# Authentication in Ad-hoc and Sensor Networks

Dissertation

Submitted to the Faculty of
Electrical Engineering and Information Technology
of the
Ruhr-University Bochum

for the
Degree of Doktor-Ingenieur

by

André Weimerskirch

Bochum, Germany, 2004

Author contact information:
aweimerskirch@web.de

Thesis Advisor: Prof. Christof Paar
Thesis Reader: Prof. Jean-Pierre Hubaux

*to my parents*

# Zusammenfassung

In der nahen Zukunft werden Mikroprozessoren in nahezu allen Geräten verwendet werden, angefangen vom mobilen Telefon bis hin zur Waschmaschine. Sobald diese Geräte über einen drahtlosen Kommunikationskanal miteinander verbunden werden, könnte ein extrem weit verzweigtes drahtloses Netzwerk ohne feste Infrastruktur oder zentrale Administration entstehen. Solch ein Netzwerk heißt Ad-hoc-Netzwerk. Es kann insbesondere dann eingesetzt werden, wenn eine zuverlässige Infrastruktur nicht verfügbar ist, z.B. nach einer Naturkatastrophe, oder wenn jene zu teuer ist. Wenn das Netzwerk aus kleinen leistungsschwachen Geräten besteht, die in der Lage sind ihre Umwelt zu überwachen, heißt ein solches Netzwerk ein Sensornetzwerk.

In dem Maße, wie Ad-hoc-Netzwerke immer mehr Teil unseres Alltags werden, könnte diese Technologie auch immer größere Gefahren im Alltag verursachen, wenn Sicherheit in Ad-hoc-Netzen nicht von Anfang an beachtet und umgesetzt wird. Ein Beispiel für eine Anwendung eines Ad-hoc-Netzwerks ist der Einsatz im Straßenverkehr, um diesen zuverlässiger und sicherer zu gestalten. Wenn diese Technologie jedoch Sicherheitslücken aufweist, so ist sie offen für Angriffe und somit eine Gefahr für die Verkehrsteilnehmer. Die Authentifizierung in Ad-hoc-Netzwerken ist ein Kernaspekt für sichere Protokolle und sichere Anwendungen in Ad-hoc-Netzen. Daher wird die Authentifizierung in Ad-hoc-Netzwerken als übergreifendes Thema dieser Arbeit behandelt.

Die Sicherheitsprobleme von Ad-hoc-Netzwerken und Sensornetzwerken unterscheiden sich wesentlich von denen herkömmlicher Netzwerke. Dies kann auf die Ressourcenbeschränkungen der mobilen Geräte und die häufigen Topologiewechsel des Netzwerkes zurückgeführt werden. Insbesondere in Sensornetzwerken sind die Geräte darüber hinaus physikalischen Angriffen wie Seitenkanalattacken ausgeliefert. Daher müssen Protokolle entworfen werden, die gegen mehrere manipulierte Geräte sowie gegen erbeutete Schlüssel immun sind.

Der Inhalt und die Resultate dieser Arbeit lassen sich wie folgt zusammenfassen: (1) Zuerst wird ein Überblick und eine Analyse verschiedener Authentifizierungsmethoden im Hinblick auf Ad-hoc-Netzwerke gegeben; (2) daraufhin folgt eine Analyse über die Eignung digitale Signaturen für Ad-hoc-Netzwerke, und ein Vergleich digitaler Signaturverfahren für Ad-hoc-Netzwerke; (3) zudem werden zwei neue extrem effiziente Authentifizierungsprotokolle für eine gegenseitige Authentifizierung in Ad-hoc und Sensornetzwerken vorgestellt, die hauptsächlich auf symmetrischen Kryptographieprimitiven basieren; und (4) letztlich wird eine Anwendung der Authentifizierung beschrieben, die Komponentenidentifikation ermöglicht. Solch eine Komponentenidentifikation kann als Maßnahme gegen gefälschte Komponenten genutzt werden, z.B. im Automobilbereich. Als weiteres Ergebnis dieser Arbeit ergeben sich die folgenden Empfehlungen: (1) Sicherheitsprotokolle sollten so stark wie möglich auf einem lokalen Ansatz basieren, in dem die Netzwerkknoten eine Vertrauensbasis zu ihrer unmittelbaren Nachbarschaft aufbauen, um aufwendige authentifizierte Rundsendungen (broadcast) zu vermeiden wie sie durch digitale Signaturen zur Verfügung gestellt werden; und (2) Protokolle sollten die Benutzung asymmetrische Kryptographie auf ein Minimum reduzieren. Die in dieser Arbeit vorgestellten Protokolle sind ein erster Schritt in diese Richtung.

