous computing environments.

Contents

1 Introduction

Yesterday is not ours to recover,
but tomorrow is ours to win or to lose.
[Lyndon B. Johnson]

We live in interesting times. Small-scale computers emerge more and more to help us manage our every-day tasks. On the one hand, traditional devices become more powerful and provide functionalities that go way beyond their original purpose. For example, mobile phones no longer provide only their literal functionality, making phone calls en route, but additionally include our calendars and tasks, serve as digital cameras, music players, and navigate us towards our destinations. On the other hand, the numbers of computers surrounding us increase significantly. Thus, mobile phones may also be used, for example, at home as a universal remote to control various electronic devices, such as the stereo system or television. Furthermore, other parts of the building equipment and appliances increasingly become computer-enabled as well and thus may be controlled and operated remotely. In these so-called smart homes aspects, such as the heating system, lights, and kitchen appliances become completely controllable using common computer technology. Prototypes of these smart homes already exist and are promoted by vendor associations, e. g. [247].

This computerization of our world is assisted by global networking capabilities, allowing for new always-on technologies, i. e. permanent connection to online services such as Google Latitude [85], which allows you to publish your current location on an online map, so that your friends are always able to locate and contact you. Localization also provides the basis for so-called location-based services, i. e. services offered to you via your phone that are located in your immediate physical surrounding.

Thus, the virtual world of cyberspace increasingly merges with our physical world. This ongoing trend in information technology towards integrating sensing, communication, and computation into the physical world offers fascinating new services. For example, the physical reality could be augmented with fitting digital information in order to provide the user with additional information about a piece of art in a museum or intuitively usable navigation system, displayed directly on the windshield of the car. Furthermore, it will be possible to continuously monitor patients or elderly people at home. This vision of a completely computerized world is commonly referred to as the paradigm of ubiquitous computing (UC) [279].

1.1 Motivation

While offering many fascinating new services, the same technologies also introduce many new risks with regard to security. Questions that arise include, for example, who controls the data that is gathered and how can we keep our data to ourselves or control who may access it? Furthermore, can we trust the technologies to exactly provide the service they advertise and are they stable enough in their operation that we may depend on them? Who is liable if something goes wrong? How can we manage security-related aspects in a world where computers are no longer visible? Thus, in order to safeguard the users from negative side-effects of the ubiquitous technology and gain their acceptance, it is very important to consider security and dependability aspects when designing and building these systems.

In this dissertation we study how security and dependability can be managed in such a future world that is fully penetrated by information technology. We would like to clarify, that we do not attempt to provide solutions for all problems related to security and dependability in ubiquitous computing, just to name and categorize them all could easily fill a book. Rather, we use the vision of ubiquitous computing to give focus and context to our research and present and discuss particular well-defined sub-problems for which we provide possible solutions or advance the state-of-the-art knowledge.

1.2 Contributions and Structure of this Work

The research contributions we present in this dissertation examine three problem areas regarding security and dependability in UC in more detail:

1. *The integration of security into UC devices* – we propose and evaluate two approaches to integrate security into wireless sensor networks, one of the core technologies in UC environments.
2. *The achievement of autonomic dependability in wide-area networking* – we determine the most promising nature-inspired approach to routing in networks and analyze theoretically as well as by means of simulation its advantages and drawbacks compared to the state-of-the-art routing approach OSPF, focusing on the autonomic realization of dependability in wide-area networking.
3. *The case for data-centric security* – we argue why traditional security concepts need to be critically challenged regarding their applicability in UC environments and present the concept of data-centric security. Furthermore, we analyze how far this concept may be realized using today's prevalent technologies.

This thesis first discusses general aspects of security and dependability before presenting the vision of UC in more detail, including its distinct characteristics

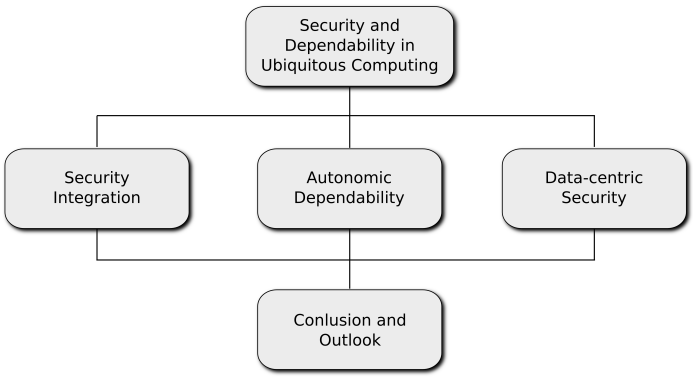


Figure 1.1: *Structure of this work*

and the challenges that arise regarding security and dependability. Based on this, we focus on our contributions. The individual concepts and operational settings for the three areas differ significantly. Therefore, to improve readability, we target each area in one chapter and try to make it as self-contained as possible[1], i. e. we present and discuss the respective foundations, related work, and, where applicable, the proposed solution or experimental results inside each of these chapters. Thus, each of the main chapters may be read independently, if the the reader is familiar with the basic concepts of security, dependability, and UC. Finally, we summarize the results of our contributions and provide possible directions for future work. The overall structure of this work as well as possible reading paths are illustrated in Figure 1.1. In more detail, the thesis is structured as follows:

In Chapter 2 we provide the necessary foundations and terminology for the reader to follow the discussions and presentations in our work. First, Section 2.1 introduces the traditional concepts regarding security and dependability. Section 2.2 provides a brief overview of the vision and research field of UC. At this, we especially highlight its distinct characteristics as well as the resulting challenges and issues focusing on security and dependability. Finally, we briefly describe, arrange, and visualize in Section 2.3 the respective research areas of our contributions in the overall context of UC environments, to provide a clear picture which aspects and challenges of UC we address.

Chapter 3 targets the integration of security technologies and concepts into resource-constrained devices, i. e. wireless sensor nodes, that represent one of the fundamental technologies that will be used in UC environments. We provide the necessary fundamentals regarding wireless sensor networks (WSN) as well as an introduction to the middleware synthesis tool Fabric in Section 3.1. Fabric will be utilized in one of our later proposals in the proof-of-concept implementation. Section 3.2 gives an overview of the research area of security

[1]Thus, it may happen that some abbreviations are defined twice in this work, e. g. "wireless sensor network (WSN)" is defined in the background in Section 2.2 as well as in the respective chapter of the contribution in Section 3.1.

in WSNs. In Section 3.3 we present and evaluate our first proposal to provision a WSN with the required security technologies, i. e. we define a security architecture for mobile self-organizing WSNs. Due to the fact that this approach to provide security results in several disadvantages, we propose and evaluate a more generic framework in Section 3.4 that allows to annotate the required security mechanisms during the application development. Section 3.5 provides a summary of the important findings and results from our two proposals.

In Chapter 4 we examine the question whether autonomic approaches to routing have the potential to increase the dependability of wide-area networks (WANs), which will become hard to manage in the face of the increasing complexity in UC. Section 4.1 provides the necessary foundations for the analysis. In detail, we recap the general routing problematic and present state-of-the-art as well as nature-inspired approaches to routing. In particular, we discuss the nature-inspired approach BeeHive as well as the state-of-the-art approach OSPF in more detail, as these two will be examined more detailed in the further analysis. We provide a survey of the published evaluations of several nature-inspired routing approaches to determine the most promising candidate, which turned out to be BeeHive, in Section 4.2. Following this, in Section 4.3 we analyze BeeHive and OSPF in more detail regarding the aspects of dependability, both theoretically and by means of simulation. Finally, Section 4.4 summarizes the results and presents our conclusions.

Chapter 5 is devoted to the question whether the security concepts that are prevalent today still represent effective security solutions with regard to UC or if data-centric security provides a superior and viable security concept. To this end Section 5.1 introduces the goals and underlying concepts of data-centric security. In Section 5.2 we analyze the concepts and technologies that are used to implement them, focusing on their realizability and effectiveness in the context of UC. Section 5.3 finally summarizes the results.

Chapter 6 concludes this dissertation with a summary of the contributions and a presentation of possible directions for future work.

Acknowledgements

A dissertation is usually not the result of countless hours spent alone in a dark room. The work presented here would not have been possible without the close cooperation and the prolific discussions and cooperations with my colleagues at the Institute of Telematics and our project partners.

The first part of the research on integrating security into ubiquitous computing devices is joint work with Holger Krahn, Prof. Dr. Dietmar Wätjen, and Prof. Dr. Stefan Fischer, while the second part is joint work with Dr. Dennis Pfisterer and Prof. Dr. Stefan Fischer.

The results on autonomic dependability in wide-area networking stem from the project BSI ASA I “Evolutionäre Algorithmen für autonome Systeme“,

which we conducted with our partners from the University Dortmund, Christine Zarges and Prof. Dr. Thomas Jansen as well as Dr. Christian Ruge and Dr. Uwe Laude at the Bundesamt für Sicherheit in der Informationstechnik (BSI).

The research on data-centric security is in parts joint work with Prof. Dr. Christian Werner.

Furthermore, my sincere gratitude goes towards several people who read part or all of my dissertation. Thank you for your endless patience, sharp eyes and informative comments and criticism.

2 Security and Dependability in Ubiquitous Computing Environments

One's mind has a way of making itself up in the background,
and it suddenly becomes clear what one means to do.
[A. C. Benson]

The following sections are devoted to providing the reader with the required concepts and terminology to understand the further presentations and discussions in this work. First, the next section presents the traditional concepts regarding security and dependability in computer networks and defines important terms. Furthermore, we present approaches to network security as they are prevalently in use today. In Section 2.2 we present the vision of UC, including a disambiguation of terms regarding related concepts. Furthermore, we highlight the distinct characteristics of UC as well as its foundations and present challenges and issues with regard to security and dependability. Finally, Section 2.3 briefly describes the research areas on which we focus in our contributions. Furthermore, we show the relations between them and visualize their place within the overall context of UC.

2.1 Security and Dependability

The need for security is as old as the formation of the first human societies. While historically security implied mostly the protection of physical assets from others, the dawn of the information age, if nothing else, has extended this need to the protection of digital information as well. Over the history, mankind has invented several methods and protocols to secure his assets. However, as constantly as new ways to attack the protection mechanisms emerge, new security techniques are developed as well. Consequently, security has become such a broad discipline, that today it is almost impossible to give an all-encompassing in-depth overview.

Accompanying the information age is also an increasing technological pervasion of our surroundings. As humanity depends more and more on these systems, the research area of dependable computing has gained momentum. Essentially, dependability centers around building up trust that a system delivers its advertised service. Originally, dependability considered only the avoidance and handling of non-malicious system faults. Lately however, other aspects have been taken into consideration as well, overlapping in parts with security directions. Though the areas of security and dependability have traditionally

evolved separately, there have been attempts to provide unifying concepts that encompass both [12].

Generally, security and dependability both represent non-functional properties of a system. Thus, they express how the system must behave while providing its functionality. Functional properties in contrast ensure the system's actions, i. e. what the system is intended to do. Thus, while security and dependability of a system are usually implicitly expect by most people, few explicitly think about them as they focus on the task at hand. This often leads to a neglect of security and dependability aspects during systems design. However, it is of special importance to include them at this stage, since retrofitting them is often burdensome if not impossible and the components that lack integrated security are usually the first point of attack.

However, perfect security for a system does not exist on principle, as long as the system is supposed to be used later on. Therefore, the discussion about security is always a discussion about risk management, e. g. what security measures still allow for a useable system and how do theses measures compare economically to the risks they address. Equally, no perfect system, i. e. without faults, can be build. Therefore, discussions about the dependability of a system are really discussions about the kind of faults possible and the threshold up to which they are manageable and thus acceptable.

In the remainder of this section we first establish a common terminology to be able to efficiently discuss the problems and solutions in this work. Furthermore, we introduced security and dependability concepts as they are commonly in use today.

2.1.1 Terminology

The vocabulary of security an dependability is rich with seemingly alike terms and definitions, leading sometimes to confusion and misunderstandings. So in order to establish a common vocabulary and a basis for understanding the further discussions, we define in the following the terms that are relevant in the context of this dissertation. We hereby focus on aspects that are important in the context of networks and distributed systems.

2.1.1.1 Security

The communication in computer networks is generally considered secure, if one or several security services are satisfied [253].

Definition 1 (Security Service) A security service defines security aspects for data or a system, that counter a certain class of threats when fulfilled. A security service makes use of one or several security mechanisms to provide the service.

Three primary security services exist: confidentiality, integrity, and availability, with secondary security services covering additional aspects. The following definitions are based on the widely accepted descriptions in [240].

Definition 2 (Confidentiality) Confidentiality protects information against unauthorized disclosure.

In other words, confidentiality guarantees that the information is not made available or disclosed to unauthorized individuals, entities, or processes. Regarding communication in computer networks, an adversary may indeed detect that two parties are communicating. However, he is not able to determine the content of the communication.

Definition 3 (Integrity) Integrity protects a system element or the system as a whole against unauthorized intentional or accidental changes.

It is important to note, that integrity cannot prevent changes, e. g. changing a message in a distributed system during transfer from sender to receiver. Yet, it ensures that changes are definitely detected. Furthermore, while integrity deals with constancy of and confidence in data values, it allows no assessment about the information that the data values represent, e. g. trustworthiness of the information. With regard to systems, integrity ensures the correct mode of operation, free from deliberate or inadvertent unauthorized manipulation.

Definition 4 (Availability) Availability guarantees that a system or a system resource is accessible and usable upon demand by an authorized entity, according to the design specifications of the system.

Providing availability thus requires a proper management and control of the system resources.

Building upon these three basic security services several others are defined to assure further aspects. The two most important ones are authenticity and access control.

Definition 5 (Authenticity) Authenticity enables verification of the identity claimed by or for an entity.

In computer networks, authenticity is generally used in one of two forms, either data origin authenticity or peer entity authenticity. Generally, authenticity cannot be achieved in this environment without integrity, ensuring that the communicated data is not altered during the transmission.

Definition 6 (Access Control) Access control prevents unauthorized usage of a system or system resource, including the prevention of using the system or system resource in an unauthorized manner.

Thus, the focus of access control is authorizing entities to use the system or system ressource in a clearly defined way or denying them access. At the same time it prevents unauthorized ways of access to the system or system ressource. Access control therefore requires authenticity to fulfill its goal.

Each of the aforementioned security services represents an abstract security goal. To obtain the respective goal technically, the services make use of one or more security mechanisms.

Definition 7 (Security Mechanism) A security mechanism provides a technical process that can be used in a system to implement a security service.

Therefore, a security mechanisms implements the technical means to detect or prevent an attack or to reset the system state after a successful attack. Examples for often used security mechanisms are encryption, digital signatures, message authentication codes, and hash algorithms. Usually several mechanisms are used in combination to implement a security service.

2.1.1.2 Dependability

The focus of dependability has broadened from the traditionally original target of non-malicious system faults to include malicious faults as well over time. A good overview and discussion about this is presented in [12]. The following definitions are based upon this as well as the definitions given in [240].

Definition 8 (Dependability) The ability of a system to deliver its service in a way that users can justifiably trust. The means to achieve this is the system's ability to avoid service failures that are more frequent and more severe than is acceptable.

Dependability is an umbrella concept that comprises availability, reliability, safety, integrity, and maintainability. Figure 2.1 (adapted from [12]) illustrates the relationship between security and dependability. In the following, we provide short definitions of these concepts as far as they were not defined in the security part already.

Definition 9 (Reliability) The system's ability to perform a required function under stated conditions for a specified period of time.

The level of robustness of the employed software and hardware, i. e. the ability to react to failures in a way that no system failure occurs or its effects are minimized, is decisive for the system's reliability.

Definition 10 (Safety) The property of a system being free from risk of causing harm to its users and system entities.

Definition 11 (Maintainability) The ability of a system to undergo modifications during its operation as well as repairs after a failure.

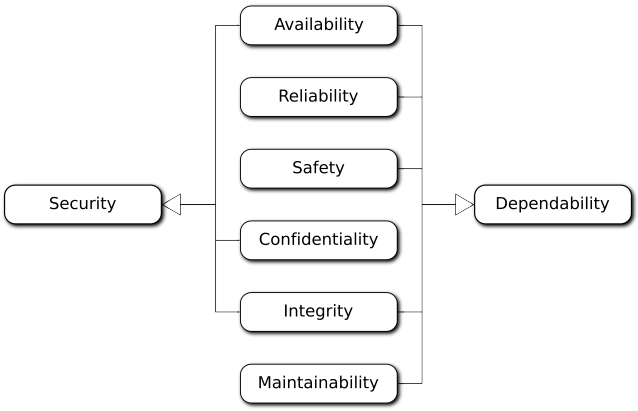


Figure 2.1: *Relationship between security and dependability*

2.1.1.3 Threats, Attacks, and Risks

Generally, threats, attacks, and risks refer to the unwanted but unavoidable events that oppose the security and dependability of a system. However, a lot of ambiguity exists in the literature regarding the exact terms, their scope, and differences. We use them throughout this dissertation as follows.

Definition 12 (Threat) A potential for violation of security, which exists when there is an entity, circumstance, capability, action, or event that could cause harm, adversely affecting the system.

In other words, a threat represents the possibility of something bad happening. It is important to note that threats may be caused accidentally, for example, by human error, equipment malfunction, or natural disaster. Though threats include intentional acts as well, we refer to these as attacks.

Definition 13 (Attack) An intentional act by which an intelligent entity, the attacker, attempts to evade security services and violate the security policy of a system.

Thus, attacks realize threats, mostly by exploiting system vulnerabilities. Attacks are generally distinguished by their point of origin into *internal* and *external attacks*, using a security perimeter (c. f. Section 2.1.2) as boundary. Internal attacks are executed by an insider, an entity that has legitimate access to the system resources but uses them maliciously. Quite often internal attacks can not be differentiated from byzantine failures [195]. Here a system still operates, but incorrectly and it is not possible to determine if the incorrect operation results from a successful attack or from failures, such as misconfiguration, hardware failures, or software bugs. External attacks in contrast are initiated from

Attack	Type	Description
Masquerade	Active	Posing as another entity, e. g. when sending messages.
Replay	Active	Retransmission of valid messages.
Message modification	Active	Altering of the message's content, or delaying or reordering messages.
Denial of service (DoS)	Active	Preventing the legitimate use of a system resource.
Information disclosure	Passive	Unauthorized publishing of confidential information.
Traffic analysis	Passive	Observation of the frequency and length of communications, in order to derive information about the entities' relationship.

Table 2.1: *Generic active and passive attacks*

outside the security perimeter, by an unauthorized or illegitimate user of the system, the so-called outsider.

Furthermore, attacks are differentiated according to their intent into *active* and *passive attacks*. Active attacks attempt to alter system resources or affect their operation, while passive attacks attempt to learn or make use of information from a system by monitoring but not affecting the system resources. Several taxonomies exist that organize attacks, e. g. [105, 158]. In the context of this thesis, we categorize attacks regarding networks according to Stallings [253]. Table 2.1 provides an overview.

Especially with regard to analysing possible attacks against a system, it is important to specify the attacker's capabilities in order to define the boundaries for the analysis. The *Dolev-Yao* model [67] is often used to model active adversaries in communication networks and distributed systems. The basic assumption is an insecure communication channel between any communicating entities. Thus, the attacker, who is assumed to be omnipresent, may execute all of the active and passive attacks mentioned above at any point in the network. However, the employed cryptographic protocols and mechanisms are beyond this ability to break without knowledge of the key. Furthermore, he can not compromise the communicating entities.

Finally, risk centers on the quantitative aspects of threats.

Definition 14 (Risk) Risk defines the probability of loss due to a successfully realized threat.

With regard to analysing risk of a system, the term *residual risk* defines the portion of an original risk or set of risks that remains after the available countermeasures have been applied to the system.

2.1.2 Concepts

The prevalent approaches to network security today center on providing security solutions for specific problems, mainly focussing on the environment. For example, virtually every organization protects its devices, services, and data in its internal network from the outside by means of a firewall. This concept of perimeter security aims at separating a trusted internal network from the untrusted external network by securing the borders of the network which the organization controlls. Firewalls analyse the data traffic trying to determine unwanted data packets or communication connections. Here, unwanted traffic is specified by security policies specified by the organization. In order to enforce these policies modern firewalls often not only filter the data traffic by analysing the connection information, e. g. the IP addresses or port numbers, but include sophisticated content filtering technologies as well, e. g. to detect attacks on services available to the outside such as web servers.

Inside the trusted networks, further security technologies are often utilized to enforce for example access control for documents, e. g. by means of encryption, or for devices or services, e. g. by requiring users to authenticate to the PC or the database.

If data traffic crosses the security boundaries of the internal network the focus of the security measures shifts to securing the communication. Secure communication protocols, such as SSL/TLS provide for example a protected tunnel through which the communication may take place. The scope may range from a single communication connection up to a protected interconnection of protected networks by means of a virtual private network (VPN).

All of the mentioned concepts above represent proactive security approaches, i. e. they consist of measures that are taken with the goal of preventing attacks from successfully compromising the system. However, it is practically impossible, to build and maintain a perfectly secure system that anticipates every possible attack. Therefore, reactive security approaches aim at detecting and reacting to attacks and other system anomalies against which proactive security measures are not feasible, due to technological difficulties or efficiency and cost issues. Security technologies for reactive security are usually summarized under the term intrusion detection system (IDS) [61, 224].

In essence, approaches to security today center around protecting the information technology, i. e. the infrastructure, closely following the military concept known as defense-in-depth. As the name implies, defense-in-depth aims at providing different layers of security. If one protection is overcome by an attacker, further technologies provide additional safeguards.

2.2 Ubiquitous Computing

Marc Weiser coined the term ubiquitous computing (UC) in his seminal article in Scientific American [279] and is therefore often referred to as the father of UC. His vision of UC in a nutshell is not concerned with a special technology by itself but rather symbolizes Information Technology (IT) that merges with the real world surroundings, offering services in an unobtrusive and intuitively useable way.

UC IT, which we will simply denote as UC technology or devices in the following, will be seamlessly integrated into everyday objects making it invisible to common awareness. Rather it will always be present in the background and people will use it unconsciously to accomplish everyday tasks. As an example of disappearing technology Weiser points out that although several motors are required in an automobile, most people will never think of starting a specific motor when starting the wipers. Furthermore, people will be able to absorb the presented information from UC devices without the need to explicitly think about it or to acquire specialized knowledge beforehand. Weiser compares this to reading a street sign. People usually do not consciously perform the act of reading when looking at it, still its information is instantly transmitted to the reader.

Weiser also refers to UC as *embodied virtuality* which is diametrically opposed to *virtual reality*. While virtual reality is based on a virtual world inside the computer, the UC devices are integrated in the real world, which effectively merges the real and the electronic, i. e. virtual, world.

Towards the realization of his UC vision Weiser sees a clear trend in [280]. He describes three epochs in computing, which he defines as waves of technological change that fundamentally alter the place of technology in our life, rather than merely changes in the technology itself. The three epochs are the mainframe era, the personal computer era, and the UC era.

- *Mainframe era* – In the mainframe era computers are run exclusively by experts and have to be shared by many users because they are a scarce resource, i. e. expensive and rare.
- *Personal computer era* – The personal computer era started around 1984 when the people using personal computers outnumbered the mainframe users. In this era people work with their own personal computer. An important thing to note is that this work occupies them completely, i. e. they are not doing anything else. However, to do so the user does not necessarily have to be an expert.
- *UC era* – In the UC era the ratio between users and computers or devices has reversed itself completely in comparison to the mainframe era. Each user is surrounded and interacts with lots of computers. The handling and interaction with them is intuitively done as the user focuses on solving his everyday problems.

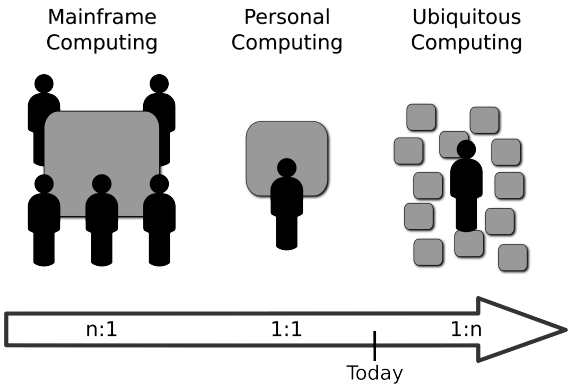


Figure 2.2: *Weiser's major trends in computing*

Due to Weiser we are currently in a transition phase between the latter two eras, in which the Internet and the distributed computing paradigm tie together the personal computers and give them access to servers, which resemble on their own the mainframe era. Figure 2.2 illustrates the eras and the corresponding computer-to-person ratio respectively.

2.2.1 Terminology

Alongside Weiser's term UC, to which we will stick in this thesis, several other terms and definitions exist in the literature that are often used synonymously or overlap in parts. Some of the widely mentioned terms in this context include Pervasive Computing, Ambient Intelligence (AmI), Mobile Computing, and Autonomic Computing. In order to differentiate between them and identify their relations a disambiguation the terms is clearly needed.

Müller et al. define UC as the intersection of the concepts mobile computing, pervasive computing, and autonomic computing, each with it's own focus but yielding important aspects into UC (cf. Figure 2.3) [177]. Furthermore, they arrange these areas into a temporal development. On the basis of traditional computing Mobile Computing set the development off towards mobile devices, making seamless global connectivity the focus of research. The following inclusion of context is the key point of Pervasive Computing. With increasing complexity of the networks and devices people can no longer administrate the IT and the need for self-organization becomes evident, focused in the research area Autonomic Computing. Furthermore, other definitions and disambiguations of UC-relating terms exist in the literature, e. g. Lyytinen and Yoo differentiate the individual terms in [161] based on the dimensions embeddedness and mobility.

The vision of AmI was coined by the IST Advisory Group of the European Union [116] during the 6^{th} Framework Program, in order to foster european

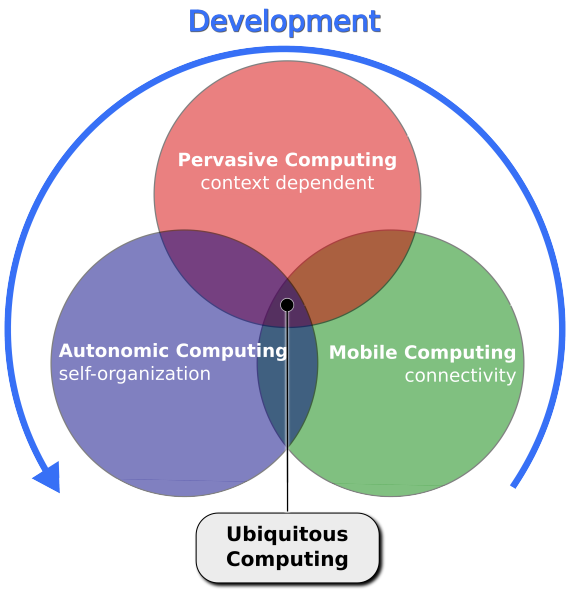


Figure 2.3: *Research areas that form ubiquitous computing.*

research. Essentially, AmI and UC describe the same vision. However, while UC focuses on the technological aspects, AmI's focus is broader, taking for example social aspects, such as the acceptance of AmI, into account as well.

We define UC as an umbrella concept that integrates various other research areas, thus following closely Müller's definition. However, we acknowledge that social issues have to be taken into account as well. This definition will become clearer when we talk about the characteristics of UC environments in the next section.

2.2.2 Characteristics and Foundations

Generally, UC environments consist of numerous context-aware devices that are able to communicate and interact with each other in order to provide services that facility the user in his everyday tasks. Salient characteristics that distinguish UC from other computer science domains are:

- *Ubiquity*: by design UC is everywhere. Permeating every aspect of the real world, it is effectively affecting every part of our lives. Thus, the traditional boundaries to computing vanish. IT is no longer confined to the computer and the cyberspace but rather integrates into the real world. Hereby, the devices may be static as well as mobile.
- *Invisibility an unobtrusiveness*: the merging of IT with the real world leads to invisible devices that offer their services in an unobtrusive way. The user should no longer be aware that he is using a computer while performing a task.

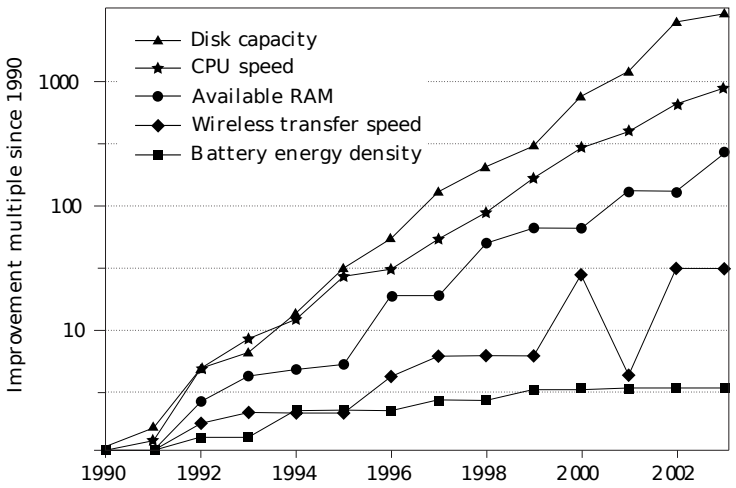


Figure 2.4: *UbiComp resource trends*

- *Context-awareness*: UC devices are able to understand their surroundings, i. e. they are context-aware. By means of various sensors they may capture and process certain aspects of the user or their surroundings.
- *Adaptability*: on the basis of the context information, UC devices adapt their operation to the user's wishes or the current situation.
- *Communication and cooperation*: all devices communicate with each other in order to cooperate and integrate specialized services into higher level services.

Weiser emphasizes in [279] three technological requirements for UC to become reality: cheap, low-power computers that include equally convenient displays, software for ubiquitous applications, and a network that ties them all together. While at the time of writing in 1991 this was clearly a vision for the future, trends in hardware and software since then seem to make this vision feasible today. Most importantly is the ongoing trend to miniaturise hardware at an exponential rate, following Moore's law [171]. However, equal technological advancements can be found for example in the areas of storage capacities and communication bandwidth as well [167]. Figure 2.4 illustrates the growth rates of different resources in the last years based on [188].

These trends give rise to the widespread deployment of small embedded systems as well as new technologies, such as radio frequency identification (RFID) tags and wireless sensor networks (WSN). Furthermore, new materials emerge that for example resemble on the surface traditional paper but are in fact electronically controllable [66] or that are bendable and thus allow to be integrated into the clothing, so-called wearable computing [160].

Regarding the network capabilities, several technologies have been developed to cover almost every usage scenario, ranging from strictly local communication via NFC or Bluetooth, to mid-range communication technologies, such as ZigBee

and WLAN, up to global coverage via GPRS, UMTS, and the Internet. These technologies tie the individual devices into a globally spanning omnipresent network.

2.2.3 Issues and Challenges

The real power of UC lies in the unrestricted flow of information, which is gathered and processed by the plethora of devices to establish their services and support the users in their everyday tasks. It is evident that the ongoing realization of UC has a profound impact on our lives and the way we interact with computers. Since the concept focuses on the complete penetration of the real world with information technology, possible applications can be envisioned wherever imagination reaches. Today several UC concepts are already in use, e. g. in the area of medicine, logistics, or smart homes [28].

However, realizing UC gives also rise to several issues and challenges. This is especially true for security and dependability. While the traditional security services, which have been defined years in advance of the UC vision, represent important pointers to reason about security in UC environments, the traditionally employed mechanisms to provide these services must be critically challenged due to the significant changes in the operational environment.

For example, by design the devices that form the UC environment must be cheap and small, in order to place them everywhere and in large quantities. Thus, these devices will be significantly constrained with regard to their on-board resources, such as CPU, memory, energy, and communication bandwidth. This strongly impacts the choice of technical security measures, such as encryption algorithms. These are mostly designed for full-scale computers and may not run on these devices. Furthermore, the heterogeneity of the devices as well as the complexity of the UC environment makes the individual programming and administrating of these environments a nearly unmanageable task.

Furthermore, the traditional security boundaries are vanishing in UC. No longer will well-defined entry and exit points exist to a network. Rather the devices communicate on demand using multiple available communication technologies and redundant communication paths. This requires a change in the traditional security assumptions and trust models. While traditional perimeter security established an internal trusted network in which devices may assume a certain security and trust level, as well as an external untrusted network, UC devices have to establish and maintain trust dynamically with their communication partner.

Another issue becomes apparent if we take for example a look at traditional authentication methods. Generally, devices or users are authenticated by either proving that they know a secret, proving the possession of a token, or providing a certain characteristic feature. Although these basic concepts may also be used in UC environments, the management of these authentication paths was traditionally handled manually, e. g. users were given credentials by the ser-

vice provider in order to use the service. The on-demand service construction and multitude of devices which UC environments comprise however, makes a manual approach infeasible. Furthermore, authentication is traditionally centered on IDs and the question arises, if this is still adequate when most of the communication takes place between devices, with no user-interaction.

Apart from the technological challenges, the socio-technological perspective, i. e. the recognition of the interaction between people and technology, must be taken into account when arguing about and designing security solutions for UC environments. Foremost, usability aspects, i. e. security solutions that don't get in the way of the systems' users, represent probably the greatest challenges for UC in this regard [58]. However, just as important are aspects that usually spark users' fears, such as the inscrutability of the complex UC services. Especially the questions if the system acts as advertised and does it provide the promised security, provide important challenges regarding the traceability of the systems operation and robustness, as well as the visualization of the employed security. Measures in this regard would allow the user to build up trust into the systems.

A profound open question in this regard is also the secure handling of personal data, commonly denoted as privacy, in order to avoid the transparent user. Mobile users will leave bits of personal data whereever they go and correlating this data may result in a complete trace of their whereabouts and actions. Therefore, systems or mechanisms that provide amongst others location privacy and data confidentiality are imperative in a UC environment.

We are well aware that we have covered only the tip of the iceberg in our discussions above. However, the topic is so vast, that whole books have been written that focus solely on security in UC, such as [41, 249]. Furthermore, several extensive studies center around deriving the impacts UC will have on our lives, e. g. [28,37,102]. For a more in-depth presentation and discussions we therefore refer to these excellent sources as well as the significant amount of research papers that have been published on numerous conferences and workshops.

In the context of this work, it suffices to say that security and dependability in UC are complicated on a technological level by the device heterogeneity, the complexity and enormous scale of UC environments, as well as the potential mobility, adaptability, and the required autonomic behaviour of the devices. On the socio-technological level the main challenges represent usability, trust in UC devices, visualization of security, as well as data and location privacy. The security community has to determine which of these security-related problems can be solved by adapting existing solutions from traditional distributed systems while the others require novel solutions.

2.3 Roadmap

Before we set out discussing some of the issues and challenges set forth in the last section in more detail and present our findings and solutions, we want to

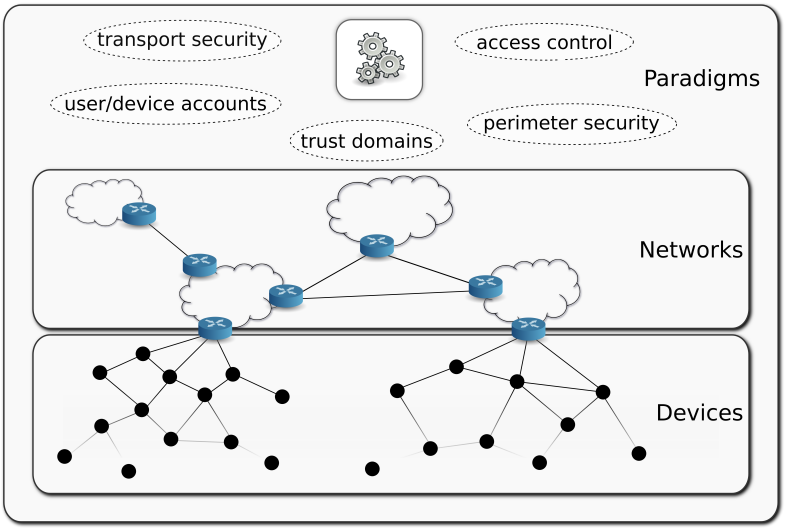


Figure 2.5: *Research focus areas*

clarify what exactly we are trying to accomplish. Since the topic of security and dependability in UC represents a vast research field, we concentrate our work on advancing the body of knowledge with regard to three problem areas:

1. *Devices*: how to build security into the devices that form the ubiquitous computing environments?
2. *Networks*: how to enable the existing WAN infrastructures to cope with the changes that result from the distinct characteristics of ubiquitous computing?
3. *Paradigms*: do we need to rethink traditional security paradigms to enable efficient security in ubiquitous computing environments?

Figure 2.5 provides a graphical representation of our research areas as well as the relations between them. We start at the bottom with the individual devices that provide the basis for UC environments. These need to be programmed with security-enabled applications. Connecting these devices are networks, which eventually comprise the well-established Internet backbones. Especially the employed routing approaches must be advanced to cope with the changes that accompany UC. Finally, we take a step back to discuss if the traditionally prevalent security paradigms still represent feasible and efficient security concepts in UC environments at large and argue the case for data-centric security. In the remainder of this section we take a closer look at each of our three target research areas.

Devices A plethora of different devices will be deployed in a multitude of operational settings when UC becomes reality. Thus, the IT is characterized by a significant device heterogeneity and high complexity. In order to enable

secure and dependable operations, security measures must be integrated into these applications, i. e. each device must be programmed comprising the appropriate security measures. However, although this is generally a problematic field, especially tiny, resource-constrained devices, such as sensor nodes, complicate the development of secure and dependable applications even further, since traditional security measures are usually too heavyweighted.

In Chapter 3 we focus on how to integrate the "right" security measures into WSNs. We discuss in detail why WSN applications require security as well as challenges that arise from their technical nature. Furthermore, we present and evaluate two approaches to provide security for WSNs.

Networks. Networking capability is a critical component in UC environments. Without the ability to communicate and cooperate, the devices will not be able to provide the services that are envisioned by UC. While several networking techniques exist, that provide special support for the wireless ad hoc communication between the devices, the existing WANs will continue to serve as the backbone of future global networks, integrating all UC environments. However, the increasing complexity and resulting diversity of service demands the WANs have to fulfill make them no longer humanly manageable.

In Chapter 4 we explore the question if nature-inspired routing strategies are able to autonomously increase the dependability and security of WANs. On the one hand we discuss in this context theoretical aspects and evaluate the different approaches in this regard. On the other hand, we actively analyse the behaviour of two selected approaches by means of simulation.

Paradigms. The prevalent security concepts today focus on protecting the infrastructure, e. g. protecting the network perimeter, the transport channel, or providing identity-based access control. However, several of the presented UC characteristics, such as the multitude of devices that form UC environments and the vanishing network perimeters, run contrary to the available security concepts. Furthermore, data is generally the most valuable asset, not the devices that store and process it. Therefore, the question arises, if solely relying on traditional security concepts is sufficient to provide security in future UC environments

In Chapter 5 we start by analysing the situation regarding security in general today and describe the impact which the realization of UC will have. Furthermore, we present the goals of a data-centric security paradigm as well as necessary concepts of this approach. Finally, we survey as to how far this approach may be realised in UC environments using today's existing technology.

3 Integrating Security into Ubiquitous Computing Devices

Nothing useful can be said about
the security of a mechanism, except
in the context of a specific application.
[Robert H. Courtney Jr.]

Security is an important aspect in UC as we have discussed in the last chapter. However, the question arises, which are the necessary security measures and how can these be integrated into the devices that form UC environments? First, the plethora of potential application scenarios seems to make it impossible to find a common denominator in form of a basic security setup that provides the necessary protection mechanisms, while not including aspects that are seldom used. Furthermore, due to the extreme device heterogeneity in UC such a setup, which surely comprises some of the well-known security algorithms, will quickly overburden some devices. Especially small, resource-constrained devices, which will be massively deployed in UC environments, represent an inhibiting factor in this regard.

In this chapter, we explore the problem of how to integrate the "right" security measures into these small devices. At this, we focus on WSNs, one of the core technologies in UC. Generally, although a lot of the envisioned application scenarios today would benefit from integrated security mechanisms, few actually concern themselves with security aspects as the technological nature of WSNs already provides a challenging environment for application developers. Since large numbers of small, severely resource-constraint devices form WSNs and have to be programmed, deployed and maintained, even well studied concepts like routing, or time-synchronization pose again challenging problems. Therefore, although security is often a desired feature, building a working system generally takes precedence over integrating security-related aspects, leaving a large part of WSN applications at risk.

In the following we provide first the basis for our further presentations and discussions in Section 3.1. Especially, we introduce at this the research area of WSNs. In Section 3.2, we discuss on the one hand the role of security in WSNs and highlight specific security challenges. On the other hand, we introduce areas of related work that aim at meeting them. Afterwards in Section 3.3, we present and explain in detail our initial approach to provide a security architecture for mobile WSNs. We evaluate this approach and discuss its achievements as well as its drawbacks. Finally, we present and evaluate in Section 3.4 a more generic technique to provide security for arbitrary WSN application scenarios.

3.1 Fundamentals

In the following we first provide a general overview of wireless sensor networks in the next section and introduce afterwards, in Section 3.1.2, the middleware synthesis framework FABRIC, which may be used to facilitate programming of the sensor nodes.

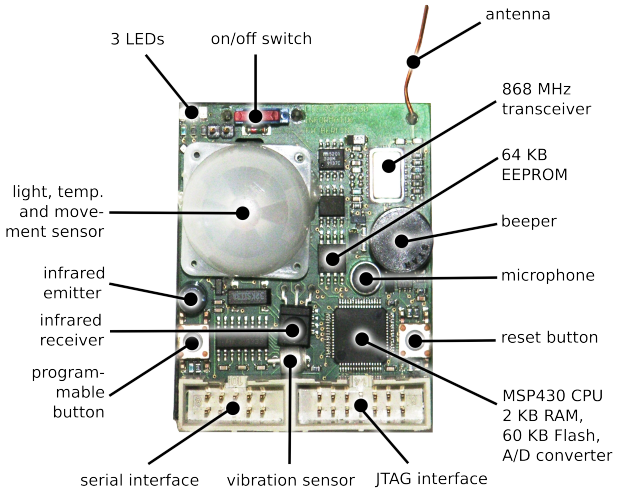
3.1.1 Wireless Sensor Networks

Wireless sensor networks (WSNs) represent one of the core technologies in UC environments, allowing UC to perceive the real world. They comprise in theory of hundreds to thousands of inexpensive and tiny nodes that allow to monitor and analyze real-world phenomena. Although the individual nodes are severely constraint in their resources and abilities, the power of WSNs arises from several interesting characteristics. First, the nodes are supposed to be so cheap that they can be deployed in large numbers, e. g. to completely cover an extended geographical area. Furthermore, due to their small size, they can be deployed unobtrusively and close to the phenomena they are intended to monitor. Another advantage lies in the potentially easy deployment of WSNs. For example, the nodes can be dropped at random, since they automatically organize themselves into a network during start-up and start to cooporate to achieve their goal. Due to this intended self-organization and the fact that the sensor nodes communicate generally wireless no expensive IT infrastructure is required.

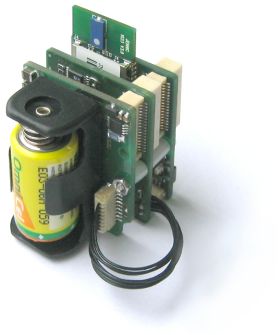
3.1.1.1 Applications

Initial research in WSNs was motivated mainly by military applications leading to well-known research projects like *Smart Dust* [202]. In its context the military application of a WSN was, among other aspects, successfully tested in the *29 palms experiment* [1]. At this, the sensor nodes were dropped from an unmanned aerial vehicle with the task to build up a network and detect and track military vehicles driving through the operation area. Another military applications represents, for example, a WSN to detect and locate snipers [244].

However, the deployment of a WSN is also advantageous in several civilian application domains. Especially, monitoring and tracking applications benefit from WSNs. Well-known examples can be found in long-term habitat and environmental monitoring applications. In habitat monitoring the objective is to track and observe wild life in real-time, keeping the human disturbances at a minimum. Hiding the sensor nodes in the animals' burrows [162, 255] or attaching them directly to animals [123, 293] provides researchers detailed information about the life of the respective animals. Information, that could not easily be gathered otherwise. The same holds true for environmental monitoring applications. For example, in [166] Martinez et al. monitor a glacier by means of a WSN. Especially, they aim at exploring the inner workings of the glacier.



(a) *ESB 430/1 sensor node.*



(b) *iSense sensor node.*

Figure 3.1: *Past and present sensor nodes.*

While human disturbance is irrelevant in this application, the environmental conditions are too harsh for humans to endure for long.

Further extensive fields of possible WSN applications represent the areas of home and office automation, structural monitoring, logistics, disaster relief, and healthcare. More details about the application examples given above as well as further examples can be found in the good survey articles [221] and [291]. While we have presented only a fraction of the possible applications, it is apparent that WSNs are applicable in a very diverse set of application domains.

3.1.1.2 Sensor Nodes

Several different sensor node hardware platforms emerged during the years. Generally, sensor nodes are characterised as severely resource-constrained devices in terms of available computational power, memory, energy, and communication bandwidth. The ESB 430/1, one of the first commercially available sensor nodes, reflects this. It is powered by three standard AAA batteries, incorporates 60 KB flash memory, 2 KB RAM, and 64 KB EEPROM. Furthermore, it uses the 16 bit microcontroller MSP 430 from Texas Instruments which is working at 4 MHz. For communication purposes, a RFM TR1001 radio is used, working at a fixed data rate of 19.2 Kbit/sec in the 868 MHz band. The node is equipped with several sensors, e. g. for measuring temperature and movement detection. The ESB 430/1 was developed at the FU Berlin in the ScatterWeb project [4] and served as our first research node. Figure 3.1a gives a descriptive illustration of the ESB 430/1.

Since the first designs, sensor nodes have continuously been enhanced and refined. An example of a state-of-the-art node represents the iSense sensor node [40], developed by the coalesenses GmbH. Compared to the ESB 430/1 it is significantly more powerful. It comprises a 32 bit RISC processor working at 16 MHz, 96 KB RAM, and 128 KB flash memory. Furthermore, it incorporates the 802.15.4 communication standard, also known as ZigBee, offering a data rate up to 250 Kbit/sec. Figure 3.1b provides an illustration of the iSense sensor node.

Usually several sensor modules are available for the different hardware platforms, ranging from generic sensors, e. g. to measure temperature, humidity, or light intensity, up to highly specialized sensors, e. g. to detect certain chemical substances. Depending on the application, actuators may be incorporated in the sensor nodes as well.

Generally, today's commercially available sensor nodes are still light-years away from the envisioned microscopic nodes, also known as Smart Dust [273]. However ongoing advances in integrated circuit design, which continually shrink their size, while simultaneously improving their capabilities, may make this vision reality in the future. However, energy supply will remain a critical issue in the foreseeable future, as the progress in this area of research is significantly slower than for example the increase in CPU speed, as we have discussed in Section 2.2.3 (cf. also Figure 2.4).

Furthermore, several application-specific sensor nodes are likely to emerge. Specialized nodes that are robust with regard to extreme environmental conditions and that carry only the required hardware, e. g. in terms of CPU, memory, and sensors, to optimally support the envisioned application are clearly needed. Since the application scenarios vary greatly, a "one size fits all"-approach is impractical. In certain cases it may also be beneficial to include a limited amount of more powerful devices in the WSN to support for example some complex computation. Thus, a great variety of different sensor nodes will be employed in future UC environments.

3.1.1.3 Challenges

Generally, the special characteristics of WSNs provide many challenges for software engineering. Since WSNs constitute in essence extremely large distributed systems consisting of tiny, resource-constrained embedded devices that communicate wireless, well-known challenges from the research areas of distributed systems, embedded systems, and wireless ad hoc networks mostly present challenges in WSNs as well. Without aiming for completeness, we highlight some of the major challenges in more detail in the rest of this section.

Resource Constraints First and foremost, the resource constraints in WSNs represent significant challenges. Compared to today's state-of-the-art computer technology, the resources available to the sensor nodes, as described in the

last section, are of several magnitudes lower, regarding CPU, memory, and communication capabilities. While it is possible to build better equipped sensor nodes, the trends towards cheap and tiny devices counteract this.

Another critical aspect in the context of resource constraints represents the limited energy that is available for each node. Especially, the wireless communication is by far the most expensive operation regarding energy consumption. Hill et al. ascertain in [101] that transmitting one bit requires as much energy as roughly 100 CPU operations. However, sensor nodes must communicate frequently in order to organize the network and cooperate in attaining their operational goal. Furthermore, depending on the application, WSNs are envisioned to operate for years and once deployed, the effort to replace the battery is often practically infeasible. While some applications may allow for energy harvesting, e. g. by using solar panels, the introduction of data aggregation inside the WSN [98, 141] and sleep cycles, i. e. times during which the nodes shut down their communication and switch the CPU into a low-power mode, represent the generally accepted methods in order to save energy. While the first solution requires more computation, the main drawback of the latter approach is that during the sleep cycles, the nodes are deaf and blind and may miss interesting events. Furthermore, synchronization of the nodes' sleeping cycles is mandatory in order to allow for communication between the nodes.

In essence, the software employed on the sensor node must make efficient use of CPU and memory while enabling low-power communication to meet the challenges arising from the resource constraints. Thus, the philosophy of getting the work done as quickly as possible and going to sleep afterwards proves beneficial for sensor nodes at large [101].