# Abstract

In the near future microprocessors will be found almost everywhere from cellular phones to washing machines and cars. Once these are connected via a (wireless) communication channel to each other and possibly to already existing static computers this could form an extremely dynamic wireless network which may not have access to an infrastructure or centralized administration. Such a network is often referred to as ad-hoc network. It is particularly useful where a reliable fixed or mobile infrastructure is not available – e.g., after a natural disaster – or too expensive. If the network consists of very small computing devices that are able to sensor their environment, such a network is called a sensor network.

As ad-hoc and sensor networks become more a part of everyday life, they could become a threat if security is not considered before deployment. For instance, ad-hoc networks might be used to increase vehicle traffic safety. However, if there are any security vulnerabilities, this technology might be open to attackers and thus endanger passengers. Authentication in ad-hoc networks is a core requirement for secure protocols and secure applications of ad-hoc networks. Thus authentication in ad-hoc networks is the focus of this work.

The security issues for ad-hoc networks and sensor networks are different than those for fixed networks. This is due to system constraints in mobile devices, frequent topology changes in the network, and the weak physical security of low-power devices. Moreover in sensor networks, the sensors are exposed to physical attacks such as power analysis and probing. Consequently, protocols need to be designed that are robust against a set of malicious devices as well as compromised secrets.

The main goals and achievements of this thesis are as follows: (1) to give an overview of authentication schemes and analyze how well they are suited to ad-hoc networks; (2) to analyze how well digital signature schemes can be used in ad-hoc networks and to compare signature schemes for this purpose;

(3) to propose two new extremely efficient authentication schemes for pairwise authentication that mainly use symmetric cryptographic primitives providing a basic form of authentication in sensor networks and certified identification in ad-hoc networks; and (4) an application of authentication providing component identification. Such component identification can be used as a countermeasure to faked components, e.g., for components of automobiles. As a result of this thesis, we recommend the following: First, protocols should be based as much as possible on an approach where trust associations are established to the local one-hop neighborhood only to avoid broadcast authentication schemes; and second, to design protocols that reduce the amount of asymmetric cryptography to a minimum. The protocols proposed in this thesis are a first step to achieve these goals.

# Preface

This thesis describes research that I conducted during the three years I worked as a researcher at the Ruhr-University Bochum. I hope that this work is of use and will be carried on.

The work I present in this thesis would not have been possible without the support of several people. First of all I would like to thank my advisor Prof. Christof Paar. He supported me all the time by his advise and by his friendship and camaraderie, and he gave me the freedom to explore several opportunities.

I am grateful to my thesis committee, especially to Prof. Hubaux for taking the time to review this work and giving me many valuable suggestions and comments. I would also like to thank Dirk Westhoff for all his advise, for giving me so many insights to wireless networks, and for supporting me at several projects.

I would like to give special thanks to all my colleagues and friends. I thank Jorge Guajardo and Thomas Wollinger for their friendship and for all the fruitful discussions and exchanging ideas. I also thank Marcus Heitmann, Sandeep Kumar, Jan Pelzl, Kai Schramm, and Marko Wolf for their support, for their friendship, and for having much fun with them. I want to thank all the other members of the COSY group for always contributing to a good and warm group atmosphere. In particular, I thank Irmgard Kühn for taking care of all the paper work hassle and being a crumple zone to the university administration.

I want to thank Sheueling Chang Shantz, Hans Eberle, and all the other people of Sun Labs for giving me the chance to spent a summer in sunny California, for teaching me their experience, and for giving me many advises. I also want to thank Prof. Ivan Damgård for giving me the chance to stay a semester at Århus University together with my girl-friend Dörte, and the European Union for giving me a stipend for the time in Denmark. Furthermore, I want to thank everyone who in some way supported me with my research: Jonathan

Hammell, Katrin Höper, Uwe Hüpping, Stefan Lucks, Christian Röpke, and Marcel Selhorst.