Self Organization and Robustness WSNs are deployed in the real world, on, in, or close to the phenomena they are supposed to monitor. Thus, in several application scenarios the application terrain is either not easily accessible by humans, characterized by harsh environmental conditions, or even considered hostile. Thus, once deployed the sensor nodes are supposed to autonomously find their neighbours and organize themselves into a network. This includes, for example, building up a routing tree, determining their locations and synchronizing their time. This self-organization is essential for other reasons as well. First, a random deployment, e. g. dropping the nodes from a low-flying airplane, is explicitly envisioned. Furthermore, the exact conditions and characteristics of the application terrain can usually not be predicted in all details before the deployment and it is impractical to configure large-scale networks manually after deployment.

In addition, WSNs are supposed to operate unattended. Thus, continuous adaptations are usually necessary during its lifetime, e. g. due to mobile or failing nodes or general changes in the application environment. Regarding the robustness of the WSN, especially failing nodes or defect ones that communicate erroneous data samples must be reckoned with and handled appropriately.

Additionally, challenges from the wireless communication domain, such as unreliable communication, unstable routing paths, or the problem of topology control [230] must be considered as well, in order to build a robust network.

Programming Abstractions A key to widespread usage of WSNs is supporting the application developer with regard the programming of the sensor nodes. In essence, these represent embedded systems, which are generally difficult to program and debug. Furthermore, the heterogeneity and mandatory optimization of the application to the specific hardware capabilities represent major challenges, as they require expert expertise in various fields of computer science. Thus, relieving the application designer from low-level aspects, such as the detailed handling of the sensors and the sending and receiving of messages, allows to programmer to concentrate on his application. Besides, including aspects, such as mobility of nodes and scalability of the employed algorithms, usually provide additional challenges for the application programmer.

Security Finally, security plays of course an important role in WSNs as well. Since WSNs enable a qualitative assessment of their surrounding orders of magnitude more detailed than today's measures, security-related aspects, such as location privacy and the protection of personal data, become high-priority challenges. Since we focus on security in this work, we discuss the security-related aspects of WSNs separately and more detailed in Section 3.2.

This concludes our overview of WSNs. For more details we kindly refer the interested reader to the good survey articles [5], [291], and [295]. While the latter two are more recent and thus provide more up-to-date works in WSNs, all represent good starting points to find in-detail information about certain aspects of WSNs.

3.1.2 Fabric

WSNs constitute massively distributed systems with only very scarce resources, as we pointed out in the last section. Especially, application development for these kind of networks is complicated and error-prone, as it requires expertise from various fields. In addition to their distributed nature, heterogeneity, energy awareness, and harsh resource constraints are constant challenges in WSNs. It is desirable to hide these complex aspects of WSNs from the developer in order to be able to focus the efforts on the target application during development. In the following we provide a brief introduction to the middleware synthesis framework Fabric, proposed by Pfisterer et al. in [199, 201]. In Section 3.4 we extend Fabric with security-related aspects, that allow for a seamless integration of security in the process of WSN application development without requiring users to be security experts. Although Pfisterer et al. include security related aspects in their description of Fabric, they use these just to

exemplify the framework's functionality. However, they did not elaborate on their realization, nor did they provide an implementation.

3.1.2.1 Overview

FABRIC supports WSN application development by generating custom-tailored middleware instances. The authors point out that a major advantage of this approach is that exactly the required functionalities are provided, while functionalities that will never be used during operation are omitted. This results in lean code, well-suited for resource-constrained devices.

The underlying idea of FABRIC is that each data type of the application may require a certain handling. To assign the appropriate handling methods to the individual data types, the application developer annotates the data type definitions with treatment aspects. A code generation processor passes both, type definitions and annotation aspects to so called modules that provide the actual functionality. These modules finally generate source code for the annotations they are in charge of, e. g. compact serialization or reliable messaging.

Therefore, only the required functionalities for the defined data types are generated by the framework, resulting in high-level data management operations for transmitting and receiving instances of the application's data structures. On the one hand this relieves application developers from dealing with low-level WSN issues while on the other hand still allowing for data type specific treatment. Finally, in order to build the resulting application binary, the synthesized middleware code along with the application code and eventual OS or firmware of sensor nodes must be compiled and linked together.

3.1.2.2 Architecture

Figure 3.2 illustrates the FABRIC architecture. On the left hand side, the required input by the application developer can be seen. It includes the actual application code, annotated data type definitions, and a target specification that defines properties of the target platform, such as the programming language and hardware type. These parameterizations are fed into an instance of the FABRIC-framework which is provided by a framework developer. An instance of FABRIC consists of the generic FABRIC generator, a framework specification, and a selection of modules providing the actual source code as depicted in the middle part. On the right hand side the generated middleware is illustrated encapsulating the low-level WSN aspects and providing the type-specific API for the application developer to use in his application.

The architecture indicates that two fundamental roles exist in FABRIC: an application developer and a framework developer. These result in different views on the framework. While the application developer's focus is on having an easy to use and flexible system to help devising his application, the framework

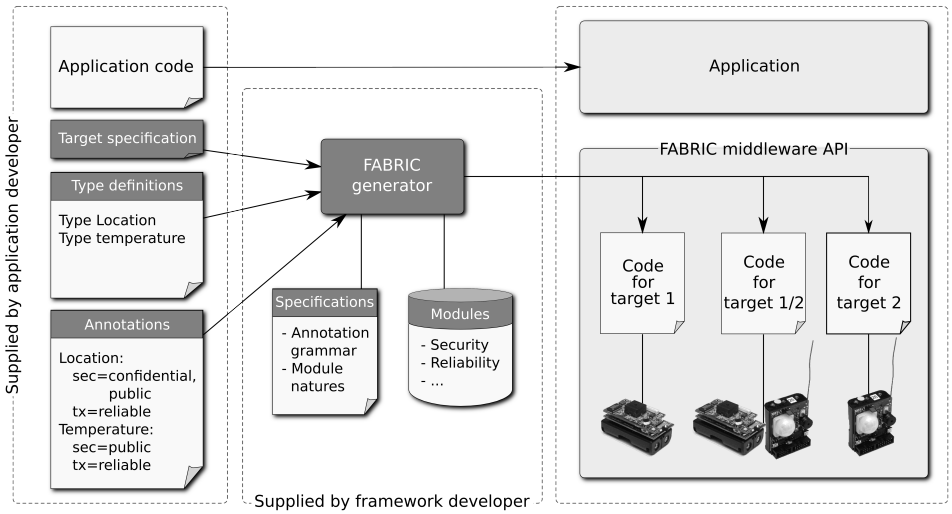


Figure 3.2: *The architecture of* FABRIC.

developer's focuses on providing the framework specification as well as sound implementations of the supported concepts, i. e. the modules.

Application Developer The application developer specifies the data types that the target application will contain and annotates them with treatment aspects that define how the data type will be handled by the middleware. Aspects are grouped inside FABRIC into domains that embrace related aspects. The concept of annotating data types is realized using a feature of the well-known XML Schema standard that allows amending most schema elements with XML documents. Annotating a data type therefore simply means attaching an XML document to each data type definition. Figure 3.3 gives an excerpt of such a specification, displaying the data type "`location`" along with the corresponding annotation from the domain "`serialize`". The annotations are contained inside an appinfo tag intended for machine readable information.

In addition to the annotated type definitions, the application developer needs to specify the targets, i. e. which hardware platform and programming language the middleware should support. Finally, he can devise the envisioned application assisted by the provided type-specific API of the middleware. Note that the treatment of each data type can be altered simply by changing its annotations leaving the application code unchanged.

Code Generation Process For each annotated aspect, a number of modules provide source code. Modules of the same aspect may target different hardware platforms, programming languages or be of different complexity. For a given annotated type definition, the framework picks the best modules based on each module's self-description. The code generation processor reads the schema,

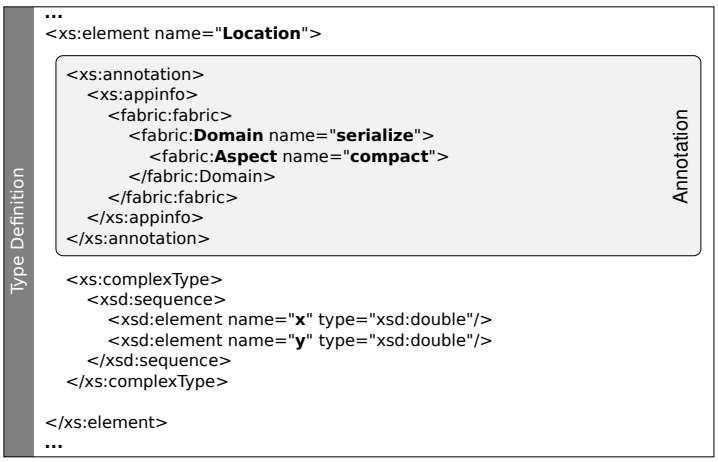


Figure 3.3: *Example* Fabric *XML-Schema Annotation.*

and passes both type definitions and annotation attributes to a set of selected modules. These then generate source code for the functionality they are in charge of, e. g. compact serialization.

With regard to the example in Figure 3.3 the generated middleware contains code to send location data using compact serialization. For a formal description and an example of the module selection we refer the interested reader to [201].

Framework Developer The framework developer provides the Fabric framework, namely the specifications and modules. As an expert in his field, he is able to provide sound implementations of the supported concepts, in form of the modules, which provide the actual source code. The specifications guide the application developer as they reference all available functionality and specify the grammar of the data type definitions and annotations.

This completes our short introduction to the middleware synthesis tool Fabric. More in-depth information about Fabric, including a discussion about middleware approaches for WSNs, details about Fabric's inner workings as well as an evaluation of the concept can be found in [199, 201].

3.2 Security in Wireless Sensor Networks

WSNs represent the eyes and ears by which UC environments will perceive the real world. The fact that they potentially permeate every aspect of the real world, including the private areas of our lives, makes the provision of security one of the fundamental challenges in WSNs.

Especially the unmatched data quality gives rise to many concerns about the privacy with regard to personal data. However, with increasing office automation WSNs will also be aware of many sensitive information that companies need to keep confidential. Finally, decisions and actions will be based increasingly automated upon the gathered data. Therefore it is essential to include security measures in order to enable the safe and secure operation of WSNs.

In general, security is an important requirement in each WSN application that collects, processes and communicates secret or private personal data, that provides security-critical functionality, such as keyless door entry, or that depends on data correctness in order to provide reasonable responses to sensed events. Furthermore, highly personalized services and law regulations may demand certain security aspects to be present as well. Ultimately, including security into WSNs is also the key towards gaining widespread users' acceptance for this new technology.

3.2.1 Requirements and Challenges

Many of the security requirements and corresponding challenges common to traditional networks are relevant in the context of WSNs as well. However, from the specific characteristics, which we have described in Section 3.1.1.3, arise on the one hand additional security requirements, while on the other hand, they make it difficult to directly apply the existing security measures today in these networks.

Requirements Regarding the security requirements, the commonly prevailing security services confidentiality, integrity, availability, and authenticity are needed in WSNs as well. However, due to the unattended operation and potential accessibility of the sensor nodes, node failure or even compromised nodes must be considered. Thus, all employed security measures must be resilient with regard to these situations and degrade gracefully in their performance or security, i. e. the damage from the attack must be contained and minimized to the greatest possible extent. Additionally, the employed security measures must continue to operate properly and provide as much security as possible in the event of single failing or compromised nodes. Furthermore, data freshness, i. e. measures to ensure that messages are recent, is of paramount importance to prevent message replay attacks. Due to the envisioned size of WSNs this is essential in achieving robustness regarding the system operations inside the network. Finally, since the sensor nodes are required to self-organize, the required operations, e. g. for localization and time synchronization, must not be vulnerable to attacks.

Challenges As a general rule, WSNs inherit many of the common challenges from distributed systems, mobile ad hoc networking, e. g. the lack of a physical infrastructure and the dependence on an insecure wireless media [46], as well

as embedded systems, e. g. see [213] for an excellent overview. Especially the latter are much pronounced in WSNs. The severe ressource limitations provide considerable obstacles with regard to securing WSNs. First, cryptographic measures must not utilize the CPU excessively in order not to interfere with the node's system operations. Furthermore, security measures must neither require much communication for coordination purposes, nor enlarge the messages significantly, since the sending of message is the most expensive operation in WSNs with regard to energy consumption. Finally, the memory needed to store key material is strictly limited, requiring sophisticated key management schemes.

Another set of challenges arises from the self-organization and unattended operation in accessible or even hostile environments. We already mentioned that the employed security measures must be able to operate in spite of failing or compromised nodes and only gracefully degrade in their performance as well as provided security. However, this requires a careful and anticipatory design of the security measures and represents in general a topic that requires further research [7]. In case of failing nodes, the application scenario may also demand the deployment of additional sensor nodes at a later point in time to compensate for the dead nodes. Regarding security, the measures must therefore allow to securely integrate new nodes at any point in time after deployment.

3.2.2 Attacker Model

Generally, the attacker model for a WSN must be evaluated with respect to its application scenario. For example, a WSN for habitat monitoring is much less likely to be attacked than a smart office or even a military-type WSN. Furthermore, the types of attackers will doubtless differ between the application scenarios, e. g. regarding their capabilities. However, a generic model, which can be evaluated in more detail within the context of the specific application is advantageous in order to assess the provided level of security.

As a basis, the Dolev-Yao [67] attacker model (cf. Section 2.1.1.3), which is widely used in distributed systems, can be used in WSNs as well. However, it must be extended to account for the special WSN characteristics. The model assumes in essence an insecure communication channel, while the endpoints are not themselves subject to attack. However, this assumption is not reasonable in WSNs. Since the sensor nodes should be as cheap as possible only critical applications with a high demand for security will utilize tamper-proof hardware, due to cost factors and the general difficulty in building such devices [9, 10]. Furthermore, the sensor nodes operate unattended in real-world settings, which an attacker may access in order to physically tamper with the nodes, e. g. to extract sensitive information. Although tampering with a sensor node is not trivial as it mostly requires to remove the sensor node from the WSN for a noticeable amount of time [21], it is also not infeasible. Therefore, in addition to the insecure communication channel the end-points must be considered untrusted in most application scenarios as well.

In addition to these general assumptions the exact capabilities and the extent of the attacker's actions need to be specified. Often the attacker is categorized with regard to his equipment as being either a node-class or laptop-class attacker and regarding his knowledge and access to the WSN as either an insider or outsider [129].

A more differentiated framework for modelling attackers in WSNs is proposed by Benenson et al. in [22]. They classify attackers according to the four parameters goals, presence, intervention, and available resources.

Generally, the *goals* of an attacker are hard to derive and specify. Therefore, the primary security services (confidentiality, integrity, and availability) are usually considered equally important.

The attacker's *presence* specifies the scope of the WSN the attacker is able to influence. Benenson et al. differentiate along this dimension the categories: local (the attacker can influence only a small portion of the network), distributed, (the attacker is either mobile or may influence different though not connected parts of the network), and global (the attacker may influence all nodes inside the WSN).

The dimension *intervention* specifies the abilities of the attacker. Categories in this dimension include in increasing order of strength:

- Eavesdrop: the attacker may only mount passive attacks.
- Crash: the attacker may additionally crash nodes, e. g. by draining them of power, destroying them physically, or by sending them corrupt messages. Essentially the node stops working afterwards.
- Disturbing: the attacker can partially disturb protocols, e. g. by physically moving nodes, selective jamming of the network, or influencing the sensor readings, for example, by holding a lighter close to a temperature sensor of a node.
- Limited passive: the attacker is able to compromise nodes, i. e. retrieve all stored information. But in order to to this, he has to remove each node from the network and take it to a laboratory for tampering.
- Passive: the attacker is able to compromise nodes in-situ, i. e. he does not have to remove them from the network. Furthermore, he is able to modify the data on the node.
- Reprogramming: the most powerful attacker regarding this parameter may reprogram nodes in order to execute arbitrary software. thus he may for example clone nodes.

Regarding the *available resources* several aspects may be differentiated concerning the attacker's funding, equipment, expertise, and time. Again this requires some guesswork during the security evaluation.

The two significant parameters in this framework with regard to the security evaluation of a specific WSN are intervention and presence, since they allow for a specific definition of the attaker. Figure 3.4 visualizes attacker types of different strengths with regard to these dimensions. The arrows describe the

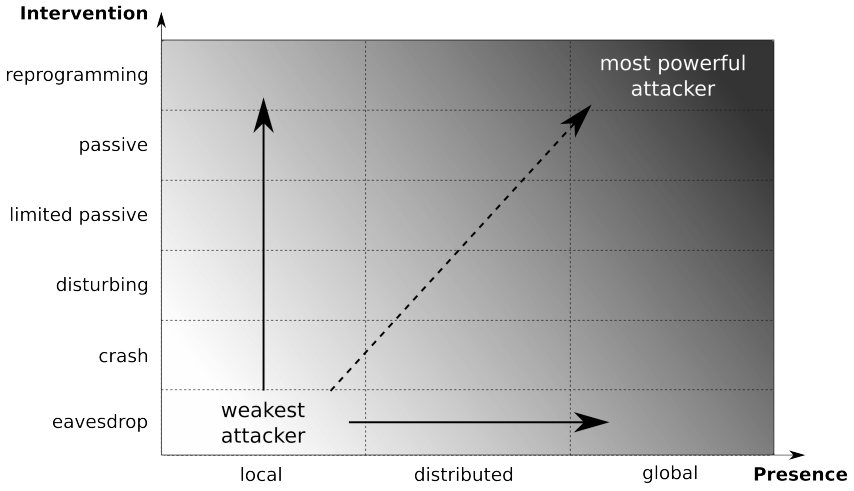


Figure 3.4: *WSN attacker model dimensions*

paths for the attacker to gain more strength. For example, the weakest attacker type may only eavesdrop on the communication in a small part of the WSN, while the most powerful attacker has global reprogramming capabilities.

3.2.3 Attack Possibilities

The commonly known two categories of active and passive attacks can also be applied to sensor networks. However, sensor networks are susceptible to more attacks than ordinary networks, due to their special characteristics. Although, several publications categorize and describe the special attack possibilities on WSNs, e. g. [95,129,222], we differentiate in the context of this work for brevity only between passive and active attacks.

Passive Attacks Passive attacks on the sensor network are relatively easy because of wireless communication. The illegitimate disclosure of information, which breaks confidentiality, presents the major threat here. Therefore, information that is exchanged between the nodes should be encrypted to ensure its confidentiality. Furthermore, by means of traffic analysis an attacker may gain important knowledge about the network structure or the occurrence of special events.

Active Attacks Active attacks allow the attacker to seriously disrupt the functioning of the WSN. Several points of attack are imaginable to influence the WSN in its operation. First, on a message level, it must be ensured that it is impossible to inject or replay messages into the WSN. Otherwise an attacker

could masquerade as a legitimate member of the network and send its own information or replay old data, which might lead to wrong decision making inside the WSN. The commonly known threat of message modification, however, does not pose a significant threat in our opinion. Due to the local broadcast of the messages, it should be almost impossible to intercept and modify a single message before any other node receives it. However, capturing a message, modifying it, and injecting it at a different place inside the WSN must be considered, though this represents a special case of packet injection. Another means of influencing the WSN originates from the way sensor networks operate. If an attacker is able to generate physical stimuli, e. g. by holding a cigarette lighter close to a temperature sensor, he can influence what data the sensor nodes collect.

A serious vulnerability arises from the physical accessibility of the sensor nodes. Communicated location information might lead attackers to find out about the positions of single nodes, which further emphasizes the need to keep the communication between the nodes confidential. Furthermore, considering that the sensor nodes are usually not tamper-proof, it is possible to physically attack accessible nodes. Though tampering sensor nodes is not an easy task, as we have already discussed in the last section, it is still possible.

Also several ways exist to start a Denial of Service (DoS) attack against a WSN [284]. For example, an attacker who is able to create an extensive amount of physical stimuli might paralyze the network or at least parts of it, as the real events drown in the artificial noise of stimuli. Furthermore, attackers who possess the location information of the nodes could capture or destroy the nodes one by one. The traditionally known jamming of the wireless medium or sensor specific DoS attacks, such as sleep deprivation torture [251], i. e. keeping a node awake in order to drain its energy resources, present threats as well.

Finally, several specialized attacks on the basic services, such as routing [129] or time synchronization [163] exist.

3.2.4 Security Solutions

During the past few years there has been an explosive growth in the research devoted to WSN security. Multiple approaches have been proposed in the literature that address various aspects of WSN security, realizing most security services. In the following we provide an overview of well-known proposals with regard to the most significant security services confidentiality, integrity, authenticity, and availability. Furthermore, we briefly discuss security solutions for fundamental WSN services as well as solutions for key management.

Confidentiality To achieve message-based confidentiality in the communication between the nodes, many of the traditional well-established cryptographic algorithms may be used to encrypt data in WSNs as well. In general, the well-known symmetric algorithms, e. g. AES [264] or RC5 [217], as well as asymmetric algorithms, e. g. RSA [219] or ECC [139,140] may be utilized. However, most

algorithms have been designed for full-scale computers and thus need to be evaluated with regard to performance when implemented on the severely resource limited sensor nodes. Especially asymmetric cryptography is often considered to perform poorly, although studies show that it is in general feasible on sensor nodes, e. g. see [29, 93, 157]. However, in particular the individual architecture of the sensor node's hardware affects the performance of the encryption algorithms. Comparative analyses of the performance of several cryptographic algorithms on different sensor node platforms present for example Law et al. in [146] and Roman et al. in [220]. A more extensive discussion about the general challenges as well as an extensive analysis of the suitability of several cryptographic algorithms for WSNs presents Kaps in his dissertation [126].

Integrity and Authenticity Integrity and authenticity of a message is often provided simultaneously by means of a message authentication code (MAC) that is attached to the message. For protection of the link layer in WSNs, i. e. providing integrity and authenticity for single hop communication, the TinySec library [128] or its successor MiniSec [159] provide the necessary mechanisms when using sensor nodes that implement TinyOS [151]. Furthermore, WSNs that operate according to the ZigBee standard [297] must provide the security measures specified in the IEEE 802.15.4 standard [145], including link-layer encryption. For this purpose, compliant sensor nodes, such as the iSense node often contain a hardware chip that provides link-layer AES encryption.

Multi-hop integrity and authenticity may be achieved by utilizing end-to-end pairwise keys between distant nodes, instead of using pairwise keys between neighbouring nodes. A special kind of multi-hop authenticity represents broadcast authentication, in which all nodes may verify that received messages originated from the claimed source and were not modified en-route. μTESLA represents the most prominent approach for broadcast authentication in WSNs. It was proposed by Perrig et al. for the SPINS protocols [196]. In essence, μTESLA encrypts messages with a symmetric key, which is made public in the subsequent time slot, in order to allow for verification on all nodes. μTESLA requires for this loose time synchronization between the nodes. The authors of LEAP [296] provide broadcast authentication by means of one-way hash chains.

Availability Availability represents and interesting research topic in WSN security, since the possibilities for DoS attacks are manifold and mostly easy to execute. Wood et al. discuss in [284] several ways to execute DoS attacks in WSNs as well as possible defenses. Furthermore, they present in [285] mechanisms to map the region in which the WSN suffers from a DoS attack. On the one hand, this allows to circumvent the affected area in the communication of the other nodes. On the other hand, defense operations may be targeted at the affected network region. Further defense proposals include for example evasion techniques that allow for sending an alarm despite an active DoS attack [42] and an approach to protect end-to-end communication against DoS attacks [64].

WSN Services Several fundamental services require protection in WSNs in order to allow for undisturbed operation. This includes especially routing and data aggregation. Functional routing is very important to guarantee the operations in WSNs when facing attacks. If the routing fails, no sensible communciation is possible between the sensor nodes. Karlof and Wagner discuss in [130] several possibilities to disrupt routing in WSNs as well as possible defenses. Further approaches to secure routing are presented for example in [260, 283]. Secure data aggregation is an equally important service because undetected compromised nodes may send erroneous data to influence the WSN. Ye et al. analyse in [289] how to detect false data that was injected by compromised nodes. A secure hop-by-hop data aggregation protocol is proposed by Yang et al. in [288]. Further approaches include for example [51, 106, 205]. A survey providing additional solutions for secure data aggregation is presented by Sang et al. in [229].

Key Management Most of the approaches described above require a specific kind of key management to operate properly. Generally, key management represents the foundation of any security infrastructure, since most cryptographic measures involve keys. Therefore, a significant body of research proposals is devoted to key management in WSNs. Generally, nodes may be provisioned with the appropriate key material before they are deployed or they may establish their key material during runtime after the deployment, according to the respective key scheme protocol. Both, asymmetric as well as symmetric key schemes can generally be used in WSNs. Although asymmetric cryptography is feasible on sensor nodes as we have discussed above, it is often considered too inefficent and consequently not employed. Therefore, the majority of key management proposals focus on symmetric key schemes.

The simplest key management technique is to provide all sensor nodes with a single network-wide key before deployment that is later used exclusively in all cryptographic operations. While this scheme requires only limited memory on the sensor nodes and avoids complex protocols, compromising a single node reveals the key to the attacker, which results in a collapse of all security measures. Furthermore, providing end-to-end security and node authentication is not possible using a network-wide key, since all nodes share the secret key.

Pairwise keys provide better security with regard to these security shortcomings, albeit at a higher cost in terms of the required memory to store the keys. In general, a pairwise key is shared exclusively by two nodes, in order to secure their communication. However, storing a pairwise key for every other member in a large WSN quickly exhausts the available memory on a sensor node. Thus, the scalability of the simple pairwise scheme is limited.

Several key management schemes have been published that establish pairwise keys while trying to remedy the scalability problem. Some schemes simply assume that during the first time intervals after the deployment of the sensor nodes no attacker is present. Thus, the nodes may use unprotected communica-

tion to establish pairwise keys during this time [8, 296]. Other approaches, the random key pre-distribution schemes, establish pairwise keys by first generating a large key pool from which they provide each node with a random subset. These schemes require in a second step a discovery protocol after deployment in which the nodes share for example the IDs of their keys to determine a pairwise key. If no common key could be found an additional protocol may generate a new pairwise key by determining a secure path via other nodes with which each of the two nodes shares a pairwise key. Schemes that operate according to this scheme are for example [50, 73]. Using these, two nodes may establish a pairwise key only with a certain probability. Furthermore, the same keys may be used several times as pairwise keys between different node pairs, which impedes node authentication. Related approaches exist that substitude the key pool for example with a pool of Blom's matrices [68] or bivariate polynomials [155]. From these keys can be derived that are bound to the node IDs and thus allow for authentication. Further key schemes aim for example at deterministically determining a pairwise key between any two nodes, e. g. [49, 147] or provide location-based pairwise keys [156].

This short overview represents only a small fraction of the proposed key management schemes for WSNs. More extensive discussions about key management schemes for WSNs can be found for example in [47, 286, 295].

Before finishing this overview we would like to present briefly the random key pre-distribution scheme Hashed Random Preloaded Subsets (HARPS) [208], which we will support in our security framework in Section 3.4. Similar to other key management schemes in this category, the nodes are configured with a number of randomly chosen keys from an initial key pool that is generated prior to the deployment of the nodes. However, HARPS additionally utilizes a hash function to diversify the keys further before they are stored on the individual nodes. Thus, a node contains a certain number of keys, identified by their ID, in a random hash depth, i. e. the hash function is repeated several times taking the hash value of the previous run as input for the next run. In addition to the keys, the nodes are provided with the used hash function as well as a selection function, i. e. using this function with a node's ID as input reveals the key IDs and hash depths of the keys that are stored on that node. Now, if two nodes want to derive a pairwise secret key they determine if both have one or more keys with the same ID in common by using the selection function. If this is the case the node that contains the key or keys in the lower hash depth adjusts those by applying the hash function to derive the appropriate keys at the required hash depth, i. e. corresponding to the hash depths of the other node's keys. Finally, the hash function is applied to the concatenation of all common keys to derive the final pairwise secret key.

3.3 A Security Architecture for Mobile Wireless Sensor Networks

During the early years of WSN research most security approaches centered around the idea that every node inside the sensor network shares a secret key with a base station. Furthermore, this base station is considered trusted, i. e. immune to attacks, and is at all times able to communicate with every node in the network, e. g. see [196]. Furthermore, the base station was also often considered resource-abundant and thus able to execute computationally demanding operations for the nodes, e. g. see [33].

However, this focus on a specific network setting limited the applicability of these WSNs security solutions. For example, the envisioned scenarios in which mobile sensor nodes need to be included or if the deployment of the WSN has to be done in an ad hoc manner, without worrying about a supporting infrastructure, are excluded.

Therefore, we have analysed different measures to secure WSNs that do not necessarily require a base station. Especially, we wanted to include in our target design mobile self-organizing nodes that are able to independently create a secure pairwise key whenever the need arises, i. e. one node enters the communication range of another node. With regard to the envisioned security services, the security design should support unambiguous authentication between the nodes as well as encrypted and integrity-assured communication of messages. Due to the ressource limitations of the nodes we especially explored lightweight cryptographic measures for these purposes. As the target hardware platform we chose the ESB 430/1 sensor nodes (cf. Section 3.1.1.2), which represented state-of-the-art sensor nodes at that time.

In the following we first present our findings regarding applicable cryptography on the ESB 430/1 and afterwards present the security architecture for our mobile self-organizing WSN, which was mostly developed in [104]. Furthermore, parts of this research have also been published in [232].

3.3.1 Cryptography on the ESB 430/1

Employing cryptographic measures on sensor nodes requires careful design, keeping the constraint nature of the resources that are available on the sensor node in mind. Since the main goal of the nodes is to fulfill their operational task, the employed cryptography must not use more than a small fraction of the available memory and processor time.

Therefore, cryptographic algorithms should have a small memory footprint, regarding the static code size as well as the dynamic memory usage during runtime. Furthermore, they should be fast in their computation. In general, the memory requirements during runtime as well as the level of security provided by the employed cryptography depends largely on the key size. However, while

longer keys provide a higher level of security, storing for example pairwise keys for numerous other nodes in the network quickly exhausts the available memory if long keys are used. Therefore, the actually required level of security should determine the key size. In the following we take a closer look into both aspects.

Key size Cryptographic mechanisms are generally considered computationally secure, if the security is based completely on keeping the key secret, not the algorithm. Thus, even though an attacker may have complete knowledge about the inner workings of the algorithm, it is infeasible for him to decrypt the ciphertext without the correct key. However, a brute force attack, i. e. systematically trying every possible key in the key space, is always viable. Thus, the security of the information protected by cryptography directly depends on the strength of the employed key, i. e. its size. Choosing the right key size is therefore of utmost importance. Discussions about and guidelines for this can be found for example in [19, 150].

In general, a cryptographic key has the right length, if

1. breaking the key is more expensive than the value of the safeguarded information, or
2. breaking the key takes a longer timespan than the information is valid or useable.

Thus, even short key sizes may provide a sufficient level of security, if these rules hold for the envisioned application. To evaluate which key size is sufficient, it is therefore required to determine the average time required to break the key by estimating the runtime the cryptographic measures require for this task. However, these depend in large parts on the complexity of the underlying cryptographic algorithm. For example, a brute force attack on a 56 bit DES key requires $5 \cdot 10^5$ MIPS years[1] [150]. Using commercially off-the-shelf available Pentium 4 (2 GHz) PCs, which are considered 4000 MIPS machines, 11 machines are required to break the DES key within one month. Nevertheless, specialized hardware may reduce this time further [65, 71].

In general, a brute force attack requires on average testing half of all possible keys in order to find the correct one. In Table 3.1 we provide an overview of several key sizes and the resulting average times for the brute force attack based on [253]. Especially, we list the two cases where the computer is able to perform one encryption per microsecond and 10^6 encryptions per microsecond. Thus, breaking a 56 bit key, which is used by DES, takes 10.01 hours if 10^6 encryptions per μsecond can be performed by the computer.

Cryptographic algorithms Most cryptographic algorithms are designed for full-scale personal computers, utilizing for example the available 32 bit registers. Since our sensor node represents a much weaker device, from a perfor-

[1] One MIPS year represents the amount of work that one computer, with the capability of processing one million instructions per second, could perform in one year

Key Size	**1 enc./μsec**	**10^6 enc./μsec**
32	35.8 minutes	2.15 milliseconds
56	1142 years	10.01 hours
64	$2.9 \cdot 10^5$ years	106 days
80	$1.9 \cdot 10^{10}$ years	19000 years
100	$2 \cdot 10^{16}$ years	$2 \cdot 10^{10}$ years
128	$5.4 \cdot 10^{24}$ years	$5.4 \cdot 10^{18}$ years
168	$5.9 \cdot 10^{36}$ years	$5.9 \cdot 10^{30}$ years

Table 3.1: *Average time required to brute force different key sizes*

mance point of view, we started out to determine first, which cryptographic algorithms provide feasible solutions for the ESB 430/1. Therefore we analysed in [104] several algorithms to determine their applicability.

Generally, asymmetric as well as symmetric cryptography could be employed to achieve security. Guajardo et al. prove in [91] that an implementation of elliptic curve cryptography, one of the fastest available asymmetric methods is possible on the ESB 430/1. Although Guajardo et al. use several available measures to optimize their implementation, e. g. by programming directly in the machine code of the CPU, the results indicate, that it seems to be infeasible to implement a fast public key system on the given hardware platform. The calculations for encryption and decryption would delay the message transmission for more than a second, which is unacceptable. Consequently we refrain from considering asymmetric cryptography further.

Focussing on symmetric cryptography algorithms, we analysed the final candidates that competed for the advanced encryption standard (AES), solicited by the National Institute of Standards & Technology (USA) (NIST) to determine the follow-on algorithm to DES. In detail, we determined based on the respective source code the necessary CPU cycles for each algorithm and estimated on this basis the required runtime for encrypting a 32 byte text on the ESB 430/1. Furthermore, we calculated the required memory each algorithm requires during runtime, when an 80 byte secret key is utilized. The overall memory requirements for each algorithm depend in large parts on the implementation and the compiler settings and can therefore not easily be estimated. Table 3.2 displays the results of our analysis. For each algorithm we denote the reference as well as the considered encryption rounds, i. e. several algorithm may be used with different rounds. Furthermore, we present the CPU cycles, required times and memory. For example, the 10-round Rijndael algorithm requires 11824 CPU cycles during encryption, which results in 12.5 milliseconds to encrypt 32 byte of data. The whole operation furthermore requires 48 byte of memory during runtime when an 80 bit secret key is utilized. The detailed descriptions and security analyses of the individual algorithms are available in [104].

The Rijndael algorithm [264] offers the best performance in terms of required time and runtime memory. However, it requires a large S-Box, i. e. a scheme

Algorithm	Ref.	Rounds	CPU Cycles		Time[1]	Memory[2]
			Setup	Encryption	[ms]	[byte]
AES/Rijndael	[264]	10	–	11824	12.5	48
Mars	[39]	32	33272	16180	34.7	252
RC5-32	[217]	12	30316	6528	29.8	114
RC6-16	[218]	20	21604	7804	27.9	96
RC6-32	[218]	20	72908	19252	58.9	188
Serpent	[27]	16	–	29212	30.9	68
Serpent	[27]	32	–	57980	61.3	68
Twofish	[235]	16	–	59972	63.4	56

[1] *Time in milliseconds required to encrypt a 32 byte text.*
[2] *Memory required during runtime in byte, using an 80 bit secret key.*

Table 3.2: *Symmetric cryptography performance on the ESB 430/1*

to implement a non-linear substitution of the bits, which adds around 10 KB to its source code. Although, it is independent of the keys, and can thus be included in the source code as constants, the resulting memory requirements for the algorithm are still rather high. Furthermore, Rijndael is not designed for a key lengths shorter than 128 bit, which may be used in sensor networks. Simply padding the key might expose new flaws. Therefore, we decided to incorporate Serpent [27] which has been explicitly constructed to support shorter key sizes. Furthermore, it shows good runtime behaviour because it can be implemented using logical operations only. A reduction of the number of rounds from 32 to 16 results in a linear speed-up. The reduction is reasonable, since Serpent has a generally high security margin [183], i. e. although 16 rounds were considered sufficient against all known types of attack, the designers nevertheless specified 32 rounds to protect the algorithm against future discoveries in cryptanalysis.

3.3.2 Architecture

The security architecture for the mobile self-organizing WSN is based on three different interacting phases. First, a pairwise key agreement to provide mutual authentication and the initial key exchange between the sensor nodes, second, the establishment of sending clusters to extend the secure pairwise communication to secure broadcast inside the communication ranges of the nodes, and third, the encrypted and authenticated communication of sensor data inside these secure broadcast domains.

3.3.2.1 Pairwise Key Agreement

To achieve pairwise key agreement we use the Blundo et al. scheme [31], which is based on a pre-distribution scheme by Blom [30]. It enables two nodes to determine a pairwise secret that is solely shared among these two nodes. The

scheme is unconditionally secure and resistant against collusion of a maximum of t users. In terms of a cooperative sensor network, this means that an attacker has to compromise more than t nodes to compromise the whole network, i. e. being able to derive all secret pairwise keys. By compromising less than t node the attacker gains no knowledge about the other keys.

The scheme requires a certificate authority (CA) which randomly generates a symmetric bivariate polynomial $f(x, y)$ of degree t over an arbitrary finite field.

$$f(x, y) = \sum_{i,j=0}^{t} a_{ij} x^i y^j \quad (a_{ij} = a_{ji})$$

The CA, ensuring that each sensor node x has an unique ID, evaluates the polynomial in the following way:

$$g_{ID}(x) = f(x, ID)$$

Thus, g_{ID} is a polynomial of degree t with a single variable x. The CA transfers the coefficients of the polynomial g_{ID} to the node with the unique identification ID, prior to the deployment of the sensor nodes. These constitute its individual key material.

Two nodes are now able to determine their specific pairwise secret by evaluating their private polynomial $g_{ID}(x)$ where x denotes the other node's identity. It can be derived directly from the symmetry of the polynomial $f(x, y)$ that both nodes calculate the same value. An efficient way to implement the Blundo et al. scheme on sensor nodes is presented in Appendix A.1.

3.3.2.2 Establishment of Sending Clusters

To communicate securely with all nodes within its communication range at once, without creating the overhead of sending every message to each node individually, every node establishes a randomly generated key within its neighbourhood. This key is used solely by this node to encrypt and authenticate its messages. If a node B receives a message M from node A, for which the content cannot be decrypted and authenticated, i. e. node B does not know the key, it calculates the pairwise secret K_{AB} for the sender and itself using the Blundo et al. scheme. This secret is then used to transfer its own sending key securely to the other node, who replies by transferring its sending key using the pairwise secret K_{AB}, which he derives, as well. Node A automatically sends its own key, because we assume that if a node could not decrypt and authenticate the other node's message, either the network is initializing or a change in the topology must have happened and thus the other node equally does not know its sending key. Figure 3.5 illustrates the protocol.

This protocol enables a node to establish its key in a new environment and get to know the other nodes' keys with just two messages per neighbour. It is not

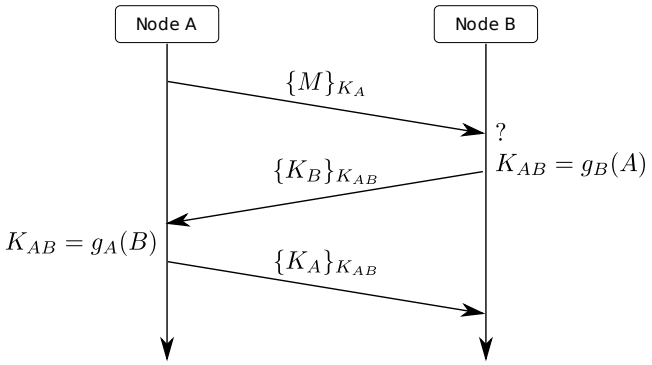


Figure 3.5: *Protocol to establish the sending cluster*

restricted to an initialization phase and can be used at any time and with any other legitimate node of the network. Thus, it supports mobile sensor nodes as well as the deployment of new nodes at a later point of time.

3.3.2.3 Secure Communication

The sensor nodes always broadcast messages to their direct neighbourhood. For encryption purposes the counter mode of operation (CTR) [69] (c. f. Figure 3.6) is used, which allows for an encryption of the message M to the ciphertext C, without changing its length. Since the transmission of messages requires by far the most energy, keeping the original message length is advantageous compared to padding it to achieve a certain necessary block size. In addition to the message itself the counter is added, which results in ordered and unique messages. The counter register is divided into two parts, s which is incremented once per message and t which is reset to zero for a new message and incremented once per block. This overhead can be avoided if both sender and receiver increment the counter after each transmission. However, due to the lossy nature of the communication in sensor networks this procedure is inappropriate. A message loss by any of the neighbours would require two additional messages, the request for the counter value and its reply. Thus, the counter value is included in every message.

It is not necessary to use the full block size of the encryption algorithm for the counter value, because it can be padded with zeros to achieve the appropriate length. Since the same counter should not be used twice, as this would support attacks, the size of the counter limits the number of messages which can be encrypted with a single key.

In order to provide integrity and authentication, we furthermore compute a MAC, using the same encryption algorithm albeit in cipher block chaining (CBC) mode [69], i. e. each block of plaintext is XORed with the previous ciphertext block prior to the encryption.

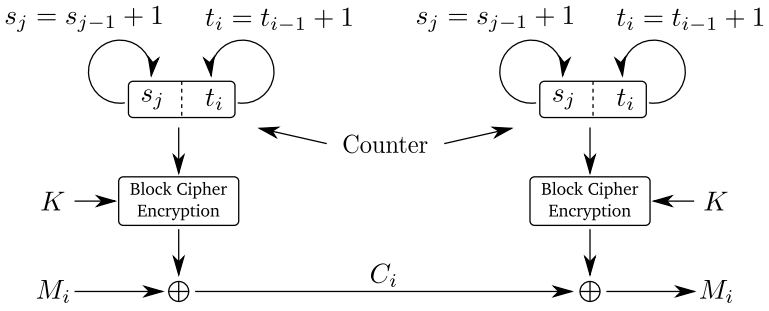


Figure 3.6: *Counter mode of operation*

3.3.3 Implementation

We have implemented a prototype of our security architecture using the ESB 430/1 sensor nodes (cf. Section 3.1.1.2). The algorithms require 17.1 KB of memory. This results mainly from the Serpent algorithm being a standard implementation, which has only been optimized with regard to volatile memory usage and speed but not for code size. During run-time the algorithms need additional 86 byte for their operations. The average delay caused by the encryption is 30.9 milliseconds per 16 byte of cleartext information.

The security degree t, i. e. the number of nodes an attacker has to compromise to successfully calculate the CA's original polynomial, is completely adjustable, but of course influences the run-time and memory requirement. For our implementation that utilizes 80 bit secret pairwise keys, $(t + 1) \cdot 10$ byte are required to store the key material. However, the coefficients can be freely distributed between volatile memory, code space and EEPROM, wherever space is available.

During our tests, we targeted a 100-node WSN and chose the security degree to be 20. Thus an attacker must compromise at least 21 nodes to being able to derive all secret pairwise keys. The coefficients are stored in memory. In addition, a node has to store the sending key of each neighbour. In our implementation nodes store up to 20 sending keys of their neighbouring nodes, which represent a rather high network density. After that they start to reuse the space by deleting the oldest key. In summary, we require approximately a quarter of the available memory on the ESB 430/1 for our security architecture. Regarding the extension of the messages, our prototype used a 4 byte MAC and a 2 byte index, which results in an extension of each message by 6 bytes.

3.3.4 Evaluation

We evaluate our security design and prototype with regard to two aspects. First, in order to assess the provided security, we perform a security analysis.

Afterwards, we evaluate the general quality of the methodology used in our approach.

Security Analysis By employing our security architecture we are able to prevent most of the threats WSNs are subject to, while considering the constraints that the sensor network demands. It provides confidentiality as well as integrity for the communicated information and ensures the authenticity of the sensor nodes. Furthermore, it minimizes the effects of compromised nodes. All deployed cryptographic algorithms are efficient in terms of run-time and memory usage and do not extend the messages significantly.

Confidentiality is achieved by encrypting the messages. This prevents any illegitimate disclosure of information. Furthermore, the MAC ensures the integrity of the messages. We use short MACs in our architecture to reduce the message overhead, which, on the one hand, enables the attacker to determine legitimate messages by brute force. On the other hand, he has no control over the message content due to the encryption of the message. Furthermore, the counter mode of operation, which is used in the encryption process, automatically adds an index to each message without adding overhead compared to simple encryption. Hereby, replay attacks are prevented, because a receiver can keep track of already used counter values, while the messages are only slightly extended.

Only legitimate nodes can join the communication, since no illegitimate node is able to derive the sending key of the other nodes. An attacker may conduct the pairwise key agreement protocol using a captured key request message from another node. However, it is only possible for him to attack the encrypted sending key, which is itself encrypted with the pairwise secret key. Thus, if no new ways to attack Serpent become known, this can be considered impossible. Furthermore, the authentication process is implicitly included in the key exchange, which only requires two additional messages.

End-to-end security can easily be established in our approach, though we did not consider it in our description. If two nodes need to communicate securely via several intermediary hops, they can derive and use their pairwise key to encrypt the message payload rather than their broadcast communication key. Thus, only the two nodes may gain access to the message.

One threat we cannot prevent is the possible capturing and compromising of individual nodes. However, our security architecture minimizes the effects of this attack. While an attacker, who successfully compromised a node, may be able to authenticate himself to the network and is consequently able to join in the communication, he can only do so using the one compromised node. Only very powerful attackers that are able to clone nodes gain more footholds inside the network. However, the communication between non-compromised nodes remains secure. This holds true for up to t compromised nodes, which is based on the security specified by the initial symmetric bivariate polynomial calculations. It is important to note that t is not necessarily linearly dependent on the actual network size. It denotes the number of nodes that need to be physically

compromised, which is by far the most expensive attack in this scenario and thus much less likely than radio-based attacks. Due to the fact that cryptographic algorithms alone cannot prevent DoS attacks, the security architecture does not address these threats any further.

In general the architecture is scalable and robust, since all operations take place inside a node's communication radius. Thus, the actual size of the network does not influence its local security associations.

Methodology Evaluation We have achieved our goal of designing and implementing a security architecture that provides the desired security service for a mobile self-organizing WSN while using only lightweight cryptography. However, all data is handled equally with regard to applying security. This is disadvantageous if only one data type requires protection while the other data types could be transmitted without. Thus, our solutions may generate unnecessary computational overhead. Furthermore, a general drawback arises from this methodology of devising the security solution for WSNs.

Generally, WSNs are applicable to a multitude of application settings, of which several will not be consistent with our target application. WSNs for long-term environmental monitoring applications, for example, most likely comprise a base station and only stationary nodes. Thus, some aspects, such as the ability to securely derive pairwise keys on demand to support node mobility, provide unnecessary overhead in these applications. However, keeping the general ressource limitations of WSNs in mind, any unnecessary overhead should be avoided.

In essence, changing the application scenario has a major impact on the required security setup in most cases. This however often results in the expensive redesign of the security architecture for each new application, which is clearly a daunting task. Furthermore, changing the hardware platform for the sensor nodes usually requires to optimize the utilized cryptographic measures with respect to this new architecture, which requires expert knowledge in the fields of cryptography as well as the programming of embedded systems.

3.3.5 Conclusion

It is without doubt possible to integrate security into resource-constrained devices, such as sensor nodes and thus secure the WSN application according to its specific requirements. However, providing a specialized security setup for a well-defined application does not present an effective approach, keeping the plethora of potential application scenarios in UC environments in mind. Rather, the process of integrating security measures should be easy to use and apply to most application scenarios. Furthermore, special consideration should be given to providing the security measures in a way that allows the application developer to include the required aspects in his application without having to

be an expert in this field. One promising possibility, which we examine in more detail in the next section, is the usage of a middleware approach.

3.4 Annotating Security during Application Development

We have learned that it is inevitable that all discussions about security must have the target application in mind and that the application developer should devise the required security measures. He knows which data needs to be secured with which kind of security service. Furthermore, he knows about the scope of these security measures, e. g. whether the whole communication needs to be secured or only a certain data type when it is communicated.

However, devising and implementing sound security measures requires significant knowledge and experience in this field [7, 282], which a large group of application developers presumably lacks. Therefore, it is beneficial to provide the application developer with easy to use tools to include security in his application while designing it.

Furthermore, a security solution for an application must always be subject to a thorough security evaluation in order to justify its security promises and to foster the application developer's awareness regarding which aspects are secure and which are at risk, thus avoiding a false sense of security. Again, this requires significant expertise in the area of WSN security.

In the following we present a solution to these predicaments. We propose a framework which allows the integration of comprehensible security into the development process of WSN applications. In detail, we show how to automatically devise application-specific security mechanisms from the characteristics of the application. Furthermore, we explain how to automatically provide a security evaluation of the possible security solutions in order to enable the application developer to choose the appropriate solution. Finally, we make these steps usable by integrating them into a middleware synthesis tool. Parts of this research have been published in [210] and [211].

The remainder of this section is organized as follows. First, we take up the subject of application-specific security again, deal with it in greater detail, and derive important aspects which should be considered when providing security for WSNs in the next section. Then we review related work in Section 3.4.2. Subsequently, in Section 3.4.3, we introduce our concept of automatically providing comprehensible security for WSNs. In detail, we explain how to derive viable security solutions from the characteristics of the envisioned application and how to evaluate their security. Furthermore, we describe how these concepts are implemented in the WSN middleware synthesis tool Fabric in Section 3.4.4. Finally, we illustrate the whole process by means of a use-case and give an evaluation of the whole concept in Section 3.4.5 and Section 3.4.6 respectively.

3.4.1 Application-specific Security

The requirements for security vary greatly between applications, mainly based on their data semantics, i. e. the meaning of data to the application. At a high level, different application domains usually have different security requirements for their data. For instance, the security requirements of structural monitoring applications differ significantly from those of health-care applications. Take for example the "Sustainable Bridges" project [164] in which the WSN application continuously monitors the condition of the bridge, in order to detect for example structural defects and to streamline visual inspection. While this application would benefit from mechanisms that guarantee correctness of the WSN data, i. e. by employing integrity and authenticity, it is clear that it could also work without these, since the data is always validated by visual inspection. Furthermore, there is no need to employ mechanisms to achieve confidentiality. In contrast, in health-care applications, such as the "Ubiquitous Support for Medical Work in Hospitals" project [18], confidentiality, authenticity, and integrity are a must, since these applications handle sensitive personal data, e. g. the patients' vital parameters. Furthermore, these applications have a much stronger requirement for data-correctness and availability since the measures taken on the basis of this data can easily be life-threatening.

Even the same data type may have different security requirements in different applications. Consider for example location data that is communicated between the nodes. Due to the physical accessibility of the nodes, the location of a node represents highly sensitive information in surveillance WSNs, such as [137]. Gaining this knowledge gives an attacker valuable information on how to attack or circumvent the WSN. However, location information in environmental monitoring is mostly used to coordinate the WSN and thus is of relatively low value from a security point of view.

Due to the scarce resources in sensor nodes, the strategy of always including all-encompassing security measures is counter-productive in WSNs. An example are the measures to prevent traffic analysis, such as those in [63]. Clearly, for a large group of WSNs traffic analysis is not a great security concern and consequently these networks need not waste resources on the nodes with measures that prevent it. Therefore, a "one size fits all"-approach is not feasible for providing security in WSNs. Rather specialized security building blocks that target exactly the security requirements are needed. Additionally, it is important to differentiate the required security on the basis of the data type. While it may be acceptable to use inefficient asymmetric cryptography to secure a highly sensitive data type, such as the update of a secret key during the WSN lifetime, other data types, such as sensor or management data, should be secured differently in order to achieve the optimal overall performance and security.

Generally, research to provide security for different aspects of WSNs is numerous as we have discussed in Section 3.2.4. Furthermore, it is mostly conducted by security experts, who design secure solutions for certain scenarios, i. e. how

to securely and efficiently establish pairwise keys when nodes are randomly distributed. While all approaches are essential contributions to explore the design space of security in WSNs, the specially chosen scenarios and assumptions make the solutions mostly only applicable to certain application scenarios. For example, Wood et al. present in [283] a family of secure routing protocols. They assume among other things that each node knows its geographic location as well as the location of important other nodes, e. g. the base station, in order to calculate distances. Furthermore, they limit a possible attacker to have only hardware capabilities that are equal to the sensor nodes. While these assumptions could be valid for a certain class of WSN applications they are unlikely to be valid for all.

Furthermore, some standard assumptions that can be found in many security proposals are partially not realistic. Gamage et al. challenge in [81] especially the often made assumptions that pairwise keys between the nodes are always needed and that the nodes are consistently deployed randomly as not being realistic for a large class of applications. Silva et al. further present in [241] some applicability drawbacks of the widely accepted class of key management approaches that are based on the random pre-distribution of keys, such as [73]. They argue that good mathematical results can be achieved when the three parameters number of neighbours, number of total nodes and number of keys in each node's keyring, can be freely adjusted. However, these parameters are strictly confined in real-world WSNs. For example, the number of neighbours depends on the application scenario, e. g. the node's communication range, the network's topology, and density, while the maximum number of keys a node can store is limited by its available memory. Therefore, the good mathematical results cannot always be transferred to real world applications.

In essence, a serious gap exists in WSNs security research between theory and practice today. Despite the great advances in theory the transfer of these concepts to real systems is not well supported.

Another reason for approaching security from the application point of view arises from the fact, that several application design parameters have a direct impact on the required security mechanisms although they concern at heart non-security related aspects of the application. This becomes especially apparent in WSN key management. For example, the communication pattern between nodes as well as the deployment strategy both have a strong impact on the pre-deployment of key material, e. g. which nodes need to share a key and which nodes do not. Furthermore, taking dependability into account, different routing concepts have for example a strong impact on the availability and robustness of the network.

In summary, although invaluable contributions exist in WSN security research, most works do not provide any assistance and guidelines for the application developers with regard to choosing the appropriate security measures for their specific application, leaving them to adapt existing solutions or devise and

implement their own. However, as we already mentioned, this is complicated and error-prone.

We believe that the application developer must be provided with an easy to use process for including security into his application based on the specific requirements of the application. Especially, it must be possible to differentiate the security requirements and resulting measures for different data types in order to achieve optimal performance. Furthermore, he must be enabled to assess the chosen security solution in order to avoid a false sense of security and thus develop an insecure WSN application. Finally, in order to be widely used, these steps must be embedded into WSN development tools. The framework, which we will present later in this chapter, targets specifically these aspects.

3.4.2 Related Work

We review in the following related work focussing on the main requirements: application-specific security provision, the security analysis of feasible security solutions for an application, and the provision of security at the middleware level.

3.4.2.1 Application-specific Security

Although current research seconds our application-centric view of security provision for WSNs, e. g. see [225,252], we are not aware of any structured approach to facilitate the application developer in the process of generating application-specific security solutions.

Close to our view of differentiating the security requirements for different data types are Slijepcevic et al. in [246]. However, they target a predefined WSN application scenario, for which they statically specify three types of data that are communicated inside the network. They formulate different security requirements and measures for each of these three types, but do not present a concept that would allow for the application developer to freely specify additional data types.

3.4.2.2 Security Analysis

Generally, the question arises how to choose the proper security solution from all feasible solutions. In general, security, as such, is hard to quantify and visualize. Discussing and evaluating the quality of different security schemes requires significant domain expertise in the area of security. Although several attempts try to establish metrics that measure and describe the achieved level of security of a system, no all-encompassing framework exists, that is a able to cover all aspects and points of view. In particular, the metric must be appropriate for the evaluated technology and the results of the evaluation must be understandable for the target audience. Thus, while a false-positive ratio may

be a good measure to judge the quality of an intrusion detection component it is not suitable to express the quality of security measures in WSNs. Additionally, a representation of the security evaluation by a mere number, i. e. security solution 1 rates 0.8 whereas solution 2 rates 0.97, gives no hint how these two solutions differ, especially regarding residual risks, and which one might be more appropriate.

The upcoming ISO/IEC 27004 standard [115] as well as the recommendations in the NIST special publication 800-55 [53] try to measure the security of a system, however, mostly along a process-oriented approach, i. e. do backup strategies exist and how complete is an implementation. In this, they are suitable to measure the security level of an organization. However, with this broad perspective, measuring security in WSNs is not satisfyingly possible with these approaches, since important technical details are not covered.

A more technical approach to measure the efficiency of security measures is based on security patterns [236]. This model-based approach provides reusable solutions to specific security problems which application developers may use to build secure systems. Since the patterns are widely utilized and evaluated in real systems, the security evaluation of the resulting system becomes much easier [100]. Although several patterns might be helpful in establishing and evaluating security in WSNs they were not widely considered with this focus yet and consequently do not cover some WSN-typical aspects. Particularly, retaining security with the possibility of compromised nodes is not included [203].

Typically, due to the lack of formal approaches, authors of specific WSN security measures evaluate their approaches by means of attack possibilities with regard to a chosen attacker profile. However, attacker profiles are often not specified in great detail and evaluations differ greatly with respect to the considered attacks. Although attack taxonomies are published in the literature, e. g. [95, 222], they do not seem to be used widely. Although we use attack possibilities in our evaluation as well, we use on the one hand a structured approach that takes all possible attack paths into account. On the other hand, the application developer is able to influence the evaluation by specifying certain characteristics of the attacker.

3.4.2.3 Providing Security at the Middleware Level

To the best of our knowledge, the integration of security into WSN applications, let alone an automated security analysis, is not supported by existing WSN middlewares leaving the developers to devise, evaluate, and implement their own security solutions. This is also backed by a current WSN middleware survey [271]. Notable exceptions to this are the TinySec [128] and MiniSec [159] libraries for the widespread operating system for sensor nodes TinyOS [151]. They provide some security building blocks, e. g. to encrypt and authenticate data packets. However, in order to be used they assume that necessary founda-

tions like key setup are handled elsewhere. Furthermore, their usage is limited to systems running TinyOS.

We go a step further and focus on a more encompassing system. It automatically determines necessary security requirements for the target application and is extensible in order to support different hardware platforms as well as different programming languages.

3.4.3 Automated Comprehensible Security Provision

We now present our framework which facilitates the application developer in integrating security into his WSN application. As already mentioned, it comprises three steps. First, possible application-specific security solutions are derived by querying simple properties of the target application from the application developer. Second, each security solution is assessed with regard to what kind and the level of security it provides. The results of the security assessments are given to the application developer as feedback in form of residual risk tables, clearly identifying for each solution the aspects that are secure against attacks as well as those that are still vulnerable. Finally, the actual security measures are provided in form of a communication middleware.

To automatically derive feasible and fitting security mechanisms as well as assessing the solutions regarding their security provision, the application developer first has to specify several application-specific properties. These provide the foundations for our framework. On the one hand, the application developer assigns abstract security specifications for each data type that needs to be handled securely during runtime. On the other hand, he specifies several application aspects, namely node and network properties as well as the attacker profile. Table 3.3 provides an overview of the necessary specifications. While most of the specifications in the first three categories are used to derive appropriate security solutions, the hardware and lifetime specifications, as well as the specifications regarding the attacker profile are used during their evaluation.

3.4.3.1 Deriving Application-specific Security

The security requirements for each data type provide the basis for the derived security solutions. At this, the application developer specifies the desired abstract security service or services for each data type, e. g. authenticity and integrity for the data type `location data`. This defines how location data will be handled security-wise when communicated in the network during operation. Furthermore, the scope is required for each data type, i. e. which nodes must be able to participate in the secure communication of this data type.

On the one hand, these specifications clearly delineate the required security technologies. On the other hand, the scope specification gives valuable hints towards selecting an appropriate key setup. As we have discussed in Section 3.2.4 a plethora of different key management schemes are proposed for WSNs. How-

Parameter	Description
Abstract security specification	
Security service	Desired security service for a data type, e. g. confidentiality.
Scope	The scope specifies who participates in the secure communication of this data type. Examples are all nodes and each node and the base station.
Node properties	
Memory	The amount of memory available on the node for storing secret keys, e. g. 200 byte.
Hardware	Do the nodes have tamper-proof hardware?
Network properties	
Nodes	Total number of sensor nodes inside the WSN.
Basestations	Total number of base stations inside the WSN.
Lifetime	Envisioned network lifetime, e. g. 1 year
Attacker profile	
Presence	Specification of the attacker's scope, i. e. which extent of the WSN is he able to influence?
Intervention	Defines the attacker's abilities, e. g. eavesdropping.

Table 3.3: *Required application properties*

ever, the choice and implementation of a fitting proposal for the target application are anything but trivial. We try to alleviate this problem by providing several options for key pre-distribution, i. e. key material that is initially stored on the sensor nodes before they are deployed. The goal is to pre-deploy all keys necessary while avoiding wasting precious memory on keys the application will never use.

However, the appropriate key management scheme largely depends on several application characteristics. For illustration, Table 3.4 defines four generic key types and Table 3.5 explicates the confinements several application properties set on these key types for single hop and multi-hop communication.

Without going into detail on the individual confinements, the general limiting factor is the available memory on the nodes that is required to store the key material. Pairwise keys are generally only applicable if the number of keys that are required multiplied by the key size require less or equal memory space than being available.

However, using these simple equations to provide the key material would only allow for deriving very basic key setups, such as pairwise keys for small WSNs. Though, especially when the nodes are deployed manually to the operational area, highly specific key setups could be devised that take the resulting topology and structure of the WSN into account. Therefore, in order to keep the provision of key material simple while still allowing for great flexibility, we limit the automatic generation of key material to providing pairwise keys between all nodes, between each node and the base station (or base stations) or to providing a network-wide key. Additionally, we include the possibility for the application

Key Type	Description
Pairwise node-node key	Each pair of nodes shares one key in order to allow for end-to-end security in their communication.
Pairwise node-bs key	Each node shares a key with each base station in order to enable secure end-to-end communication between each pair.
Group key	A group key is used by each node inside a group to communicate its data securely to the other group members, allowing for a secured intra-group broadcast. A group key may secure the single hop communication, e. g. if the nodes establish sending keys to secure broadcasts to all neighbours, or provide end-to-end security between group members in multi-hop communication.
Network-wide key	A network-wide key is shared between all nodes in the WSN including the optional base stations. Thus, it secures the communication link, but does not provide end-to-end security.

Table 3.4: *Basic symmetric key types*

developer to specify a key topology that defines which nodes share a pairwise or a group key. Thus, prior to deployment he may completely specify the network topology from the point of view of security relations between the nodes and the nodes will automatically be provisioned with the required key material. Note that such a specification usually requires a manual deployment of the nodes. For example, specifying the node IDs that should share a common secret key with an aggregating node guarantees that these nodes, which could be deployed manually to a certain area, get provisioned with the secret key of the node that aggregates the data for that region in the WSN. Thus, during runtime, each node is able to communicate securely in an end-to-end fashion with the aggregating node. Furthermore, we include HARPS [208] as a possible key scheme, if the available memory on the nodes is not sufficient to support simple pairwise keys between all nodes but the application requires this key type. Generally, it is possible to include more key management schemes in our framework or to exchange for example HARPS with another scheme.

During the process of deriving possible security solutions, our framework evaluates for each option, if it provides a feasible solution. In general, this will be the case for more than one option. Thus, the framework specifies each possible solution and hands them to the next stage, the evaluation phase.

3.4.3.2 Evaluating WSN Security

Often several different security solutions can be synthesized that provide appropriate solutions, however, each with its specific strengths and weaknesses. Furthermore, the choice of a solution often has consequences for the application

	Pairwise keys		Group	Network-wide
	Node-node	**Node-bs**	**key**	**key**
Single Hop				
Communication scope				
All nodes	–	–	✓	✓
Node-to-node	$(\checkmark)^*$	–	✓	✓
Node-to-bs	–	✓	✓	✓
Deployment strategy				
Manually	$(\checkmark)^*$	✓	✓	✓
Random	$(\checkmark)^\dagger$	$(\checkmark)^\S$	–	✓
Node mobility	$(\checkmark)^\dagger$	$(\checkmark)^\S$	$(\checkmark)^o$	✓
In-network aggregation	✓	–	✓	✓
Additional nodes				
Manual	$(\checkmark)^+$	✓	✓	✓
Random	–	$(\checkmark)^\S$	–	✓
Multi-hop				
Communication scope				
All nodes	–	–	–	✓
Node-to-node	$(\checkmark)^\dagger$	–	$(\checkmark)^o$	✓
Node-to-bs	–	$(\checkmark)^\S$	$(\checkmark)^-$	✓
Node-to-group	$(\checkmark)^\dagger$	–	$(\checkmark)^o$	✓
Deployment strategy				
Manually	$(\checkmark)^\dagger$	$(\checkmark)^\S$	✓	✓
Random	$(\checkmark)^\dagger$	$(\checkmark)^\S$	–	✓
Node mobility	$(\checkmark)^\dagger$	$(\checkmark)^\S$	$(\checkmark)^o$	✓
In-network aggregation	$(\checkmark)^\dagger$	–	$(\checkmark)^o$	✓
Additional nodes				
Manual	$(\checkmark)^+$	$(\checkmark)^\S$	✓	✓
Random	–	$(\checkmark)^\S$	–	✓

$(\checkmark)^*$ applicable if *number of neighbours* · *keysize* ≤ *available memory*
$(\checkmark)^\dagger$ applicable if *total number of nodes* · *keysize* ≤ *available memory*
$(\checkmark)^\S$ applicable if *number of base stations* · *keysize* ≤ *available memory*
$(\checkmark)^o$ applicable if *number of groups* · *keysize* ≤ *available memory*
$(\checkmark)^-$ applicable if the base station knows the group key
$(\checkmark)^+$ applicable if dead nodes are cloned and re-deployed.

Table 3.5: *Applicability of key types for certain application properties.*

logic itself. For example, if the application developer chooses a solution that is based on one network-wide key, he will not be able to use end-to-end encryption between the base station and a particular node based on the selected scheme. Furthermore, it may be necessary to consider residual risks in the application logic, such as for example replayed packets, if the security solution does not provide any measures to prevent this.

Therefore, in order to support the application developer in his choice, it is important to present him a qualitative security evaluation for each possible

solution rather than merely presenting all possible solutions or choosing one in a black box fashion.

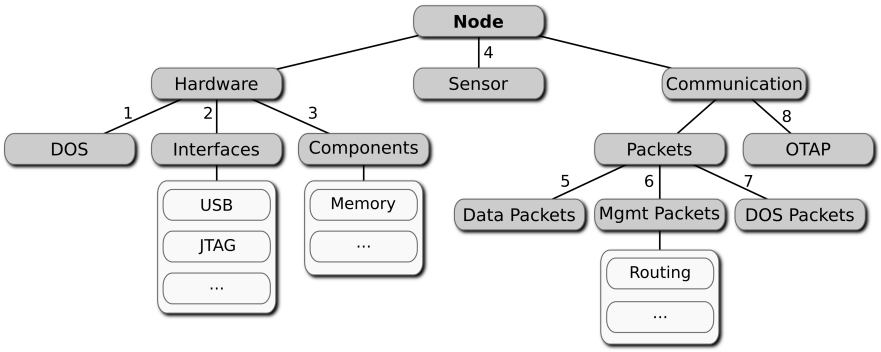
In the following we go into detail about how we evaluate the security solutions as well as the qualitative feedback we provide to the application developer in form of residual risk tables.

Security Evaluation For the security evaluation of the derived security solutions we use a formal and structured approach to model threats to the system, similar to Attack Trees presented by Schneier in [234]. First, all attacks on a system are arranged in one or several tree structures, with the targets modelled as roots and possible ways to attack these modelled as leaves. Then each attack path can be evaluated with respect to its likelihood, the necessary effort for the attacker, or other qualitative measures. Finally, specific techniques can be devised and implemented in order to prevent likely attack paths.

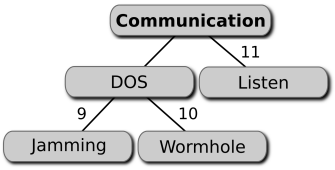
Generally, we classify attacks into insider and outsider attacks in this dissertation. While it is characteristic for insider attacks that the attacker controls a valid member of the network, outsider attacks are always possible, even without control of a valid member. In the evaluation of the security solutions we are mainly concerned with outsider attacks. This is a reasonable choice, as we evaluate the initial security solution for the WSN and there must not be any successful attacks during the pre-deployment phase. Otherwise the whole security solution falls apart.

Three important targets for outsider attacks exist in each WSN: the nodes, the communication, and the environment. Attacks on nodes target valid members of the WSN either by destroying, controlling, or influencing their data or behaviour. Attacks on the communication target the communication in parts or as a whole, and focus either on disturbing it or eavesdropping on it. Finally, attacks on the environment target the WSN in parts or as a whole by influencing its operation area, e. g. by triggering wrong sensor readings. A simple example of this is holding a burning lighter next to a temperature sensor or increasing the heating inside a building. On this basis we model three attack trees, which are illustrated in Figure 3.7. The numbers denote the possible attack paths.

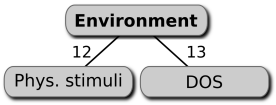
However, we do not include all possible attack paths in our evaluation since no proper defenses exist against some attacks. For example, consider DoS attacks. Some research exists that aims at lessening the effects of these attacks, e. g. see [284]. However, a complete defense against a sufficiently provisioned attacker is not feasible. Therefore, DoS attacks are for now excluded from the further analysis. Following the same reasoning we exclude all attacks on the environment from our analysis. Additionally, we do not include attack paths that can be avoided by best-practice programming guidelines, such as to disable the nodes' programming interfaces and over-the-air programming (OTAP) capabilities during operation.



(a) *Target: sensor node*



(b) *Target: communication*



(c) *Target: operational environment*

Figure 3.7: *WSN Attack Trees* (numbers denote the different attack paths)

Table 3.6 gives an overview of the different attack paths, including the fundamental attacking efforts and goals of the attack. The attack paths we consider in our further evaluation are highlighted, namely the paths 3, 5, 6, and 11. We now focus on real attack types along these paths and possible defences. Generally, we assume that the foremost goal of an attacker is to become an insider using the outsider attack paths discussed above. Once an insider, he may influence the WSN directly or stage more sophisticated attacks, such as Sybil attacks [184]. Regarding his capabilities and presence, we consider the specifications which the application developer may choose from the categories derived by Benenson et al. in [22] (c. f. Section 3.2.2).

Becoming an insider can be achieved in three ways. First, if the communication is not secure the attacker instantly learns the communicated data and influencing the nodes at will is trivial via message injection and modification. Furthermore, nodes which are controlled by the attacker may be added to the WSN. In case of a secured communication valid messages can be replayed to influence or at least to confuse nodes. Additionally, the attacker may try to break the employed cryptography, which can only succeed if the key size is too small. We assume the actual algorithm to be secure and implemented properly. Finally, he may attack the nodes physically in order to compromise them, thus again gaining access to the WSN. Ideally, he analyses the traffic beforehand in order to isolate important nodes in the network, such as data sinks or nodes close by. Table 3.7 gives an overview of the considered attacks and defences.

Path	Effort	Attack
Attacks on Nodes		
1	Find & access node	Destroy the node's hardware
2	Find & access node	Compromise the node by accessing it through live interfaces
3	Find & access node	Compromise the node by accessing its hardware directly
4	Find & access node	Influence the node's sensor readings
5	Addressable node	Influence the node's information
6	Addressable node	Influence the node's behaviour
7	Addressable node	Sleep deprivation torture DoS with malformed packets
8	Addressable node	Reprogram node via OTAP
Attacks on the Communication		
9	Attacker model	Jam communication medium
10	Attacker model	Redirect traffic via wormhole
11	Attacker model	Eavesdrop on the communication
Attacks on the Environment		
12	Attacker model	Influence sensor readings
13	Attacker model	Destroy operation area

Table 3.6: *WSN attack paths*

We use this evaluation logic to automatically assess the derived security solutions with regard to their quality. This is done first by matching the security annotations specified by the application developer in addition to the derived security specifications with the attack possibilities to find paths in the attack tree that are secure. In a second step we evaluate the efforts and risks of the insecure paths to determine the possible effects of the attacks. For this purpose we assume that communicating wireless with a node requires significantly less effort than finding and accessing it. Furthermore, Becher et al. [21] describe that compromising a node usually requires expert knowledge as well as costly equipment. Furthermore, often the nodes need to be removed from the network for a significant time-span. Therefore, we denote the effort to compromise a node as being high. The risks are mainly affected by the chosen key setup as well as the attacker's presence and capabilities.

Residual Risk Tables Finally, in order to facilitate the application developer in choosing a solution, we present these quality assessments in form of residual risk tables. These tables display for each possible security solution the remaining risks, after the specified countermeasures have been applied. In detail, we present attack paths of high and medium risk, as well as paths that are secure. Furthermore, an indication is given of how much effort a particular attack requires. For example, a security solution that utilizes pairwise keys is much less vulnerable to node compromise compared to a solution based on a network-

Attack	Target	Defence
Message insertion	Nodes	Authenticity
Message modification	Nodes	Integrity
Message replay	Nodes	Message uniqueness
Physical attacks	Nodes	Tamper-proof hardware
Eavesdropping	Communication	Confidentiality
Traffic analysis	Communication	Confidentiality (whole packet) or measures from [63]
Cryptanalysis	Cryptography	Sufficient key size

Table 3.7: *Considered external attacks and defences*

wide key, since the attacker gains significantly less influence by compromising one node. In addition to the security assessment further information about the security solution may be presented to the application developer, such as that in general a solution is possible but inefficient. An example situation in which this is helpful is presented in the use-case.

An example of a residual risk table for the use-case we describe in Section 3.4.5 is given in Table 3.8. At this, for example, no confidentiality was annotated for the data type and thus the risk of an eavesdropping attack is high. However, keep in mind, that this is not a problem for an application, in which the respective data type does not comprise sensitive information. Thus, the final interpretation of the residual risk tables must be done by the application developer.

Based on the security evaluations for each feasible security solution, the application developer is now able to either choose one proposed solution or discard the choices and go back to modify his annotations, e. g. to include further security services.

3.4.3.3 Middleware Integration

Up to this point our framework represents a stand-alone approach for devising and evaluating application-specific security solutions. However, the realization of the security measures still requires expert knowledge in the field of security. Therefore, it is beneficial to include the whole process into a WSN development tool to provide the necessary implementations of the security measures.

Generally speaking, our scheme can be integrated with systems synthesizing middleware systems or virtual machines as well as complete applications, such as RUNES [56] or ATG [13]. As a first step, we have implemented support for our middleware synthesis framework Fabric [200] that supports WSN application development by generating custom-tailored middleware instances for different target platforms. We provide more details about the actual architecture and implementation in the next section.

Attack	Effort	Risk
High Risks		
Eavesdropping	small	The content of your data and routing messages is readable.
Message Replay	small	Valid messages can be replayed.
Compromise node	high	1 node required to access to all routing messages.
Medium Risks		
Traffic Analysis	small	An attacker can analyse the traffic, though only in his local coverage area.
Not at Risk		
Message correctness	small	Messages are authentic and can not be modified undetected.
Cryptography	medium	Average time to break a 128 bit key: >58 thousand years [10^7 keys/sec].
General Comments		
–		

Table 3.8: *Example residual risk table*

3.4.4 Implementation

We have implemented a prototype of our approach for the middleware synthesis framework Fabric. In particular, we have devised and implemented three aspects. First, we have integrated the security annotation of the data types and the visualization of the residual risk tables into Fabriclipse. Fabriclipse is an Eclipse plugin that is being developed at the moment to facilitate the data annotation process for Fabric. Second, we have implemented the security logic, which determines feasible security solutions and assesses their security. It is integrated into Fabriclipse as well. Finally, we have implemented a security module allowing Fabric to generate the required security-related source code. These parts allow for specifying and generating the security-enabled middleware which will then be used by the application developer to devise his application.

3.4.4.1 Fabriclipse Integration and Workflow

Figure 3.8 illustrates the architecture of our implementation. The application developer interacts only with Fabriclipse, using the GUI for annotating the data and choosing the appropriate security solution, based on the displayed qualitative feedback, i. e. the residual risk tables. The security logic, including the invocations of Fabric during the validation of potential security solutions as well as the security assessments are transparent to him. Rather it appears to him that the system changes from the annotation step directly to the residual risk tables. In the following we briefly address each step in the order of the actual workflow.

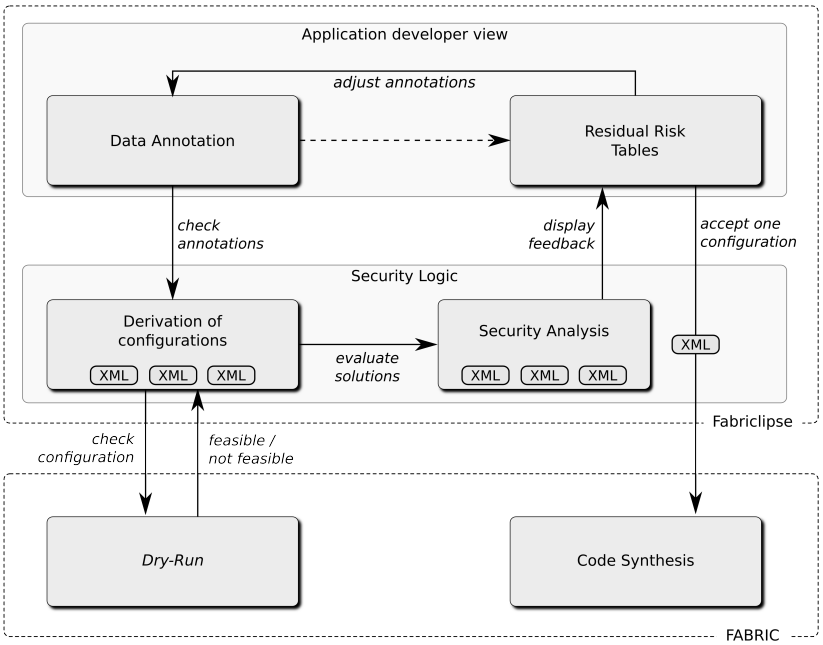
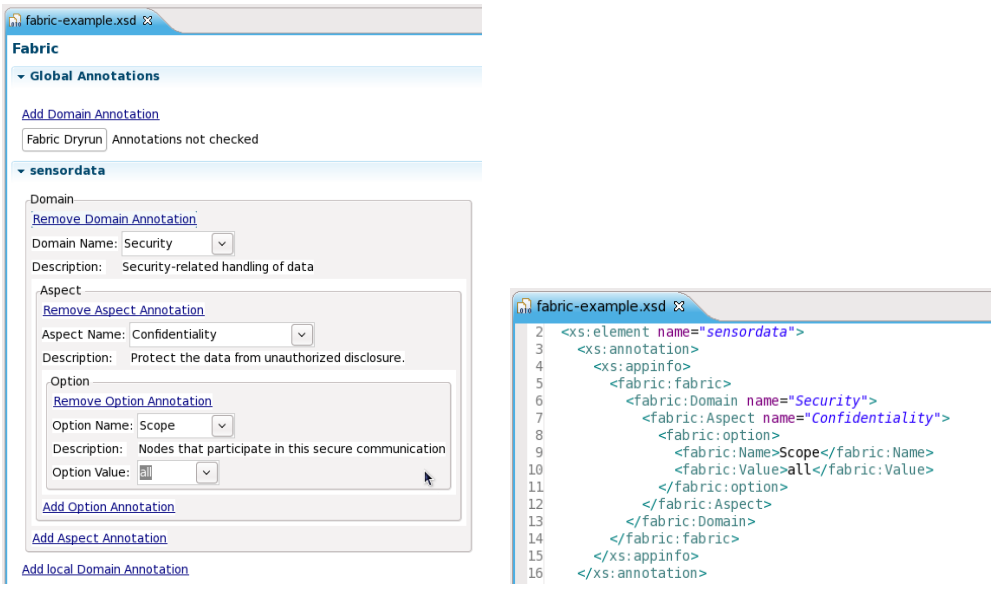


Figure 3.8: *Implementation architecture*

Data Annotation Generally, we follow FABRIC's annotation nomenclature. Therefore, all security-related annotations are located in the domain security. The application properties, such as the security services, constitute aspects, which in turn have options. For example, an annotation for the data type `sensordata` may comprise the aspect `confidentiality` and the option `scope` with the value `all`. This annotation translates into the requirement that the data type `sensordata` should be sent encrypted using a key that is known to or which can be derived by all nodes inside the WSN. The annotation process, using this example, is depicted in Figure 3.9. Fabriclipse allows for both, GUI-assisted annotation in which the possible values are provided in drop-down menus (c. f. Figure 3.9a), as well as the possibility to edit the XML file directly (c. f. Figure 3.9b).

Derivation of Configurations and Fabric Dry-run Based on the specified annotations, the security logic derives possible security solutions. First, it determines the necessary key schemes that are specified by the annotations as well as possible alternative key schemes. Returning to the example above, the `all` annotation in the option `scope` requires a key scheme in which all nodes know the encryption key or are able to derive it during runtime. This requirement could be fulfilled for example with either a network-wide key or by using HARPS. Following this, the security logic determines if the required cryptographic measures are implemented in FABRIC's security module. In addition to providing the necessary functionality, these have to provide the implementation in the



(a) *GUI-supported editing* **(b)** *Direct editing of the XML file*

Figure 3.9: *Security annotations in Fabriclipse*

required programming language as well as being applicable to the destination hardware. The security logic validates for each solution if FABRIC is able to generate the required source code. For this purpose, we have extended FABRIC slightly to support the dry-run functionality. In essence, FABRIC operates unaltered in this mode, except that no source code is generated.

Security Analysis and Residual Risk Tables The resulting feasible security solutions are then subject to the security evaluation. By means of the attack trees (c. f. Section 3.4.3.2) unprotected attack paths are determind. The results are given as feedback to the application developer in form of the residual risk tables. Figure 3.10 visualizes an example in Fabriclipse. Now, the application developer may either accept one security solution, in which case FABRIC is invoked with the respective configuration, or he may discard all proposed solutions and go back to adjust his annotations.

3.4.4.2 Fabric Security Module

In order to enable FABRIC to generate the security-related source code, we have implemented a security module that provides various functionalities. Generally, all functionality is available for the programming languages Java and C. Regarding sensor node hardware we support at the moment only the iSense platform.

First, the security module is able to generate different key schemes. The prototype supports pairwise keys, network-wide keys, and the key management

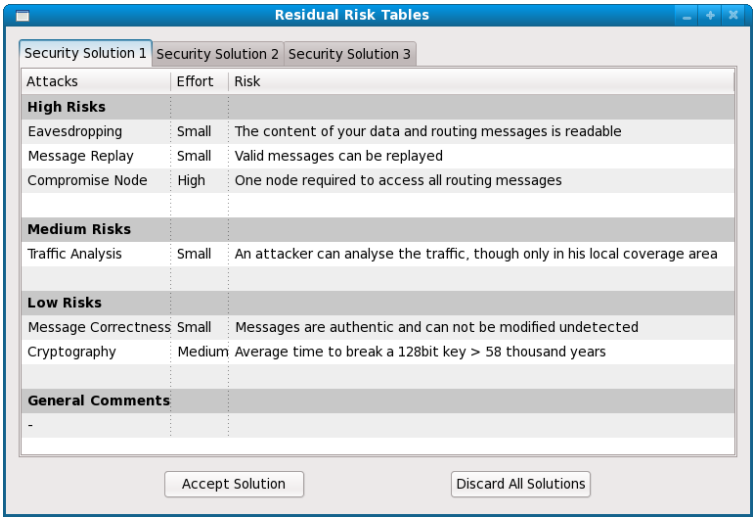


Figure 3.10: *Residual risk tables in Fabriclipse*

scheme HARPS. Furthermore, it supports customizable source code packages. This is necessary, for example, if pairwise keys are specified between certain node IDs. In this case the resulting source code can be customized during the compilation process. Figure 3.11 illustrates this by giving an excerpt of an example source code. Here, the source code contains symmetric keys for each node such that it shares a unique key with each base station. In this case this results in ten keys per node. Since the key material is different for each node participating in this application, the source code has to be compiled for each node ID individually in order to include only the appropriate keys for that single ID into the binary.

Furthermore, we have implemented the encryption algorithm AES, which may operate in the three modes CTR to support confidentiality, Cipher Block Chaining Message Authentication Code (CBC-MAC) for authenticity and integrity, and Counter with CBC-MAC (CCM) [281] which combines all three security services. AES was mainly chosen because the iSense sensor nodes are ZigBee-compliant and thus comprise a hardware AES module. For the Java programming language, we implemented these concepts in software.

The security module is completely self-contained, i. e. no specific order is necessary regarding the previous or following modules which may be generated by Fabric due to respective annotations. Figure 3.12 exemplifies the general API Fabric generates. Generally, a custom send method is provided for each data type, e. g. `sendSensordata(data)`. Internally the send method contains all the annotated functionalities, such as the serialization and encrypt functionality. The amount of utilized modules as well as their order is irrelevant from the point of view of the security module since each module uses a standardized format for input and output, i. e. two buffers, one for input and one for output, as well as the size of the data inside the in-buffer and out-buffer.

```
1  #define FABRIC_NODE_TO_BS_KEY_COUNT 10
2  #define FABRIC_NODE_TO_BS_KEY_BYTES 16
3
4  #if NODE_ID == 0
5   #define FABRIC_NODE_TO_BS_0_KEY {0x75, 0x70, 0x4a, 0x25, 0x79,\
6   0x3a, 0x3c, 0x34, 0x5d, 0x66, 0x7d, 0x6b, 0x57, 0x65, 0xa4, 0x44}
7    ...
8   #define FABRIC_NODE_TO_BS_9_KEY {0x41, 0x27, 0x67, 0x48, 0x30,\
9   0x71, 0x62, 0x59, 0x63, 0x7b, 0x53, 0x3c, 0x4c, 0x6a, 0x47, 0x58}
10
11 #elif NODE_ID == 1
12  #define FABRIC_NODE_TO_BS_0_KEY {0x79, 0x6a, 0x47, 0x58, 0x33,\
13  0x34, 0x75, 0x5d, 0x77, 0x6b, 0x2e, 0x3c, 0x7e, 0x79, 0x4d, 0x4d}
14   ...
15  #define FABRIC_NODE_TO_BS_9_KEY { 0x42, 0x2f, 0x72, 0x5b, 0x74,\
16  0x2c, 0x39, 0x6e, 0x2d, 0x2f, 0x4d, 0x27, 0x59, 0x4f, 0x2e, 0x55}
17 ...
18 #endif
19
20 unsigned char FABRIC_NODE_TO_BS_KEYS[FABRIC_NODE_TO_BS_KEY_COUNT]
21           [FABRIC_NODE_TO_BS_KEY_BYTES]
22   = { FABRIC_NODE_TO_BS_0_KEY, ... ,  FABRIC_NODE_TO_BS_9_KEY };
```

Figure 3.11: *Customizable source code*

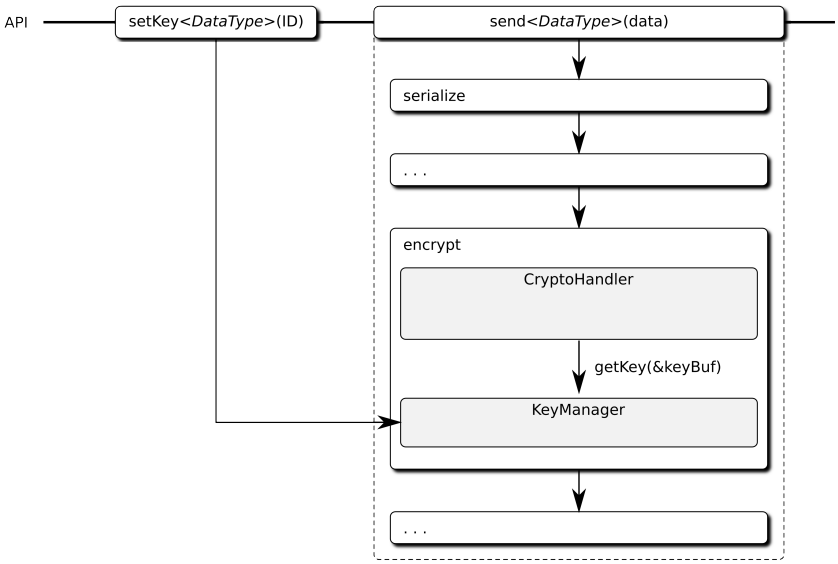


Figure 3.12: *Security measures in the generated API*

Regarding the security module, the application developer first has to specify the key for the next call of the `send<DataType>(data)` method. For this purpose the API contains the function `setKey<DataType>(ID)` which is also customized for each data type, e. g. `setKeySensordata(ID)`. Calling this method with the ID of the destination node sets the corresponding key for the next `sendSensorData(data)` call.

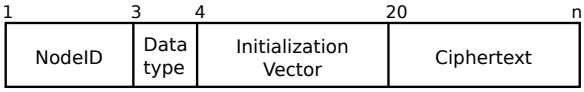


Figure 3.13: *Ciphertext format*

Setup	Routing Data	Sensor Data
1	128 bit network-wide key	128 bit pairwise keys
2	HARPS 128 bit pairwise keys	

Table 3.9: *Use-case: two feasible key schemes*

In order to allow for an automatic decryption process after an encrypted message has been received, the destination node ID as well as the encrypted data type are included in the message in front of the ciphertext. Furthermore, the message comprises the initialization vector which is required for the decryption. Figure 3.13 visualizes the ciphertext format. The node ID requires 2 bytes, the data type 1 byte, and the initialization vector 16 byte. After that the ciphertext is appended. It comprises all necessary parts, e. g. the encrypted data or the concatenation of the data and corresponding CBC-MAC.

3.4.5 Use-Case

To further illustrate the framework described above and to evaluate its usefulness, we now look at the whole process by means of a use-case. We consider a structural monitoring application, similar to the one investigated in [252]. In this scenario we have two data types: routing data and sensor data. Both must be transmitted with authenticity and integrity guarantees, while confidentiality is not required. Furthermore, routing data must be accessible to all nodes, while sensor data is only transmitted in an end-to-end fashion from each node to the base station.

Here, we assume 200 employed nodes which are not tamper-proof but also not easily accessible, e. g. embedded in the walls. Furthermore, we envision a long network lifetime and assume that an attacker is able to mount communication as well as physical attacks, though only locally. For the sake of brevity, we limit ourselves to these characteristics in this example.

Based on these specifications our framework generates and evaluates possible security solutions that are able to satisfy the given characteristics. In the following we discuss two of these choices.

Both solutions provide the required services based on the keys given in table Table 3.9. While solution 1 provides a network-wide key to secure routing data and a pairwise key between each node and the base station to secure the sensor data, solution 2 utilizes HARPS pairwise keys for both data types.

Attack	Effort	Risk
High Risks		
Eavesdropping	small	The content of your sensordata and routing messages is readable.
Message Replay	small	Valid messages can be replayed.
Compromise node	high	1 node required to gain access to all routing messages. An attacker gains full control over the sensor data messages from a compromised node. Other sensor data messages remain secure.
Medium Risks		
Traffic Analysis	small	An attacker can analyse the traffic, though only in his local coverage area.
Not at Risk		
Message correctness	small	Messages are authentic and can not be modified undetected.
Cryptography	medium	Average time to break a 128 bit key: >58 thousand years [10^7 keys/sec].
General Comments		
–		

(a) *Security solution 1*

Attack	Effort	Risk
High Risks		
Eavesdropping	small	The content of your data and routing messages is readable.
Message Replay	small	Valid messages can be replayed
Medium Risks		
Traffic Analysis	small	An attacker can analyse the traffic, though only in his local coverage area.
Compromise node	high	Several nodes required to access all routing and sensordata messages.
Not at Risk		
Message correctness	small	Messages are authentic and can not be modified undetected.
Cryptography	medium	Average time to break a 128 bit key: >58 thousand years [10^7 keys/sec].
General Comments		
Routing messages: a new MAC will be created for each communication hop using the pairwise key of the communication neighbours. This may result in high computing costs.		

(b) *Security solution 2*

Table 3.10: *Residual risk tables*

The residual risk tables for both solutions are given in Figure 3.10. They illustrate respectively the risks associated with each option and give additional information for the solution, if necessary. The application developer can accept one of the options if all the attacks he thinks likely are handled or he can discard both and specify further security properties in the security annotations.

In this use-case it is likely that the application developer selects the first security solution, although the risk potential regarding the compromised nodes is higher. However, the WSN is supposed to operate for a long time and therefore the solution should be as energy-efficient as possible. Furthermore, the nodes are not easily accessible and the overall likelyhood of an attack in this scenario is relatively low. However, keep in mind that this is our personal view and other application developers might choose differently.

3.4.6 Evaluation

We achieve several important goals with our approach. Most importantly, we support application developers in devising sound security solutions for their applications without requiring them to become security experts. Since they are able to specify the security requirements in an abstract form and receive a detailed security evaluation of possible solutions, they may make an informed decision about which security solution is best for their target application. Furthermore, they are made aware of possible open issues in the security solution, for example reply attacks may be possible despite the employed security measures. Having this knowledge the application developer is able to handle those issues in the application logic itself, thus increasing the robustness of the resulting application.

Therefore, on the one hand our approach alleviates the integration of security into WSN applications, while on the other hand it allows for the overall quality of the employed security in WSN applications to increase. The latter aspect results from the additional knowledge the application developer is been given, as well as the strict division of labor. While security experts implement the individual algorithms and provide the logic of how to use them, the actual users are guided to feasible solutions and may use them in a straight forward way. Eventually, they benefit from the sound implementations done by experts.

Especially with regard to the device heterogeneity in UC environments, the possibility of programming large numbers of heterogeneous devices in one process is advantageous. The different devices will comprise the same security measures, e. g. for encryption, optimally adjusted to the specific hardware and implemented in the required programming language. Thus, the application developer does not need to worry about the interoperability of the different devices.

Regarding useability, especially the integration of our approach with Fabric results in a very easy to use system. Deriving appropriate security solutions requires only the specification of the corresponding abstract annotations. Fur-

thermore, changing a security solution may be easily achieved by changing the annotations. The integration into Fabriclipse further increases useability since the application developer is provided with a GUI and may stay in his accustomed programming environment, i. e. Eclipse.

Another advantage of our approach is its modularity. Experts may implement new security measures and integrate them into the framework or they may fix erroneous implementations without having to change the general logic. Particularly application developers who are security experts, may integrate custom security concepts and measures to speed up the integration of security into their applications. However, it would also be possible to devise and implement custom security measures inside the application and only annotate desired aspects that complete the custom solution. Likewise it is possible to completely pass on the automated security generation and security assessment by not annotating any security-related aspects at all.

One aspect to keep in mind is that our approach is only able to provide the quality and extensiveness of the underlying expert implementations. This holds true for the expressiveness of the security evaluation as well. Thus, changes or the integration of new concepts into the framework should be done with great care.

Evaluating our approach from a technical point of view is almost impossible, since different implementations of the security measures generally result in different code sizes and runtimes. Furthermore, several compiler settings have a great influence on both aspects as well. Nevertheless, we have compared parts of the generated source code from Fabric with independent manual implementations of equal measures. Especially the FlegSense project [112] has implemented some aspects of our prototype, such as the key management scheme HARPS on the iSense sensor nodes as well. Both implementations, though independently devised, resulted in roughly the same code size.

3.4.7 Summary and Outlook

We have presented a framework that facilitates the application developer in integrating security concepts and measures into his application. It is based on the premise that the application properties define most of the significant security aspects. In essence, it provides comprehensible security and thus helps the application developer in choosing the optimal security solution for his application. Furthermore, the integration into Fabric allows for the automatic provision of the required source code.

In the future, our approach would benefit from additional aspects. In particular additional information regarding the resulting code size as well as approximate runtimes of the security solutions would enable application developers to determine the advantages and disadvantages of the security solutions even better and chose the most sufficient solution. Furthermore, we would like to extend the scope of the security evaluation to include DoS and internal attacks as well.

Other possible extensions may comprise dependability aspects. For example, the routing strategy has a great influence on the robustness of the network. Thus, it may be beneficial for data types that must be received at the destination to use flooding instead of the normal routing scheme. We think it is worthwhile to determine which other factors influence the dependability as well and make them available for annotation in the FABRIC-domain dependability.

3.5 Conclusion

We have shown that the integration of meaningful security into WSNs needs to be considered with regard to the respective target application. While it is possible to design and implement a specific security solution for a strictly defined target application, as we have shown in our first approach, this methodology results in a significant amount of work for each additional target application scenario and always requires expert knowledge. Therefore, we have devised a more generic approach that allows for application developers to integrate sound security solutions into their applications without limiting the scope of possible application scenarios nor the requirement for the developer to become a security expert. We have shown that this approach yields many advantages.

Although we have focused on WSNs in this research, the same reasoning holds true for UC environments, where a vast quantity of possible application scenarios exists. Furthermore, especially regarding the device heterogeneity in UC environments the establishment of security aspects greatly benefits from a middleware-based approach. We consider in particular the integration of our approach and FABRIC to be of great use in integrating into UC environments as well.

4 Autonomic Dependability in Network Routing

The next thing in technology
is not just big but truly huge:
the conquest of complexity
[Andreas Kluth]

Networking capability is one of the key building blocks to fully establish UC environments, as we have discussed earlier. Hereby, several levels of networking can be differentiated: local networks which the devices form by interconnecting, the connection of these to WANs, and the interconnections of the WANs into a globally spanning network. Several networking technologies exist, that specially take the UC devices' unique features and constraints into consideration. However, the existing WANs operate with long established technologies and thus may become the critical component when UC environments increasingly emerge. Especially the increasing complexity, e. g. in traffic volume and service demands, or drastically increasing connection links may lead to massive problems with regard to the manageability and efficiency of these networks. Therefore, new concepts and strategies must be evaluated to enable the existing WANs to cope with emerging UC environments.

Industry as well as researchers have started to address these manageability problems amongst others under the names of *autonomic computing* or *organic computing*. In essence, both envision systems that adapt autonomously to their environments and are thus able to optimize their operations. In this context, nature-inspired strategies, i. e. strategies that seek inspiration from nature to solve technological problems, have become one focus of research. This follows the reasoning that despite nature being an inherently complex system, it has without doubt been doing an extraordinary job maintaining a diversity of life for millions of years. In this context several successful strategies evolved to solve complex problems, e. g. the foraging of animals. Expectations are high, that some of these strategies may prove beneficial to ensure operation in the face of growing complexity in computer science as well.