Finally, but most important I want to thank my parents and my girl-friend Dörte for always supporting me and for encouraging me.

Thank you all!

André

# Contents

Zusammenfassung iv

Abstract vi

Preface viii

1 Introduction 1

1.1 Motivation . . . 2

1.2 Main Contributions . . . 3

2 Ad-hoc and Sensor Networks 5

2.1 Ad-hoc Networks . . . 5

2.2 Sensor Networks . . . 9

2.3 Embedded Systems . . . 11

2.4 Potential Applications . . . 11

2.5 Security Issues . . . 13

2.5.1 Security Goals . . . 13

2.5.2 Restrictions . . . 14

2.6 Network and Adversary Model . . . 17

2.6.1 Network Model . . . 17

2.6.2 Device Model . . . 18

2.6.3 Adversarial Model . . . 18

2.7 Categorization . . . 19

3 Background and Related Work 23

3.1 Authentication . . . 23

3.1.1 Basic Authentication Schemes . . . 24

3.1.2 Mutual and Broadcast Authentication . . . 28

3.1.3 Further Authentication Schemes . . . 30

3.1.4 Overview . . . 33

3.2 Key Distribution . . . . . . . . . . . . . . . . . . . . . . . . . . 35

3.2.1 Secure Channel . . . . . . . . . . . . . . . . . . . . . . . 35

3.2.2 Distributed Public-Key Infrastructure . . . . . . . . . . . 37

3.2.3 Key Pre-Distribution Schemes . . . . . . . . . . . . . . . 38

3.3 Cryptography on Embedded Systems . . . . . . . . . . . . . . . 39

3.4 Security Protocols . . . . . . . . . . . . . . . . . . . . . . . . . . 40

3.5 Existing Technology . . . . . . . . . . . . . . . . . . . . . . . . 44

3.6 Authentication Models . . . . . . . . . . . . . . . . . . . . . . . 46

**4 Signature Schemes in Ad-hoc Networks 50**

4.1 Security Protocols and Digital Signatures . . . . . . . . . . . . 51

4.2 Performance . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54

4.3 Digital Signatures for Security Protocols . . . . . . . . . . . . 56

4.3.1 Security Model . . . . . . . . . . . . . . . . . . . . . . . 57

4.3.2 Required Level of Security . . . . . . . . . . . . . . . . . 58

4.3.3 Performance . . . . . . . . . . . . . . . . . . . . . . . . . 60

4.4 Recommendations . . . . . . . . . . . . . . . . . . . . . . . . . . 61

4.4.1 Performance . . . . . . . . . . . . . . . . . . . . . . . . . 61

4.4.2 Topology of the Network . . . . . . . . . . . . . . . . . . 62

4.4.3 Security Relationships . . . . . . . . . . . . . . . . . . . 64

**5 Efficient Authentication in Ad-hoc and Sensor Networks 68**

5.1 Zero Common-Knowledge Recognition . . . . . . . . . . . . . . 69

5.1.1 Recognition . . . . . . . . . . . . . . . . . . . . . . . . . 70

5.1.2 Network And Adversarial Model . . . . . . . . . . . . . . 72

5.1.3 General Recognition Protocols . . . . . . . . . . . . . . . 73

5.1.4 Zero Common-Knowledge Protocol . . . . . . . . . . . . 74

5.2 Identity Certified Authentication . . . . . . . . . . . . . . . . . 79

5.2.1 Network and Adversarial Model . . . . . . . . . . . . . . 80

5.2.2 Outline . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81

5.3 Identity Certified Authentication Protocol . . . . . . . . . . . 83

5.3.1 Security . . . . . . . . . . . . . . . . . . . . . . . . . . . 85

5.3.2 Certificate Lifespan . . . . . . . . . . . . . . . . . . . . . 86

**6 Efficiency Analysis of Authentication Protocols 88**

6.1 Energy Consumption . . . . . . . . . . . . . . . . . . . . . . . . 88

6.1.1 Device Architecture . . . . . . . . . . . . . . . . . . . . . 88

6.1.2 Energy Model . . . . . . . . . . . . . . . . . . . . . . . . . 89

6.1.3 Energy Map of the ZCK Protocol . . . . . . . . . . . . . 91

6.1.4 Energy Map of the IC protocol . . . . . . . . . . . . . . 93