With regard to communication networks, the employed routing mechanism determines in large parts the dependability of the network. Furthermore, it presents the central point where human administrators may influence and control the networks' operations. Therefore, the question arises, whether nature-inspired strategies may be incorporated here to increase the dependability of networks in an autonomic way. Thus, relieving the human administrators from the burdensome task of managing the increasingly complex network in detail.

Several nature-inspired routing protocols have been proposed in the literature, see for example [276] for a good overview. However, the respective evaluations are mostly focused on performance-related questions. Furthermore they are not fully comparable due to different evaluation settings, provide partially questionable results, or the results are irreproducible since important details about the evaluation are not given by the authors. Therefore, the above question cannot be answered with certainty based on these evaluations.

To remedy this problem, we review the published evaluations, select the most promising nature-inspired routing approach, and analyse it further in comparison to the state-of-the-art routing protocol.

The remainder of this chapter is structured as follows. In the next section we first aquaint the reader with the necessary foundations for our analysis. We then survey the published evaluations of nature-inspired routing approaches, point out problematic areas, and determine the most promising approach. In the following we analyse this approach in comparison to a state-of-the-art routing protocol with regard to dependability. Finally, we try to provide an answer to the question about the role nature-inspired methods can play in network routing and point out future work.

4.1 Fundamentals

We now provide the necessary foundations for the further analysis in this chapter. Specifically, we give an overview of the autonomic computing initiative as well as the research field of organic computing in the next section. Furthermore, we briefly review the problem of routing in communication networks, give an overview of state-of-the-art routing algorithms and explain which categories of nature-inspired approaches present alternative approaches in Section 4.1.2. Finally, we describe OSPF, the state-of-the-art technique for intra-domain routing today in Section 4.1.3, as well as the nature-inspired approach to routing, BeeHive in Section 4.1.4.

4.1.1 Autonomic and Organic Computing

IBM started the Autonomic Computing (AC) initiative as a response to the growing complexity in the IT industry [111, 135]. The term is derived from human biology and describes at heart a vision where autonomic systems manage themselves according to high-level goals given by administrators. On this account, IBM anticipates that systems incorporate four fundamental "self"-properties:

- *Self-configuration:* systems adapt automatically to dynamically changing environments, following high-level policies.
- *Self-optimization:* systems monitor and tune resources automatically to improve their own performance and efficiency.

- *Self-healing:* systems automatically detect, diagnose, and repair localized software and hardware problems.
- *Self-protection:* systems anticipate, detect, identify, and protect themselves from malicious attacks.

With the focus on business applications, IBM defines a roadmap of five consecutive steps for realising self-managing systems [84]: step one describes the current status of IT technology, where human administrators manage the systems manually. Steps two through four successively integrate more sophistication, until fully self-managing systems are available at step five.

Organic Computing (OC) was first introduced in [269] as the research counterpart to IBM's business initiative. While both agree on the principle that self-organization will characterize future IT systems, the scope of OC is considerably wider than the scope of AC.

The central focus of OC is the analysis of information processing in biological systems and its transfer into IT systems. According to its vision, the newly designed systems will behave life-like by adapting autonomously to changes in the environment. For this purpose, the systems have self-x properties, similar to those described by the AC initiative. However, man-machine interaction, robustness and trust in the reliability and the protection of privacy are additional focal points in the OC initiative [231].

With regard to this thesis, we do not differentiate further between AC and OC. However, we utilize several of the self-x properties in our evaluation of the routing approaches' autonomic behaviour Section 4.3.

4.1.2 Network Routing

Routing is one of the core topics in networking and has been studied by researchers since the dawn of packet switched networks, e. g. see [169]. It represents a key feature in networks, focusing on the process of moving a data packet from a source to a destination node, mostly via several intermediary nodes. Generally, this presents a complex and difficult optimisation problem, first, in finding possible paths from source to destination and second in finding the best next path segment to forward the data packet. Routing is therefore of special significance in networks, since it determines in large parts its performance and robustness.

4.1.2.1 State-of-the-Art

Numerous approaches exist, that try to optimise the choice of routing paths. In IP-based WANs, we generally differentiate between intra-domain and inter-domain routing. Figure 4.1 respresents an example section of a global-spanning WAN. It displays three network domains, each having an internal routing backbone and being interconnected by gateway routers. We refer to the routing

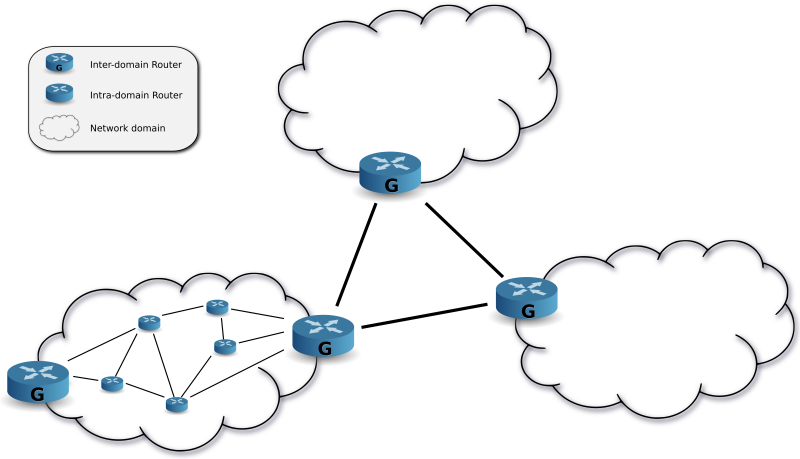


Figure 4.1: *Sample domain structure of a wide area network*

inside a domain as intra-domain routing, while the routing between domains is referred to as inter-domain routing.

Routing domains usually represent network segments that are administered by a single authority[1]. This authority has complete control of the network and may decide autonomously all aspects of the domain, e. g. its physical structure or which routing algorithm is deployed. Despite the autonomy, OSPF constitutes nowadays state-of-the-art routing protocol for intra-domain traffic. While the optimisation criteria for intra-domain routing is mostly the network's performance, the optimisation criteria for inter-domain routing is largely defined by costs and politics. Therefore, a different routing protocol is usually employed for inter-domain routing. Here, BGP constitutes nowadays the de facto standard.

Open Shortest Path First (OSPF) is specified in its current version, OSPFv2, in RFC 2328 [176]. OSPF is a link-state routing algorithm, i. e. it broadcasts the cost for each link and calculates the shortest paths to each destination on the basis of all link costs . We give a more detailed description of OSPF in Section 4.1.3.

The Border Gateway Protocol (BGP), in the latest version 4, is specified in RFC 4271 [214]. From BGP's point of view, a WAN only consists of several interconnected domains, each with a unique number, the network prefix. BGP is a path vector protocol, i. e. instead of periodically transmitting the costs for the links, it always sends the whole path to a destination. The definition of the rules, according to which the optimal path is chosen, is outside the scope of BGP. Therefore, rules may contain arbitrary aspects, such as monetary or

[1] We abstain in this thesis from using the name "Autonomous Systems" as these domains are usually called, in order to avoid confusion with the autonomous systems that are envisioned by the AC and OC initiatives.

political aspects. For example, the rule "traffic from this domain should never pass through domain XYZ" could be incorporated.

4.1.2.2 Nature-inspired Routing

Several promising nature-inspired approaches to routing have been proposed by researchers. A good overview is presented by Wedde and Farooq in [276]. In the context of packet switched networks, we include the following two categories of nature-inspired routing approaches into our analysis: algorithms that are based on *swarm intelligence* as well as algorithms that are based on *evolutionary algorithms*.

Swarm Intelligence Routing algorithms based on swarm intelligence mimic the behaviour patterns of animal swarms in nature, e. g. ants or bees. A basic assumption in this context is that although the individuals possess only limited capabilities, their collective behaviour leads to an intelligence each individual alone would not be capable of. However, the individuals benefit from this, for example for foraging.

Definition 15 (Swarm, Swarm System) A swarm is defined as a group of loosely structured, interacting agents [132]. A swarm system is a group of individuals (agents) that are able to interact and communicate among them and with the environment. At this, they may change the environment or the swarm.

Regarding routing in networks, swarm-based approaches are realized as agent systems. Each agent represents an individual and is implemented as a special kind of data packet that travels through the network. On its way it assesses the links' quality parameters and updates the routing data structures on the nodes accordingly. Well-known swarm-based approaches to routing are AntNet and BeeHive.

AntNet [43] is a routing approach based on ant colony optimization (ACO). ACO describes a discrete optimization technique that is oriented on the foraging behaviour of ants. During foraging, ants communicate indirectly via chemical substances, so-called pheromones, which are laid continuously on the paths the ants travel. The amount of pheromones depends hereby on the quality of the food source. After some time the pheromones on the paths dissolve. Although ants choose their path in a probabilistic fashion, they generally tend to follow paths with high pheromone concentrations. In [86] an experiment was conducted that proved the potential of ACO to solve the shortest path problem.

AntNet uses this technique for routing purposes. It's goal is to increase the overall network performance by load-balancing the traffic in a probabilistic fashion over all available links. In order to determine the link characteristics, AntNet employs two kinds of agents, forward ants and backward ants. They commu-

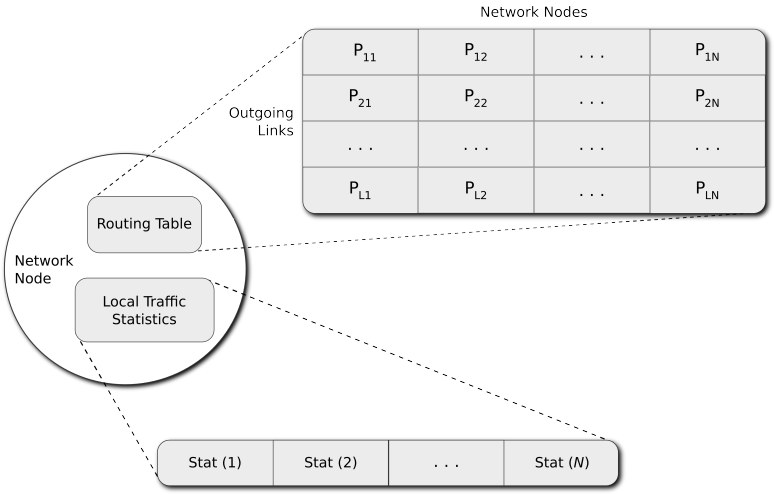


Figure 4.2: *Data structures in AntNet*

nicate indirectly via two data structures that are located at each node in the network, see Figure 4.2.

- *Routing table*: for each destination d and neighbour k each node stores in his routing table the probability $P(k, d)$ which reflects the goodness of using node k as the next hop towards the destination d. The table has the size $L \times N$, whereby L denotes the amount of neighbours and N denotes the number of nodes inside the network.

- *Local traffic statistics*: an array $\mathcal{M}_s = (\mu_{sd}, \sigma^2_{sd}, \mathcal{W}_{sd})$ defines the statistical model for the distribution of traffic from the individual node s to all destination nodes d. At this, μ_{sd} reflects the average and σ^2_{sd} the variance of the required travelling time of a data packet from node s to d. Additionally, the shortest travelling time is stored in a time window $\mathcal{W}_{sd}$, which serves as a short-term memory. Based on these values the travel time as well as the stability of the route can be estimated.

In regular intervals each node s sends out forwards ants $F_{s \to d}$ to different destinations d. The choice of the destination node is probabilistic and based on the distribution of the traffic.

Forward ants use the same queues as normal data packets to measure the delay. On their path to the destination they store each intermediary node and the state of the links in a stack structure. Generally, forward ants choose one of the nodes they have not yet visited as the next hop. If all hops have already been visited the forward ant chooses one randomly and deletes the resulting circle from the memory stack. Upon reaching its destination, a backward ant $B_{d \to s}$ is generated that receives all travel information from the forward ant and starts its way back towards the originating node. The forward ant is eliminated since it is no longer needed. Backward ants use higher priority sending queues to speed up the dissemination of the gathered information in the network.

The backward ant travels the path of the forward ant in reversed order and updates the data structures regarding the entries that concern the destination node d on each node.

Several enhanced versions of AntNet exist. Foremost, the authors of AntNet propose an enhanced version called AntNet-CO [44]. In this version both, forward and backward ants use higher prioritized queues in order to propagate the routing information faster. The queueing delay is estimated.

Furthermore, Barán and Sosa propose several enhancements for AntNet [15,16]. We denote their version as AntNet* in the following. First, they enhance the initialization of the routing tables by including knowledge about the network topology. Essentially, the probability values for the entries that have a direct neighbour as destination and next hop, i. e. P_{dd} for $d \in N$, are increased, while the probabilities for all other entries, i. e. P_{di} for $u \notin N$, are initialized equally, the standard approach in AntNet. Furthermore, they increase the performance during network failures. While the next hop is chosen at random during failure situations in AntNet, AntNet* recalculates the probabilities based on the already learned values. The same improvement is applied to the situation where links are re-established. With regard the next hop to which ants travel, AntNet* decides randomly if it is chosen probabilistically or deterministically, in which case the best routing entry towards the destination is chosen. Finally, AntNet* limits the amount of ants to four times the size of the network nodes and eliminates backward ants that can not reach their destination due to network failures.

AntNet-local has been proposed by Liang et al. in [152]. AntNet requires global information on all nodes, such as the total number of nodes in order to fill the routing table. However, this is unrealistic in real networks. Furthermore, if the network changes, e. g. nodes are added, each node in AntNet must extend its routing table with the required columns and rows. Liang et al. explored the behaviour of an AntNet version that utilizes only locally available information. AntNet-local differs from AntNet with regard to the following aspects:

- The routing table comprises only entries for direct neighbours
- Each node utilizes a buffer to store an arriving ant's information, i. e. the timestamp and ID of the previous node

Choosing the next hop is done in a deterministic fashion if the destination is a direct neighbour. Otherwise it is chosen based on the probabilities in the routing table. It is obvious that these decisions are much more uncertain, since entries exist only for the direct neighbours and the respective probabilities are not differentiated with respect to the different destinations.

The travelling time is calculated incrementally. The forward ant does not keep the information about the hops and delays internally in its memory stack but stores this information together with its ID in the buffer of the next node. Eventually, this entry is read and deleted by the backward ant.

Algorithm 1 Basic operation of an Evolutionary Algorithm

1: randomly initialize the population
2: evaluate the population
3: **repeat**
4: select the best individuals according to their fitness
5: apply diversification operators to generate the new population
6: evaluate the population
7: **until** finishing criteria is met

Beehive is a swarm-based routing algorithm developed by Farooq and Wedde [76, 275, 277]. It is based on the foraging principles of a honey bee colony. We give a detailed description of BeeHive in Section 4.1.4.

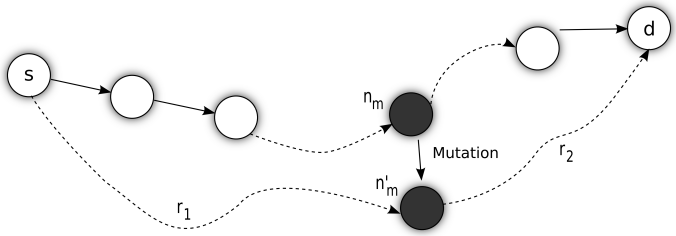
Evolutionary Algorithms Evolutionary Algorithms (EAs) constitute general randomised search heuristics that replicate the concepts of evolution in nature. Evolution in this context is considered as a process, influenced by chance, that allows for species to adapt to their surroundings. The basic assumption is, that the next generations of individuals will be gradually better adapted to their surroundings than the current generation. Furthermore, it is assumed that this refinement can be measured. It constitutes the fitness of the individuals. Based on these assumptions we define an EA as follows:

Definition 16 (Evolutionary Algorithm) An evolutionary algorithm[2] maximises a fitness function by gradually adapting a randomly generated population via selection and diversification operations until a finish criteria is met.

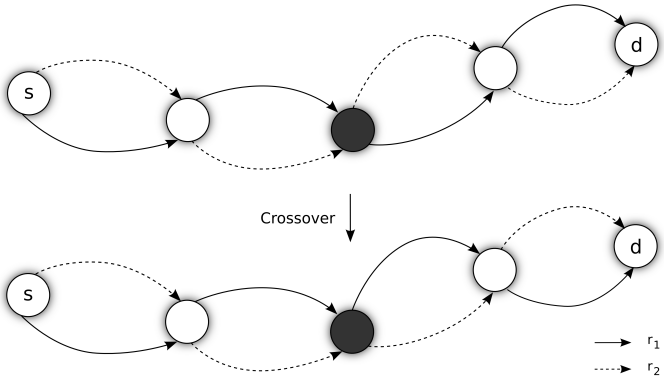
The basic operation of a EA is given in Algorithm 1. With regard to routing EA-based algorithms are realized as agent systems, similar to swarm-based approaches. Each agent represents an individual and is, from a technical point of view, a special kind of data packet that travels through the network. The quality of the traveled path, e. g. the end-to-end delay or the average link capacity utilisation, defines its fitness. If an individual is selected to be a parent for the next generation, the corresponding path will serve as the basis for the diversification. Generally, mutation and crossover strategies are used in this process.

Beyer and Schwefel point out in [25], that the choice of fitting mutation and crossover strategies is generally problem-dependent. However, they also present some general rules for choosing an appropriate diversification strategy. To foster the general understanding of these concepts, we give a short example, how the mutation and crossover strategies may look like in the routing context. The descriptions as well as the illustrations are adapted from the EA-based routing approach GARA (cf. [180]).

[2] Although some publications denote this concept as a genetic algorithm, the commonly used term today is evolutionary algorithm. We do not differentiate further between the two, though small differences exist, and stay for the sake of consistency with the term evolutionary algorithm.



(a) *Mutation operation*



(b) *Crossover operation*

Figure 4.3: *Diversification operations in evolutionary algorithms*

- *Mutation:* The first step in the mutation process is the selection of a random node n_m on the path from source node s to destination node d. Next, a neighbour node n'_m of n_m is chosen at random and the shortest paths from s to n'_m (r_1) and from n'_m to d (r_2) are computed using the Dijkstra algorithm. If the two paths r_1 and r_2 have nodes in common, the mutation will be discarded. Otherwise, the concatenation of r_1 and r_2 represents the mutated path. Figure 4.3a illustrates this process.
- *Crossover:* The crossover operation exchanges sub-paths between two selected paths. Both paths must have the same source node s and destination node d, as well as at least one other path-internal node in common. From the common internal nodes one is randomly chosen to be the crossover point. Eventually, the two sub-paths from the crossover point to the destination node are exchanged. Figure 4.3b visualizes the crossover operation for the two paths r_1 and r_2.

For more details on EAs that go beyond this short description, we refer the interested reader to [24] and [120]. We include in our further analysis the two EA-based routing approaches GARA and DGA.

The Genetic Adaptive Routing Algorithm (GARA) is specified in [179, 180]. It represents a source routing algorithm, i. e. the whole path from source to

Destination	Path	Frequency	Delay	Fitness
2	(1 3 2)*	7232	50	0.7
	(1 3 4 2)	2254	60	0.2
	(1 3 4 5 2)	1039	70	0.1
6	(1 8 6)*	20983	100	0.4
	(1 10 11 6)	34981	105	0.6
8	(1 8)*	30452	40	0.9
	(1 7 8)	3083	40	0.1

Figure 4.4: *Example routing table in GARA*

destination is specified in the data packet at the sender. Therefore, each node needs to know the whole topology of the network.

GARA maintains a routing table at each node similar to the example depicted in Figure 4.4 (adapted from [180]). For a destination, several complete paths are stored. Furthermore, each entry consists of the frequency of the particular path being used in the past as well as its measured delay. Finally, the fitness of each path is included, which determines the probability of using each path for future data packets. Initially, the routing table is empty. For each new destination a standard path (marked with a * in the example) is calculated using Dijkstra's shortest path algorithm based on a hop count metric. The delay for each path is calculated from the communication latency of special delay query packets that are sent periodically on each path.

After each evaluation of the fitness values, alternative paths are computed by applying diversification operations to the existing paths with a certain probability. We already described GARA's diversification operations in the general description of EAs (c. f. Figure 4.3). Furthermore, in order to keep the routing table from overflowing, selection operations are executed whenever the routing table exceeds a predefined threshold. GARA contains two selection strategies: (1) for all alternative paths to a destination, the path with the lowest fitness is deleted, (2) all paths to a destination that the node rarely communicates with are deleted. Thus, each node contains only good routes to destinations he frequently corresponds with. An extension to GARA, which additionally allows for the exchange of routes between neighbouring nodes, is proposed by Munetomo et al. in [181].

The Distributed Genetic Algorithm (DGA) is based on the concepts of GARA as well as AntNet and AntNet-local. It is described by Liang et al. in [153,154]. The authors' goal was to find a routing algorithm that does not require global information while still finding appropriate paths in the network.

Analogous to AntNet, DGA utilizes forward and backward agents. However, they travel trough the network following a pre-defined path. This path is defined as a string of next-hop offsets, e. g. {1, 5, 0, 4, 2, 3, 5} over the interval [0, L],

Agent ID	Fitness	Travel time (ms) and Node ID
95	0.32	$(3, J)$, $(9, C)$, $(21, W)$
234	0.39	$(1, B)$, $(7, A)$, ..., $(432, Y)$
...	...	...
31	0.71	$(5, C)$, $(9, K)$, ..., $(871, X)$

Figure 4.5: *Example routing table in DGA*

where L is selected to enable indexing of the maximum amount of neighbours a node may have. On entering a node, the next offset is used to determine the next hop, using a clockwise count which starts with the arriving link. If this results in choosing the arriving link, the node randomly chooses another and updates the string entry accordingly (deterministic mutation).

If the string is finished, the forward agent is changed into a backward agent which returns to the source node. This may also happen if no next hop can be determined. In this case the string is truncated accordingly. In contrast to AntNet, the backward agent changes only the routing table at the source node.

In addition to the routing table, DGA stores in each node a population of agents, which is generated randomly during startup of the network. Its size is proportional to the square of the total number of nodes inside the network. At the beginning half of these agents are sent into the network. The routing table (c. f. Figure 4.5) is initially empty. After four agents have been returned from their destinations, their fitness is evaluated. It measures the popularity of nodes in terms of the relative frequency α_k^i describing how often data packets are forwarded from node i to node k as well as the required travelling times t_k which are recorded by the agents. Thus, agents that discover shortest paths to frequently used destinations are favoured. In summary, the fitness function f has the following form:

$$f = \frac{\sum\limits_{\forall \text{destinations } k} \alpha_k^i \cdot t_k}{\sum\limits_{\forall \text{destinations } k} t_k}. \tag{4.1}$$

Based on their fitness, the two best agents are chosen and stored in the routing table or their fitness values are updated if they already exist. Upon reaching a certain size, the two worst agents are deleted from the routing table. Using one-point crossover and mutation two new agents are generated based on the two selected agents. At this, the mutation operator randomly chooses one element in the string and adds or subtracts a random integer ensuring that the result still resides in the interval $[0, L]$. Due to the crossover operation agents may have a different size. The new agents are included in the population while the worst two agents are eliminated. Eventually four agents are chosen randomly from the population and sent to explore the network.

The next hop for a packet is chosen deterministically based on the travelling times of the entries, i. e. the neighbour with the shortest travelling time. If no entry exists, the neighbour with the highest fitness is chosen.

Additionally to these basic concepts, the authors enhanced DGA further. On the one hand, each node sends the best agents, according to their fitness, to its neighbours in fixed intervals. On the other hand, an aging mechanism prevents stagnation of the routing entries. During the update of the routing table the fitness of each entry is multiplied by a constant $c \in (0, 1)$ and the travelling time is divided by c.

4.1.3 OSPF

The routing protocol OSPF was developed by the IETF for the intra-domain routing setting. The current standard OSPFv2 (OSPF version 2) for IPv4 (IP version 4) based networks is specified in RFC 2328 [176], with several extensions and details being furthermore specified in additional RFCs[3]. For IPv6 (IP version 6) based networks, OSPFv3 (OSPF version 3) is specified in [55]. We cover only OSPFv2 in this dissertation and refer to it for readability as OSPF, unless stated otherwise.

OSPF constitutes a link-state routing protocol, i. e. the shortest path between two nodes is computed on the basis of all links and their associated costs which are known to every node in the network. In order to gain this knowledge, the nodes regularly exchange their lists of neighbours and link conditions.

At a high level, the operation of OSPF follows this scheme:

1. In order to discover their neighbours, routers send *hello packets* out on all interfaces. Based on certain information inside this hello packet two routers decide if they become neighbours.
2. Each router sends *link-state advertisements* (LSAs) to all of it's neighbours. These LSAs contain all of the router's links, including their state, as well as the router's neighbours.
3. Upon receiving an LSA from a neighbour the router includes the data in it's link-state database and forwards the LSA to all of it's neighbours. Thus, all routers learn the network topology and are able to build identical link-state databases.
4. Eventually, a shortest path first (SPF) algorithm is employed by each node to generate a loop-free graph describing the shortest path to every other router in the network. Note that the shortest path is based on the link costs, i. e. the lowest cost path, rather than the shortest hop count.

   The SPF algorithm for a node A is displayed in Algorithm 2. We use the following notation:

   - N: denotes the set of nodes for which a shortest path is known.

[3] See Appendix B.1 for an overview.

Algorithm 2 Shortest path first algorithm for a node A

```
1:  /* initialisation */
2:  N = {A}
3:  for all nodes v do
4:     if v is neighbour of A then
5:        D(v) = c(A, v)
6:     else
7:        D(v) = ∞
8:     end if
9:  end for
10:
11: /* iteration */
12: repeat
13:    select node w ∉ N with minimal D(w)
14:    N = N + w
15:    for all neighbours v of w, whereas v ∉ N do
16:       D(v) = min( (D(v), D(w) + c(w, v)) )
17:    end for
18: until all nodes n ∈ N
```

- $c(i, j)$: represents the costs associated with the link between the nodes i and j. For simplicity we assume that $c(i, j) = c(j, i)$.
- $D(v)$: denotes the costs for the path to node v.

5. In a stable network topology OSPF becomes rather quiet after this step. Though hello packets are still exchanged regularly between neighbours as keepalives, LSAs are only seldom retransmitted, after a configurable timeout.

6. If changes occur inside the network, e. g. a router or link going down, each of the neighbouring routers sends out a new LSA that reflects this change. Based on this information all routers will recalculate the shortest paths and update their routing tables.

Generally, OSPF is designed as a deterministic routing protocol, i. e. the next hop is always chosen deterministically to be the next node on the shortest path to the destination. However, it also supports *equal-cost multipath* (ECMP) routing. Thus, if multiple equal-cost routes to a destination exist, OSPF discovers and uses all of them to transfer data packets.

OSPF is suitable for large-scale networks, despite the facts that LSAs are flooded and all routers have to keep track of all links inside the network. To achieve scalability, OSPF incorporates an *area* concept, which allows for a hierarchical structuring of the network. The major advantage of this approach is the significant reduction in the amount of network bandwidth consumed by routing updates. Inside each area, LSAs are exchanged as described above. However, the area border routers summarize the internal information when sending an LSA on the outside interface. This results in only one entry in the non-area node's routing tables regardless of the actual area size.

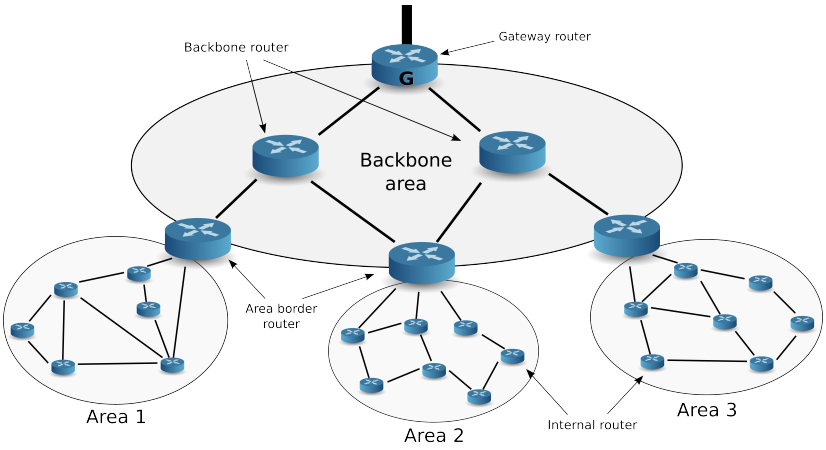


Figure 4.6: *OSPF area concept*

Figure 4.6 illustrates an example of the area concept. Areas are arranged into a two-level hierarchy whereby one or more areas are attached to the central backbone area.

Another feature explicitly integrated into the OSPF protocol is the authentication of routing packets. Different types of authentication schemes are supported by OSPF and can be configured on a per network area basis. This prevents rogue routers from advertising unauthorized routes, allowing only trusted routers to participate in the network routing.

Finally, OSPF is able to integrate routes provided by another routing protocol, such as BGP which allows for an easy integration into the global internet.

This concludes our introduction to OSPF. For more information about the protocol we refer the interested reader to [175] and [176] as well as the respective RFCs that define further aspects of OSPF. An overview of these is provided in Appendix B.1.

4.1.4 BeeHive

Wedde and Farooq have developed the swarm-based routing algorithm BeeHive based on the foraging principles of a honey bee colony [76, 275, 277].

BeeHive uses agents to assess the links' qualitative characteristics and update the routing information on the nodes. At this, only forward agents exist in BeeHive, unlike in AntNet, though two different types: *short-distance bees* and *long-distance bees*. Short-distance bees explore only the direct neighbourhood of a node (up to a predefined number of hops), while the long-distance bees explore the whole network. Their lifetime is limited as well by a hop limit. The differentiation between these two types is motivated by the fact, that only few

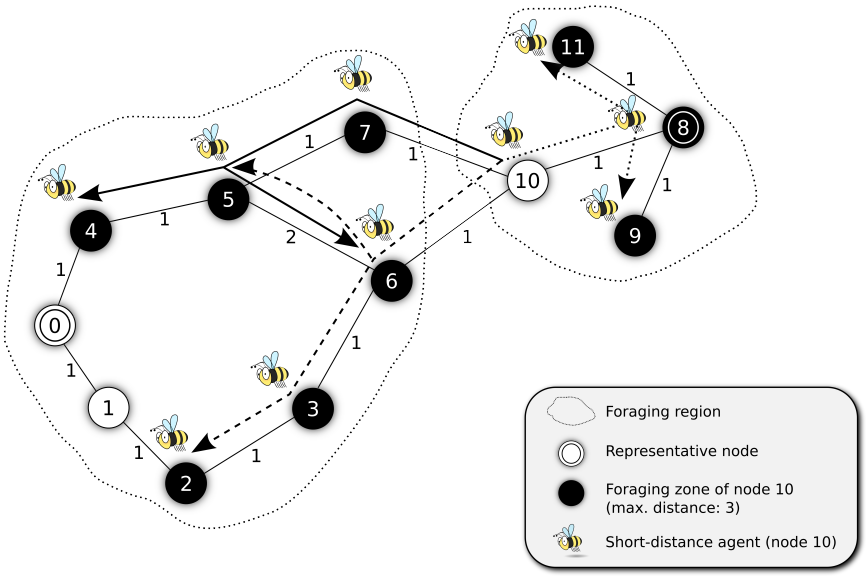


Figure 4.7: *The region and zone concept in Beehive*

bees explore long-distance while most stay in the close neighbourhood of the hive.

BeeHive is by design a hierarchical routing protocol. For this purpose, the network is divided into *foraging regions*. These partitions stay fixed during the lifetime of the network. Each foraging region has a representative node which is usually the node with the smallest IP. If this node fails during the network's lifetime, the node with the next higher IP will assume the position as representative node. Farooq has shown in [76] that more elaborate selection processes do not yield better performance results while leading to significantly more administrative effort.

In addition to the foraging regions, each node maintains its personal *foraging zone*. This contains all the nodes in the direct neighbourhood of the node and is determined by the range that the short-distance bees travel. It may span over multiple foraging regions.

Figure 4.7 gives an example of this. Two foraging regions are illustrated, the representative nodes being 0 and 8. Furthermore, the foraging zone of node 10 is indicated by the dark nodes.

BeeHive determines the foraging regions autonomically during the start-up phase, by means of the short-distance bees. In detail, the following three steps will be repeated until the whole network is partitioned into foraging regions:

1. In the beginning each node assumes that it is the representative node of the foraging region and sends out its short-distance bees to propagate this belief.
2. If node i receives a short-distance bee from node s, claiming that s is the representative node, i decides on the basis of the IP addresses if it resigns his status. If the IP address of s is smaller than its own, it joins the foraging region of s and abandons its efforts.
3. If i receives at a later time a short-distance bee from s announcing that s resigned its status and joined another foraging region, i restarts the whole process.

These steps are repeated until the network is divided into foraging regions and foraging zones. Finally, each node informs all other nodes in the network to which foraging region it belongs. If link of node failures occur in the network the foraging regions may need to reshaped. In this case the initialisation process is repeated. BeeHive detects link or router failures by means of hello packets each node sends at regular intervals to its neighbours.

During normal operation, each regular node sends out short-distance bees while the representative nodes send out long-distance bees. When a bee is received at a node, it updates the routing information and is flooded to all neighbours except the one from which it arrived. This process repeats until the bee's lifetime exceeds or if another replica of this bee has already been received at the node. Bees generally have no destination. Rather the flooding algorithm is a variant of a breadth first search algorithm. In order to disseminate the routing information quickly, bees are always communicated via priority queues.

On their way they assess each link's overall delay and update the routing information on the next node. However, since bees only move forward, they have to estimate the trip time back to their originating node via the traveled path. For example, a bee originates at node s and arrives at node k on the link from the intermediate node i. It estimates the time t_{ks} as the sum of the queue delay q_{ki}, the transmission delay x_{ki}, and the propagation delay p_{ki} of the link (k, i). Furthermore, it adds to this the estimated travel time t_{is} from node i to node s. The transmission delay and the propagation delay are static link characteristics while the queue delay is directly dependent on the size of the queue (l_{ki}). It is approximated with regard to the available bandwidth of the link (b_{ki}). This results in the following formula:

$$t_{ks} = \underbrace{\frac{l_{ki}}{b_{ki}}}_{q_{ki}} + x_{ki} + p_{ki} + t_{is} \tag{4.2}$$

During the initialisation phase the nodes approximate the static link characteristics by means of the hello packets that are sent on each link. The bee estimates the queue delay for a link by observing the size of the queue for the normal traffic.

For packet switching each node s maintains three routing tables:

1. *Intra foraging zone* table: for each destination d inside the foraging zone and each neighbour k on a path to these an entry $P_{kd} = (q_{kd}, p_{kd})$ is stored. It contains the queue delay q_{kd} and the propagation delay p_{kd} which a packet will experience on its way to node d via node k. The tables is updated by short-distance bees.
2. *Inter foraging region* table: for each representative node d in the network and each neighbour k on a path to these an entry $P_{kd} = (q_{kd}, p_{kd})$ is stored. This table is updated by long-distance bees.
3. *Inter foraging membership* table: an entry exists for each node in the network that displays its foraging region. This table is built during the initialisation phase.

The choice of the next hop for a packet with the destination d is based on the goodness g_{kd} of each neighbour k of node s ($k \in \mathcal{N}_s$). In this process, the node checks first if d resides inside its foraging zone. In this case, the goodness calculations are based on the values from the intra foraging zone table. Otherwise, the foraging region of the destination node is determined from the inter foraging membership tables and the calculations are based on the entries in the inter foraging region.

The aim of the goodness is to reflect the actual situation in the network. Therefore, the goodness for a neighbour should be high if the corresponding link's overall delay is low and vice versa. The critical factor here is the queue delay during high network utilisation or respectively the propagation delay during low utilisation. The calculation of the goodness value considers this:

$$q_{kd} = \frac{1/\left(p_{kd} + q_{kd}\right)}{\sum_{l=1}^{|\mathcal{N}_s|} 1/\left(p_{ld} + q_{ld}\right)} \tag{4.3}$$

Eventually, the next hop is chosen in a probabilistic manner, based on the goodness values. More precisely, BeeHive uses a method called stochastic sampling with replacements. This ensures that the probability of choosing neighbour k with goodness g_{kd} is at least $\frac{g_{kd}}{\sum_{i=1}^{|\mathcal{N}_s|} g_{id}}$.

Due to the aim of reflecting the actual network situation, the values p and q which provide the basis for the goodness calculation need to be updated frequently. This is done by the bees. More precisely, the short-distance bees update the delays in the intra foraging zone table, while the long-distance bees update the delays in the inter foraging region table. Furthermore, the bees communicate with each other in order to incorporate the experience of different paths in their estimate.

Figure 4.8 visualizes the communication of the bees. Here, three paths exist from node s to node k. Thus, s launches three replicas of the same bee on each of the available paths. Each estimates the link's delay as described above. On

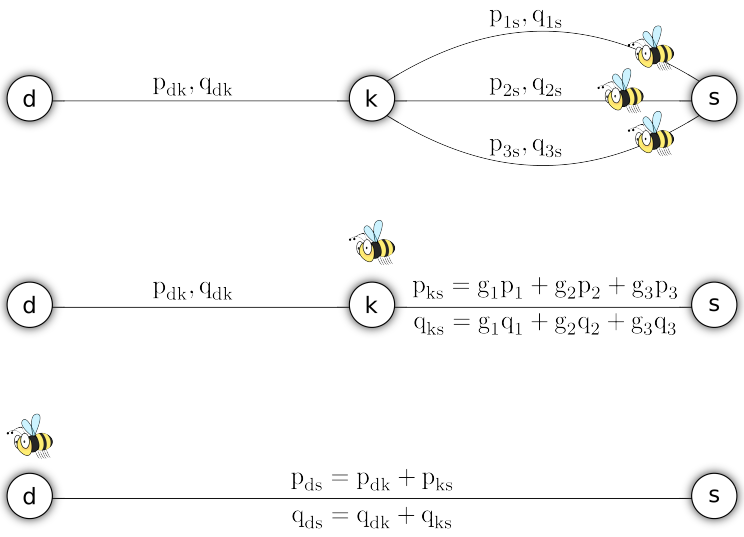


Figure 4.8: *Communication paradigm of the bee agents*

node k the three bees actualize the entries in the routing table and exchange their estimates. Only the bee that arrived first is allowed to continue its flight. It incorporates the two other estimates proportional to the quality of the paths they traversed:

$$q := \sum_{i \in \mathcal{N}_k} (q_{is} \cdot g_{is}) \qquad \text{and} \qquad p := \sum_{i \in \mathcal{N}_k} (p_{is} \cdot g_{is}) \tag{4.4}$$

This concludes our introduction to BeeHive. For more information about the algorithm as well as performance evaluations by its authors we refer the interested reader to [76].

4.2 Survey of the Published Evaluations

Each of the nature-inspired routing protocols mentioned in the fundamentals was evaluated by the respective authors, mostly by means of simulation. In order to get a first impression of their quality and to be able to focus the further analysis on a promising candidate, we first survey the published evaluations and point out problematic areas.

Generally, the nature-inspired approaches concentrate on performance evaluations, making them applicable for intra-domain routing. We are not aware of

any nature-inspired approach that targets inter-domain routing. Furthermore, it is highly unlikely that all domain operators switch to an autonomic routing algorithm, since the basis for the inter-domain routing decision centers mostly on monetary aspects and contracts between carriers. Consequently, we focus our analysis on intra-domain routing as well.

Furthermore, we exclude GARA from the further analysis, since it represents a source routing approach. This results in the whole path being specified in each data packet, which is clearly not a scaleable solution. It is therefore in our opinion not a useable approach for intra-domain routing.

Based on the published results we have compiled a summary of the evaluations in Table 4.1. In order to determine their comparability we extracted several important aspects that describe the evaluations. Foremost, we were interested in which algorithms were used for comparison. Furthermore, we looked which network topologies were used, how traffic was generated, and if network failures were taken into account. Finally, we listed the used evaluation parameters.

For the sake of brevity, we focus in the following overview on the most interesting findings. The detailed survey is available in [209].

Algorithms: No all-encompassing evaluation of all approaches exists. However, due to the complexity of this task, this does not come as a big surprise. The most comprehensive evaluation is done by the authors of BeeHive, being also the most recent. Generally, all authors evaluated their approach in comparison to older versions of their own protocol and AntNet.

Regarding OSPF, only a simplified version that emulated OSPF's general aspects was considered, both in the AntNet and BeeHive evaluations. Essentially, on the basis of statically assigned link costs that mirror the physical characteristics of the links, OSPF represents a simple SPF algorithm. None of the more elaborate features of OSPF were taken into account. Concerning this, especially Di Caro et al. acknowledge that this functional limitation penalizes OSPF with regard to its performance. Barán and Sosa do not give any details about the employed version of OSPF in the evaluation of AntNet*. However, since their work is based on AntNet, it is likely that OSPF is only considered in its basic functionality as well.

Network topologies: The choice of network topologies in the evaluations is heavily influenced by the AntNet evaluation, where the specially designed SimpleNet and the two real world network topologies NSFNet and NTTNet were used. Especially NTTNet was adopted by all authors. However, its description and graphical representation differ greatly among the publications. We must therefore assume that it is unlikely that the same network was used in all evaluations.

Scalability with regard to network topologies consisting of more than 150 nodes was only considered in the BeeHive evaluation. Especially, it revealed among other interesting aspects, that AntNet and AntNet-CO do

not scale well in these grand topologies and display the worst results of all tested approaches, including the impaired OSPF version, in these settings.

Traffic generation: Traffic was only considered on an abstract flow-based level in all evaluations. None analysed realistic traffic profiles or modeled realistic Internet protocols. The most elaborate discussions can be found in the AntNet and BeeHive evaluations. Both consider mainly session-based traffic in which several different types exist to model several traffic situations and distributions.

Network failures: The robustness of the algorithms in the face of network failures was analysed by all evaluations except AntNet and AntNet-CO. However, the BeeHive evaluation included AntNet in this analysis. Regrettably, OSPF was excluded in this, as the authors of BeeHive deemed it to be too inferior in its performance compared to BeeHive and AntNet.

Generally, the individual evaluations are not fully comparable due to the significant differences in the simulation settings. Especially, the comparison to the state-of-the-art routing protocol OSPF was never in the focus of the evaluations. Apart from that, the evaluation results are partially not comprehensible, since important details of the evaluations are omitted in the descriptions. Thus, an objective assessment of each algorithm's efficiency is not possible on this basis.

However, it is possible to deduce an approximate ranking regarding the performance of the evaluated approaches. Especially, the evaluations conducted by Farooq et al. in [76] are quite extensive and provide a good basis for this.

In general, the performance results of the swarm-based approaches outperformed the EA-based approach DGA with the exception of AntNet-Local, showing mostly the lowest performance of all approaches. The improved versions of AntNet, AntNet-CO and AntNet* achieved furthermore better results than the original version of AntNet. Yet, the results for AntNet* raise several questions since they contradict in parts the AntNet results. The best results among all approaches were observed for BeeHive.

As a result of our survey we decided to focus on BeeHive for our further evaluation as it seems to provide the highest performance of the nature-inspired routing approaches.

4.3 Analysis

The survey of the published evaluations revealed several general shortcomings, resulting in the dilemma that no objective assessment of the nature-inspired approaches is possible on this basis. Especially the direct comparisons of the respective nature-inspired approaches to OSPF were done only rudimentary, if at all. Furthermore, the focus of the published evaluations was generally on performance issues and not on dependability. While dependability comprises performance aspects, a more comprehensive analysis of BeeHive as the

	Nature-inspired routing approaches					
	AntNet[1]			**AntNet-Local**	**BeeHive**	**DGA**
Considered publications	[15, 16, 43, 44]			[152]	[76, 275, 277]	[153, 154]
Algorithms used for comparison						
OSPF	✓[2]	✓[2]	✓	–	✓[2]	–
AntNet	✓	✓	✓	✓	✓	–
AntNet-CO	–	✓	–	–	✓	–
AntNet*	–	–	✓	–	–	–
AntNet-Local	–	–	–	✓	–	✓
BeeHive	–	–	–	–	✓	–
DGA	–	–	–	–	✓	✓
Evaluated network topologies						
SimpleNet	✓	–	✓	–	✓	–
NSFNet	✓	–	✓	–	–	–
NTTNet	✓	✓	✓	✓	✓	✓
Randomly generated	–	✓	–	–	✓	–
Large-scale	–	–	–	–	✓	–
Traffic specification						
Session-oriented	✓	✓	–	–	✓	–
Sessionless	–	–	–	–	✓	–
Poisson distribution	–	–	–	✓	–	✓
Undefined	–	–	✓	–	–	–
Taking failures into account						
Link-/router failure	–	–	✓	✓	✓[3]	✓
Evaluation parameters						
Throughput	✓	✓	✓	✓	✓	✓
Delay	✓	✓	✓	–	✓	✓
Network utilization[4]	✓	✓	–	–	–	–
Routing overhead	✓	✓	–	–	✓	–
Queue size	–	–	–	✓	–	–
Session delay	–	–	–	–	✓	–
Complete sessions	–	–	–	–	✓	–
Delivery rate	–	–	–	–	✓	✓
Routing loops	–	–	–	–	✓	–
Jitter	–	–	–	–	✓	–
Agent processing time	–	–	–	–	✓	–
Suboptimal overhead	–	–	–	–	✓	–
Dropped packets	–	–	–	–	–	✓

[1] *AntNet family: AntNet, AntNet-CO, and AntNet*.*
[2] *in a simplified version.*
[3] *OSPF was excluded from this evaluation.*
[4] *only data packets.*

Table 4.1: *Summary of the published evaluations.*

most promising approach in comparison to OSPF regarding all aspects of dependability is indispensable. In particular the aspects availability, reliability, integrity, and maintainability (cf. Section 2.1.1.2) of the routing approaches need to be evaluated.

Availability and reliability focus on performance aspects in the routing context, such as how well the network is utilized and how robust the algorithms handle failures inside the network. We analyse these aspects in Section 4.3.1.

In Section 4.3.2 we focus our analysis on the evaluation of the approaches' characteristics regarding system integrity, i. e. the protection of the routers as well as the messages they exchange against intentional or accidental change or manipulation.

Finally, we analyse maintainability aspects of the routing approaches in Section 4.3.3. In particular, we are interested in the extend to which both approaches operate autonomously, addressing the growing complexity in UC environments.

We neglect the safety aspect of dependability in our further evaluations, since routing functionality is generally free from risk of causing harm the user or the system.

4.3.1 Performance

Farooq provides in [76] an extensive performance evaluation of BeeHive in comparison to other nature-inspired routing approaches. For reference he also included OSPF in these evaluations. However, we question some of his results. Especially the following aspects need clarification:

OSPF's performance results: the performance results of OSPF were considerably inferior to the results BeeHive achieved. However, the evaluation of OSPF was done using a simplified basic version. Therefore, we put forth the hypothesis that a realistic OSPF configuration with optimized link costs, should increase its performance results significantly.

Realistic results: Farooq's simulation environment abstracted from real networks. Neither did he include realistic protocols, such as IP and TCP, nor realistic traffic patterns. While abstract simulations do provide important insight into a protocol and its performance in certain situations, it is questionable if these findings can be generalized to real world environments.

For example, the widely used TCP protocol comprises several mechanisms to provide a reliable transport service in unreliable networks. This results, among others, generally in a higher percentage of delivered packets, since TCP retransmits dropped packets. Thus, the influence of the routing algorithm on this performance parameter decreases significantly. Furthermore, multipath routing schemes in general might even decrease

the TCP performance in certain situations [290] resulting in the need to adapt TCP to improve the results [148].

Therefore, we think it prudent to compare both approaches' performance in a simulation environment that models realistic network characteristics.

Robustness: no evaluation exists that compares the behaviour of OSPF and BeeHive during network failures. Farooq excluded OSPF from this evaluation claiming that the performance of OSPF would be inferior to BeeHive's performance since the results were already significantly worse in a faultless network. This must be validated in order to soundly determine both algorithms' robustness with regard to network failures.

We analyse these three aspects by means of simulation. Therefore, we first describe our simulation environments in the next section and afterwards provide the detailed settings for each simulation and discuss the respective results.

4.3.1.1 Simulation Environments

We use the C++-based discrete event simulation framework OMNeT++ version 3.3 [267, 268]. Additionally, we utilise Internet-related functionality, such as the protocols IP, TCP, and UDP, from the corresponding INET framework version 20061020 [3]. It also provides two implementations of OSPF.

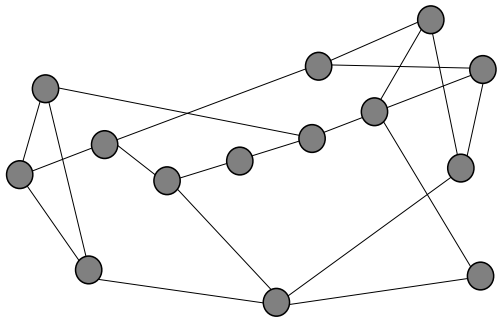
Generally, we reiterate each simulation ten times to avoid choosing an unfavourable seed for the random number generator by accident. Thus, the performance results always represent the average of ten independent simulation runs. Each simulation run lasts 1030 seconds. During the first 30 seconds, the routing algorithms have exclusive access to the network in order to initialize their routing tables. No application traffic is generated during this time.

With regard to network topologies, we model two real world WANs that are commonly used in the evaluations of nature-inspired routing approaches as well: NSFNet and NTTNet.

NSFNet represents the old US T1 backbone consisting of fibreoptic links that allow for a bandwidth of 1.544 Mbit/sec It comprises 14 nodes that are connected with 21 bidirectional links, each having a transmission delay between 4 and 20 milliseconds. Figure 4.9 illustrates the topology and summarizes the characteristics.

NTTNet models the Japanese backbone or, more precisely, the fibreoptic backbone of the company NTT. NTTNet comprises 57 nodes that are connected with 81 bidirectional links. The bandwidth of the links is uniformly 6 Mbit/sec while the transmission delays range between 1 and 5 milliseconds. Figure 4.10 illustrates the topology and summarizes the characteristics.

Although both WANs are rather old and do not represent the current state-of-the-art regarding for example, link bandwidth, we chose these for two reasons.

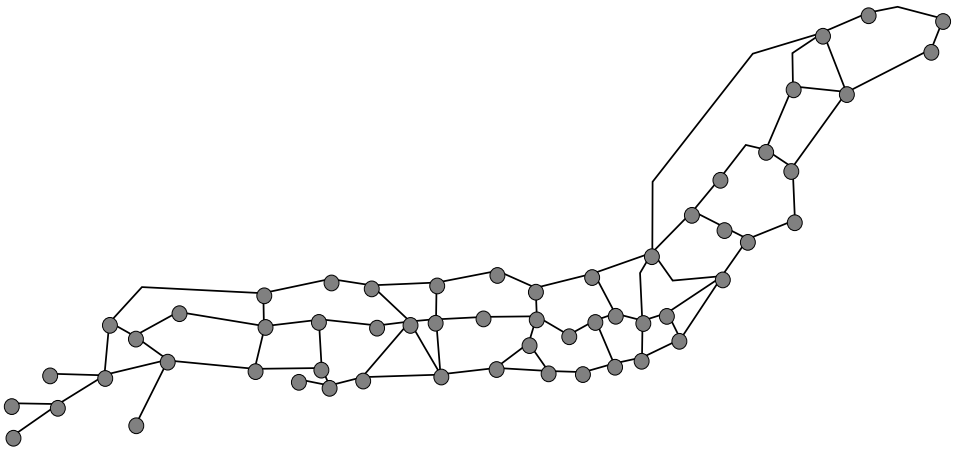


(a) *Topology*

Number of nodes:	14	Bandwidth:	1,5 MBit/s
Bidirectional links:	20	Link delay:	4–20 ms

(b) *Characteristics*

Figure 4.9: *NSFNet*



(a) *Topology*

Number of nodes:	57	Bandwidth:	6 MBit/s
Bidirectional links:	81	Link delay:	1–5 ms

(b) *Parameters*

Figure 4.10: *NTTNet*

First, they are widely used in the evaluations of the nature-inspired routing approaches. Second, they allow us to overload the network while still keeping the time and memory requirements of the simulations reasonable.

Parameter	Description
Percentage delivered	The amount of packets (in percent) that arrived at their respective destination on the application layer. The routing algorithm should maximise this value.
Throughput	The average throughput (in Mbit/sec) denotes the average traffic that arrived at its destination on the application layer per second. The routing algorithm should maximise this value.
Complete sessions	The amount of sessions (in percent) that were transferred completely, i. e. all packets arrived successfully at their destination application. Note, that while TCP retransmits dropped packets, a single dropped UDP packet renders a UDP session incomplete. The routing algorithm should maximise this value.
Packet delay	The average time (in seconds) the packet traveled from sender to destination. Hereby, we measure the packet delay from sending to receiving application. Thus, it includes for example the time that is required for a TCP retransmission of the same packet if it was dropped previously. The routing algorithm should minimize this value.
Jitter	The average jitter (in milliseconds). Jitter results from the variable packet delays of the data packets and denotes their average difference. Especially time critical applications, such as VoIP or video streaming, require low jitter values in order to provide a good replay quality. The routing algorithm should minimize this value.
Session delay	The average time (in seconds) that is required to finish a session completely. The routing algorithm should minimize this value.
Routing overhead	The bandwidth (in percent) that is consumed solely by the routing algorithm. The routing algorithm should minimize this value.

Table 4.2: *Performance parameters collected during the simulations*

For our experiments we have two simulation environments available: the simulation environment Farooq devised and used in his simulations in [76] and our newly devised simulation environment that incorporates realistic Internet functionality. Both network topologies are available in the two environments. Furthermore, we collect the performance parameters illustrated in Table 4.2 in both environments.

Farooq's simulation model Farooq provided us with the original simulation model he used in his experiments in [76]. We are grateful for this, since this allows us to use his implementation of BeeHive, eliminating a great potential source of error. Regarding OSPF, we made only minor changes to the model in order to allow for using link costs when calculating the shortest paths between nodes.

Parameter	Description
MSIA	Mean session inter-arrival time
MPIA	Mean packet inter-arrival time in a session
sessionSize	Size of the session
packetSize	Size of the packets

Table 4.3: *Traffic generation parameters in Farooq's simulation environment*

The traffic is specified on a flow-level, i. e. the communication is represented by an abstract continuous flow of packets between the nodes. The parameters displayed in Table 4.3 are used to model the flow. Generally, each node generates traffic. At this, the destination is chosen alternating completely at random or at random while favouring the previous destination by a probability of 40%. We do not consider network failures in his model.

For a more detailed description of Farooq's simulation environment we refer the interested reader to [76].

Realistic simulation model In addition to Farooq's simulation model we devised and implemented our own model striving for more realism. The significant difference to Farooq's model lies in the fact that we model the complete protocol stack as it exists in the Internet today. Furthermore, we use the Quagga OSPF implementation from the INET framework for OMNeT++ , which is a real routing software [2] that was adapted to run inside OMNeT++. A small drawback results from the fact that the adaptation layer does not support the use of ECMP routing which will have a slightly negative impact on the OSPF results. Though it is impossible to quantify this impairment, the results will still be meaningful, since they represent the worst case scenario, in which no equal-cost paths exist inside the network.

In order to include BeeHive in our model, we implemented a wrapper that allows for Farooq's version of BeeHive to run in our model making only minor adaptations necessary. We tested our implementation by also porting Farooq's traffic generator to our model. We found no significant discrepancies in the attained results in comparison to his simulation model and thus assume that our wrapper works correctly. For completeness we provide the results in Appendix B.2.

Both described network topologies are available for use in this model as well. In order to model realistic traffic, we looked at the characteristics of real Internet traffic, which comprise, amongst other aspects, self-similarity [149] and heavy-tailed distributions [77]. Similar to Farooq's model, each node generates traffic, though the destination is always chosen at random. We based the traffic generator on ReaSE [82, 83] which considers these known characteristics of real Internet traffic. In detail, we specify the fundamental traffic mix as illustrated in Table 4.4. Based on this, we devise four traffic profiles that successively increase the network load until the network is overloaded. Finally, we include the

Application	Percentage[1]	Protocol
Backup	1.57	TCP
Interactive server	4.71	TCP
Email server	4.19	TCP
Streaming server	1.05	UDP
Web server	11.52	TCP
Misc	20.42	UDP
DNS	56.54	UDP

[1] of the total traffic.

Table 4.4: *Traffic mix*

possibility to disable and re-enable nodes during the simulation at configurable times, to simulate router failures.

4.3.1.2 Evaluation 1 – Optimized link costs

The performance results of OSPF should increase significantly if the link costs are optimized. This should become especially apparent with increasing traffic volume, since the traffic load should be balanced better across the network, resulting in a longer time before single network links become overloaded. In this first evaluation we analyse this hypothesis.

OSPF bases the calculation of the shortest paths solely on the given link costs. Thus, changing the link costs results in different shortest paths and consequently a different network utilization and overall performance. The link weight setting problem, i. e. finding the optimal link costs, is a well discussed topic in the literature, e. g. [78, 79, 215, 223].

Optimizing link costs represents an external optimization method, i. e. an administrator or management system may calculate the optimal link costs for the available traffic and update the router configurations accordingly, either before or during operation. However, each changing of the link costs results in a recalculation of the shortest paths in the network. Therefore, this optimization should be used sparingly during operation as it results in unstable routes during the recalculation time. Another problem represents the fact that the traffic has to be known in advance to optimize the link costs accordingly. It is evident that great changes in the traffic result in a different utilization of the network. Thus, optimized link costs for one traffic type might not be optimal for another.

Generally, the link weight setting problem is NP-complete [78]. However, several optimization strategies are proposed and discussed in the literature, e. g. using local search algorithms [78, 79], evolutionary algorithms [38, 72], or heuristics [228].

We optimize the link costs via simulation, using the cost metric from [79], where $\emptyset_a$ denotes the cost for a link a, $c(a)$ denotes the capacity of link a, and x denotes the maximum load during the simulation:

$$\emptyset_a(x) = \begin{cases} 1 & \text{for} \quad 0 \leq x/c(a) < 1/3 \\ 3 & \text{for} \quad 1/3 \leq x/c(a) < 2/3 \\ 10 & \text{for} \quad 2/3 \leq x/c(a) < 9/10 \\ 70 & \text{for} \quad 9/10 \leq x/c(a) < 1 \\ 500 & \text{for} \quad 1 \leq x/c(a) < 11/10 \\ 5000 & \text{for} \quad 11/10 \leq x/c(a) < \infty \end{cases} \tag{4.5}$$

After each simulation run, the utilization $x/c(a)$ of each link a is evaluated and the corresponding cost is updated according to this cost metric. Overall, we did 100 optimization iterations to keep the total optimization time reasonable. While this does not represent an exhaustive optimization, the simulation results, which we present in the following, already show a clear trend. From now on, we refer to OSPF using uniform link costs of one as *UnitOSPF* and to OSPF using optimized link costs as *Adv.OSPF*.

Simulation settings In order to analyse the effect of optimized link cost isolated from other factors, we use Farooq's simulation model for this analysis. This guarantees valid results for BeeHive, allowing us to unambiguously assess Farooq's claim that OSPF generally performs worse than BeeHive.

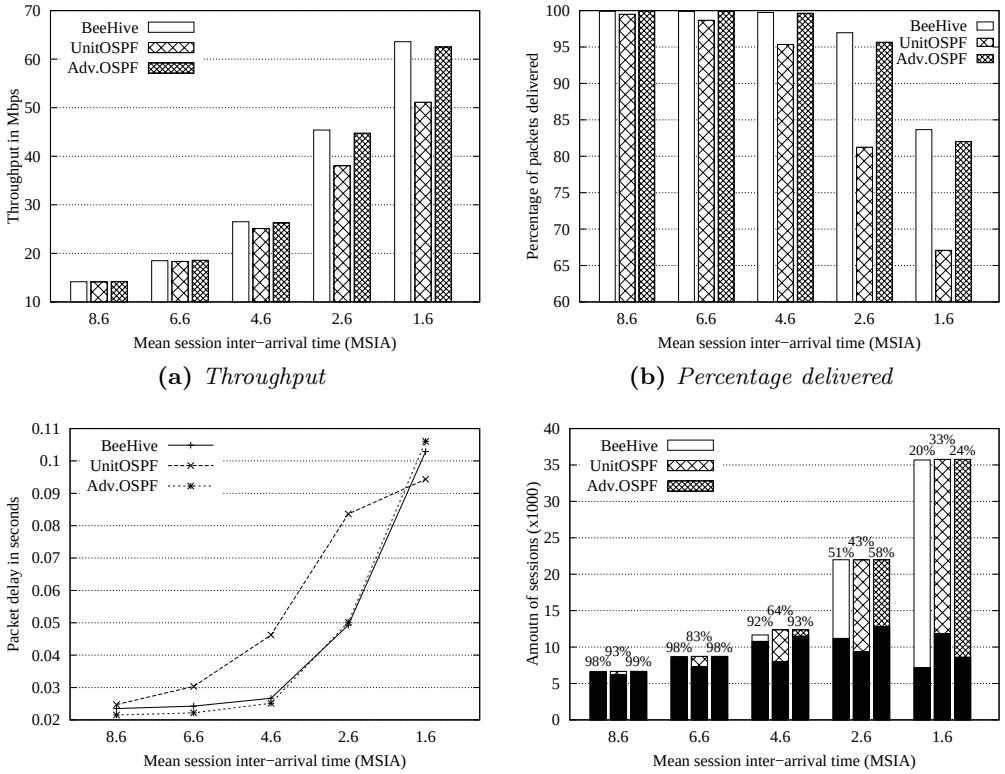
Furthermore, we use in accordance to Farooq the network topology NTTNet and specify the traffic equally to his settings. Each node generates traffic with MPIA = 0.005 sec, sessionSize = 2130000 bit, and packetSize = 512 byte. The buffer size in each router is set to 1000 packets. The value for MSIA is gradually decreased from 8.6 to 1.6, in order to continuously increase the traffic load.

We are especially interested in the performance parameters throughput, the percentage of successfully delivered packets, the packet delay, and the total amount of created and successfully finished sessions.

Results In general, the results support our hypothesis that the performance of OSPF increases significantly, if it is provided with optimized link costs.

Figure 4.11 visualises the simulation results in detail. In each diagram the x-axis denotes the five different traffic situations, which are generated by gradually decreasing the MSIA from 8.6 to 1.6. Thus, the network load increases from left to right.

Adv.OSPF outperforms BeeHive by a slight margin in low-traffic situations and performs only slightly worse in high-traffic situations regarding throughput, percentage delivered, and packet delay which are illustrated in Figure 4.11a,



(a) *Throughput*

(b) *Percentage delivered*

(c) *Packet delay*

(d) *Total and completed sessions*

Figure 4.11: *Simulation 1 results*

Figure 4.11b, and Figure 4.11c respectively. Furthermore, the session analysis reveals that Adv.OSPF performs generally better than BeeHive, i. e. more sessions are completed successfully. Though the difference is only minimal during low traffic situations, the performance gap widens with increasing traffic. Figure 4.11d reflects this. It displays the total number of generated and successfully completed sessions. Each shaded column displays the number of total sessions, while the successfully completed sessions are displayed in black. The percentage of completed sessions is given above each column for the sake of clarity. In the cases where nearly 100% of the sessions are completed, the first column always displays the results for BeeHive, the second UnitOSPF, and the third Adv.OSPF respectively.

With increasing traffic load, UnitOSPF performs gradually worse than BeeHive. The fact that it achieves the best results regarding packet delay in the overload situation results from the fact that 33% of all packets are dropped when using UnitOSPF, compared to only 17% for Beehive and 18% for Adv.OSPF. Generally, the packets that account for high delay values are those with longer routes, i. e. packets that have to travel longer distances in the network. However, these are statistically more likely to get dropped than packets with short routes. Since we do not consider the delays of dropped packets in the packet

delays, the mean packet delay is therefore lower in UnitOSPF, compared to BeeHive and Adv.OSPF.

At first it is also remarkable that UnitOSPF completes the most sessions successfully in the overload situation, while it performs generally worse than BeeHive and Adv.OSPF. However, this can be explained by looking at the general strategies of the routing variants and the characteristics of NTTNet. UnitOSPF always uses the shortest path regarding the hop count while BeeHive and Adv.OSPF statistically use longer routes. Furthermore, all links in NTTNet have almost equal characteristics, i. e. the same bandwidth and only slightly varying link delays. Therefore, since the overload situation is equally distributed across the whole network and not limited to single links, again the probability of a packet drop statistically increases with each additional link the packet has to travel, penalizing longer routes.

In summary we observe that while Farooq is correct in his claim that OSPF in a default configuration performs worse than BeeHive, a more thorough analysis proves that optimized link costs enable OSPF to partly outperform BeeHive. Furthermore, there seems to be further potential for improving the performance of Adv.OSPF since we did not perform an exhaustive optimization of the link costs. This is also supported by the results in [79] that observed a minimal difference of only three percent between the performance using completely optimized link costs and the general optimum that is possible.

However, due to the fact that the traffic has to be known in advance of the optimization and the NP-completeness of the optimization problem, optimizing link costs might not be possible in every situation. From this point of view BeeHive provides the more efficient routing approach.

4.3.1.3 Evaluation 2 – Realistic protocols and traffic

The simulation environment we used in simulation 1 modelled the network abstractly. No real protocols or traffic profiles were incorporated. While this evaluation provides important first insights into a new protocol and its performance, the question arises if the results can be generalized to real world networks.

Although Farooq analyses in [76] the performance results of a real world BeeHive implementation in comparison to a real implementation of OSPF as well, he evaluates both using only a small network topology and in a situation that clearly favours multipath routing approaches. Thus, as expected, BeeHive clearly outperforms OSPF. Farooq takes these results as proof that BeeHive generally outperforms OSPF. However, we question the generality of these results and analyse the impact of realistic protocols and traffic in this second evaluation.

Simulation settings Again, we analyse the behaviour of BeeHive, UnitOSPF, and Adv.OSPF. However, this time we use our simulation environment that models real Internet protocols, such as IP, TCP, and UDP. Furthermore, we use our predefined traffic profiles that reflect a more realistic traffic mix. Similarly to the first evaluation, we use NTTNet as the network topology and increase the traffic load during the simulations from low to overload, i. e. from left to right in the visualisations.

Results We illustrate the general results of this evaluation in Figure 4.12. The individual diagrams are arranged similar to those of simulation 1. Merely the traffic load if no longer specified by the MSIA but our traffic profile.

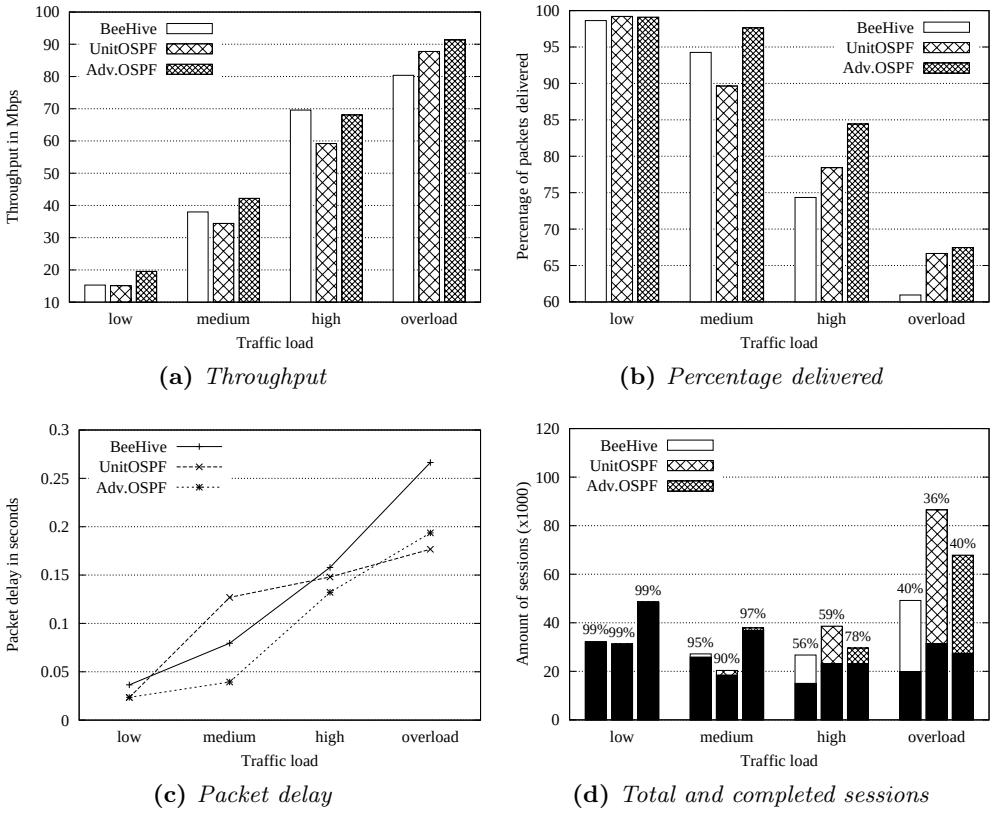
Generally, the results show that the model comprising a realistic protocol stack as well as traffic yields different performance results than from the abstract model, operating with flow-based traffic. Adv.OSPF clearly outperforms BeeHive in every aspect, with the exception of slightly lower throughput during high traffic load. However, we expect this situation to change if we optimize the link costs further.

An interesting results is that BeeHive performs worst in the overload traffic situation. While we see the same effect as in simulation 1 of UnitOSPF showing the lowest packet delay and highest amount of completed sessions, BeeHive falls behind UnitOSPF in terms of throughput and percentage delivered as well, though Adv.OSPF performs still better than UnitOSPF on both accounts.

Furthermore, the average packet delay using BeeHive in the low traffic situation is with 36 milliseconds 13 milliseconds higher than UnitOSPF and Adv.OSPF, which both result in an average packet delay of 23 milliseconds.

We are curious if this performance degradation results from including TCP in the simulation and thus explore the performance results in more detail. With regard to TCP traffic, we take a closer look at the email sessions of the traffic mix. In detail, the email session models a request-response communication, in which a total of 41000 bytes are sent via TCP. Furthermore, the application processing time between the requests adds up to a total of 3 seconds.

Figure 4.13 displays the simulation results regarding the session delay (Figure 4.13a) and the total and completed sessions (Figure 4.13b). Only the delays of finished TCP sessions are considered. Nevertheless, since TCP retransmits dropped packets, the delays of the individual sessions vary significantly. Therefore, we change the visualization from displaying merely the mean session delay to visualizing three aspects of the session delays. The columns represent the range in which the lower 80% of the session delays are located. The median of all session delays, i. e. the delay of 50% of the sessions is lower or equal to this value, are displayed by the horizontal line inside each column. Finally, the maximum session delay is given above each column.



(a) *Throughput*

(b) *Percentage delivered*

(c) *Packet delay*

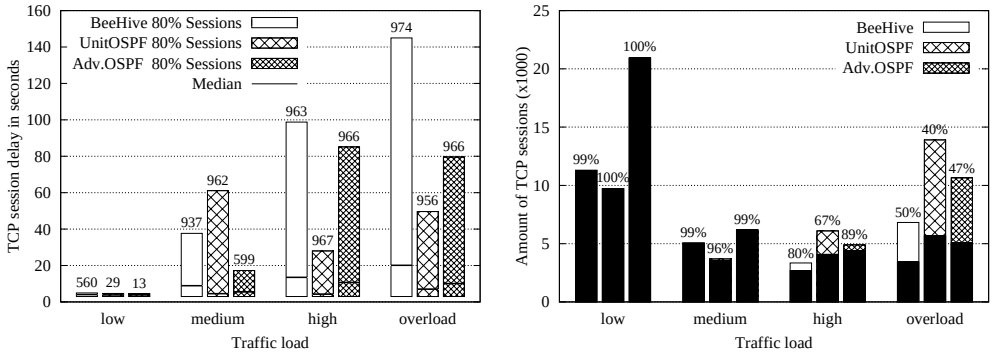
(d) *Total and completed sessions*

Figure 4.12: *Simulation 2 results*

On first inspection, it is striking that remarkably high session delays up to 974 seconds occur. So, before delving into discussing the results in detail, we would like to spend a moment explaining the underlying reasons for this.

If a packet is dropped, TCP waits until the retransmission timeout (RTO) timer expires. The initial value is set to 1 second. This complies to the recommendation in [238], in which the author analysed in detail the influence of the initial RTO on TCP's performance. After each packet drop the RTO is doubled, following the specification of the TCP retransmission timer in [192]. The maximum RTO is defined to be 240 seconds in the simulation environment, following [35]. Finally, the maximum number of retransmissions is set to 12. Therefore, in the worst case, i. e. if a packet is dropped repeatedly for 12 times, the RTO delay alone adds up to 1215 seconds. To this, the transmission, queueing, and processing delays on the path to the packet's destination have to be added as well, resulting in potentially long-running sessions.

Generally, several causes for packet retransmission exist. Especially when using multipath routing schemes, the TCP-internal measurement of the round trip time (RTT), needed to set the RTO appropriately, is never accurate due to the different paths used. If the delays of different paths differ significantly this may lead to the premature timeout and retransmission of the packet that travels on



(a) *Email session delay* **(b)** *Total and completed email sessions*

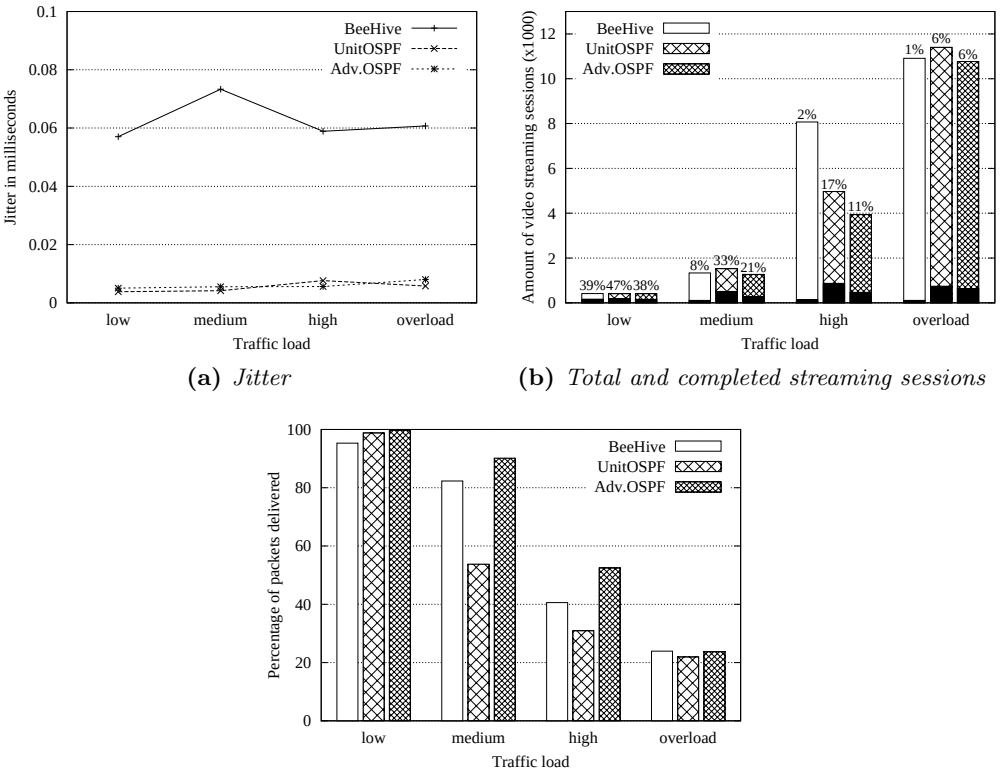
Figure 4.13: *Simulation 2 detailed TCP session results*

the long-delay path [287]. Furthermore, since NTTNet comprises links with a bandwidth of only 6 Mbit/sec, packet drops occur frequently. Thus, high session delays may occur at times.

Still, it is remarkable, that even with a low traffic load, the maximum session delay takes up to 560 seconds for BeeHive, compared to 29 and 13 seconds for UnitOSPF and Adv.OSPF respectively. This could be an indication of the RTT problem described above. However, these extremely high values are the exception. The median of all delays using BeeHive is 4.0 seconds, which is almost as good as the median of 3.8 seconds for both UnitOSPF and AdvOSPF. However, with increasing traffic load, the median using BeeHive increases faster than using UnitOSPF and Adv.OSPF. Eventually, in the overload traffic situation, the median using BeeHive is twice as high as the median using Adv.OSPF and almost three times as high as the median using UnitOSPF.

Thus, including TCP seems to be at least in parts responsible for the performance degradation of BeeHive, beginning with a high traffic load as well as the higher average delay in low traffic situations. One reason behind this is that BeeHive is able to handle less sessions during the simulation time than its competitors due to the longer session times. We do not generate a new session at a fixed interval on each node, but model a number of applications, that start a new session at random during a specified time interval, after their current session is finished, either successfully or broken.

To validate these findings, we examine the UDP traffic in more detail as well. Furthermore, we are interested in the performance both routing approaches show regarding time critical streaming traffic. Thus, we take a closer look at the streaming application from our traffic mix, which models the streaming of a high-quality compressed audio stream. In detail, it generates a constant bit rate stream of 96 Kbit/sec. At this, 256 byte UDP packets are sent every 21.3 milliseconds. The total size of the stream is chosen randomly between 512 KB and 2 MB. Furthermore, we model a 0.5 second buffer to account for jitter.



(a) *Jitter*

(b) *Total and completed streaming sessions*

(c) *Percentage of packets delivered per session*

Figure 4.14: *Simulation 2 detailed streaming session results*

Thus, if a the next packet in the stream is not received after 0.5 seconds, we consider the stream broken.

Figure 4.14 illustrates the detailed simulation results for the streaming application. Figure 4.14a clearly shows that the multipath strategy of BeeHive results in significantly higher jitter values compared to UnitOSPF and Adv.OSPF. Clearly the multipath routing strategy of BeeHive takes its toll regarding jitter. However, despite being higher, the jitter is still acceptable for our streaming application.

More amazing are the low values for the completed streaming sessions, illustrated in Figure 4.14b. Even during low traffic, no approach allows for even half of the started streaming sessions to successfully transfer all packets. However, since NTTNet is generally low equipped regarding the links' bandwidth compared to today's standards, packets are likely to get dropped. Especially streaming applications suffer from this, since they have rather high requirements for the network to work properly. Still, despite packet drops, the received stream might still be useable.

Therefore, we illustrate in Figure 4.14c the percentage of the packets that are delivered on average per stream. During low network load, the streams

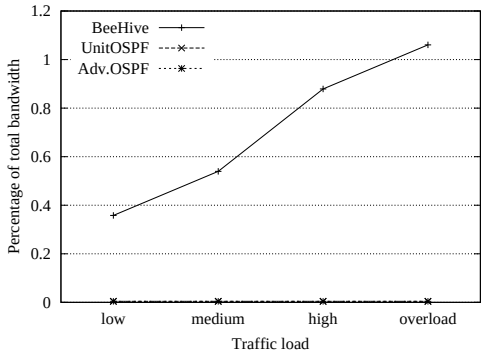


Figure 4.15: *Simulation 2 routing overhead*

should generally be useable, though the quality is worst when using BeeHive as the routing algorithm. In medium traffic load situations only BeeHive and Adv.OSPF allow for usage of the streaming application. Although the audio quality suffers noticeable with packet delivery rates from only 80 to 90%, the usage of packet loss concealment rechniques, e. g. [194], might still provide a limited useable audio stream. We seriously doubt that a stream with less than 60% of the packets available is of any use to the receiver. We therefore do not evaluate the details of the streaming results further.

However, since BeeHive's performance regarding packet delivery is better compared to UnitOSPF beginning with medium network load, we take this as an indication that indeed TCP seems to be responsible for the overall performance degradation of BeeHive.

We conclude our second evaluation with the analysis of the routing overhead, i. e. the percentage of the total available bandwidth that each approach requires for its routing messages. The results are displayed in Figure 4.15. Due to the fact that the network in this simulation is error-free, OSPF becomes rather quiet after the initial setup of the routing tables. The default Quagga configuration for retransmitting LSAs is every five seconds. However, in BeeHive every node sends a bee agent every second, resulting in a significantly higher bandwidth that is required. Furthermore, the graph for BeeHive is increasing over time, which results from the fact that in addition to the bee agent that is sent every second, another bee is generated after a certain amount of data packets has been generated at the node. Thus, increasing traffic results in increasing routing overhead in BeeHive. However, the required bandwidth of BeeHive is generally acceptable in our case since it generally requires less than 1% of the total available bandwidth, with the exception of the overload situation. Still, if BeeHive should be employed in future WANs that will have much higher link bandwidths (even today the German research network XWin comprises some links that have a bandwidth of 1 Tbit/sec) and are thus able to handle much more traffic, the effect of this behaviour should be further evaluated.

In summary, we note that the results of the abstract simulation model are not transferable one-to-one into real world networks. Especially the TCP-based traffic seems to have a negative impact on BeeHive's overall performance. Generally, Adv.OSPF outperforms BeeHive in all aspects. Furthermore, UnitOSPF shows better results than BeeHive in overload traffic situations.

4.3.1.4 Evaluation 3 – Network failures

Our final evaluation targets mainly the robustness of the routing algorithms since Farooq excluded OSPF from these evaluations. Furthermore, in order to validate the previous results, we want to compare the behaviour of the two routing approaches on a second network topology with different characteristics.

Simulation settings We have chosen the network topology NSFNet (cf. Figure 4.9) for this simulation for two reasons. First, it represents in contrast to NTTNet a smaller topology with significantly less alternative paths between any two nodes. We are mainly interested in the question if BeeHive still outperforms UnitOSPF in a network topology that is not heavily interconnected. Furthermore, although NSFNet is widely considered in the evaluations of nature-inspired routing approaches, Farooq did not include it in his evaluations.

We use NSFNet in our realistic simulation environment, in which we generate the traffic utilizing our four traffic profiles, that increase the network load from low to an overload situation.

In detail, we evaluate the characteristics of BeeHive, UnitOSPF, and Adv.OSPF in three scenarios:

1. For reference and in order to be able to compare the results to the second evaluation, we analyse the approaches' performance in an error-free network.
2. In the second scenario we introduce a failure situation. At $t_1 = 300$ seconds router 6 fails and restarts at $t_2 = 600$ seconds.
3. In the third scenario we aggravate the failure situation further. In addition to router 6 we let router 10 fail at $t_3 = 400$ seconds, which almost partitions the network. Router 10 restarts similarly to router 6 at $t_4 = 600$ seconds.

Figure 4.16 illustrates the locations of the failing routers in NSFNet.

Results We illustrate the results of the first, error-free simulation scenario in Figure 4.17. It is instantly obvious that the characteristics of NSFNet are problematic for BeeHive. Generally, Adv.OSPF again outperforms BeeHive in every performance aspect. Furthermore, except for the low traffic situation, where BeeHive performs slightly better than UnitOSPF regarding throughput and

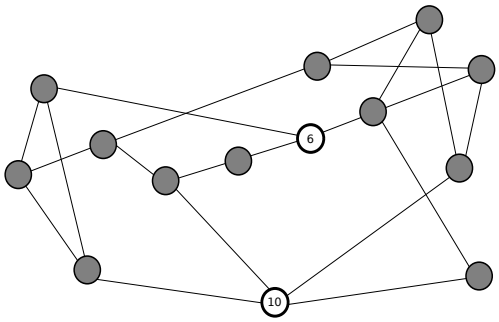


Figure 4.16: *Failing routers in the NSFNet topology*

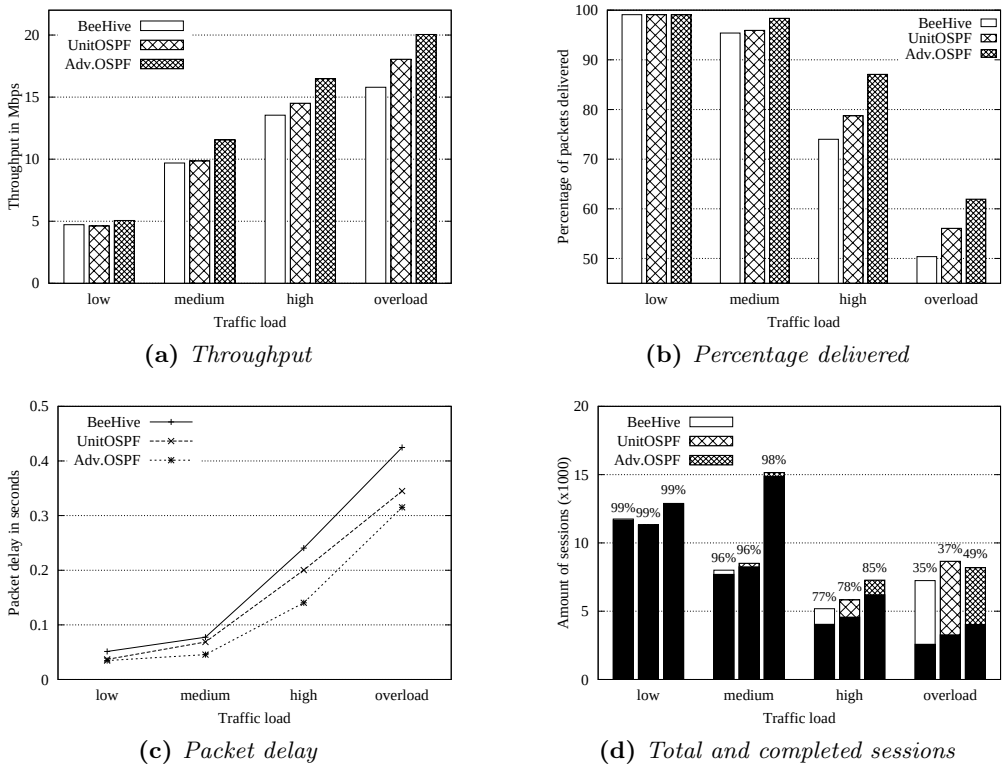
the completed sessions, it performs worst in every other traffic situation. Especially with increasing traffic load the performance gap becomes wider between BeeHive and the two OSPF variants.

We believe this is a direct consequence from NSFNet's topological structure. While in NTTNet usually several only slightly different paths or sub-paths exist between two nodes, NSFNet is marked by the fact that although several paths exist bestween any two nodes, mostly only one good path sticks up. Therefore, the probabilistic routing decision of BeeHive which spreads traffic over all available paths has a negative impact on the performance in this setting.

The same general trends are apparent in the results for the two failure situations as well. Though the performance of BeeHive suffers more from the network failures. Especially in failure situation 2 it performs worst of all candidates regarding all performance aspects. We visualize the results in Figure 4.18. At this, we display the results for failure situation 1 on the left and directly oppose the respective results of failure situation 2 on the right side.

In order to evaluate how fast OSPF and BeeHive react to the failure situations, we illustrate the throughput per second on the application layer in Figure 4.19. Each row displays a traffic profile, increasing from low to overload. Furthermore, the left column displays the results in failure situation 1, while the right column displays failure situation 2. To increase readability, we include at this point only the graphs for BeeHive and UnitOSPF. Furthermore, the diagrams include for reference the throughput graphs of BeeHive and OSPF in the error-free network, which are denoted by their names. The graphs in the failure situations on the other hand are marked with a star, e. g. UnitOSPF*.

Both approaches react instantly to the failure as well as the re-entry of the router. Generally, the time each algorithm needs to notice the router failure and re-entry depends on its configuration. Both recognize a failing router when no answer is received for four consecutive hello packets. Consequently, the required time is determined by the frequency of the hello packets, which is configurable in both approaches. In the simulation, both are set to 1 second, which explains the equal behaviour in this regard.



(a) *Throughput* **(b)** *Percentage delivered*

(c) *Packet delay* **(d)** *Total and completed sessions*

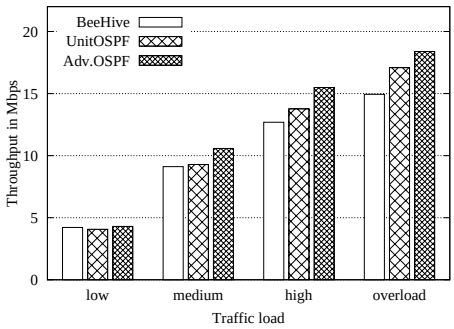
Figure 4.17: *Simulation 3 error-free network results*

Furthermore, it shows clearly that BeeHive handles the first failure better than OSPF in low and medium traffic situations. However, this reverses in high and overload traffic situations, which supports our earlier observation that OSPF outperforms BeeHive in overload situations. During the second failure, BeeHive's performance generally falls short of UnitOSPF's performance.

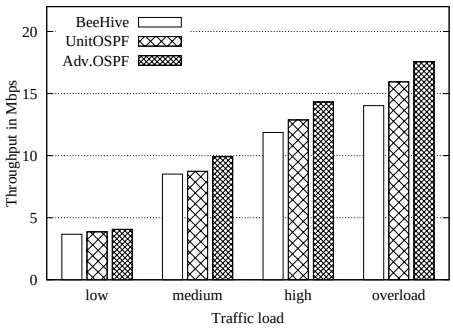
After the failures, the performance of each approach recovers quickly and evens out approximately at the level of the error-free performance curve. The only exception constitutes BeeHive in failure situation 2 with a low traffic load, where the throughput stays for unknown reasons significantly lower compared to the error-free curve.

Finally, the routing overhead results are similar to those of evaluation 2 (c.f. Figure 4.15) which is why we abstain from a graphical representation. The overhead of OSPF centers around 0.01% of the total available bandwidth, while BeeHive's overhead increases gradually from 0.5 to 1.6%.

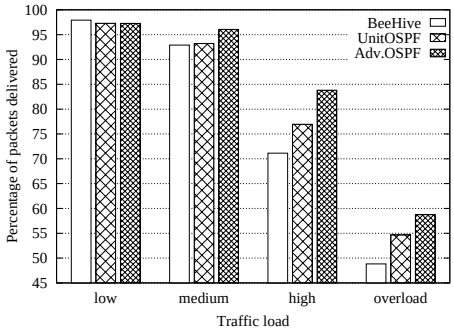
In summary, we record that both approaches show robust behaviour regarding network failures. While the performance is understandably lower during the failure situation, it quickly recovers after the failure has been fixed. Furthermore, BeeHive's performance suffers if the network topology does not comprise



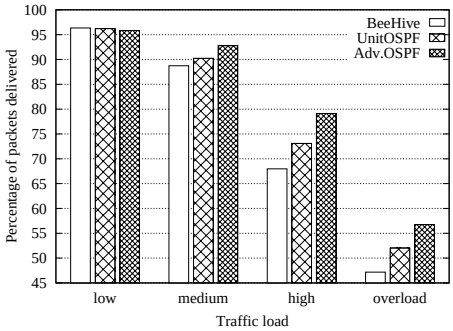
(a) *Throughput (fail 1)*



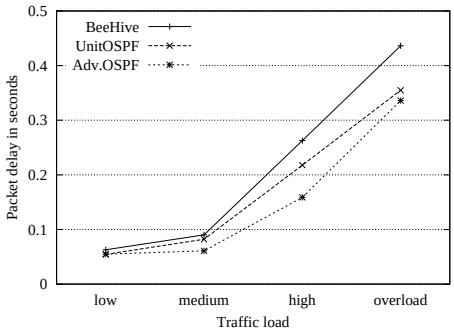
(b) *Throughput (fail 2)*



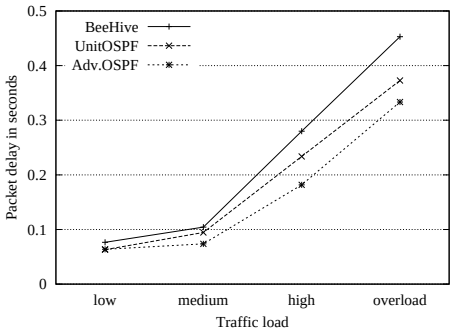
(c) *Percentage delivered (fail 1)*



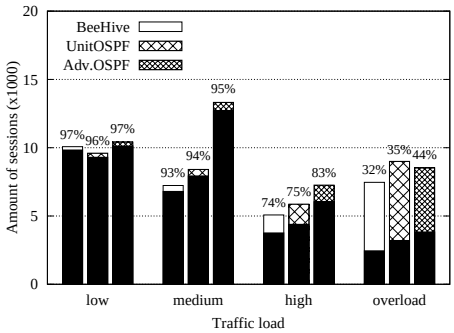
(d) *Percentage delivered (fail 2)*



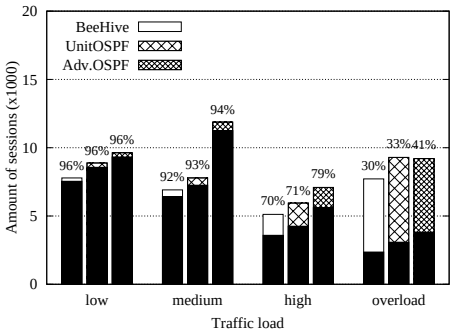
(e) *Packet delay (fail 1)*



(f) *Packet delay (fail 2)*

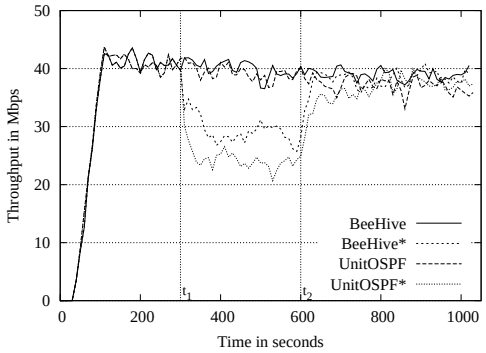


(g) *Total and completed sessions (fail 1)*

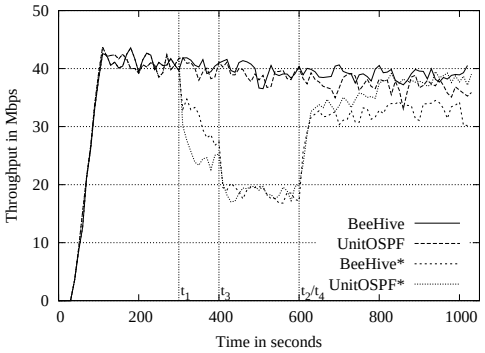


(h) *Total and completed sessions (fail 2)*

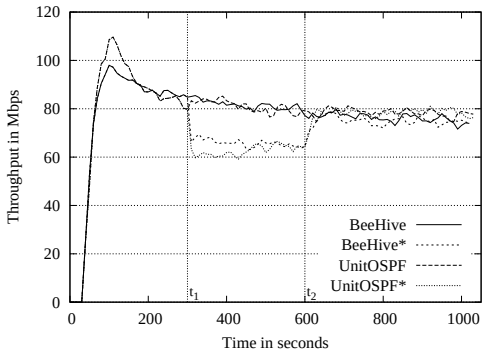
Figure 4.18: *Simulation 3 failure situations results*



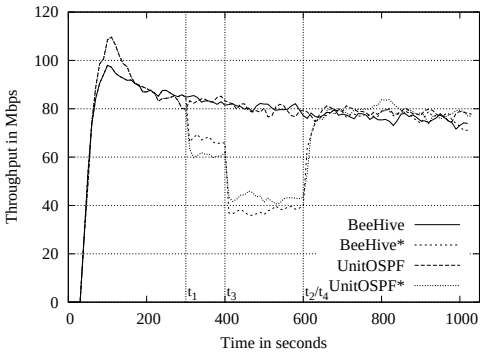
(a) *Failure 1, low traffic*



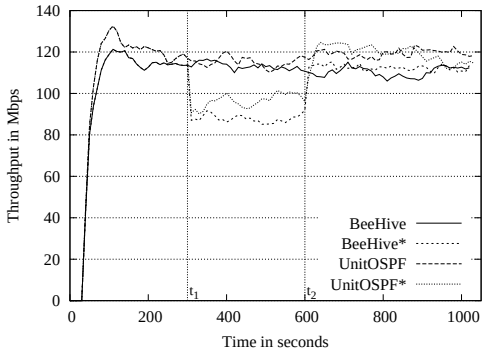
(b) *Failure 2, low traffic*



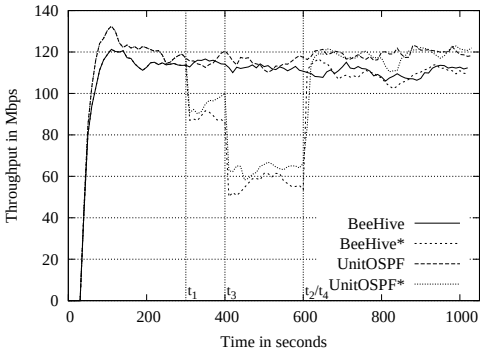
(c) *Failure 1, medium traffic*



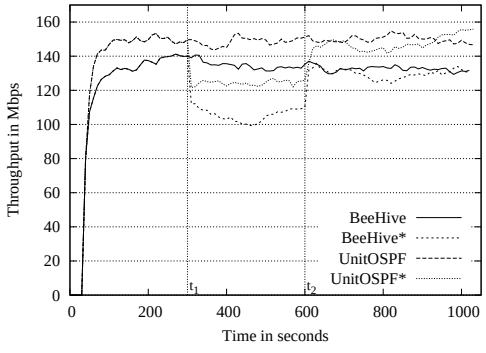
(d) *Failure 2, medium traffic*



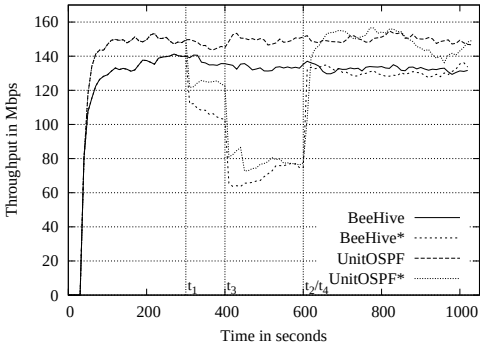
(e) *Failure 1, high traffic*



(f) *Failure 2, high traffic*



(g) *Failure 1, overload traffic*



(h) *Failure 2, overload traffic*

Figure 4.19: *Simulation 3 throughput per second during failure situations*

several alternative paths between the nodes that are similar in terms of delay and hop count.

4.3.1.5 Summary

We evaluated three hypotheses in the performance evaluations, regarding which we compared the performance of OSPF and BeeHive. In summary, the results show that a realistic configuration of OSPF using optimized link costs results in better or at least equally good performance results compared to Beehive. Thus, especially Farooq's assumption that BeeHive always outperforms OSPF is invalidated from a purely performance oriented point of view.