6.1.5 Energy Map of Traditional Protocols . . . . . . . . . . . 94

6.2 Efficiency Analysis . . . . . . . . . . . . . . . . . . . . . . . . . 97

6.3 Comparison of the Schemes . . . . . . . . . . . . . . . . . . . . 97

6.3.1 Operational Complexity . . . . . . . . . . . . . . . . . . . 99

6.3.2 Device Lifetime . . . . . . . . . . . . . . . . . . . . . . . . . 100

6.3.3 Comparison of the Schemes . . . . . . . . . . . . . . . . . 103

**7 Component Identification 107**

7.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 107

7.1.1 Assumptions . . . . . . . . . . . . . . . . . . . . . . . . . . 109

7.1.2 Adversarial Model . . . . . . . . . . . . . . . . . . . . . . 111

7.2 Asymmetric Component Identification . . . . . . . . . . . . . . 112

7.2.1 Initialization of a Car . . . . . . . . . . . . . . . . . . . . 115

7.2.2 Installation of a Component . . . . . . . . . . . . . . . . 116

7.2.3 The Running Vehicle . . . . . . . . . . . . . . . . . . . . . 118

7.2.4 Demounting of a Component . . . . . . . . . . . . . . . . 120

7.3 Distributed Component Identification . . . . . . . . . . . . . . . 120

7.4 Symmetric Component Identification . . . . . . . . . . . . . . . 123

7.4.1 Installation of a New Component . . . . . . . . . . . . . 124

7.4.2 Running System and Demounting of a Component . . . 125

7.5 Features and Enhancements . . . . . . . . . . . . . . . . . . . . 125

7.6 Security . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 128

7.7 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 129

**8 Summary and Outlook 130**

**Bibliography 132**

**Curriculum Vitae 141**

**Index 148**

# List of Tables

2.1 Ad-hoc network categorization . . . . . . . . . . . . . . . . . . . . 22

3.1 Authentication protocols . . . . . . . . . . . . . . . . . . . . . . . 34

3.2 Protocol efficiency . . . . . . . . . . . . . . . . . . . . . . . . . . 34

4.1 Key size recommendation for high security level [58, 76] . . . . . 54

4.2 Execution times for signature operations based on ECDSA and RSA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55

4.3 Recommended key sizes for protocols on the transport layer . . 59

6.1 Energy map of the ZCK protocol per protocol step and per involved entity . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91

6.2 Energy map of the IC protocol per protocol step and per involved entity . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93

6.3 ZCK and IC properties . . . . . . . . . . . . . . . . . . . . . . . . 97

6.4 ZCK and IC efficiency . . . . . . . . . . . . . . . . . . . . . . . . 97

6.5 Comparison between ZCK and Guy Fawkes protocol . . . . . . 98

6.6 Operational complexity of ZCK and UDH . . . . . . . . . . . . 99

6.7 Operational complexity of IC and CDH . . . . . . . . . . . . . . 100

# List of Figures

2.1 Ad-hoc network cloud . . . . . . . . . . . . . . . . . . . . . . . . 7

2.2 Combination of ad-hoc network clouds . . . . . . . . . . . . . . . 7

4.1 Comparison of RSA and ECDSA key sizes . . . . . . . . . . . . 55

4.2 Comparison of RSA and ECDSA signature sizes . . . . . . . . . 56

4.3 Comparison of RSA and ECDSA based certificate sizes . . . . . 57

4.4 Average running time per packet . . . . . . . . . . . . . . . . . . 61

4.5 Trade-off between RSA and ECDSA with respect to the number of intermediate nodes $n$ . . . . . . . . . . . . . . . . . . . . . . . . 62

4.6 Traffic distribution per route length, e.g. with 100m vs. 250m transmission radius . . . . . . . . . . . . . . . . . . . . . . . . 63