Furthermore, we showed in evaluation 2 that the performance results using an abstract simulation model are not representative for a real world network. In particular the inclusion of TCP-based traffic has a negative impact on BeeHive's performance results. Thus, a more thorough analysis of BeeHive's performance regarding TCP traffic in order to find measures to improve it is advisable. Although several research proposals exist that adapt TCP's behaviour to meet the requirements of multipath routing schemes, e. g. [148,287], from a pragmatic point of view, changing the multipath routing scheme to perform well with the widely deployed standard protocols increases the realistic applicability of these approaches. It is highly unlikely that widely deployed protocols, such as TCP will be changed significantly or replaced completely in the near future.

With regard to streaming traffic, BeeHive performed generally worse than Adv.OSPF. However, concerning the applicability of BeeHive in future networks, this is of special importance. In the generally ongoing process of network convergence, telecommunication providers increasingly substitute their traditional circuit-switched telephone systems with VoIP solutions. Furthermore, more and more comprehensive all-round packages are offered, including, in addition to telephone and internet services, television over data networks (IPTV). Therefore, the importance of good performance results regarding streaming traffic is growing. This correlates with the latest forceasts about the development of global IP traffic. For example, Cisco's June 2009 forecast estimates a 367% compound anual growth rate for multimedia traffic in the timeframe 2008 to 2013. Furthermore, by the end of this time period the sum of all forms of video traffic are estimated to account for over 91 percent of global consumer traffic [54]. In essence, we provided only the first step in the evaluation and the algorithms' characteristics regarding streaming content should be analysed further.

Focussing on VoIP, Bohge and Renwanz present in [32] for example a framework that allows for a realistic qualitative evaluation of VoIP sessions. Essentially, they provide the possibility to feed arbitrarily encoded sound files into an OMNeT++ VoIP traffic generator and measure the quality of the received sound file using the ITU recommendation for perceptual evaluation of speech quality (PESQ) [256], which has been developed specifically to analyse end-to-end voice

quality in real world telecommunication networks. An analysis of BeeHive and OSPF using this framework would yield a more detailed assessment regarding VoIP quality.

Similar future work in the context of evaluating the streaming quality may include the integration of objective and subjective perceptual evaluation of multimedia video quality (PEVQ) as specified in [257] and [258] respectively.

A general lesson we learned during the evaluations is that using simply the shortest path for the routing decision is advisable in situations where the network is overloaded. Especially the results of evaluation 2 account for this. It is not clear if situations where this holds can be described in a way that allows for the routing algorithms to switch their operations to shortest path routing accordingly. This is an interesting open question for future research.

Finally, both routing approaches show robust behaviour with regard to network failures.

Advisable future work includes a detailed analysis of both algorithm's scalability characteristics as well as a performance comparison using state-of-the-art network characteristics. We did not include this in the evaluation at hand due to the respective enormous CPU and memory requirements of such a simulation. Especially with regard to the discussed UC characteristics this is indispensable.

Furthermore, we did not include in our analysis more elaborate traffic engineering techniques that are used in OSPF-based networks today. Especially, the usage of Multiprotocol Label Switching (MPLS) allows to choose the shortest path to a destination on the basis of additional metrics than solely the link costs, e. g. the currently available bandwidth of the paths. It relies on the use of the OSPF traffic extensions [131], which allow in addition to bandwidth considerations the definition of other special link or tunnel attributes [60]. A comparison of BeeHive and MPLS on top of OSPF would be interesting.

4.3.2 Operational Integrity

Routing is a critical part in networking. If the routing system is disrupted either by malicious intent or by accident, the network may become useless. Therefore, in addition to good routing performance to guarantee the network's availability and its reliable operation in spite of possible network failures, the operational integrity of the routing system as a whole must be ensured as well.

Generally, routers incorporate the information inside every routing message they receive as they put unconditional trust in the sender and the message content. Thus, without mechanisms to protect and verify the integrity and authenticity of the exchanged messages it is trivial to manipulate the routing operation externally, e. g. by sending forged or modified messages to the routers. Since these messages can be sent from any host inside the network the attack potential is significant. Furthermore, a misconfigured router that is started up deliberately or by accident can bring an entire network to a standstill by

reporting incorrect route information to the other routers. Therefore, in order to provide a serious solution for real world deployment a routing algorithm must include or allow for external mechanisms to protect its operation from deliberate or inadvertent unauthorized manipulation.

In the following we first set the boundary of our analysis by giving an overview of relevant attacks in the routing context and specifying the attacker's capabilities. We then evaluate which security measures are available for both protocols as well as the extend to which they provide the necessary protection for the routing system with regard to the described attacks.

4.3.2.1 Attacks

We consider attacks on routing algorithms on several levels: the packet level, the information level, as well as the protocol level. Although the attack potential for the router itself, i. e. the machine running the routing algorithm, is quite significant, as it provides a possibility for an external attacker to become an insider, such attacks are outside the scope of this dissertation. Furthermore, we do not consider vulnerabilities and the resulting attack possibilities that arise from faulty implementations of the routing algorithms.

As a foundation we assume a Dolev-Yao attacker who is able to execute the full range of active and passive attacks. Moreover, without applying security measures, outsiders are given the same active attack possibilities as insiders. They can attack the routing on all abstraction levels at will. For example, at packet level they may monitor, drop, delay, reorder, or replay the routing packets sent by legitimate routers. Furthermore, they may execute DoS attacks, e. g. by flooding a router with bogus packets. On the information level they may for example modify valid routing packets, injecting fake routing packets into the system by masquerading as legitimate router, or pose as a new router, distributing arbitrary routing information.

Additionally, an attacker may execute protocol specific attacks. For example, three insider modification attacks that target the OSPF protocol are described in [52, 270]. Here, some of the features designed to make OSPF more robust, especially with regard to the handling of LSAs, can also be exploited by an insider to attack the routing operation.

Seq++ attack. Each LSA has a sequence number to distinguish if an LSA is newer or similar to an already known LSA. An attacker may change the routing information inside an LSA he received, recalculate the checksums, and increase the sequence number before flooding it again. As a consequence all other routers will incorporate his falsified information as they believe the LSA to be newer. Eventually, the bogus LSA will reach the actual originator, which will respond with a new LSA that contains the correct information. If the attacker keeps up his attack, this leads to an unstable network topology, where the routing information constantly changes.

Maximum age attack. The LSA age provides a mechanism to delete old information from the system. If an attacker modifies the age of a received LSA to MaxAge (default setting is 1 hour in OSPF) while keeping the sequence number intact and re-injects this LSA into the network, it causes all other routers to delete the corresponding LSA from their database. Eventually the originator of the LSA receives the bogus LSA and responds with a correct one. Similar to the seq++ attack, this results in an unstable network topology if the attacker continues his attack.

Maximum sequence number attack. The LSA sequence number represents a 32 bit value. If an attacker modifies the routing information and sets the sequence number of a received LSA to the maximum value, it will be considered the "freshest" by all other routers. The originator has to take two steps in his response: first purge this LSA by generating an LSA with MaxAge set and then generate a new LSA including the correct information and reinitialized sequence number. Again, the effects are similar to the seq++ attack if the attacker keeps his attack up.

Although these three attacks may result in an unstable topology, we think them unlikely to be executed, since an inside attacker who, for example, floods the network with arbitrary routing information will attain an unstable topology situation at a much faster rate.

The attacker's motivation for attacking the routing can be very diverse. For example, changing the packet flow may be beneficial in order to enable him to monitor data packets he would otherwise not have access to. Another reason may be the resulting higher monetary costs for the provider or decreased network performance. Destructive minds may simply want to damage the routing by creating unstable topologies, e. g. by flooding the network with arbitrary routing messages.

Further discussions about attacks on routing, including those that are outside the scope of this analysis, can be found in the literature, especially [17] and [108] provide good overviews.

4.3.2.2 Security Measures

Security measures that protect the routing against the mentioned attacks can take various forms. Generally, we focus on techniques that protect the integrity and authenticity of the routing. At this, we differentiate the approaches along two dimensions. We describe and evaluate proactive approaches that are based on cryptographic schemes and reactive approaches that derive from IDSs. Furthermore, we evaluate each with regard to the level on which it provides security, the packet level, the information level, or the protocol level.

Establishing security measures to ensure integrity and authenticity limit the possible attacks for outsiders and insiders. Security on the packet level narrows the possible active attacks for outsiders down mainly to replaying packets and

DoS attacks. Furthermore, they may drop, delay, or reorder the packets, if this is physically possible. However, packet level security does not restrict insiders. Defenses against inside attacks have to be established on the information level.

OSPF Security OSPF includes proactive security measures on the packet level to achieve authenticity and integrity of the exchanged routing messages [176], although the standard refers mainly to authenticity. For this purpose, the OSPF packet header includes an authentication type field, and a 64-bit data field, which the different authentication schemes may use to store their data. Three authentication types are specified in the standard: null authentication, simple password, and cryptographic authentication.

Null authentication essentially disables this feature. It provides no security against any of the mentioned attacks.

Simple password authentication specifies that a 64 bit password is included in every message, albeit in clear text. This guards against accidental threats, such as routers inadvertently joining the routing domain, since each router has to be configured with the correct password before it may participate in the routing. However, simple password authentication does not provide any security against attacks, since it is vulnerable to passive attacks. An attacker who is able to capture a routing message instantly learns the password and is consequently able to actively attack the routing.

Cryptographic authentication provides the only mode that allows for message authenticity and integrity. It specifies that a shared secret key must be configured in all routers participating in the routing. This key is used to generate and verify a message digest that is appended to the end of the packet. The message digest is the result of a one-way function with the message and the secret key as input. Since the secret key is never transmitted over the network, an attacker is consequently not able to learn it by capturing a packet, although no confidentiality is provided. Furthermore, since our Dolev-Yao attacker is not able to break cryptographic measures, this authentication mode provides the required security on the packet level and consequently protects the routing operation against external attacks.

Additionally, replay attacks are impeded by integrating a non-decreasing sequence number in the packet. Although it is possible to replay each packet until the sequence number changes, this still provides long-term protection against replay.

From a technical point of view, the OSPF standard specifies only the MD5 algorithm [216] for use with the cryptographic authentication, which has been reported successfully broken in late 2008 [262]. Thus, it should not be used anymore.

While cryptographic authentication theoretically protects the message during transport between neighbouring routers, it provides no security for the included routing information, i. e. on the information level. Any router may alter the in-

formation inside the LSA before sending it on. Thus, protection against internal attackers is out of the scope of this measure. To remedy this problem, Murphy et. al propose as an extension to OSPF the application of digital signatures [182]. They hereby address proactive end-to-end authentication and integrity on the information level, protecting the actual routing information that is flooded via routers which may themselves be faulty or compromised.

Their basic idea is to add digital signatures to LSAs that are flooded through the network while keeping the neighbor-to-neighbor authentication algorithm on the packet level, such as the keyed MD5 which is proposed in OSPF's cryptographic authentication. By signing each LSA the originator is always identifiable and end-to-end integrity and authentication for LSA data is ensured.

As a prerequisite to using this security extension the public keys of all legitimate routers have to be made available inside the network. A new LSA type (Public Key LSA) is defined for this purpose, using the standard OSPF flooding mechanism to distribute the public keys. In order to assure the identity of the router and its public key, a trusted third party, the so-called CA, generates for this purpose certificates for all routers. Each router is configured with the CA's public key and thus able to verify each certificate that contains a valid key. Hence, if a router receives a new public key, it can verify the signature inside the certificate to assure the identity of the sending router. Finally, successfully verified public keys are used to verify the origin and integrity of LSAs.

Murphy et. al specify the RSA/MD5 algorithm [121] for generating and verifying the digital signature, although in principle other asymmetric algorithms may be used as well. Furthermore, they require signature and key formats according to the specification in [70].

From a security point of view, this proposal provides the greatest security for OSPF. It provides, in addition to the transport security ensured by the cryptographic authentication, end-to-end authenticity and integrity for the actual routing information. Therefore, the impact of faulty or compromised routers inside the network is reduced significantly. They can still distribute incorrect information, delay, drop, or reorder other routing messages, but cannot alter other router's information. Furthermore, if faulty routing information is detected, the digital signature pinpoints exactly the responsible router.

A drawback of this approach is the necessary central CA. A successfull attack on the CA renders the whole approach insecure. Furthermore, we are not aware of any publication that quantifies the performance loss that results from using digital signatures in detail. However, asymmetric cryptography is generally up to 1000 times slower than symmetric cryptography [233]. Therefore, we expect a significant performance loss in LSA generation and verification, compared to standard OSPF operations.

In addition to the discussed proactive security measures, the JiNao distributed intrusion detection system is presented in [52, 122]. It provides reactive security at the protocol level by stateful protocol analysis. For this purpose, JiNao

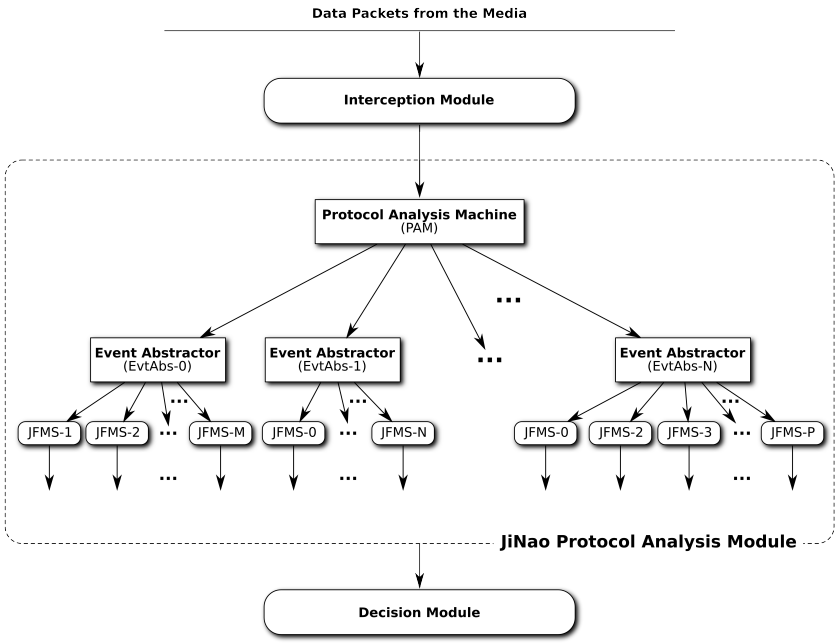


Figure 4.20: *JiNao protocol analysis model architecture*

includes a protocol analysis module that is able to perform a real-time protocol analysis based on a timed finite state machine that represents the real-time behaviour of OSPF.

Figure 4.20 illustrates the protocol analysis module. Each intercepted routing packet is first given to the Protocol Analysis Machine that determines its type and forwards it to the appropriate Event Abstractor. Each event abstractor is responsible for a certain type of LSA and performs some global checks, such as checking the LSA checksum. Furthermore, it comprises of several JiNao Finite State Machines (JFSMs) that analyse the LSA further, once the global checks succeed. Each JFSM models a certain pattern, such as a known attack or normal behaviour. Based on the event string they receive, their internal state machine advances or stops. Eventually, one JFSM should finish its state machine and report its findings. If no JFSM finished, the protocol analysis module concludes an unknown attack and reports the event sequence.

JiNao furthermore contains a prevention module acting like a firewall. If corrupt LSAs are found, they are dropped and not passed on to OSPF.

The authors have analysed the protocol analysis module in [52] by constructing JFSMs for the three OSPF protocol attacks seq++, maximum age, and maximum sequence number. Their experiments verify that all three attacks are successfully detected. Furthermore, unknown attacks can be detected if JFSMs for normal behaviour are specified. Compared to proactive security measures, JiNao protects the routing against insider attacks, without the extra cost of

for example digital signatures. However, the authors of JiNao did not evaluate other attacks, such as a router masquerading as another router. Furthermore, they did not elaborate on the complexity of constructing a JFSM that reflects normal behaviour.

All publications considered, security in the context of OSPF is a widely discussed topic in the literature. Since the late 1990s several proposals to secure OSPF were published, in addition to the approaches we discussed above, e. g. [97, 109, 110, 292]. We focused in our descriptions on what we believe to be the most significant ones, since none of the other proposals seems to have made a lasting impact. None have been embraced by the Internet community and resulted in an Internet standard, which is mostly the basis on which vendors include functionality in their products. Furthermore, security for OSPF is still considered in parts an open problem [170] and is also still subject to active research, especially within the IETF. For example, Bhatia et. al are currently enhancing the cryptographic authentication of OSPF in the Internet-Draft [26] by adding support for the algorithms defined in the NIST Secure Hash Standard (SHS) [266] using the Hashed Message Authentication Code (HMAC) [265] mode. Presumably this is done due to the discovered insecurity of MD5.

Although we focus on OSPFv2 in this dissertation, we still want to mention how the security considerations for OSPF will presumably change in the long-term perspective. With the increasing adoption of IPv6 [62] OSPFv3 will replace OSPFv2. All security measures have been removed from OSPFv3 as it relies on the security services mandatory in IPv6 networks. So in order to ensure integrity, authentication, and confidentiality of the routing exchanges, the IP authentication header [133] and the IP encapsulating security payload [134] are utilized as described in [92].

BeeHive Security The BeeHive protocol does not comprise any security mechanisms. However, proactive as well as reactive security extensions have been proposed.

Wedde et al. specify in [278] BeeHiveGuard a proactive security extension to BeeHive that is based on standard cryptography. It utilizes RSA [219] to generate digital signatures. One signature is used to protect the integrity of the agent, i. e. it cannot be modified and impersonated. Additionally, two further signatures are used to protect the integrity of the routing information which the agent carries, i. e. the propagation delay and the queuing delay. Thus, BeeHiveGuard comprises packet level security as well as information level security.

Wedde et al. analysed their approach by simulation. Especially they analysed active attacks, such as modification, masquerading, flooding, and dropping attacks. The experiments reveal that BeeHiveGuard protects BeeHive against all considered attacks. However, due to the extensive usage of digital signatures the operational costs for BeeHiveGuard cause significantly worse performance results compared to BeeHive without security extensions. The authors state that

the average processing cost for the agents amount to 52100% up to 160400% compared to standard BeeHive. Furthermore, the communication costs amount to 133% up to 1390% compared to BeeHive.

Due to this unacceptable performance degradation, Wedde et al. come to the conclusions that new security mechanisms need to be devised to efficiently protect nature-inspired routing algorithms. The same hypothesis was also published beforehand in [294].

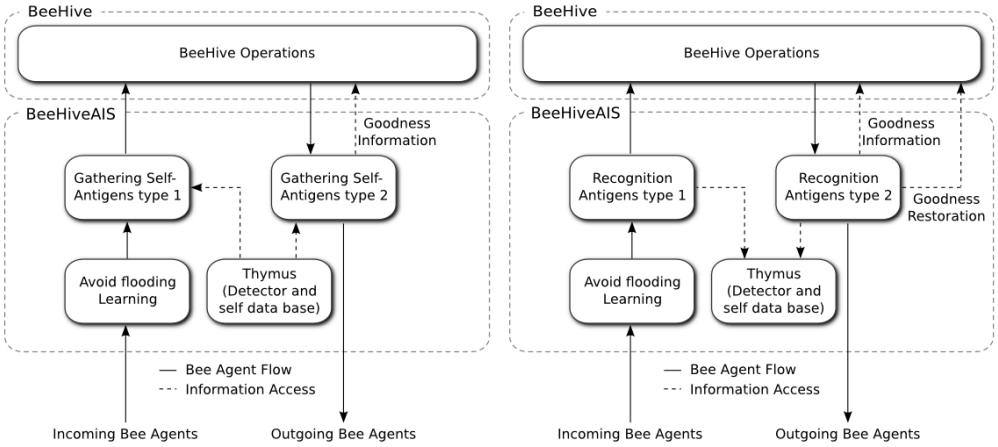
BeeHiveAIS [274] provides a framework to detect and handle threats to BeeHive in a reactive manner, based on the concepts of an *Artificial Immune System* (AIS).

AIS represent a nature-inspired approach to IDS as they mimic the human immune system in order to detect and defend against known and unknown incidents. Generally, a differentiation between self and non-self cells provides the basis of anomaly detection. With regard to BeeHive routing, self-antigenes are representations of the routing agents' normal behaviour while antigenes (non-self) represent anomalous behaviour. They are detected by antibodies, which are generated randomly on the basis of the self-antigenes. At this, antibodies whose affinity, based on the Hamming distance, to the self-antigenes is above a defined threshold are discarded.

A full description of the interesting AIS research field is out of the scope of this dissertation. However, we recommend the work of Kim et al. to the interested reader, who provide in [136] a good overview of AISs as well as a categorization of most published approaches.

The basic operation of BeeHiveAIS consists of three distinct phases: initialization, learning, and protection. During the *initialization phase* the patters of agent movements, which form the foraging zones and foraging regions are learned. In the *learning phase* typical patterns in the data traffic are learned. It operates under the assumption that at this time no attacker and no malicious node are present in the network. Combined, these first two phases result in a repository of self-antigenes, that reflect the normal situation in the network. These are built in each node and provide the basis for antibodies that reflect the anomalous behaviour and are generated to counter the different threats. Finally, during the *protection phase* the network is monitored and the various security threats are detected and countered through the respective antibodies.

The learning and protection phase are illustrated in Figure 4.21a and Figure 4.21b respectively. Generally type 1 and type 2 self-antigenes are differentiated. Type 1 self-antigenes protect the integrity of the agent. They consist of tuples of (source address, neighbour, hops). Type 2 self-antigenes protect the integrity of the propagation and queuing delays. They consist of the average goodness values of the path to a destination via a specific neighbour. The values are gathered over a sliding window of the delay of five subsequent bee agents, allowing to derive an upper and lower bound to both values. If an antibody



(a) *Learning phase* **(b)** *Protection phase*

Concept		Description
Self-Antigenes	type 1	Correct incoming direction and region of the bee agent
	type 2	Correct propagation and queueing delay tendencies
Antigenes	type 1	Incorrect incoming direction and region of the bee agent
	type 2	Incorrect propagation and queueing delay tendencies
Antibody	type 1	Pattern that can detect antigenes of type 1
	type 2	Pattern that can detect antigenes of type 2

(c) *BeeHiveAIS Concepts*

Figure 4.21: *BeeHiveAIS operation phases*

of either type matches an antigene the corresponding bee agent is dropped. Table 4.21c summarizes the BeeHiveAIS concepts.

In addition to the antibodies, a protection against DoS attacks in form of an upper bound to routing table updates is included in BeeHiveAIS. This essentially defines a minimum time between two consecutive arriving bee agents. An agent arriving before this time will be discarded and the routing table not be updated.

Wedde et al. analysed BeeHiveAIS by means of simulation in comparison to BeeHiveGuard. In detail, they evaluated an impersonating scenario, in which a node fakes his address in order to divert traffic to himself, a tampering scenario, in which a node tampers with the delay values of a bee agent in order to alter the goodness of other nodes, a combined attack, in which several nodes collude in the attack, and a DoS attack. They come to the conclusion that BeeHiveAIS provides the same level of security as BeeHiveGuard while keeping the processing and control overhead 200 and 20 times smaller respectively. However,

they note that further evaluations regarding the scaleability of BeeHiveAIS are required.

One aspect the authors of BeeHive do not elaborate on, is how to ensure that no attacker is present during the learning phase. However, this clearly is a critical point in real world networks. One possibility could be to disconnected the network from outsiders in the beginning. However, it is questionable if BeeHiveAIS is able to learn the patters of realistic traffic in this case. We presume that it rather determines realistic traffic as anomalous after the network has been reconnected. Thus, other ways to achieve this must be identified and evaluated, e. g. shielding the network from attackers somehow or filtering out anomalous traffic, before BeeHiveAIS presents an applicable solution for real world networks.

Generally, BeeHive could also benefit from security that is provided at the IP layer, similar to OSPFv3 that requires the mandatory security measures in IPv6. However, we are not aware of any research in this direction.

4.3.2.3 Summary

Both routing approaches have been subject to security evaluations and several security solutions have been proposed. Based on the results from the respective evaluations, we conclude that both can theoretically be secured on all levels with regard to integrity and authenticity. However, some aspects demand further analysis.

Only the proactive security measures at the packet level, which are included in the OSPF standard have been embraced by the Internet community, despite the fact that numerous approaches to secure OSPF further have been published in the literature. Of these, only the cryptographic authentication provides any reasonable security. Although we cannot say how network providers secure their routing today, when using OSPF, we presume that most providers have experts monitoring the networks closely and reacting to anomalies. This is reasonable, since OSPF was never intended to run completely autonomous. On this assumption, cryptographic authentication may indeed provide a good compromise between provided security and extra operational cost. However, a detailed study of current routing products of vendors, such as Cisco, could shed more light on the security possibilities that are available in today's routing hardware. Furthermore, interviewing big providers might provide some answers, though we deem it unlikely that most will talk freely about such a sensitive topic.

Much higher are the security requirements for BeeHive, since it is supposed to operate completely autonomously. Based on the respective results, BeeHiveAIS seems to provide a quite powerful security measure that fulfills this mission. However, security aspects in BeeHive, including BeeHiveAIS, were so far only analysed by its authors, demanding clearly for further independent evaluations.

4.3.3 Autonomic Behaviour

The evaluation of the approaches' autonomic behaviour is aimed at answering the question if human administrators may be relieved from the increasingly burdensome task of managing the network in the future. Especially, we want to identify if BeeHive represents an improvement over OSPF. Furthermore, in the latter case we also want to differentiate the areas in which this proves true.

The autonomic computing initiative as well as the organic computing research community have established several self-x properties that describe autonomic systems. Generally, we define a self-x property as follows.

Definition 17 (Self-X Property) A self-x property describes the ability of a system to provide a certain capability independently.

Much ambiguity exists, however, with regard to the exact definitions of individual self-x properties, such as self-configuration, self-management, and self-organization, since they are usually defined slightly different within the context of the application domain at hand. A discussion about the different perceptions of self-x properties in different research areas present, for example, Fromm and Zapf in [80].

To avoid ambiguity in the following discussions, we therefore define first those properties that are relevant in our context and afterwards assess the algorithms' characteristics in their regard. Figure 4.22 provides an overview of the individual properties that are important in our context, their operational focus, as well as their relations. Although both represent individual properties, slef-configuration and self-stabelization interact significantly.

In the assessment we include in addition to the actual routing algorithm the security extensions we have described in the last section. Especially, we assume that OSPF has the cryptographic authentication enabled, while BeeHive is protected using BeeHiveAIS.

Generally, if all individual self-x properties are present in a system, they add up to a self-managing system.

Definition 18 (Self-Managing) A system is self-managing if it adapts its operation to meet the characteristics of its environment, without external intervention, thus ensuring its operation even in the face of failures and attacks.

However, since a meaningful assessment of the routing approaches regarding this generic definition is not possible, we divide the analysis into two focus areas: configuration, which includes optimizing it for performance, as well as operational stability, which includes robustness with regard to network failures and defense against attacks. Undoubtedly, interdependencies exist between these two areas. For example, reacting to network failures is usually not possible without being able to change the configuration.

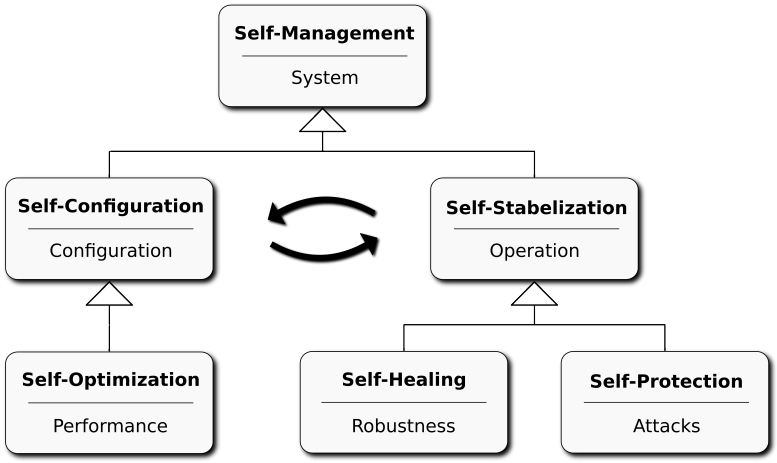


Figure 4.22: *Self-X properties and their focus*

4.3.3.1 Configuration

The configuration of the routing approaches comprises the initial setup of the network as well as the routing algorithm and the configuration of the routing tables.

Definition 19 (Self-Configuration) Self-configuration enables a system to independently adjust its configuration parameters.

OSPF was not designed to build its initial configuration autonomous. Thus, it must be configured with regard to the network structure, i. e. areas must be configured, and link costs must be set, if the default setting of one, which equals a hop-count metric, is not wanted. In order to increase the performance this is advisable, as we have shown in Section 4.3.1.2. Furthermore, several protocol parameters, such as the frequency of the hello packets, the time interval to regard a neighbouring router as down, need to be configured.

BeeHive on the other hand determines the network structure by itself during the formation of the foraging regions. Furthermore, the link characteristics are measured during runtime. No external initial configuration is necessary. However, several protocol aspects of BeeHive have to be specified, equally to OSPF, via parameters, such as the frequency of bee agents or hello packets.

Based on their initial configuration, both approaches establish their routing tables without further external guidance. Both find neighbouring routers by means of hello packets and determine routes to all destinations inside the network.

Regarding security, OSPF again requires the manual configuration of the passwords, whereas BeeHiveAIS learns autonomous. Yet, it requires a time span in which no attacker must be present in the network. However, the authors do not

elaborate on how this might be done. We presume that at present significant manual effort is required to ensure this attacker-free environment.

Definition 20 (Self-Optimization) Self-optimization constitutes continuous monitoring and adaptation of the configuration with regard to optimizing the performance of the system.

BeeHive performs continuous optimization by adapting the goodness values for the individual nodes with regard to the current situation inside the network. Since these provide the basis for the routing decisions, the overall network performance is continuously optimized.

An optimization measure that both algorithms include is the update of the routes if better paths become available. This could happen for example, if a network error has been fixed and the path is again available for routing.

Apart from this, OSPF does not include any further measures to optimize its configuration. Merely the link costs might be updated by an administrator or management system with externally optimized link costs, which will prompt OSPF to update the routes.

Generally, both protocols do not include any measures to omptimize their protocol, such as adapting the frequency of routing messages or bees according to the situation at hand inside the network.

4.3.3.2 Operational Stability

Operational stability characterises the convergence of the network into a stable routing situation, in an error-free network as well as in the face of network failures and attacks on the routing.

Definition 21 (Self-Stabilization) Self-stabilization describes the ability of a system to (1) reach an acceptable state from an arbitrary state in a limited amount of time and (2) to keep this state independently.

Both approaches are self-stabilizing in a sense that the routers find each other and legal routes are determined for each destination, in a limited amount of time. Furthermore, in an error-free environment, the stable routing situation is kept indefinitely.

Important subaspects in the context of self-stabilization regarding failure situations are self-healing and self-protection. We distinguish between them in terms of intentional versus accidental failures. At this, self-healing focusses on accidental network failures, while the focus of self-protection lies on intentional failures, i. e. attacks on the system.

Definition 22 (Self-Healing) Self-healing defines the ability of a system to reach an acceptable state, starting from an unacceptable one, in a limited amount of time.

Both approaches are self-healing. They autonomously detect network failures and adapt the routing entries accordingly. By excluding the failed system or network link, they ensure the operation of the remaining network as far as possible.

Definition 23 (Self-Protection) Self-protection enables the system to detect, identify, and protect itself against attacks, in order to maintain the overall system integrity.

Self-protection at its full extend is not available today. Measures exist, for example, for detecting and identifying attacks, such as checking the message integrity to determine unauthorized modifications or monitoring the system by means of an IDS. However, a completely autonomous response is usually not available, especially reconfiguring the employed security measures, such as to increase the system's overall security or to attain specialized protection against previously unknown attacks that goes beyond detection, is still an open research topic.

Nevertheless, the OSPF protocol contains limited measures for self-protection, originally included for fault tolerance. In the context of attacks on the routing algorithms (cf. Section 4.3.2.1) we mentioned that if erroneous LSAs arrive at the legitimate router, it will send our the correct information to heal this "error"[4]. Furthermore, the usage of the cryptographic authentication could be considered limited self-protection, since on its basis external attacks can be identified and successfully prevented. However, it requires manual configuration to achieve this.

No automatic measures to protect OSPF against insider attacks with good performance are available.

Although the core BeeHive algorithm does not contain any measures for self-protection its authors present with BeeHiveAIS a solution that provides a remedy to this. However, we are not sure about the extend to which BeeHiveAIS is able to operate unsupervised in case of attacks, as only a few have been evaluated. This needs to be evaluated further, ideally independent from its authors. Furthermore, apart from detecting and dropping erroneous packets, it provides no advanced security measures.

4.3.3.3 Problems of Autonomic Behaviour

Generally, increasing autonomic behaviour decreases the possibilities to influence the employed algorithms. In terms of the characteristics of today's networks this may lead to several problems.

For example, OSPF is well integrated into the global Internet structure, being able to redistribute domain-external routes from BGP and vice versa. We are

[4]This situation could also arise due to a misconfigured router, in which case these measures could also be classified as self-healing measures.

not aware of any possibility, manually or autonomic, to interact with BeeHive's operation.

Another interesting aspect represents the specification and provision of guarantees about the network. Today, most customers negotiate service level agreements (SLA) with the service providers, which specify a certain level of service, e. g. a certain amount of bandwidth that is guaranteed to be available for a customer. The autonomic concept of BeeHive clearly clashes with this established process.

Finally, the introduction of an autonomic routing approach yields a certain loss of control about the network. Since no manual intervention is possible, it must be well tested. However, a certain amount of residual risk remains, since it is generally hard to devise a completely error-free design that accounts for all failure situations and attacks.

4.3.3.4 Summary

In summary, both approaches work autonomic during their runtime, in error-free network conditions as well as with regard to failures. However, while BeeHive does not need any initial configuration to operate with good performance, optimizing it continuously, an external initial configuration and optimization is mandatory for OSPF to work efficiently. Furthermore, both approaches may be configured with regard to their internal protocol behaviour, if the default settings are not sufficient. Regarding attacks, some autonomic measures are available for both approaches, though we believe this aspect has to be evaluated in more detail for a proper qualitative assessment.

4.4 Conclusion

In this chapter, we have evaluated the question, if nature-inspired routing approaches have the potential to provide autonomic dependability in network routing. On this account, we briefly surveyed nature-inspired approaches to network routing and have chosen the most promising candidate, BeeHive, for further anaylsis. At this, we gradually evaluated the characteristics of BeeHive in comparison to the state-of-the-art routing protocol OSPF regarding the individual aspects of dependability.

Focussing on availability and robustness we challenged some of the results that were published in the evaluation of BeeHive. In detail we have invalidated the conclusion that OSPF performs generally worse than BeeHive. Furthermore, we have proven that simulation results that were attained in an abstract simulation model are not one-to-one representative for real world networks.

Our results document that a realistic configuration of OSPF, i. e. using optimized link costs, generally outperforms BeeHive. Yet BeeHive performs gener-

ally better than UnitOSPF, with the exception of the overload traffic situation. Looking at the results in more detail has reveiled that BeeHive does not operate optimally with TCP-based traffic. Here further research is needed. At best, optimizing BeeHive in this regard would render changing the well-established TCP protocol unnecessary. However, maybe the step towards UC makes this neccessary.

Additionally, regarding streaming applications our results attest the worst performance for BeeHive's during low traffic situations and generally worse results than Adv.OSPF. Furthermore, the jitter values were significantly higher in Beehive. In the face of the estimated growth of streaming traffic a more detailed evaluation focussing on streaming traffic is unavoidable.

Problematic on account of optimizing the link costs are the facts that the optimization is generally NP-hard and the traffic has to be known in advance. However, under the assumption that real world network providers have detailed knowledge about their network and the traffic demands, it it reasonable to assume, that today optimized link costs are utilized in general. Still, regarding future UC networks this will probably change, due to the increasing complexity. Thus, the question arises if measures can be found that allow for an autonomic online optimization of the link costs. Furthermore, it is well worth evaluating additional traffic engineering approaches that operate on top of OSPF, such as MPLS. A further evaluation of BeeHive in comparion to these is therefore advisable.

Finally, we have shown that BeeHive and OSPF react almost equally good to network failures, though OSPF provides on average the better perfomance during the failure situation.

The analysis of the operational integrity has reveiled that, although several approaches have been proposed to secure the operation of OSPF, only the basic measures providing hop-based integrity and authenticity for the routing messages seem to have made an impact. Security for BeeHive has only been targeted by the authors of BeeHive so far. Independent in-detail evaluations are advisable to attain a qualitative assement.

Finally, we have evaluated the extent of autonomic behaviour both approaches show. BeeHive clearly distinguishes itself regarding the initial configuration and continuous optimization during runtime. OSPF was not designed to operate autonomous on these accounts. Thus, BeeHive represents a step forward to conquering the growing complexity.

Despite our evaluation results, we have to admit, that we cannot give a definitive answer to the initial question what role nature-inspired methods can play in network routing. Although we have shed more light on the applicability of BeeHive in future WANs and advanced the knowledge on this account. Especially, some earlier results and conclusions about BeeHive, specifically regarding the comparison to OSPF, are no longer justified. Furthermore, we have revealed several areas, in which further work is advisable.

5 Towards Data-Centric Security

The value of what is being protected
affects the measures taken to protect it.
[Basic principle of security]

Today we live in a world where the amount of digital data is constantly growing, both in volume and diversity. A recent study [113] estimates that the amount of digital data will increase by the factor of 10 every five years. At this, the increasing amount of personal data in digital form is of special importance. Being of a certain economic value, it is mostly stored outside the users' control on websites or by external companies. Users ultimately have to trust these to appropriately handle the personal data. Equally important and problematic is the handling of confidential or sensitive data in organizations, such as blueprints for a new technology or customer data. It is often shared with other companies during cooperation or the outsourcing of operations, leaving its protected haven.

However, today personal and confidential data can quite often be found, for example by a simple Google search, on lost USB drives, or on auctioned hard drives, which clearly demonstrates that data is not always handled as securely as it should be. For some time now, this led researches to warn about a denied oblivion of data [250] and to propose that the future information society and technology need to relearn how to forget [168].

Especially organizations should pay special regard to data security, since they are also often subject to legal and compliance requirements. For example, the EU Directive 95/46/EC [75], the Health Insurance Portability and Accountability Act (HIPAA) [263], or the Sarbanes-Oxley legislation [237] all dictate a specific handling of certain data with regard to security. Quite often usage restrictions are part of these obligations as well. For example, the EU Directive 95/46/EC dictates that personal data may only be used for the originally intended purpose. The organization is responsible for the compliance, regardless if the data resides on their own servers or the IT infrastructure of partners that may perform some outsourced operations. Still, the recurring news about unintentional data leakage everywhere indicate that the employed security mechanisms do not cover the security requirements appropriately. The ramifications for organizations may range from damaging the business reputation up to a struggle for the continued existence of the company.

It is undisputed, that generally all data should be handled securely. However, striving for the absolute maximum security regarding all data is a bad strategy, as the costs explode and the usability diminishes. Rather, providing "good enough" security is a feasible and meaningful approach. Therefore, due

to the fact that different data is not equal in its value, security should ideally be applied according to the old security principle: *"the value of what is being protected affects the measures taken to protect it"*. Yet, as we have discussed in Section 2.1.2, security concepts today center mostly on the technologies used to store and handle data, not the data itself, though additionally, a few applications have specialized, built-in security measures. A differentiation in the security for different data types is barely possible using these technologies in distributed and complex IT environments. Furthermore, large organizations often do not even know which kinds of data are present and where they are stored. Therefore, due to the cost of providing high security throughout the IT environment, sensitive or confidential data is often not handled as secure as it should be.

The movement towards UC will aggravate these problematic trends as well as create several new challenges (cf. Section 2.2.3). The mountains of digital data will grow even faster, both in quantity and quality, with the massive deployment of smart devices and sensor networks. Furthermore, due to the high numbers of employed devices, network complexity increases dramatically. Managing their security on a per device level is doomed to fail, demanding for autonomic behaviour. Thus, a plethora of autonomously operating end devices interact dynamically, offering their specialized services or composing higher-level services on-the-fly from other services. In these IT environments the information flow will no longer have any boundaries and data will generally be shared, stored and accessed outside the control of its owner.

It is obvious, that the traditional focus of security on the infrastructure, containing concepts such as perimeter security, ID-based access control to data, or secure communication tunnels between two trusting devices, cannot cope entirely with this increasing complexity as well as the quantity and quality of data on their own. Trying to achieve security with device-centric solutions alone will quickly become an unmanageable task. While they will still have their application domains, we generally need to rethink our dominating security paradigms. This has also repeatedly been said in the security community lately, e. g. in [57, 119, 207].

Clearly, although a healthy IT infrastructure is vitally important, the real asset in today's as well as in future IT environments is the data and not the devices that gather and process it. Therefore, it is reasonable that the focus of security should likewise be data-centric rather than device-centric. However, we have two problems that accumulate. Already today legal obligations demand for better security. However, the existing security infrastructures are not well equipped to handle the task. Furthermore, emerging UC environments will aggravate the problem by several orders of magnitudes. Thus, it's high time to question the existing security systems and evaluate the benefits and impeding factors of data-centric security.

The term data-centric security is used quite often today. However, apart from research done by IBM [89, 99] that focuses on business process integration of

data-centric security, we are not aware of a distinct analysis and definition of necessary elements and aspects as well as inhibiting factors and technological feasibility.

Therefore, we define in the following the goals and necessary concepts needed to establish data-centric security. In addition we provide a survey on existing technologies that are mainly used today to implement the required concepts and discuss, to what extent they provide useable solutions in future UC environments. In particular, we highlight open issues that need further research. Parts of this research have been published in [212].

5.1 Goals and Concepts

The focus of data-centric security is to provide at all times the fitting security level for each data set, allowing for continuous optimal data security, regardless where the data is transferred, stored, or accessed. At this, the data itself specifies the required security level, not the available infrastructure. In fact, the infrastructure should automatically adapt its provided security to the requirements of each data set.

Data-centric security has several advantages compared to the traditional device-centric focus of security. First, data-centric security effectively prevents data leakage, since the data is protected according to its needs at all times. For example, a high value data type may only be stored on a mobile device using strong encryption. Now, if the mobile device is lost or stolen the stored data remains secure. Furthermore, the complexity of the security management decreases since not a plethora of devices have to be configured with regard to security. Rather the focus is on providing standardized building blocks that provide the security that is required by the data.

Another major advantage of data-centric security, especially for organizations, is the clear division of labor and the resulting ease of integrating the security specification into business processes [99]. In particular, security requirements are specified in an abstract and technology agnostic form, while the specific implementation is left to the infrastructure. This allows on the one hand to enable the decision makers and users of the data to specify the correct security requirements for the data, e. g. to meet compliance requirements. On the other hand, the experts on security technologies can focus on providing sound implementations of the required concepts, without having to decide which level of security is appropriate for a particular device that handles several different data types. Finally, evidence of compliance is provided automatically, if the data history is recorded, since this provides a complete audit trail.

Generally, data-centric security must go beyond mere transmission and storage security as well as access control. It must also include measures to describe and enforce usage rules, in order to comply, for example, to legal obligations, such as that the data may only be used for the originally intended purpose.

Thus, in order to establish data-centric security, four aspects are essential on a conceptual level: data classification, meta data, security specification, and security enforcement.

Data Classification The foundation of data-centric security is data classification, i. e. the data owner has to evaluate which types of data need to be protected and why. Generally, not all data will require protection, but the subset that is important, e. g. for the company's operation, or that is of sensitive nature needs to be secured. The importance is expressed as the data value, i. e. a high value implies high security requirements.

For example, a location system that monitors the usage of an office building may produce two different data types. On the one hand, statistical data about the monthly usage of certain rooms or services may be classified as low value data and thus shared freely within the company's network. On the other hand, detailed data that allows to see which person is present in which room using which service at a specific time, is of high value, due to privacy concerns. Therefore, it may only be used in strictly specified situations, such as to locate all employees in an emergency or to determine the user after a service has been used maliciously. Security for this data type may be strict access control, e. g. by encrypted storage, the prohibition of the data flowing outside the company's core network, and the usage constraint on the specified situations.

Additionally, legal obligations may demand specific security requirements for data in certain situations. An example of this can be found in the payment card industry's security standard [193] that requires encryption for identity data when it is combined with the account number, but not if it appears by itself, as Bayuk describes in [20]. Therefore it is imperative to include the context of the data usage, e. g. the underlying business process, when classifying data.

Today, the classification of data is generally a very difficult thing to do, since data permeates the whole IT systems and is stored in several different places, such as file servers, email inboxes, or databases. Therefore, on the one hand it is hard to identify all relevant data and on the other hand the evaluation is complicated, since different technologies have to be used to access the data. Regarding the qualitative challenges in UC environments this paints a bleak picture about the feasibility of data classification in these environments. However, keeping the service orientation of UC environments in mind, specifying the data value during creation according to high level policies provides a decentralized solution to this problem. For example, the location system described above knows the purpose for which it gathers data and is therefore able to label it according to a global policy.

Meta Data In order to describe and assess the data quality and value, the meta data specifies additional information about the data. In addition to the data origin, it must include the whole lifecycle of the data, covering its gen-

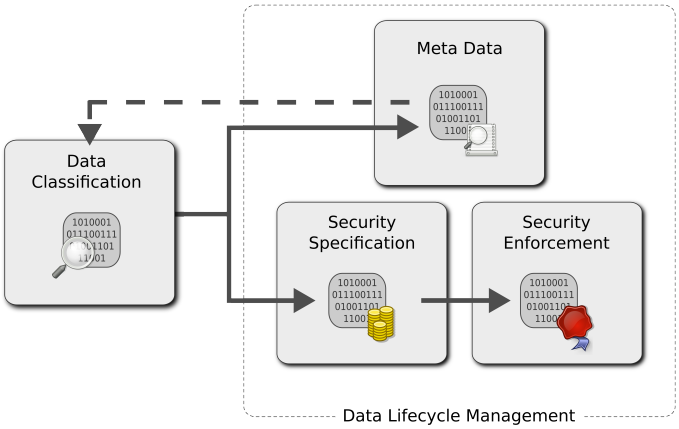


Figure 5.1: *Overview of data-centric security concepts*

eration, each transmission and every usage or alteration until its lifetime is exceeded or its purpose is fulfilled.

In the context of future UC environments this is especially important regarding two problem areas. First, the quality and relevance of data becomes measureable and assessable, which facilitates the classification. Second, a complete data history greatly facilitates accountability as the data history provides a complete audit trail. At this it also sustains the dependability of UC services.

Although we focus in this chapter on data-centric security, UC devices must also be confident that the data they process is genuine and accurate in order to build dependable services. Therefore, measures that allow the devices to build up trust in the accuracy of data, such as traceable quality, are equally important as measures to protect data.

Security Specification Based on the data value and potential legal obligations, the security goals for the data must be specified and transferred into security policies, i. e. guidelines and constraints that regulate the usage, handling, and flow of the data. These may include for example the time frame during which the data may be used or is up-to-date, how it must be stored, and under what conditions it may be accessed. Generally, the policies must be inseparably attached to the data at all times during its lifetime, specifying how it must be handled.

Security Enforcement Finally, the policies must be enforced during the whole lifetime of the data, regardless whether it is transferred, stored, or accessed. Thus, the automatic provisioning of security by the employed infrastructure as well as the compliance of the end devices must be verifiable.

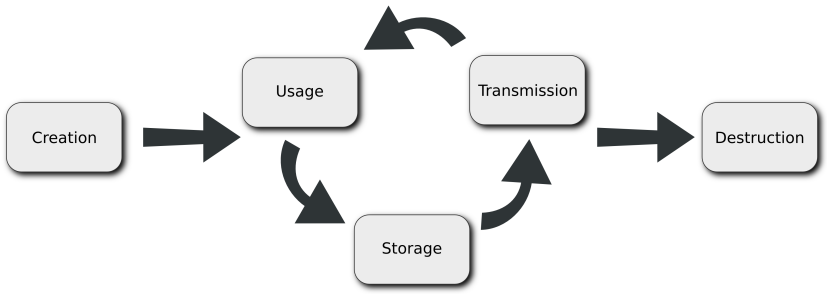


Figure 5.2: *Data lifecycle*

Figure 5.1 illustrates the conceptual parts and their relations. We refer to the combination of the latter three concepts as *Data Lifecycle Management* (DLM) and focus our further discussions in the next section on its details.

5.2 Data Lifecycle Management

Data lifecycle management centers around specifying and controlling the whole lifecycle of data. At this, it comprises all steps in the life of data beginning with its creation, storage, each transmission, and every usage, including alterations, until its lifetime is exceeded or its purpose is fulfilled. In particular, the destruction of the data is explicitly included and specified. Figure 5.2 illustrates the lifecycle of data. With regard to security, requirements can be specified for each step, such as to allow only encrypted storage or that the data should never leave the internal network.

This concept is sometimes also referred to as Information Lifecycle Management (ILM). Although this has been a hot topic in organizations for some time now, ILM has mostly been limited to the area of data storage [198]. At this, additionally to data security, other concerns, such as data backup and recovery play an important role in ILM. However, this limited application area will not be sufficient in the future. Rather, DLM needs to be incorporated into the IT environment as a whole, in order to provide sufficient security in future IT environments.

In contrast to today's security concepts, an IT environment that fully utilizes DLM, requires few device-specific security configurations. Rather, these can largely be replaced by the dynamic and autonomic fulfillment of the security requirements specified alongside the data. This allows on the one hand for a more fine-grained security configuration, optimally adjusted for each data set, and on the other hand, it reduces the complexity of the security management. Furthermore, changes in the required security handling for a specific data type can easily be achieved by simply changing its security requirements.

In the following, we will delve into the details of the individual concepts that make up DLM, the meta data as well as the security specification and enforcement. Furthermore, we review which technologies are used today to implement the concepts and comment on their applicability in future UC environments.

5.2.1 Meta Data

In UC environments the vast quantity of devices will create an immense data flood. At this, typically more than one source will provide similar information, albeit maybe of different quality or slightly different focus. Furthermore, due to the opaque and complex communication patterns between the autonomous devices and on-the-fly service provisions, the specific type of data as well as its origin and quality will be hard to establish. Furthermore, especially in science, given results are only credible and considered reliable if the process that produced them is completely traceable.

We would like to illustrate especially the last point by giving an example that will be common in UC environments. Imagine a widespread WSN that provides high resolution spatial and temporal environmental monitoring capabilities. Due to energy and bandwidth constraints it is generally advisable to filter and aggregate data inside the WSN, although this introduces the risk of potentially loosing information. Especially the capturing of high-frequency phenomena using resource constraint devices requires extensive in-network processing [90]. However, the resulting data can only be interpreted correctly by external users, if the operations on the data are clearly marked and traceable.

Therefore, in order to enable reasoning about the relevance of a data set, its quality, and thus, its value, it is imperative to provide information about the data, including the authoritative source of the data and its history.

Research on *data provenance*, which is gaining momentum lately, focuses on this subject, e. g. see [174, 243]. Typically, the exact definition of data provenance differs slightly with regard to its application domain. Similar to Simmhan et al. [243] we define data provenance as:

Definition 24 (Data Provenance) Information that helps to determine the derivation history of a data set, starting from its original source.

Generally, several domain-specific provenance models exist, e. g. [34, 114], that typically look at data provenance from different angles. In [242, 243] Simmhan et al. present a taxonomy of general data provenance techniques. Apart from possible applications of data provenance, such as facilitating traceable data quality, data reliability, and the provision of an audit trail, they also present the design space of data provenance schemes and highlight important design questions.

First, the subject of provenance needs to be decided upon, e. g. the data or the process. Furthermore, the question arises, which granularity of provenance

information is sufficient for the applications that handle the data type? Second, a model for representing the provenance data needs to be established, specifying the syntax and semantics as well as providing the possibility to query the provenance data. Both design decisions have a great impact on the third design questions, the storage of the provenance data. Especially the aspects overhead and scaleability play an important role in this context.

Regarding data-centric security, the subject of provenance clearly needs to be the data. However, taking the general quantitative increase of data in UC environments into account, this results in a tremendous overhead, questioning its scalability and general performance. Therefore, we believe that the "right" granularity of provenance information should be determined on the basis of the application domain, in order to keep the overhead to a minimum. For example, the surveillance system we described above gets by with only two types of provenance data, mapping each data set to one of the provenance types, if the method to gather the data does not change over time. Problems arise in this scheme, however, if the data crosses application domains, which may have different granularity needs.

In order to represent the provenance data in a standardized structured data model, Moreau et al. propose in [172, 173] the open provenance model. It aims at providing a generic model to specify provenance unambiguously. According to its authors, it allows on the one hand for the documentation of complex distributed applications that comprise different technologies. On the other hand, it provides a compatibility layer to allow provenance information to be exchanged between different systems. Finally, a query mechanisms is included, allowing to identify information and processes independent of the used technologies. Still, the applicability of the open provenance model in UC environments, especially regarding its application on resource constrained devices, needs to be studied further. Nevertheless, Park and Heidemann present in [189] a provenance model for WSNs that is compatible to the open provenance model, which gives first positive hints towards its applicability in UC environments.

From an architectural point of view, the question arises where to store the provenance data. Storing it alongside the data will overload the communication channels, due to the significant overhead. Yet, since seamless connectivity is an underlying assumption in UC, we believe that it may be stored at any available place inside the network, as long as the data contains an unambiguous link to it. Furthermore, storage should be associated with the data provider, in order to establish a decentralized and thus more scaleable solution.

Finally, although disregarded in the taxonomy, the security of the provenance data itself must be addressed, since only reliable provenance data is of any use. For example, if an attacker is able to alter existing provenance data or insert additional entries it becomes useless. Generally, integrity and availability are the security services desired foremost in this context. However, confidentiality may also be a vital concern, for example, when the process that generated the data is a trade secret.

Mainly, the research on provenance data security is still in its infancy, offering an interesting field for further research. Nevertheless, first publications that highlight open research issues in provenance security exist. Hasan et al. study security aspects of provenance focussing on data storage [96]. Additionally, Braun et al. reason in [36] that provenance data has different security needs compared to data in general and therefore requires its own security model.

5.2.2 Security Specification

Based on its value and potential legal obligations, the security goals for the data must be specified and transferred into policies, i. e. guidelines and constraints that regulate the usage, handling, and flow of the data. Generally, policies can be devised for each step in the data lifecycle, including a predefined lifetime either along a temporal dimension or based on the fulfillment of a condition. Furthermore, the policies must be inextricably bound to the data during its lifetime in order to govern it and to provide the infrastructure with the specific security requirements.

5.2.2.1 Policy Specification

Several definitions exist for a security policy, originating mostly from the context of the different application domains [240]. For our purpose we define a security policy as follows.

Definition 25 (Security Policy) A security policy represents a high-level statement, specifying rules to achieve the security goals and objectives for an asset, independent of the security mechanisms.

In the context of DLM mostly two kinds of security policies are relevant: access control policies that specify the device or devices that are allowed to access the data and data restrictions, formulating constraints regarding its usage and responsibilities about how the data is to be handled.

The traditional standard technique for protecting documents is access control [107], which has been studied in the community for decades. The XML-based language XACML [186] is widely used today focussing on expressing access control policies. However, this narrow focus, rather than the ability to specify high-level usage rights for the data, such as “the data may only be used for purpose ABC”, makes it only applicable to application scenarios, in which access policies provide sufficient expressiveness.

Several more encompassing rights expression languages (RELs) exist in the context of digital rights management (DRM), the most popular being ODRL [206] and XrML [272] according to [94, 117]. These can be used for the purpose of describing diverse high-level security policies and digital contracts. Both are based on XML as well and cover a wide variety of operational aspects, providing

the functionality for various possible application scenarios. Furthermore, both are extensible in order to adapt them to any specific purpose.

However, Jamkhedkar et al. note in [118] that their generic focus is also their main drawback. They are so extensive and complex that considerable effort has to be taken to understand and use them. Additionally, Pretschner et al. describe several important shortcomings in [103, 204]. First, RELs originate from the DRM context and are thus best applicable for audiovisual content. Second, their internal semantics are not always precise, leaving ambiguity in the specified policies. Third, the conversion of policies in different formats seems to be an open issue right now.

Finally, the applicability of full-blown RELs in UC environments represents an open issue. We are not aware of any project that employs a standard REL on resource constrained embedded devices.

Additionally, to these standard technologies, several approaches have been proposed that explicitly take the UC characteristics into account, though mostly focussing on special subareas. For example, Kagal et al. propose in [125] the policy language Rei. Patwardhan et al. provide an architecture and proof of concept implementation in [190] that proves Rei's ability to provide security in a mobile, though infrastructure-based, UC environment. The policy language Ponder2 for autonomous pervasive environments was recently proposed by Twindle et al. in [261]. Also focussing on mobile autonomic networks are Aljnidi et al. with a policy system they propose in [6]. A security policy language specially designed for WSNs is proposed in [165].

Despite these research accomplishments, a standard for defining security policies that takes most aspects of UC into account, has yet to emerge. Although an all-encompassing framework might be hard to devise, given the complexity of UC, interoperable standards for well-defined areas are clearly needed to address general policy issues, such as combining, merging, decomposing, migrating, and negotiating different security polices between different frameworks. Though we are not aware of any attempts to define sensible areas, this is especially important when taking the general service orientation in UC environments into account. For example, imagine a service that combines several data sets from other services to provide a higher-level service. If it crosses application domain boundaries at this, mechanisms to combine or at least retain the original policies are clearly needed.

5.2.2.2 Binding Policies to the Data

In order to control the whole lifecycle of the data it must not be possible to separate it from its policies. Otherwise, on the one hand no data-specific security can be applied to the data set. On the other hand, policies might easily be changed for other devices' data. Karjoth et al. defines this in [127] for example as the *"sticky" policy* paradigm, though they do not discuss ways to bind the policies to the data.

Generally, the binding can be achieved by employing cryptographic techniques, resorting to access control. We would like to point out especially two families of encryption schemes in this context: traditional public key encryption and attribute-based encryption.

Traditional Public key Encryption Public key encryption can easily be used to sign a hash generated from the data and its policies. However, public key cryptography needs an entity that manages the keys and provides a basis for reasoning about the keys' trust. One possibility is employing a public key infrastructure (PKI). However, with regard to scalability many PKI implementations do not perform well [245]. Furthermore, keeping the growing complexity in UC environments in mind, the manageability of a PKI is generally a daunting task. Therefore, a "web of trust" approach [45] might be a better fitting solution for UC environments due to its distributed nature. It is important to note that the reasoning about trust in the keys differs at this, compared to PKIs, since it is based on trust between individual devices and not in central certificate authorities. However, research in computational trust, though numerous, still does not provide a mature and useable solution due to some researchers, e. g. [142, 144].

Generally, in both approaches keys are bound to an entity's identity, which is not an optimal solution considering the numerous entities that will be present in UC environments. Furthermore, data can only be encrypted for one entity at a time, resulting in a high communication overhead if more than one client should access the data.

Attribute-based Encryption In contrast, attribute-based encryption (ABE) schemes do not encrypt the data for a specific entity, but for all clients that have certain properties or attributes. These may include arbitrary aspects, like the device type, its operational context or the history of a device. During encryption the combination of attributes serves as the key and a device is able to decrypt the data if its attributes match exactly the combination specified during the encryption.

Based on the original proposal of ABE [227] several different variants were proposed in the literature, e. g. [23, 87, 88], each having its own strengths and weaknesses. For example, Bethencourt et al. propose in [23] the ciphertext-policy ABE, in which the encryption takes place by specifying an access tree structure over the chosen attributes. Each client's private key is furthermore associated with arbitrary attributes expressed as strings. The client is able to decrypt a ciphertext, if its attributes pass through the ciphertext's access structure.

The drawback of most ABE is that usually a trusted third party manages the set of attributes, the private decryption keys, and the validation of the devices' properties. However, one central authority is hard to image in UC environments. However, Müller et al. propose in [178] a distributed ABE scheme to remedy this drawback.

Using ABE, the specification and binding of the security policies becomes essentially a matter of access control. However, it allows for significantly more fine-grained access control possibilities and simplifies the encryption at the data originator to specifying the desired properties for each data fragment, compared to traditional approaches. Furthermore, it blurs the lines between policy specification and enforcement, as it actually combines both. Since a device only gets access to the data if its attributes correspond to the chosen set during encryption, security policies are automatically enforced during decryption. For example, policies, such as "device type XYZ may access this data, however, only if it is located in the core network" may be specified as attributes during encryption and only compliant devices are able to decrypt the data.

Generally, both schemes to bind the polices to the data suffer from an insider attack. Once a device has been given legitimate access to the data, it may strip the policies from the data. Since we cannot prevent this a priori, detecting this lies within the focus of security enforcement, which we cover in the next section.

5.2.3 Security Enforcement

Compliance to the specified policies must be enforced wherever the data resides. Generally, the enforcement of the security policies represents both, the most important aspect as well as quite often the most complicated aspect of DLM. Without proper enforcement of the policies, the whole concept becomes questionable.

Chadwick and Lievens discuss in [48] the enforcement of policies in distributed applications from an architectural point of view. They propose three protocols, that enhance either the application protocol, provide a new security layer directly underneath the application, or use a back channel model for policy enforcement. Furthermore, they discuss the individual merits of each approach. However, a complete evaluation of the tree protocols is still subject to future work.

In general, Pretschner et al. differentiate in [204] between direct policy enforcement and observation. Direct enforcement is possible if the owner of the data can access the device to validate it or provide the application software. In cases where this is not possible, the observation of significant parameters might prove a device's compliance or noncompliance as well.

5.2.3.1 Direct Enforcement

The enforcement of access control policies is a well studied problem in computer science [107], including in addition to traditional ID-based access control other aspects, such as geo-spatial access control [11]. Especially the usage of ABE presents a neat process of specifying and enforcing arbitrary access control policies. However, the enforcement of usage restrictions and responsibilities is generally more difficult.

DRM is probably the best known technology to directly enforce restrictions on digital data and is rather popular in the content industry for enforcing digital copyright. Yet, the different existing DRM systems are typically closed proprietary and incompatible systems [143]. This results mainly from the fact that they try to enforce their policies in an untrusted environment and therefore resort to using closed proprietary hard and software. However, the closed proprietary and incompatible architectures clearly clash with the openness of UC environments, where communication based on established standards will play an important role.

Some argue that employing *trusted computing* (TC) techniques [259] may provide a solution to overcome the limitation of proprietary DRM systems. If provable unmodified software runs on a provable unmodified system, the compliance to the policies can reasonably be assumed. Therefore, while employing TC techniques does provide a trusted basis which alleviates several problems, it is unlikely that the vast number of different devices that constitute UC environments will all contain a trusted hardware chip. Thus, it may be available in some cases though not in general. Albeit, TC has rather strong requirements for several always available PKIs [14], which is not even realized in today's networks.

Almost the same direction as TC is taken by research on remote attestation, e. g. see [239, 226]. It aims at remotely verifying certain aspects of the device's hardware or software and deduce from this the compliance or noncompliance to the policies. Usually, remote attestation requires a standardized application software on each device and the ability to determine critical aspects about the hardware and operating system. Though with numerous different devices in UC environments this seems to be an unmanageable task.

Finally, several approaches try to establish trust relationships between devices and utilize these to make security-related decisions, e. g. see [124]. Still, establishing and maintaining trust between devices in UC environments is a nontrivial task, especially if the interactions are dynamic and short-lived. For example, Pavlovic argues in [191] that common trust systems can easily be subverted by attacking high-ranking nodes in the network. On this account he reasons that trust should mainly stem from private trust vectors that the devices manage themselves, including public trust notions only for refinement. Additionally, trust between devices is inherently concerned with IDs, which cannot be guaranteed to be unique and unchanging in UC environments and is thus contrary to the goal of data-centric security. Furthermore, as we have already noted earlier, some researchers question the applicability of trust at all [144], or at least do not deem the current schemes mature enough [142].

We believe however, that interactions that occur on a regular basis could profit from trust establishment. Especially the available provenance data might provide additional means to establish trust in the data itself, as shown in [59]. Furthermore, research on physically unclonable functions (PUF), e. g. [187,254], provides the means to unambiguously identify and authenticate physical de-

vices. PUFs use the unique physical characteristics of the integrated circuits of the devices to derive for example secret keys. Although two devices may be logically identical, the small variances in the hardware, caused mainly by manufacturing variances, are unique and almost impossible to copy or model. Due to the fact, that the hardware characteristics are not changeable they provide an excellent source to unambiguously identify a device. Nevertheless, taking the prospect of a plethora of deployed devices into account, the scalability of ID-based approaches is still questionable.

5.2.3.2 Observation

In situations where direct enforcement is not possible, the observation of significant parameters may provide the means to determine a device's compliance or non-compliance to the specified security policies. Pretschner et al. mention in [204] that they are not aware of any existing implementations to observe a device's compliance without direct access, due to the difficulty of obtaining trustworthy signals.

However, for audiovisual data types the usage of *watermarking* technologies [185, 197] might provide a means to detect a postiori a device's noncompliance.

In general, a postiori control of compliance in form of audits is quite common in real world businesses. Therefore, we believe it to be an interesting field of future research. A first step in this direction provide Etalle and Winsborough who transfer this idea into collaborative distributed computing, by proposing A Posteriori Policy Enforcement (APPLE) in [74]. They argue that preventive policy systems are too restrictive with regard to unforeseeable situations, i. e. situations in which acting in disagreement with the specified security policies is appropriate. APPLE allows for this, but logs these actions in order to hold the device or user accountable after the fact. An example situation in which such a system may prove beneficial is the case of an emergency, in which otherwise private data is accessed to save persons at risk.

5.3 Summary and Discussion

In the previous two sections we have presented the goals and advantageous of data-centric security, specified the concepts necessary to establish data-centric security, and surveyed technologies that may be used to implement these concepts. Due to the vastness of the underlying research fields we have selected typically used technologies as well as those that specifically target UC environments. Therefore, we do not make the claim to have presented an exhaustive list of all possible technologies. We have conducted the whole analysis from the perspective of how applicable are the concepts and technologies to UC environments.

In summary, it is obvious that several of the concepts necessary to establish data-centric security are still subject to further research before we may witness more than isolated and strictly limited data-centric security solutions. Especially the policy specification and enforcement represent problematic areas. We believe that especially with regard to policy-based systems, standards need to be established in order to allow for a permeation of these throughout the IT infrastructure.

Furthermore, regarding the security enforcement, we believe that leveraging the distinctive UC-specific characteristics for enforcement present interesting and worthwhile questions for future research. For example, the high numbers of devices that form UC environments and their context awareness are two aspects that may prove useful to encourage compliant behaviour of nodes. In the following we like to sketch two ideas in this context.

The basic assumptions are that devices generally distrust each other and that services keep a public log of all generated data types. Now, if devices that use the data give public commitments about their data usage, an interesting research project would be to explore if nodes are able to determine other misbehaving nodes on the basis of the meta data of the data sets and the public records.

The second idea centers around the belief that uncertified information is in essence nothing else than a rumour and cannot be trusted. It may address the problem of clients stealing data and pretending to be the originating service. Though we generally cannot prevent unauthorized copying and sharing of data by clients without employing TC techniques, we can try to discourage this behaviour. By binding a special token to the data or its meta data, a service proves its genuineness. Such a token may be based on the attestation of a service's data by long-established neighbouring services that built up trust among each other. Thus, a malicious device may only share the original bundled data, which can be traced back to the legitimate originator or it may share the unbundled data. However, it cannot prove the correctness of this to other devices. Therefore, any device that is interested in correct data will naturally use the data of the provable service.

Finally, though we have not discussed this topic further in our discussions above, the realization of an adaptive infrastructure presumably requires further research efforts. While adaptive security in data storage and transmission seem to be achievable without too much effort, we deem global adaptive policy-based data usage to be the most problematic area. Such systems have to be established on all end-systems in order to allow for the advantageous of data-centric security to take effect.

6 Conclusion and Outlook

We should all be concerned about the future
because we will have to spend the rest of our lives there.
[Charles F. Kettering]

Ubiquitous computing is gradually realized and this ongoing trend will change our lives in the most profound fashion. Therefore, it is of utmost importance to ask questions, analyse problems, and design solutions regarding security and dependability in future UC environments. If we miss to target this development right now at the beginning, some systems will surely be deployed that may have a grave impact on our personal freedom or the stability of our surrounding. Furthermore, retrofitting security and dependability aspects into existing applications and systems is at best a daunting task and is also generally opposed by security practitioners. Therefore, we have targeted our research at three problem areas regarding the management of security and dependability in UC in this dissertation.

First, we have addressed the requirement to integrate security measures into heterogeneous devices that provide the foundation for UC. Considering the distinct characteristics of resource-constrained devices, i. e. sensor nodes, we have designed and implemented a security architecture for a mobile WSN. Although we have achieved our goals of securing that particular application scenario, this methodology of providing security is not optimal. Changes in the application scenario or the employed sensor node hardware may require a complete redesign of the security concepts as well as of the respective implementations. Thus, we have designed and implemented a more generic framework that allows for the integration of the optimal security concepts and measures into arbitrary WSN application scenarios without the requirement for the application developer to become a security expert. The underlying premise of this concept is that the appropriate security concepts and measures can be derived by looking at the application properties. In our framework, the application developer specifies only abstract security services for the data types in his application and some important application properties. These annotations are then evaluated by the framework and possible security setups are generated and evaluated regarding the level of achieved security as well as the residual risks. This information is finally presented to the application developer, who is thus able to choose the optimal security solution for his application. Furthermore, he is aware of potential risks and may handle those inside the application logic. Finally, we have integrated this framework into the middleware synthesis tool Fabric to additionally provide the application developer with the required source code for the security solution. The evaluation of the framework shows the individual

advantages of this approach. In essence, we believe that the provision of comprehensible security at a middleware level significantly facilitates the integration of security into UC devices.

In our second contribution we have focused on the possibility of autonomically increasing the dependability of wide-area networks that may be achieved when employing a nature-inspired routing approach. First, we have determined the most promising nature-inspired routing approach, BeeHive, based on the published literature. Following this, we have gradually evaluated it compared to the state-of-the-art routing approach OSPF. At this, we have assessed both approaches characteristics regarding the individual aspects that define dependability. The overall results show that BeeHive does not clearly outperform OSPF in the performance-related aspects, despite the commonly published statements from its authors. However, BeeHive clearly wins when considering the necessary autonomic behaviour, that may be required in order to conquer the complexity of UC. In addition to these main results, we have identified several open questions, that require further analysis.

Our final focus of research has been arguing the case for data-centric security and evaluating its feasibility. Traditional security concepts will not suffice to establish security and protect the data in future UC environments alone, though they will still have their application domains. The concept of data-centric security provides a more reasonable focus of security and has the potential to provide a solution to this predicament. We have established the necessary concepts and discussed commonly used technologies to implement them. Furthermore, we have evaluated them regarding their applicability to future UC environments. Generally, it is apparent, that several diverse research directions need to be combined in order to establish data-centric security. Furthermore, the results document that the implementation technologies for several of the necessary concepts are still subject to further research, especially when the characteristics of UC have to be considered. Thus, the main contribution of our work is a structuring of the necessary concepts and technologies, which may serve as the groundwork from which further research may be conducted and combined, in order to achieve data-centric security in the future.

Generally, our research in this work has advanced the common knowledge in the respective contribution areas. However, several paths to extend our research exist. Although we have presented and discussed possible future work within each of our contributions, we would like to take the opportunity and again highlight important aspects here.

We would like to extend our framework to provide comprehensible security along several dimensions. First, we contemplate to provide more differentiated feedback about the possible security setups, especially focussing on code size and runtime. This would allow the application developer to further fine-tune his application, particularly if optimized versions of the security setups regarding both aspects are available. Furthermore, including dependability aspects would greatly improve the framework.

In the context of autonomic dependability in network routing, BeeHive has to be analysed further, especially regarding TCP and realistic streaming traffic. Furthermore, scalability test are indispensable. Finally, a comparison with more elaborate traffic engineering concepts, such as MPLS would facilitate a deeper understanding of BeeHive's advantages and drawbacks.

Finally, we would like to extend the presented ideas in the final discussion of the data-centric security research and evaluate their usefulness in detail. We believe that such simple measures could play an important role in securing future UC environments.

Appendix

A Appendix to Integrating Security into Ubiquitous Computing Devices

A.1 Implementation Details for the Blundo et al. Scheme

This appendix illustrates a way to efficiently implement the Blundo et al. scheme on sensor nodes, as presented in [232]. Two concepts are combined: first, the Horner rule minimizes the multiplication effort and second, the choice of the field determines the speed of the modular reduction.

For an efficient implementation, the *ID*'s of the nodes can be reduced to a certain range, e.g. $0 < ID < 2^{16}$. If the evaluation is done using the Horner rule [138] (see also Figure A.1) only multiplications between a short *ID* and a longer field element have to be done, which allows for a much faster modular arithmetic than the general case.

$$\begin{aligned} a_n ID^n + a_{n-1} ID^{n-1} + \ldots + a_0 \quad &= \\ (\ldots(a_n ID + a_{n-1})ID + a_{n-2})ID + \ldots) + a_0 & \end{aligned}$$

Figure A.1: *Horner rule*

The choice of the field F determines furthermore the range and the speed of the evaluation of the common secret. The scheme can be implemented using any field. Consequently, a field, which suits the existing hardware platform well, should be chosen. The arithmetic of $GF(p)$ is supported by the existing hardware multiplier of the MSP 43071 processor. Therefore, this choice is faster than an implementation using $GF(2^m)$.

The following example assumes that the identities are limited to $0 < ID < 2^{16}$ and illustrates the usage of a 80 bit Generalized Mersenne Prime, which allows for an efficient modular arithmetic [248], $p = 2^{80} - 2^{64} - 2^{32} - 1$. First a normal multiplication of a 16 bit identity and a 80 bit coefficient is calculated. Afterwards one can rearrange the 96 bit result $r = \sum_{i=0}^{5} r_i 2^{16i}$ to $s = \sum_{i=0}^{4} r_i 2^{16i}$ and $t = 2^{64} r_5 + 2^{32} r_5 + r_5$. From the special form of p follows that $s + t = r \bmod p$. The advantage is that the reduction modulo p can be calculated by a single 80 bit addition and at most two 80 bit subtractions. This is substantially faster than any standard method that assumes two equally long operands.

B Appendices to Autonomic Dependability in Network Routing

B.1 RFCs regarding OSPF

RFC	Title	Author	Date	Info	Status
2328	OSPF Version 2	J. Moy	04/1998	Obs. RFC2178	Std.
5340	OSPF for IPv6	R. Coltun, D. Ferguson, J. Moy, A. Lindem	07/2008	Obs. RFC2740	Prop. std.
5252	OSPF-Based Layer 1 VPN Auto-Discovery	I. Bryskin, L. Berger	07/2008	Errata	Prop. std.
5250	The OSPF Opaque LSA Option	L. Berger, I. Bryskin, A. Zinin, R. Coltun	07/2008	Obs. RFC2370	Prop. std.
5243	OSPF Database Exchange Summary List Optimization	R. Ogier	05/2008		Inf.
5187	OSPFv3 Graceful Restart	P. Pillay-Esnault, A. Lindem	06/2008	Errata	Prop. std.
5185	OSPF Multi-Area Adjacency	S. Mirtorabi, P. Psenak, A. Lindem, Ed., A. Oswal	05/2008	Errata	Prop. std.
5088	OSPF Protocol Extensions for Path Computation Element (PCE) Discovery	JL. Le Roux, Ed., JP. Vasseur, Ed., Y. Ikejiri, R. Zhang	01/2008	Errata	Prop. std.
4973	OSPF-xTE & Experimental Extension to OSPF for Traffic Engineering	P. Srisuresh, P. Joseph	07/2007		Exp.
4972	Routing Extensions for Discovery of Multiprotocol (MPLS) Label Switch Router (LSR) Traffic Engineering (TE) Mesh Membership	JP. Vasseur, Ed., JL. Leroux, Ed., S. Yasukawa, S. Previdi, P. Psenak, P. Mabbey	07/2007		Prop. std.
4970	Extensions to OSPF for Advertising Optional Router Capabilities	A. Lindem, Ed., N. Shen, JP. Vasseur, R. Aggarwal, S. Shaffer	07/2007		Prop. std.
4940	IANA Considerations for OSPF	K. Kompella, B. Fenner	07/2007	Errata	Best current practice

RFC	Title	Author	Date	Info	Status
4915	Multi-Topology (MT) Routing in OSPF	P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, P. Pillay-Esnault	06/2007		Prop. std.
4813	OSPF Link-Local Signaling	B. Friedman, L. Nguyen, A. Roy, D. Yeung, A. Zinin	03/2007		Exp.
4812	OSPF Restart Signaling	L. Nguyen, A. Roy, A. Zinin	03/2007	Errata	Inf.
4811	OSPF Out-of-Band Link State Database (LSDB) Resynchronization	L. Nguyen, A. Roy, A. Zinin	03/2007		Inf.
4750	OSPF Version 2 Management Information Base	D. Joyal, Ed., P. Galecki, Ed., S. Giacalone, Ed., R. Coltun, F. Baker	12/2006	Obs. 1850	Prop. std.
4577	OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)	E. Rosen, P. Psenak, P. Pillay-Esnault	06/2006	Upd. RFC4364, Errata	Prop. std.
4552	Authentication/Confidentiality for OSPFv3	M. Gupta, N. Melam	06/2006		Prop. std.
4222	Prioritized Treatment of Specific OSPF Version 2 Packets and Congestion Avoidance	G. Choudhury, Ed.	10/2005		Best current practice
4203	OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)	K. Kompella, Ed., Y. Rekhter, Ed.	10/2005	Upd. RFC3630	Prop. std.
4167	Graceful OSPF Restart Implementation Report	A. Lindem	10/2005		Inf.
4136	OSPF Refresh and Flooding Reduction in Stable Topologies	P. Pillay-Esnault	07/2005		Inf.
4063	Considerations When Using Basic OSPF Convergence Benchmarks	V. Manral, R. White, A. Shaikh	01.04.05		Inf.
4062	OSPF Benchmarking Terminology and Concepts	V. Manral, R. White, A. Shaikh	04/2005		Inf.
4061	Benchmarking Basic OSPF Single Router Control Plane Convergence	V. Manral, R. White, A. Shaikh	04/2005		Inf.
3883	Detecting Inactive Neighbors over OSPF Demand Circuits (DC)	S. Rao, A. Zinin, A. Roy	10/2004	Upd. RFC1793	Prop. std.
3630	Traffic Engineering (TE) Extensions to OSPF Version 2	D. Katz, K. Kompella, D. Yeung	09/2003	Upd. RFC2370, Upd. RFC4203	Prop. std.
3623	Graceful OSPF Restart	J. Moy, P. Pillay-Esnault, A. Lindem	11/2003		Prop. std.
3509	Alternative Implementations of OSPF Area Border Routers	A. Zinin, A. Lindem, D. Yeung	04/2003		Inf.

RFC	Title	Author	Date	Info	Status
3166	Request to Move RFC 1403 to Historic Status	D. Meyer, J. Scudder	08/2001		Inf.
3137	OSPF Stub Router Advertisement	A. Retana, L. Nguyen, R. White, A. Zinin, D. McPherson	06/2001		Inf.
3101	The OSPF Not-So-Stubby Area (NSSA) Option	P. Murphy	01/2003	Obs. RFC1587	Prop. std.
2844	OSPF over ATM and Proxy-PAR	T. Przygienda, P. Droz, R. Haas	05/2000		Exp.
2740	OSPF for IPv6	R. Coltun, D. Ferguson, J. Moy	12/1999	Obs. RFC5340, Errata	Prop. std.
2676	QoS Routing Mechanisms and OSPF Extensions	G. Apostolopoulos, S. Kama, D. Williams, R. Guerin, A. Orda, T. Przygienda	08/1999	Errata	Exp.
2370	The OSPF Opaque LSA Option	R. Coltun	07/1998	Obs. RFC5250, Upd. RFC3630	Prop. std.
2329	OSPF Std.ization Report	J. Moy	04/1998		Inf.
2178	OSPF Version 2	J. Moy	07/1997	Obs. RFC1583, Obs. RFC2328	Draft Std.
2154	OSPF with Digital Signatures	S. Murphy, M. Badger, B. Wellington	06/1997		Exp.
1850	OSPF Version 2 Management Information Base	F. Baker, R. Coltun	11/1995	Obs. RFC1253, Obs. RFC4750	Draft Std.
1793	Extending OSPF to Support Demand Circuits	J. Moy	04/1995	Upd. RFC3883	Prop. std.
1765	OSPF Database Overflow	J. Moy	03/1995		Exp.
1745	BGP4/IDRP for IP—OSPF Interaction	K. Varadhan, S. Hares, Y. Rekhter	12/1994		Hist.
1587	The OSPF NSSA Option	R. Coltun, V. Fuller	03/1994	Obs. RFC3101	Prop. std.
1586	Guidelines for Running OSPF Over Frame Relay Networks	O. deSouza, M. Rodrigues	03/1994		Inf.
1585	MOSPF & Analysis and Experience	J. Moy	03/1994		Inf.
1584	Multicast Extensions to OSPF	J. Moy	03/1994		Hist.
1583	OSPF Version 2	J. Moy	03/1994	Obs. RFC1247, Obs. RFC2178	Draft Std.
1403	BGP OSPF Interaction	K. Varadhan	01/1993	Obs. RFC1364	Hist.
1370	Applicability Statement for OSPF	Internet Architecture Board, L. Chapin	10/1992		Hist.

RFC	Title	Author	Date	Info	Status
1364	BGP OSPF Interaction	K. Varadhan	09/1992	Obs. RFC1403	Prop. std.
1253	OSPF Version 2 Management Information Base	F. Baker, R. Coltun	08/1991	Obs. RFC1252, Obs. RFC1850	Prop. std.
1252	OSPF Version 2 Management Information Base	F. Baker, R. Coltun	08/1991	Obs. RFC1248, Obs. RFC1253	Prop. std.
1248	OSPF Version 2 Management Information Base	F. Baker, R. Coltun	07/1991	Obs. RFC1252, Upd. RFC1349	Prop. std.
1247	OSPF Version 2	J. Moy	07/1991	Obs. RFC1131, Obs. RFC1583, Upd. RFC1349	Draft Std.
1246	Experience with the OSPF Protocol	J. Moy	07/1991		Inf.
1245	OSPF Protocol Analysis	J. Moy	07/1991		Inf.
1131	OSPF specification	J. Moy	10/1989	Obs. RFC1247	Prop. std.

B.2 BeeHiver Wrapper – Testresults

In this appendix we present the test results which we gained during development of the BeeHive wrapper. Remember that Farooq implemented a very abstract simulation environment, lacking typical features, such as TCP/IP protocols, IP addresses, or realistic router queues. Thus, in order to integrate the existing implementation of BeeHive into our realistic simulation environment we implemented a wrapper that translates, for example, IP addresses into integer node-IDs that are used in the original BeeHive implementation.

We do not think that the wrapper affects BeeHive's performance. However, to ensure this, we have ported Farooq's abstract traffic generator to our simulation environment as well. This allows us to execute the same simulations as Farooq has executed, albeit in our realistic simulation environment.

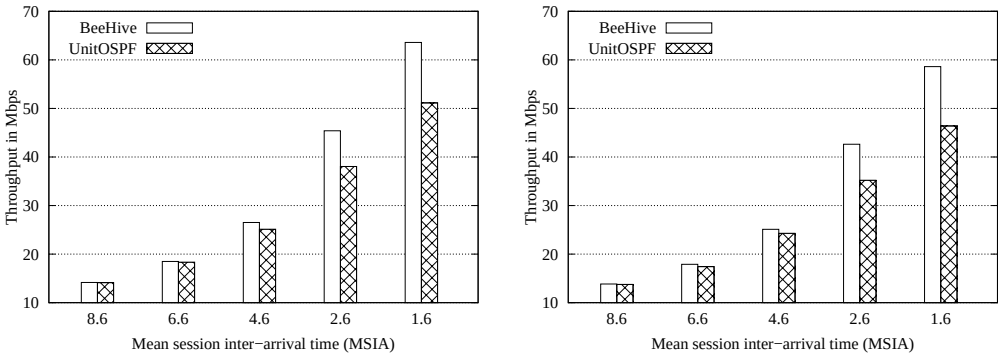
For our tests, we use the network topology NTTNet and specify the traffic in accordance to Farooq settings. Thus, each node generates traffic with MPIA = 0.005 sec, sessionSize = 2130000 bit, and packetSize = 512 byte. The buffer size in each router is set to 1000 packets. The value for MSIA is gradually decreased from 8.6 to 1.6, in order to continuously increase the traffic load.

We have analysed the performance results with regard to the parameters throughput, the percentage of successfully delivered packets, and the packet delay. In the following we review each of these parameters comparing the results using Farooq's simulation environment to the results from our simulation environment. Keep in mind that we do not expect 100% equal results, since the simulation environments differ significantly. However, the general characteristics of the algorithms should be similar and thus visible in the graphs.

The results regarding throughput are displayed in Figure B.1. In essence, they show a very similar behaviour for OSPF and BeeHive in both simulation environments, though both algorithms perform slightly worse in the realistic model, especially with increasing traffic.

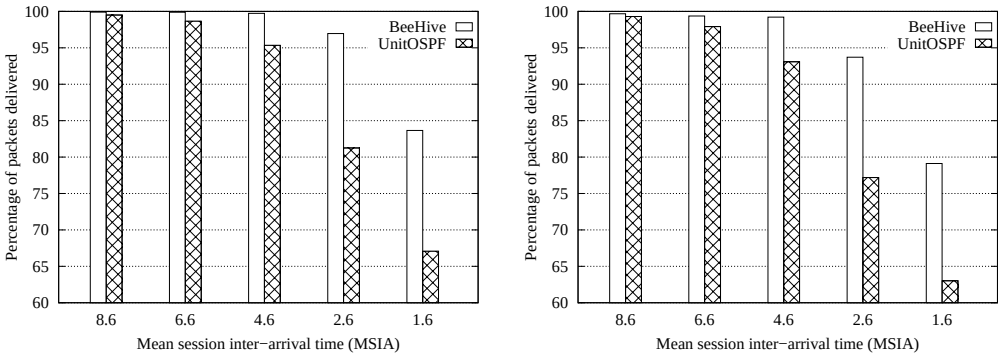
Figure B.2 illustrates the results with regard to the percentage of the data packets that are delivered successfully. Again no significant differences exist in the performance values. However, both algorithms again perform slightly worse in the realistic simulation model.

Finally, the delay results, which are illustrated in Figure B.3, ascertain the tendency that in essence the behaviour stays the same, though the performance is slightly worse in the realistic model. One reason for this performance decrease results from the realistic data packets our simulation environment uses. For example, an UDP packet increases the packet length at least by 28 byte, i. e. 8 byte for the UDP header and 20 byte for the IP header. Thus, the individual packets are generally larger in the realistic simulation environment. In summary, the results confirm our assumption that we ported BeeHive correctly to the realistic simulation model.



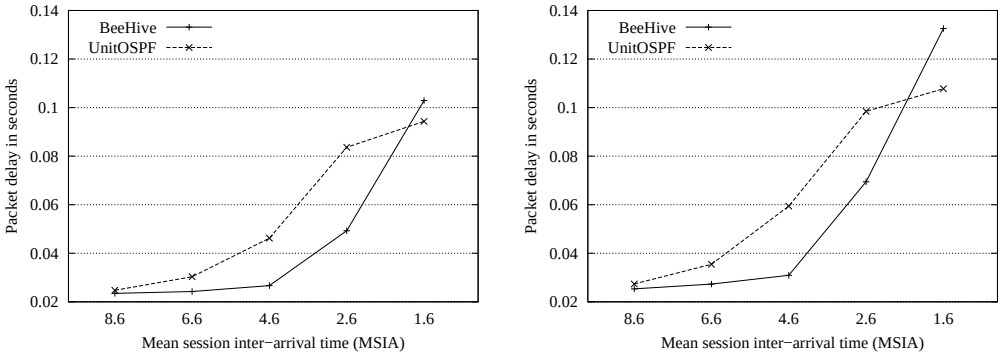
(a) *Farooq's model* **(b)** *Realistic model*

Figure B.1: *BeeHive wrapper simulation results: throughput*



(a) *Farooq's model* **(b)** *Realistic model*

Figure B.2: *BeeHive wrapper simulation results: percentage delivered*



(a) *Farooq's model* **(b)** *Realistic model*

Figure B.3: *BeeHive wrapper simulation results: delay*

Curriculum Vitae

My CV is available upon request.

Personal Publications

[1] Buschmann, A., Werner, C., Schmidt, S., and Fischer, S. Protokollunterstützung für SOAP Web Services auf mobilen Geräten. *PIK - Praxis der Informationsverarbeitung und Kommunikation, Themenheft Web Services 27*, 3 (2004).

[2] Fischer, S., Walther, U., Schmidt, S., and Werner, C. *Linux-Netzwerke: Aufbau, Administration, Sicherung*, 2 ed. Millin Verlag, 2005.

[3] Hellbrück, H., Lipphardt, M., Pfisterer, D., Ransom, S., and Fischer, S. Praxiserfahrungen mit MarathonNet - Ein mobiles Sensornetz im Sport. *PIK - Praxis der Informationsverarbeitung und Kommunikation 29*, 4 (2006), 195 – 202.

[4] Lipphardt, M., Hellbrück, H., Pfisterer, D., Ransom, S., and Fischer, S. Practical experiences on mobile inter-body-area-networking. In *Proceedings of the ICST 2nd international conference on Body area networks (BodyNets'07)* (2007).

[5] Ransom, S., Pfisterer, D., and Fischer, S. Comprehensible security synthesis for wireless sensor networks. In *Proceedings of the 3rd international workshop on middleware for sensor networks (MidSens'08)* (2008).

[6] Ransom, S., Pfisterer, D., and Fischer, S. Making Security Useable in Wireless Sensor Networks. In *Proceedings of the International Workshop on Sensor Network Engineering (IWSNE'08)* (2008).

[7] Ransom, S., and Werner, C. Towards Data-Centric Security in Ubiquitous Computing Environments. In *Proceedings of the First International Workshop on Embedded Data-Centric Systems (EDACS'09)* (2009).

[8] Schmidt, S., Krahn, H., Fischer, S., and Wätjen, D. A Security Architecture for Mobile Wireless Sensor Networks. In *Proceedings of the 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS'04)* (2004).

List of Tables

List of Figures

List of Algorithms

List of Definitions

List of Abbreviations

ABE	Attribute-Based Encryption
AC	Autonomic Computing
ACO	Ant Colony Optimization
AES	Advanced Encryption Standard
AIS	Artificial Immune System
AmI	Ambient Intelligence
API	Application Programming Interface
APPLE	A Posteriori Policy Enforcement
BGP	Border Gateway Protocol
BSI	Bundesamt für Sicherheit in der Informationstechnik
CA	Certificate Authority
CBC	Cipher Block Chaining
CBC-MAC	Cipher Block Chaining Message Authentication Code
CCM	Counter with CBC-MAC
CPU	Central Processing Unit
CTR	Counter Mode
DES	Data Encryption Standard
DGA	Distributed Genetic Algorithm
DLM	Data Lifecycle Management
DNS	Domain Name System
DoS	Denial of Service
DRM	Digital Rights Management
EA	Evolutionary Algorithm
ECC	Elliptic Curve Cryptography
ECMP	Equal-cost Multipath
GARA	Genetic Adaptive Routing Algorithm
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HARPS	Hashed Random Preloaded Subsets
HIPAA	Health Insurance Portability and Accountability Act

HMAC	Hashed Message Authentication Code
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
ILM	Information Lifecycle Management
IP	Internet Protocol
IPTV	Internet Protocol Television
IT	Information Technology
ITU	International Telecommunication Union
JFSM	JiNao Finite State Machine
Kbit/sec	Kilobit per Second
KB	Kilobyte
LEAP	Localized Encryption and Authentication Protocol
LSA	Link-state Advertisement
MAC	Message Authentication Code
Mbit/sec	Megabit per Second
MB	Megabyte
MIPS	Million Instructions Per Second
MPIA	Mean Packet Inter-arrival Time
MPLS	Multiprotocol Label Switching
MSIA	Mean Session Inter-arrival Time
μ**TESLA**	"micro" version of the Timed, Efficient, Streaming, Loss-tolerant Authentication Protocol
NFC	Near Field Communication
NIST	National Institute of Standards & Technology (USA)
OC	Organic Computing
ODRL	Open Digital Rights Language
OS	Operating System
OSPF	Open Shortest Path First
OTAP	Over-the-air Programming
PC	Personal Computer
PESQ	Perceptual Evaluation of Speech Quality
PEVQ	Perceptual Evaluation of Video Quality
PKI	Public Key Infrastructure
PUF	Physically Unclonable Function
ReaSE	Realistic Simulation Environments for OMNeT++
REL	Rights Expression Language

RFC	Requests for Comments
RFID	Radio Frequency Identification
RTO	Retransmission Timeout
RTT	Round Trip Time
SHS	Secure Hash Standard
SLA	Service Level Agreement
SPF	Shortest Path First
SPINS	Security Protocols for Sensor Networks
SSL	Secure Sockets Layer
Tbit/sec	Terabit per Second
TC	Trusted Computing
TCP	Transport Control Protocol
TLS	Transport Layer Security
UC	Ubiquitous Computing
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
VoIP	Voice over IP
VPN	Virtual Private Network
WAN	Wide Area Network
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language
XrML	eXtensible rights Markup Language

Bibliography

[1] The 29 Palms Experiment: Tracking Vehicles with a UAV-Delivered Sensor Network. Experiment Website: `http://robotics.eecs.berkeley.edu/~pister/29Palms0103/`.

[2] Quagga Routing Software Suite. Website: `http://www.quagga.net/`.

[3] INET Framework for OMNeT++ 3.x. Website: `http://www.omnetpp.org/pmwiki/index.php?n=Main.INETFramework`., 2009.

[4] AG CST, Freie Universität Berlin. Scatterweb. Project homepage: `http://cst.mi.fu-berlin.de/projects/ScatterWeb/`.

[5] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. Wireless sensor networks: a survey. *Computer Networks 38*, 4 (2002), 393–422.

[6] Aljnidi, M., and Leneutre, J. A security policy system for mobile autonomic networks. In *Proceedings of the 1st International Conference on Autonomic Computing and Communication Systems (Autonomics'07)* (2007).

[7] Anderson, R. *Security Engineering – A Guide to Build Dependable Distributed Systems*. Wiley, 2001.

[8] Anderson, R., Chan, H., and Perrig, A. Key Infection: Smart Trust for Smart Dust. In *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP'04)* (2004).

[9] Anderson, R., and Kuhn, M. Tamper Resistance – A Cautionary Note. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce* (1996).

[10] Anderson, R., and Kuhn, M. Low Cost Attacks on Tamper Resistant Devices. In *Proceedings of the 5th International Workshop on Security Protocols* (1997).

[11] Atallah, M. J., Blanton, M., and Frikken, K. B. Efficient techniques for realizing geo-spatial access control. In *Proceedings of the 2nd Symposium on Information, computer and communications security (ASI-ACCS'07)* (2007).

[12] Avivzienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing 1*, 1 (2004), 11–33.

[13] Bakshi, A., Prasanna, V. K., Reich, J., and Larner, D. The abstract task graph: a methodology for architecture-independent programming of networked sensor systems. In *Proceedings of the Workshop on End-to-end, Sense-and-respond Systems, Applications and Services* (2005).

[14] Balfe, S., Gallery, E., Mitchell, C. J., and Paterson, K. G. Challenges for trusted computing. *IEEE Security and Privacy 6*, 6 (2008), 60–66.

[15] Barán, B., and Sosa, R. A New Approach for AntNet Routing. In *Proceedings of the 9th International Conference on Computer Communications and Networks (ICCCN'00)* (2000).

[16] Barán, B., and Sosa, R. AntNet: Routing Algorithm for Data Networks based on Mobile Agents. *Inteligencia Artificial 5*, 12 (2001), 75–84.

[17] Barbir, A., Murphy, S., and Yang, Y. Generic Threats to Routing Protocols. RFC 4593 (Informational), Oct. 2006.

[18] Bardram, J. Hospitals of the Future-Ubiquitous Computing Support for Medical Work in Hospitals. In *Proceedings of the 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications (UbiHealth'03)* (2003).

[19] Barker, E., Barker, W., Burr, W., Polk, W., Smid, M., and US National Institute of Standards & Technology. Recommendation for Key Management - Part 1: General (Revised). NIST Special Publication 800-57, Mar. 2007. Available online. `http://csrc.nist.gov/groups/ST/toolkit/documents/SP800-57Part1_3-8-07.pdf`.

[20] Bayuk, J. Data-centric security. *Computer Fraud & Security 2009*, 3 (Mar. 2009), 7–11.

[21] Becher, A., Benenson, Z., and Dornseif, M. Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks. In *Proceedings of the 3rd International Conference on Security in Pervasive Computing (SPC'06)* (2006), pp. 104–118.

[22] Benenson, Z., Cholewinski, P. M., and Freiling, F. C. *Wireless Sensor Network Security*. IOS Press, 2008, ch. Vulnerabilities and Attacks in Wireless Sensor Networks, pp. 22–43.

[23] Bethencourt, J., Sahai, A., and Waters, B. Ciphertext-Policy Attribute-Based Encryption. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'07)* (2007).

[24] Beyer, H.-G., Brucherseifer, E., Jakob, W., Pohlheim, H., Sendhoff, B., and Binh To, T. VDI/VDE-Richtlinie 3550, Blatt 3: Evolutionäre Algorithmen – Begriffe und Definitionen. In *VDI/VDE-Handbuch Regelungstechnik*, VDI Verein Deutscher Ingenieure and (VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA)), Eds. Beuth Verlag, Berlin, 2003. Available online. `http://ls11-www.informatik.uni-dortmund.de/people/beyer/EA-terminologie/term.html`.

[25] Beyer, H.-G., and Schwefel, H.-P. Evolution strategies – A comprehensive introduction. *Natural Computing: an international journal 1*, 1 (2002), 3–52.

[26] Bhatia, M., Manral, V., Fanto, M., White, R., Li, T., Barnes, M., and Atkinson, R. OSPFv2 HMAC-SHA Cryptographic Authentication. Internet Draft (draft-ietf-ospf-hmac-sha-04.txt), 6 May 2009. Work in Progress. Available online. `http://www.ietf.org/internet-drafts/draft-ietf-ospf-hmac-sha-04.txt`.

[27] Biham, E., Anderson, R. J., and Knudsen, L. R. Serpent: A New Block Cipher Proposal. In *Proceedings of the 5th International Workshop on Fast Software Encryption (FSE'98)* (1998).

[28] Bizer, J., Dingel, K., Fabian, B., Günther, O., Hansen, M., Klafft, M., Möller, J., and Spiekermann, S. TAUCIS – Technikfolgenabschätzung: Ubiquitäres Computing und Informationelle Selbstbestimmung. Studie im Auftrag des Bundesministeriums für Bildung und Forschung, July 2006. Available online. `https://www.datenschutzzentrum.de/taucis/ita_taucis.pdf`.

[29] Blass, E.-O., and Zitterbart, M. Towards Acceptable Public-Key Encryption in Sensor Networks. In *Proceedings of the 2nd International Workshop on Ubiquitous Computing (IWUC'05)* (2005).

[30] Blom, R. An Optimal Class of Symmetric Key Generation Systems. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT'84)* (1984).

[31] Blundo, C., Santis, A., Herzberg, A., Kutton, S., Vaccaro, U., and Yung, M. Perfectly-Secure Key Distribution for Dynamic Conferences. In *Proceedings of the 12th International Cryptology Conference (CRYPTO'92)* (1992).

[32] Bohge, M., and Renwanz, M. A realistic VoIP traffic generation and evaluation tool for OMNeT++. In *Proceedings of the 1st International OMNeT++ Workshop* (2008).

[33] Bohge, M., and Trappe, W. An authentication framework for hierarchical ad hoc sensor networks. In *Proceedings of the 2nd ACM Workshop on Wireless Security (WiSe'03)* (2003).

[34] Bowers, S., McPhillips, T. M., Ludäscher, B., Cohen, S., and Davidson, S. B. A Model for User-Oriented Data Provenance in Pipelined Scientific Workflows. In *Proceedings of the International Provenance and Annotation Workshop (IPAW'06)* (2006).

[35] Braden, R. Requirements for Internet Hosts - Communication Layers. RFC 1122 (Standard), Oct. 1989. Updated by RFCs 1349, 4379.

[36] Braun, U., Shinnar, A., and Seltzer, M. Securing Provenance. In *Proceedings of the 3rd USENIX Workshop on Hot topics in Security (HotSec'08)* (2008).

[37] Bundesamt für Sicherheit in der Informationstechnik. *Pervasive Computing: Entwicklungen und Auswirkungen.* SecuMedia Verlags-GmbH, 2006.

[38] Buriol, L., Resende, M., Ribeiro, C., and Thorup, M. A Hybrid Genetic Algorithm for the Weight Setting Problem in OSPF/IS-IS Routing. *Networks 46*, 1 (2005), 36–56.

[39] Burwick, C., Coppersmith, D., D'Avignon, E., Gennaro, R., Halevi, S., Jutla, C., Jr, S. M. M., O'Connor, L., Peyravian, M., Luke, J., Peyravian, O. M., Stafford, D., and Zunic, N. MARS - a candidate cipher for AES. NIST AES Proposal, 1999. Available online. `http://www.research.ibm.com/security/mars.ps`.

[40] Buschmann, C., and Pfisterer, D. iSense: A modular hardware and software platform for wireless sensor networks. Technical report, 6. Fachgespräche Drahtlose Sensornetze der GI/ITK Fachgruppe Kommunikation und Verteilte Systeme, 2007.

[41] Buttyan, L., and Hubaux, J.-P. *Security and Cooperation in Wireless Networks: Thwarting Malicious and Selfish Behavior in the Age of Ubiquitous Computing.* Cambridge University Press, 2007.

[42] Cagalj, M., Capkun, S., and pierre Hubaux, J. Wormhole-based anti-jamming techniques in sensor networks. *IEEE Transactions on Mobile Computing 6*, 1 (2007), 100–114.

[43] Caro, G. D., and Dorigo, M. AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research 9* (1998), 317–365.

[44] Caro, G. D., and Dorigo, M. Two Ant Colony Algorithms For Best-Effort Routing In Datagram Networks. In *Proceedings of the 10th International Conference on Parallel and Distributed Computing and Systems (PDCS'98)* (1998).

[45] Caronni, G. Walking the web of trust. In *Proceedings of the 9th International Workshops on Enabling Technologies (WETICE '00)* (2000).

[46] Carvalho, M. Security in mobile ad hoc networks. *IEEE Security and Privacy 6*, 2 (2008), 72–75.

[47] Çamtepe, S. A., and Yener, B. *Wireless Sensor Network Security.* IOS Press, 2008, ch. Key Management in Wireless Sensor Networks, pp. 110–141.

[48] Chadwick, D. W., and Lievens, S. F. Enforcing "sticky" security policies throughout a distributed application. In *Proceedings of the 2008 Workshop on Middleware Security (MidSec'08)* (2008).

[49] Chan, H., and Perrig, A. PIKE: peer intermediaries for key establishment in sensor networks. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)* (2005).

[50] Chan, H., Perrig, A., and Song, D. Random key predistribution schemes for sensor networks. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P '03)* (2003).

[51] Chan, H., Perrig, A., and Song, D. Secure hierarchical in-network aggregation in sensor networks. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS'06)* (2006).

[52] Chang, H.-Y., Wu, S. F., and Jou, Y. F. Real-time protocol analysis for detecting link-state routing protocol attacks. *ACM Transactions on Information and System Security (TISSEC) 4*, 1 (Feb. 2001), 1 – 36.

[53] Chew, E., Swanson, M., Stine, K., Bartol, N., Brown, A., and Robinson, W. Performance Measurement Guide for Information Security. NIST Special Publication (SP) 800-55 (Rev.1), July 2008.

[54] Cisco Systems. Cisco Visual Networking Index: Forecast and Methodology, 2008-2013. Cisco Whitepaper C11-481360. Available online. `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf`, June 2009.

[55] Coltun, R., Ferguson, D., Moy, J., and Lindem, A. OSPF for IPv6. RFC 5340 (Proposed Standard), July 2008.

[56] Costa, P., Coulson, G., Gold, R., Lad, M., Mascolo, C., Mottola, L., Picco, G., Sivaharan, T., Weerasinghe, and N., Zachariadis, S. The runes middleware for networked embedded systems and its application in a disaster management scenario. In *Proceedings of the 5th International Conference on Pervasive Computing and Communications (PerCom'07)* (2007).

[57] Coviello, A. Thinking Security. Keynote at RSA Conference 2008, 2008. Transcript available online. `http://www.rsa.com/innovation/docs/coviellokeynote2008.pdf`.

[58] Cranor, L. F., and Garfinkel, S., Eds. *Security and usability: designing secure systems that people can use.* O'Reilly Media, 2005.

[59] Dai, C., Lin, D., Bertino, E., and Kantarcioglu, M. An Approach to Evaluate Data Trustworthiness Based on Data Provenance. In *Proceedings of the 5th VLDB Workshop at Secure Data Management (SDM'08)* (2008).

[60] De Ghein, L. *MPLS Fundamentals.* Cisco Press, 2006.

[61] Debar, H., Dacier, M., and Wespi, A. Towards a taxonomy of intrusion-detection systems. *Computer Networks 31*, 8 (1999), 805–822.

[62] Deering, S., and Hinden, R. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), Dec. 1998. Updated by RFC 5095.

[63] Deng, J., Han, R., and Mishra, S. Countermeasures Against Traffic Analysis Attacks in Wireless Sensor Networks. In *Proceedings of the 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)* (2005), pp. 113–126.

[64] Deng, J., Han, R., and Mishra, S. Defending against path-based dos attacks in wireless sensor networks. In *Proceedings of the 3rd ACM Workshop on Security of Ad hoc and Sensor Networks (SASN'05)* (2005).

[65] Diffie, W., and Hellman, M. Exhaustive cryptanalysis of the NBS Data Encryption Standard. *IEEE Computer 10*, 6 (1977), 74–84.

[66] Ditlea, S. The electronic paper chase. *Scientific American 285*, 5 (2001), 50–55.

[67] Dolev, D., and Yao, A. On the security of public key protocols. *IEEE Transactions on Information Theory 29*, 2 (1983), 198–208.

[68] Du, W., Deng, J., Han, Y. S., and Varshney, P. K. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)* (2003).

[69] Dworkin, M. Recommendation for Block Cipher Modes of Operation. NIST Special Publication 800-38A, 2001. Available online. `http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf`.

[70] Eastlake 3rd, D., and Kaufman, C. Domain Name System Security Extensions. RFC 2065 (Proposed Standard), Jan. 1997. Obsoleted by RFC 2535.

[71] Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics and Chip Design.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1998.

[72] Ericsson, M., Resende, M., and Pardalos, P. A Genetic Algorithm for the Weight Setting Problem in OSPF Routing. *Journal of Combinatorial Optimization 6*, 3 (2002), 299–333.

[73] Eschenauer, L., and Gligor, V. D. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02)* (2002).

[74] Etalle, S., and Winsborough, W. H. A posteriori compliance control. In *Proceedings of the 12th ACM symposium on access control models and technologies (SACMAT'07)* (2007).

[75] European Parliament. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. Official Journal of the European Communities, 23 Nov. 1995. Available online: `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:EN:NOT`.

[76] Farooq, M. *From the Wisdom of the Hive to Intelligent Routing in Telecommunication Networks - A Step towards Intelligent Network Management through Natural Engineering.* PhD thesis, Universität Dortmund, 2006.

[77] Floyd, S., and Paxson, V. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking (TON) 9*, 4 (2001), 392–403.

[78] Fortz, B., Rexford, J., and Thorup, M. Traffic Engineering With Traditional IP Routing Protocols. *IEEE Communications Magazine 40*, 10 (2002), 118–124.

[79] Fortz, B., and Thorup, M. Internet Traffic Engineering by Optimizing OSPF Weights. In *Proceedings of the 19th Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)* (2000).

[80] Fromm, J., and Zapf, M. "Selbst"-Eigenschaften in verteilten Systemen. *Praxis der Informationsverarbeitung und Kommunikation (PIK) 28*, 4 (2005), 189–198.

[81] Gamage, C., Bicakci, K., Crispo, B., and Tanenbaum, A. S. Security for the Mythical Air-Dropped Sensor Network. In *Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC'06)* (2006).

[82] Gamer, T. ReaSE - Realistic Simulation Environments for OMNeT++. Website: `https://projekte.tm.uka.de/trac/ReaSE/`.

[83] Gamer, T., and Scharf, M. Realistic Simulation Environments for IP-based Networks. In *Proceedings of 1st International Workshop on OMNeT++* (2008).

[84] Ganek, A., and Corbi, T. The dawning of the automatic computing era. *IBM Systems Journal 42*, 1 (2003).

[85] Google. Google Latitude. Project website: `http://www.google.com/intl/en_us/latitude/intro.html`, 2009.

[86] Goss, S., Aron, S., Deneubourg, J.-L., and Pasteels, J. M. Self-organized Shortcuts in the Argentine Ant. *Naturwissenschaften 76*, 12 (1989), 579–581.

[87] Goyal, V., Lu, S., Sahai, A., and Waters, B. Black-box accountable authority identity-based encryption. In *Proceedings of the 15th Conference on Computer and Communications Security (CCS'08)* (2008).

[88] Goyal, V., Pandey, O., Sahai, A., and Waters, B. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th Conference on Computer and Communications Security (CCS'06)* (2006).

[89] Grandison, T., Bilger, M., O'Connor, L., Graf, M., Swimmer, M., Schunter, M., Wespi, A., and Zunic, N. Elevating the Discussion on Security Management: The Data Centric Paradigm. In *Proceedings of the 2nd International Workshop on Business-Driven IT Management (BDIM'07)* (2007).

[90] Greenstein, B., Mar, C., Pesterev, A., Farshchi, S., Kohler, E., Judy, J., and Estrin, D. Capturing high-frequency phenomena using a bandwidth-limited sensor network. In *Proceedings of the 4th international conference on embedded networked sensor systems (SenSys'06)* (2006).

[91] Guajardo, J., Blümel, R., Krieger, U., and Paar, C. Efficient Implementation of Elliptic Curve Cryptosystems on the TI MSP 430x33x Family of Microcontrollers. In *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography (PKC'01)* (2001).

[92] Gupta, M., and Melam, N. Authentication/Confidentiality for OSPFv3. RFC 4552 (Proposed Standard), June 2006.

[93] Gura, N., Patel, A., Wander, A., Eberle, H., and Shantz, S. C. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In *Proceedings of the 6th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'04)* (2004), pp. 119–132.

[94] Guth, S. *Digital Rights Management.* Springer Berlin / Heidelberg, 2003, ch. Rights Expression Languages, pp. 101–112.

[95] Han, S., Chang, E., Gao, L., and Dillon, T. Taxonomy of Attacks on Wireless Sensor Networks. In *Proceedings of the 1st European Conference on Computer Network Defence (EC2ND'05)* (2005).

[96] Hasan, R., Sion, R., and Winslett, M. Introducing secure provenance: problems and challenges. In *Proceedings of the 2007 ACM Workshop on Storage Security and Survivability (StorageSS'07)* (2007).

[97] Hauser, R., Przygienda, T., and Tsudik, G. Lowering security overhead in link state routing. *Computer Networks 31*, 9 (1999), 885–894.

[98] Heidemann, J., Silva, F., Intanagonwiwat, C., Govindan, R., Estrin, D., and Ganesan, D. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the 18th ACM symposium on Operating systems principles (SOSP'01)* (2001).

[99] Hennessy, S. D., Lauer, G. D., Zunic, N., Gerber, B., and Nelson, A. C. Data-centric security: Integrating data privacy and data security. *IBM Journal of Research and Development 53*, 2 (2009).

[100] Heyman, T., Scandariato, R., Huygens, C., and Joosen, W. Using security patterns to combine security metrics. *Proceedings of the International Workshop on Secure Software Engineering (SecSE'08)* (2008).

[101] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K. System architecture directions for networked sensors. *SIGOPS Operating Systems Review 34*, 5 (2000), 93–104.

[102] Hilty, L., Behrendt, S., Binswanger, M., Bruinink, A., Erdmann, L., Fröhlich, J., Köhler, A., Kuster, N., Som, C., and Würtenberger, F. Das Vorsorgeprinzip in der Informationsgesellschaft. Auswirkungen des Pervasive Computing auf Gesundheit und Umwelt. Studie des Zentrums für Technologiefolgen-Abschätzung, TA 46/2003, 2003.

[103] Hilty, M., Pretschner, A., Schaefer, C., and Walter, T. Enforcement for Usage Control – An Overview of Control Mechanisms. Tech. Rep. 1-ST-18, DoCoMo Euro-Labs, July 2006.

[104] Holger Krahn. Analyse leichtgewichtiger kryptographischer Algorithmen für Sensornetzwerke. Diplomarbeit, Institut für Betriebssysteme und Rechnerverbund, Technische Universität Braunschweig, Feb. 2004.

[105] Howard, J., and Longstaff, T. A Common Language for Computer Security Incidents. SAND D98-8667, Sandia National Laboratories, Livermore, CA, USA, 1998.

[106] Hu, L., and Evans, D. Secure Aggregation for Wireless Networks. In *Symposium on Applications and the Internet Workshops (SAINT'03)* (2003).

[107] Hu, V. C., Ferraiolo, D. F., and Kuhn, D. R. Assessment of Access Control Systems. NIST Interagency Report 7316, Sept. 2006.

[108] Huang, D., Cao, Q., Sinha, A., Schniederjans, M. J., Beard, C., Harn, L., and Medhi, D. New architecture for intra-domain network security issues. *Communications of the ACM 49*, 11 (2006), 64–72.

[109] Huang, D., Sinha, A., and Medhi, D. A double authentication scheme to detect impersonation attack in link state routing protocols. In *Proceedings of the International Conference on Communications (ICC'03)* (2003).

[110] Huang, H., and Wu, S. F. An integrated solution to protect link state routing against faulty intermediate routers. In *Proceedings of the 10th Network Operations and Management Symposium: Management of Integrated End-to-End Communications and Services (NOMS'06)* (2006).

[111] IBM White Paper. An architectural blueprint for autonomic computing (4th edition). Tech. rep., IBM Corporation, June 2006.

[112] Institut für Telematik Universität zu Lübeck, Institut für Telematik Universität Karlsruhe, and coalesenses GmbH. FleGSens - Sichere und flexible Grenz- und Liegenschaftsüberwachung durch drahtlose Sensornetze. Project homepage: `http://www.itm.uni-luebeck.de/projects/flegsens/index.html?lang=de`, 2009.

[113] International Data Corporation. The Diverse and Exploding Digital Universe, An Updated Forecast of Worldwide Information Growth Through 2011. Available online. `http://www.emc.com/collateral/analyst-reports/diverse-exploding-digital-universe.pdf`, 2008.

[114] International Organization for Standardization. Information and documentation — a reference ontology for the interchange of cultural heritage information. ISO 21127:2006, 2006.

[115] International Organization for Standardization (ISO). Information technology – Security techniques – Information security management measurements. ISO/IEC 27004. Standard under Development. http://www.iso.org.

[116] IST Advisory Group (ISTAG). Ambient Intelligence - From Vision to Reality. Available online. ftp://ftp.cordis.lu/pub/ist/docs/istag-ist2003_consolidated_report.pdf, 2003.

[117] Jamkhedkar, P. A., and Heileman, G. L. Digital rights management architectures. *Computers and Electrical Engineering 35*, 2 (2009), 376–394.

[118] Jamkhedkar, P. A., Heileman, G. L., and Martínez-Ortiz, I. The problem with rights expression languages. In *Proceedings of the Workshop on Digital Rights Management (DRM'06)* (2006).

[119] Jaquith, A. Data Security: Whose Job Is It Really? CSO Online. Available at http://www.csoonline.com/article/487261, 30 Mar. 2009.

[120] Jong, K. A. D. *Evolutionary Computation: A Unified Approach*. MIT Press, 2006.

[121] Jonsson, J., and Kaliski, B. Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447 (Informational), Feb. 2003.

[122] Jou, Y. F., Wu, S. F., Frank, Y., Shyhtsun, J., Wu, F., Gong, F., Cleaveland, W. R., and Sargor, C. Architecture design of a scalable intrusion detection system for the emerging network infrastructure. Tech. rep., CDRL A005, MCNC Information Technologies Division, Research Triangle Park, N.C. 27709, Apr. 1997.

[123] Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L. S., and Rubenstein, D. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. *ACM SIGOPS Operating Systems Review 36*, 5 (2002), 96–107.

[124] Kagal, L., Finin, T., and Joshi, A. Trust-Based Security in Pervasive Computing Environments. *IEEE Computer 34*, 12 (2001), 154–157.

[125] Kagal, L., Finin, T., and Joshi, A. A policy language for a pervasive computing environment. In *Proceedings of the 4th nternational Workshop on Policies for Distributed Systems and Networks (POLICY'03)* (2003).

[126] Kaps, J.-P. *Cryptography for Ultra-Low Power Devices*. PhD thesis, Worcester Polytechnic Institute, May 2006.

[127] Karjoth, G., Schunter, M., and Waidner, M. Platform for enterprise privacy practices: Privacy-enabled management of customer data. In *Proceedings of the 2nd International Workshop on Privacy enhancing technologies (PET'02)* (2002).

[128] Karlof, C., Sastry, N., and Wagner, D. TinySec: a link layer security architecture for wireless sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04)* (2004).

[129] Karlof, C., and Wagner, D. Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks 1*, 2-3 (2003), 293–315.

[130] Karlof, C., and Wagner, D. Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures. In *Proceedings othe 1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA'03)* (2003).

[131] Katz, D., Kompella, K., and Yeung, D. Traffic Engineering (TE) Extensions to OSPF Version 2. RFC 3630 (Proposed Standard), Sept. 2003. Updated by RFC 4203.

[132] Kennedy, J., Eberhart, R. C., and Yuhui, S. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

[133] Kent, S. IP Authentication Header. RFC 4302 (Proposed Standard), Dec. 2005.

[134] Kent, S. IP Encapsulating Security Payload (ESP). RFC 4303 (Proposed Standard), Dec. 2005.

[135] Kephart, J. O., and Chess, D. M. The Vision of Autonomic Computing. *IEEE Computer 36*, 1 (2003), 41–50.

[136] Kim, J., Bentley, P., Aickelin, U., Greensmith, J., Tedesco, G., and Twycross, J. Immune System Approaches to Intrusion Detection – A Review. *Journal of Natural Computing 6*, 4 (2007), 413–466.

[137] Kim, Y., Kang, J., Kim, D., Kim, E., Chong, P. K., and Seo, S. Design of a fence surveillance system based on wireless sensor networks. In *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems (Autonomics'08)* (2008).

[138] Knuth, D. *The Art of Programming*, vol. 2 Seminumerical Algorithms. Addison-Wesley, 1969.

[139] Koblitz, N. Elliptic Curve Cryptosystems. *Mathematics of Computation 48*, 177 (1987), 203–209.

[140] Koblitz, N., Menezes, A., and Vanstone, S. The state of elliptic curve cryptography. *Designs, Codes and Cryptography 19*, 2-3 (2000), 173–193.

[141] Krishnamachari, B., Estrin, D., and Wicker, S. B. The impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCSW'02)* (2002).

[142] Krukow, K., Nielsen, M., and Sassone, V. Trust models in ubiquitous computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 366*, 1881 (2008), 3781–3793.

[143] Ku, W., and Chi, C.-H. Survey on the Technological Aspects of Digital Rights Management. In *Proceedings of the 7th International Conference on Information Security (ISC'04)* (2004). (ext. version).

[144] Langheinrich, M. When trust does not compute – the role of trust in ubiquitous computing. In *Workshop on Privacy at Ubicomp 2003* (2003).

[145] LAN/MAN Standards Committee of the IEEE Computer Society. *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs).* Institute of Electrical and Electronics Engineers, Inc., 2006.

[146] Law, Y. W., Doumen, J., and Hartel, P. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Transactions on Sensor Networks 2*, 1 (2006), 65–93.

[147] Lee, J., and Stinson, D. R. Deterministic Key Predistribution Schemes for Distributed Sensor Networks. In *Proceedings of the 11th International Workshop on Selected Areas in Cryptography (SAC'04)* (2004).

[148] Lee, Y., Park, I., and Choi, Y. Improving TCP Performance in Multipath Packet Forwarding Networks. *Journal of Communications and Networks 4*, 2 (2002), 148–157.

[149] Leland, W. E., Taqqu, M. S., Willinger, W., and Wilson, D. V. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transactions on Networking 2*, 1 (1994), 1–15.

[150] Lenstra, A. K., and Verheul, E. R. Selecting Cryptographic Key Sizes. *Journal of Cryptography 14*, 4 (2001), 255–293.

[151] Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., and Culler, D. *Ambient Intelligence*. Springer, Berlin, 2005, ch. TinyOS: An Operating System for Sensor Networks, pp. 115–148.

[152] Liang, S., Zincir-Heywood, A. N., and Heywood, M. I. The Effect of Routing under Local Information using a Social Insect Metaphor. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)* (2002).

[153] Liang, S., Zincir-Heywood, A. N., and Heywood, M. I. Intelligent Packets For Dynamic Network Routing Using Distributed Genetic Algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'02)* (2002).

[154] Liang, S., Zincir-Heywood, A. N., and Heywood, M. I. Adding more intelligence to the network routing problem: AntNet and Ga-agents. *Applied Soft Computing 6*, 3 (2006), 244–257.

[155] Liu, D., and Ning, P. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)* (2003).

[156] Liu, D., and Ning, P. Location-based pairwise key establishments for static sensor networks. In *Proceedings of the 1st ACM Workshop on Security of ad hoc and Sensor Networks (SASN'03)* (2003).

[157] Lopez, J. Unleashing public-key cryptography in wireless sensor networks. *Journal of Computer Security 14*, 5 (2006), 469–482.

[158] Lough, D. L. *A Taxonomy of Computer Attacks wityh Applications to Wireless Networks*. PhD thesis, Virginia Polytechnic Institute and State University, Mar. 2001.

[159] Luk, M., Mezzour, G., Perrig, A., and Gligor, V. MiniSec: a secure sensor network communication architecture. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN'07)* (2007).

[160] Lukowicz, P., Timm-Giel, A., Lawo, M., and Herzog, O. WearIT@work: toward real-world industrial wearable computing. *IEEE Pervasive Computing 6*, 4 (2007), 8–13.

[161] Lyytinen, K., and Yoo, Y. Introduction to Issues and challenges in ubiquitous computing. *Communications of the ACM, SPECIAL ISSUE: Issues and challenges in ubiquitous computing 45*, 12 (2002), 62–65.

[162] Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., and Anderson, J. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (WSNA'02)* (2002).

[163] Manzo, M., Roosta, T., and Sastry, S. Time synchronization attacks in sensor networks. In *Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks (SASN'05)* (2005).

[164] Marrón, P. J., Saukh, O., Krüger, M., and Grosse, C. Sensor network issues in the Sustainable Bridges project. In *European Projects Session of the Second European Workshop on Wireless Sensor Networks (EWSN 2005)* (2005).

[165] Marsh, D. W., Baldwin, R. O., Mullins, B. E., Mills, R. F., and Grimaila, M. R. A security policy language for wireless sensor networks. *Journal of Systems and Software 82*, 1 (Jan. 2009), 101 – 111.

[166] Martinez, K., Riddoch, A., Hart, J., and Ong, R. *Intelligent Spaces*. Springer, 2006, ch. A sensor web for glaciers, pp. 125–138.

[167] Mattern, F. The vision and technical foundations of ubiquitous computing. *Upgrade 2*, 5 (Oct. 2001), 2–6.

[168] Mayer-Schönberger, V. Useful Void: The Art of Forgetting in the Age of Ubiquitous Computing. Tech. Rep. RWP07-022, Havard University, John F. Kennedy School of Government, 2007.

[169] McQuillan, J., and Walden, D. C. The ARPANET Design Decisions. *Computer Networks 1*, 5 (1977), 234–289.

[170] Montgomery, D., and Murphy, S. Toward secure routing infrastructures. *IEEE Security and Privacy 4*, 5 (2006), 84–87.

[171] Moore, G. Cramming more components onto integrated circuits. *Electronics 38*, 8 (1965), 114–117.

[172] Moreau, L., Freire, J., Futrelle, J., McGrath, R., Myers, J., and Paulson, P. The Open Provenance Model. Tech. rep., University of Southampton, 2007.

[173] Moreau, L., Freire, J., Futrelle, J., McGrath, R. E., Myers, J., and Paulson, P. The open provenance model: An overview. In *Proceedings of the 2nd International Provenance and Annotation Workshop (IPAW'08)* (2008).

[174] Moreau, L., Groth, P., Miles, S., Vazquez-Salceda, J., Ibbotson, J., Jiang, S., Munroe, S., Rana, O., Schreiber, A., Tan, V., and Varga, L. The provenance of electronic data. *Communications of the ACM 51*, 4 (Apr. 2008), 52–58.

[175] Moy, J. *OSPF: Anatomy of an Internet Routing Protocol*. Addison-Wesley Professional, 1998.

[176] Moy, J. OSPF Version 2. RFC 2328 (Standard), Apr. 1998.

[177] Müller, G., Accorsi, R., Höhn, S., Kähmer, M., and Strasser, M. *Die Informatisierung des Alltags – Leben in smarten Umgebungen*. Springer, 2007, ch. Sicherheit im Ubiquitous Computing: Schutz durch Gebote?, pp. 127–142.

[178] Müller, S., Katzenbeisser, S., and Eckert, C. Distributed Attribute-Based Encryption. In *Proceedings of the 11th International Conference on Information Security and Cryptology (ICISC'08)* (2008).

[179] Munetomo, M. Network routing with the Use of Evolutionary Methods. In *Computational intelligence in telecommunication networks*, W. Pedrycz and A. V. Vasilakos, Eds. CRC Press LLC, 2001, pp. 287–302.

[180] Munetomo, M., Takai, Y., and Sato, Y. An Adaptive Network Routing Algorithm Employing Path Genetic Operators. In *Proceedings of the 7th International Conference on Genetic Algorithms (ICGA'97)* (1997).

[181] Munetomo, M., Takai, Y., and Sato, Y. A Migration Scheme for the Genetic Adaptive Routing Algorithm. In *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics* (1998).

[182] Murphy, S., Badger, M., and Wellington, B. OSPF with Digital Signatures. RFC 2154 (Experimental), June 1997.

[183] Nechvatal, J., Barker, E., Bassham, L., Burr, W., Dworkin, M., Foti, J., and Roback, E. Report on the Development of the Advanced Encryption Standard (AES). *Journal of Research of the National Institute of Standards and Technology 106*, 6 (2001), 511–577. Available online. `http://nvl.nist.gov/pub/nistpubs/jres/106/3/j63nec.pdf`.

[184] Newsome, J., Shi, E., Song, D., and Perrig, A. The Sybil Attack in Sensor Networks: Analysis & Defenses. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN'04)* (2004).

[185] Nikolaidis, A., Tsekeridou, S., Tefas, A., and Solachidis, V. A survey on watermarking application scenarios and related attacks. In *Proceedings of the International Conference on Image Processing* (2001).

[186] OASIS. eXtensible Access Control Markup Language TC v2.0 (XACML). OASIS Standard, Feb. 2005. Available online. `http://www.oasis-open.org/specs/`.

[187] Öztürk, E., Hammouri, G., and Sunar, B. Towards Robust Low Cost Authentication for Pervasive Devices. In *Proceedings of the 6th Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM 2008)* (2008), pp. 170–178.

[188] Paradiso, J. A., and Starner, T. Energy Scavenging for Mobile and Wireless Electronics. *IEEE Pervasive Computing 4*, 1 (2005), 18–27.

[189] Park, U., and Heidemann, J. Provenance in sensornet republishing. In *Proceedings of the 2nd International Provenance and Annotation Workshop: Provenance and Annotation of Data and Processes (IPAW'08)* (2008).

[190] Patwardhan, A., Korolev, V., Kagal, L., and Joshi, A. Enforcing policies in pervasive environments. In *Proceedings of the 1st International Conference on Mobile and Ubiquitous Systems (MobiQuitous'04)* (2004).

[191] Pavlovic, D. Dynamics, Robustness and Fragility of Trust. In *Proceedings of the 5th International Workshop on Formal Aspects in Security and Trust (FAST'08)* (2008).

[192] Paxson, V., and Allman, M. Computing TCP's Retransmission Timer. RFC 2988 (Proposed Standard), Nov. 2000.

[193] Payment Card Industry (PCI). Requirements and Security Assessment Procedures (Version 1.2). Data Security Standard, Oct. 2008. Available online. `https://www.pcisecuritystandards.org/security_standards/pci_dss_download.html`.

[194] Perkins, C., Hodson, O., and Hardman, V. A Survey of Packet Loss Recovery Techniques for Streaming Media. *IEEE Network Magazine 12*, 5 (1998), 40–48.

[195] Perlman, R. *Network layer protocols with Byzantine robustness.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Aug. 1988.

[196] Perrig, A., Szewczyk, R., Wen, V., Culler, D. E., and Tygar, J. D. SPINS: Security protocols for sensor networks. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MOBICOM'01)* (2001).

[197] Petitcolas, F. A. *Digital Rights Management.* Springer Berlin / Heidelberg, 2003, ch. Digital Watermarking, pp. 81–92.

[198] Petrocelli, T. *Data Protection and Information Lifecycle Management.* Prentice Hall, 2005.

[199] Pfisterer, D. *Comprehensive Development Support for Wireless Sensor Networks.* PhD thesis, Institute of Telematics, University of Luebeck, Germany, 2007.

[200] Pfisterer, D., Buschmann, C., Hellbrück, H., and Fischer, S. Data-type centric middleware synthesis for wireless sensor network application development. In *Proceedings of the Fifth Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2006)* (2006).

[201] Pfisterer, D., Hellbrück, H., and Fischer, S. FABRIC: Towards data type-centric middleware synthesis. In *Proceedings of the International Conference on Distributed Computing (DCOSS'06)* (2006).

[202] Pister, K., Kahn, J., and Boser, B. SMART DUST: Autonomous sensing and communication in a cubic millimeter. Project Homepage: `http://robotics.eecs.berkeley.edu/~pister/SmartDust/`.

[203] Platon, E., and Sei, Y. Security software engineering in wireless sensor networks. *Progress in Informatics 5* (Mar. 2008), 49–64.

[204] Pretschner, A., Hilty, M., Schütz, F., Schaefer, C., and Walter, T. Usage Control Enforcement: Present and Future. *IEEE Security and Privacy 6*, 4 (2008), 44–53.

[205] Przydatek, B., Song, D. X., and Perrig, A. SIA: secure information aggregation in sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys'03)* (2003).

[206] R. Iannella. Open Digital Rights Language (ODRL), Version 1.1. Web resource. `http://odrl.net`, 2002.

[207] Rajan, B. The Enlightened Era of Enterprise Security. *Proceedings of the 4th International Conference on Information Systems Security (ICISS'08)* (2008). (Invited Talk).

[208] Ramkumar, M., and Memon, N. HARPS: Hashed random preloaded subset key distribution. *Cryptology ePrint Archive* (2003).

[209] Ransom, S. Evolutionäre Algorithmen für autonome Systeme – Arbeitspaket 2.2.1: Bestandsaufnahme vorhandener Evaluationen. Tech. rep., Institute of Telematics, University of Lübeck, 2008. Available on demand from the author.

[210] Ransom, S., Pfisterer, D., and Fischer, S. Comprehensible security synthesis for wireless sensor networks. In *Proceedings of the 3rd international workshop on middleware for sensor networks (MidSens'08)* (2008).

[211] Ransom, S., Pfisterer, D., and Fischer, S. Making Security Useable in Wireless Sensor Networks. In *Proceedings of the International Workshop on Sensor Network Engineering (IWSNE'08)* (2008).

[212] Ransom, S., and Werner, C. Towards Data-Centric Security in Ubiquitous Computing Environments. In *Proceedings of the First International Workshop on Embedded Data-Centric Systems (EDACS'09)* (2009).

[213] Ravi, S., Raghunathan, A., Kocher, P., and Hattangady, S. Security in embedded systems: Design Challenges. *ACM Transactions on Embedded Compututer Systems 3*, 3 (2004), 461–491.

[214] Rekhter, Y., Li, T., and Hares, S. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), Jan. 2006.

[215] Rexford, J. *Handbook of Optimization in Telecommunications*. Springer Science + Business Media, 2006, ch. Route optimization in IP networks.

[216] Rivest, R. The MD5 Message-Digest Algorithm. RFC 1321 (Informational), Apr. 1992.

[217] Rivest, R. L. The RC5 Encryption Algorithm. In *Proceedings of the 2nd International Workshop on Fast Software Encryption (FSE'94)* (1994).

[218] Rivest, R. L., Robshaw, M. J. B., and Yin, Y. L. The RC6 Block Cipher. NIST AES Proposal, 2000.

[219] Rivest, R. L., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, 2 (1978), 120–126.

[220] Roman, R., Alcaraz, C., and Lopez, J. A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes. *Mobile Networks and Applications 12*, 4 (2007), 231–244.

[221] Römer, K., and Mattern, F. The design space of wireless sensor networks. *IEEE Wireless Communications 11*, 6 (2005), 54–61.

[222] Roosta, T., Shieh, S. P., and Sastry, S. Taxonomy of Security Attacks in Sensor Networks and Countermeasures. In *Proceedings of the 1st International Conference on System Integration and Reliability Improvements (SIRI'06)* (2006).

[223] Roughan, M., Thorup, M., and Zhang, Y. Traffic engineering with estimated traffic matrices. In *Proceedings of the 3rd Conference on Internet Measurement (IMC'03)* (2003).

[224] Sabahi, F., and Movaghar, A. Intrusion Detection: A Survey. In *Proceedings of the 3rd International Conference on Systems and Networks Communications (ICSNC'08)* (2008).

[225] Sabbah, E., Kang, K.-D., Abu-Ghazaleh, N., Majeed, A., and Liu, K. An application-driven approach to designing secure wireless sensor networks. *Wireless Communications & Mobile Computing 8*, 3 (2008), 369–384.

[226] Sadeghi, A.-R., and Stüble, C. Property-based attestation for computing platforms: caring about properties, not mechanisms. In *Proceedings of the 2004 Workshop on New Security Paradigms (NSPW'04)* (2004).

[227] Sahai, A., and Waters, B. Fuzzy Identity-Based Encryption. In *Proceedings of the 24th annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'05)* (2005).

[228] Salles, R. M., and Rolla, V. G. Efficient Routing Heuristics for Internet Traffic Engineering. *Computer Communications 30*, 9 (2007), 1942–1952.

[229] Sang, Y., Shen, H., Inoguchi, Y., Tan, Y., and Xiong, N. Secure Data Aggregation in Wireless Sensor Networks: A Survey. In *Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT '06)* (2006).

[230] Santi, P. Topology control in wireless ad hoc and sensor networks. *ACM Computing Surveys 37*, 2 (2005), 164–194.

[231] Schmeck, H. Organic computing - a new vision for distributed embedded systems. In *Proceedings of the 8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)* (2005).

[232] Schmidt, S., Krahn, H., Fischer, S., and Wätjen, D. A Security Architecture for Mobile Wireless Sensor Networks. In *Proceedings of the 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS'04)* (2004).

[233] Schneier, B. *Applied Cryptography*. John Wiley & Sons, 1996.

[234] Schneier, B. Attack Trees: Modeling Security Threats. *Dr. Dobb's Journal* (Dec. 1999).

[235] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., and Ferguson, N. Twofish: A 128-Bit Block Cipher. NIST AES Proposal, 1998. Available online. `http://www.schneier.com/paper-twofish-paper.html`.

[236] Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., and Sommerlad, P. *Security Patterns: Integrating Security and Systems Engineering*. Jon Wiley & Sons, 2006.

[237] Securities and Exchange Commission. Sarbanes Oxley SEC Rules. Available online. `http://www.sarbanes-oxley.com/section.php?level=1\&pub_id=SEC-Rules`.

[238] Seddigh, N. Studies of TCP's Retransmission Timeout Mechanism. In *Proceedings of the International Conference on Communications (ICC'01)* (2001).

[239] Seshadri, A., Perrig, A., van Doorn, L., and Khosla, P. SWATT: Software-based Attestation for Embedded Devices. In *Proceedings of the IEEE Symposium on Security and Privacy* (2004).

[240] Shirey, R. Internet Security Glossary, Version 2. RFC 4949 (Informational), Aug. 2007.

[241] Silva, R., Pereira, N., and Nunes, M. Applicability Drawbacks of Probabilistic Key Management Schemes for Real World Applications of Wireless Sensor Networks. In *Proceedings of the Third International Conference on Wireless and Mobile Communications (ICWMC'07)* (2007).

[242] Simmhan, Y. L., Plale, and Gannon, D. A Survey of Data Provenance Techniques. Tech. Rep. IUB-CS-TR618, Computer Science Department, Indiana University, Bloomington IN 47405, 2005.

[243] Simmhan, Y. L., Plale, B., and Gannon, D. A survey of data provenance in e-science. *ACM SIGMOD Record 34*, 3 (2005), 31–36.

[244] Simon, G., Maróti, M., Lédeczi, A., Balogh, G., Kusy, B., Nádas, A., Pap, G., Sallai, J., and Frampton, K. Sensor network-based countersniper system. In *Proceedings of the 2nd international conference on embedded networked sensor systems (SenSys'04)* (2004).

[245] Slagell, A., Bonilla, R., and Yurcik, W. A survey of PKI components and scalability issues. In *Proceedings of the 21st International Conference on Performance, Computing, and Communications (PCC'06)* (2006).

[246] Slijepcevic, S., Potkonjak, M., Tsiatsis, V., Zimbeck, S., and Srivastava, M. B. On Communication Security in Wireless Ad-Hoc Sensor Networks. In *Proceedings of the 11th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'02)* (2002).

[247] SmartHome Paderborn e.V. SmartHome Paderborn. Project website: `http://www.smarthomepaderborn.de/index.html`, 2009.

[248] Solinas, J. Generalized Mersenne Numbers. Technical Report CORR-9939, Dept of C&O, University of Waterloo, Canada, 1999.

[249] Stajano, F. *Security for Ubiquitous Computing.* John Wiley and Sons, 2002.

[250] Stajano, F. Will Your Digital Butlers Betray You? In *Proceedings of the 2004 ACM Workshop on Privacy in the electronic society (WPES'04)* (2004), pp. 37–38.

[251] Stajano, F., and Anderson, R. J. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Proceedings of the 7th International Workshop on Security Protocols* (1999).

[252] Stajano, F., Cvrcek, D., and Lewis, M. Steel, Cast Iron and Concrete: Security Engineering for Real World Wireless Sensor Networks. In *Proceedings of the 6th International Conference on Applied Cryptography and Network Security (ACNS'08)* (2008).

[253] Stallings, W. *Cryptography and Network Security: Principles and Practice*, 3rd ed. Prentice Hall, 2002.

[254] Suh, G. E., and Devadas, S. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual conference on design automation (DAC'07)* (2007).

[255] Szewczyk, R., Mainwaring, A. M., Polastre, J., Anderson, J., and Culler, D. E. An analysis of a large scale habitat monitoring application. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04)* (2004).

[256] Telecommunication Standardization Sector of ITU (ITU-T). Recommendation P.862: Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. Available online. http://www.itu.int/rec/T-REC-P.862/en, Feb. 2001.

[257] Telecommunication Standardization Sector of ITU (ITU-T). Recommendation J.247 : Objective perceptual multimedia video quality measurement in the presence of a full reference. Available online. http://www.itu.int/rec/T-REC-J.247-200808-P/en, Aug. 2008.

[258] Telecommunication Standardization Sector of ITU (ITU-T). Recommendation P.910: Subjective video quality assessment methods for multimedia applications. Online Available. http://www.itu.int/rec/T-REC-P.910/en, Apr. 2008.

[259] Trusted Computing Group. Trusted Computing Group – Backgrounder. Web Resource, 2005.

[260] Tubaishat, M., Yin, J., Panja, B., and Madria, S. A secure hierarchical model for sensor network. *ACM SIGMOD Record 33*, 1 (2004), 7–13.

[261] Twidle, K., Dulay, N., Lupu, E., and Sloman, M. Ponder2: A Policy System for Autonomous Pervasive Environments. In *Proceedings of the 5th International Conference on Autonomic and Autonomous Systems (ICAS'09)* (2009).

[262] US-CERT. MD5 vulnerable to collision attacks. Vulnerability Note VU#836068. Available online. http://www.kb.cert.org/vuls/id/836068, 30 Dec. 2008.

[263] US Department of Health and Human Services. Standards for Privacy of Individually Identiable Health Information, 2002. Available online. http://www.ihs.gov/adminmngrresources/hipaa/index.cfm?module=faq.

[264] US National Institute of Standards & Technology. Advanced Encryption Standard. FIPS PUB 197, Nov. 2001. Available online. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[265] US National Institute of Standards & Technology. The Keyed-Hash Message Authentication Code (HMAC). FIPS PUB 198, Mar. 2002. Available online. http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf.

[266] US National Institute of Standards & Technology. Secure Hash Standard (SHS). FIPS PUB 180-2, Aug. 2002. Available online. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf.

[267] Varga, A. The OMNeT++ Discrete Event Simulation System. In *Proceedings of the European Simulation Multiconference (ESM'01)* (2001).

[268] Varga, A., and Hornig, R. An overview of the OMNeT++ simulation environment. In *Proceedings of the 1st international conference on Simu-*

lation tools and techniques for communications, networks and systems & workshops (Simutools'08) (2008).

[269] VDE / ITG / GI. Organic Computing – Computer- und Systemarchitektur im Jahr 2010. Tech. rep., VDE / ITG / GI - Positionspapier, 2003.

[270] Vetter, B., Wang, F., and Wu, S. An experimental study of insider attacks for ospf routing protocol. In *Proceedings of the 5th International Conference on Network Protocols (ICNP'97)* (1997).

[271] Wang, M.-M., Cao, J.-N., Li, J., and Das, S. K. Middleware for Wireless Sensor Networks: A Survey. *Journal of Computer Science and Technology 23*, 3 (2008), 305–326.

[272] Wang, X., Lao, G., DeMartini, T., Reddy, H., Nguyen, M., and Valenzuela, E. XrML – eXtensible rights Markup Language. In *Proceedings of the 2002 ACM Workshop on XML Security (XMLSEC'02)* (2002).

[273] Warneke, B., Last, M., Liebowitz, B., and Pister, K. S. J. Smart Dust: Communicating with a Cubic-Millimeter Computer. *Computer 34*, 1 (2001), 44–51.

[274] Wedde, H., Timm, C., and Farooq, M. BeeHiveAIS: A Simple, Efficient, Scalable and Secure Routing Framework Inspired by Artificial Immune Systems. In *Proceedings of the 9th Conference on Parallel Problem Solving from Nature (PPSN'06)* (2006).

[275] Wedde, H. F., and Farooq, M. A Performance Evaluation Framework for Nature Inspired Routing Algorithms. In *Proceedings of the Workshops on Applications on Evolutionary Computing (EvoWorkshops'05)* (2005).

[276] Wedde, H. F., and Farooq, M. A comprehensive review of nature inspired routing algorithms for fixed telecommunication networks. *Journal of Systems Architecture 52*, 8 (2006), 461–484.

[277] Wedde, H. F., Farooq, M., and Zhang, Y. BeeHive: An Efficient Fault Tolerant Routing Algorithm Inspired by Honey Bee Behavior. In *Proceedings of the 4th International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS'04)* (2004).

[278] Wedde, H. F., Timm, C., and Farooq, M. BeeHiveGuard: A Step Towards Secure Nature Inspired Routing Algorithms. In *Proceedings of the Workshops on Applications on Evolutionary Computing (EvoWorkshops'06)* (2006).

[279] Weiser, M. The computer for the 21st century. *Scientific American 265*, 3 (1991), 94–104.

[280] Weiser, M., and Brown, J. S. The Coming Age of Calm Technology. In *Beyond Calculation: The Next Fifty Years of Computing*, P. J. Denning and R. M. Metcalfe, Eds. Springer-Verlag, 1997, pp. 75–85.

[281] Whiting, D., Housley, R., and Ferguson, N. Counter with CBC-MAC (CCM). RFC 3610 (Informational), Sept. 2003.

[282] Whittaker, J. Why Secure Applications are Difficult to Write. *IEEE Security and Privacy 1*, 2 (2003), 81–83.

[283] Wood, A. D., Fang, L., Stankovic, J. A., and He, T. SIGF: a family of configurable, secure routing protocols for wireless sensor networks. In *Proceedings of the 4th ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'06)* (2006).

[284] Wood, A. D., and Stankovic, J. A. Denial of service in sensor networks. *IEEE Computer 35*, 10 (2002), 54–62.

[285] Wood, A. D., Stankovic, J. A., and Son, S. H. JAM: A Jammed-Area Mapping Service for Sensor Networks. In *Proceedings of the 24th IEEE Real-Time Systems Symposium (RTSS'03)* (2003).

[286] Xiao, Y., Rayi, V. K., Sun, B., Du, X., Hu, F., and Galloway, M. A Survey of Key Management Schemes in Wireless Sensor Networks. *Computer Communications – Special Issue on Security on Wireless Ad Hoc and Sensor Networks 30*, 11-12 (2007), 2314–2341.

[287] Yabandeh, M., Zarifzadeha, S., and Yazdani, N. Improving performance of transport protocols in multipath transferring schemes. *Computer Communications 30*, 17 (2007), 3270–3284.

[288] Yang, Y., Wang, X., Zhu, S., and Cao, G. SDAP: A Secure Hop-by-hop Data Aggregation Protocol for Sensor Networks. In *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '06)* (2006).

[289] Ye, F., Luo, H., Lu, S., and Zhang, L. Statistical En-route Detection and Filtering of Injected False Data in Sensor Networks. *IEEE Journal on Selected Areas in Communications 23*, 4 (2005), 839–850.

[290] Ye, Z., Krishnamurthy, S., and Tripathi, S. Effects of multipath routing on TCP performance in ad hoc networks. In *Proceedings of the Global Telecommunications Conference (GLOBECOM'04)* (2004).

[291] Yick, J., Mukherjee, B., and Ghosal, D. Wireless sensor network survey. *Computer Networks 52*, 12 (2008), 2292–2330.

[292] Zhang, K. Efficient protocols for signing routing messages. In *Proceedings of the Symposium on Networks and Sistributed Systems Security (NDSS'98)* (1998).

[293] Zhang, P., Sadler, C. M., Lyon, S. A., and Martonosi, M. Hardware design experiences in ZebraNet. In *Proceedings of the 2nd international conference on embedded networked sensor systems (SenSys'04)* (2004).

[294] Zhong, W., and Evans, D. When Ants Attack: Security Issues for Stigmergic Systems. Tech. Rep. CS-2002-3, Departement of Computer Science, University of Virginia, 2002.

[295] Zhou, Y., Fang, Y., and Zhang, Y. Securing Wireless Sensor Networks: A Survey. *IEEE Communications Surveys 10*, 3 (2008), 6–28.

[296] Zhu, S., Setia, S., and Jajodia, S. LEAP: efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)* (2003).

[297] ZigBee Alliance. Zigbee specification, Version 1.0. `http://www.zigbee.org`.

Index