4.7 Trade-off between broadcast and neighborhood approach . . . . 67

5.1 Zero Common-Knowledge message recognition protocol . . . . . 75

5.2 Zero Common-Knowledge recognition protocol example . . . . . 77

5.3 Zero Common-Knowledge entity recognition protocol . . . . . . 78

5.4 Zero Common-Knowledge mutual message recognition protocol . 79

5.5 Identity Certified authentication example . . . . . . . . . . . . . 81

5.6 Time intervals for hash-chains . . . . . . . . . . . . . . . . . . . 82

5.7 Identity Certified authentication protocol . . . . . . . . . . . . . 85

6.1 Trade-off between the ZCK and UDH protocol . . . . . . . . . . 104

6.2 Trade-off between IC and CDH . . . . . . . . . . . . . . . . . . 105

7.1 Life cycle of a component . . . . . . . . . . . . . . . . . . . . . . 115

7.2 Key check . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 117

7.3 Proof of origin . . . . . . . . . . . . . . . . . . . . . . . . . . . . 117

7.4 System check with one verifier . . . . . . . . . . . . . . . . . . . 118

7.5 System check with cyclic verifier . . . . . . . . . . . . . . . . . . 119

# List of Algorithms

3.1 Public-key message authentication . . . . . . . . . . . . . . . . . . 24

3.2 Public-key entity authentication . . . . . . . . . . . . . . . . . . . 25

3.3 Symmetric server message authentication . . . . . . . . . . . . . 25

3.4 Symmetric server entity authentication . . . . . . . . . . . . . . 25

3.5 Symmetric message authentication . . . . . . . . . . . . . . . . . 26

3.6 Symmetric entity authentication . . . . . . . . . . . . . . . . . . . 26

3.7 Hybrid message authentication . . . . . . . . . . . . . . . . . . . . 26

3.8 Hybrid entity authentication . . . . . . . . . . . . . . . . . . . . . 27

3.9 Entity authentication with time-stamp . . . . . . . . . . . . . . . 27

3.10 Fiat-Shamir authentication . . . . . . . . . . . . . . . . . . . . . . 28

3.11 Mutual entity authentication . . . . . . . . . . . . . . . . . . . . . 29

3.12 Asymmetric MAC . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29

3.13 Lamport's one-time password . . . . . . . . . . . . . . . . . . . . . 30

3.14 TESLA broadcast authentication . . . . . . . . . . . . . . . . . . . 31

3.15 Guy Fawkes authentication . . . . . . . . . . . . . . . . . . . . . . 31

3.16 Guy Fawkes bidirectional authentication . . . . . . . . . . . . . 32

3.17 Remote User authentication . . . . . . . . . . . . . . . . . . . . . . 33

5.1 General entity recognition protocol . . . . . . . . . . . . . . . . 74

7.1 Challenge-response identification . . . . . . . . . . . . . . . . . . 114

7.2 Mutual challenge-response check of a symmetric key . . . . . . . 114

7.3 Proof of origin . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116

7.4 Demounting . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 120

7.5 List distribution . . . . . . . . . . . . . . . . . . . . . . . . . . . . 121

7.6 Selection of a random number . . . . . . . . . . . . . . . . . . . . 121

7.7 Selection of a random number by commitments . . . . . . . . . 122

7.8 Selection of a random component . . . . . . . . . . . . . . . . . . 122

7.9 Proof of origin for the symmetric case . . . . . . . . . . . . . . . 125

7.10 Key update . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 127

# 1 Introduction

Today microprocessors can be found almost everywhere. It is a well established trend to embed microprocessors in electronic devices such as cellular phones, televisions, and video recorders. In the future, small processors might be embedded into even more devices and structures like clothing, eyeglasses, buildings, and barcodes. Once these devices are equipped with a wireless radio, they could form an extremely widespread network that connects many devices of our environment, possibly including already existing computing devices such as desktop computers, notebooks, and Personal Digital Assistants (PDAs). This all-embracing network of mobile and static devices must be self-organized and should neither rely on a fixed infrastructure nor a centralized administration as devices may be introduced to and removed from the network in a highly dynamic fashion. In fact, in the general case it is assumed that each node relies on its neighboring nodes to keep the network connected, e.g., each node routes data packets for its neighbors. Furthermore, each node might take advantage of services offered by other nodes. This type of network is called an ad-hoc network. It is particularly useful where a reliable fixed or mobile infrastructure is not available – e.g., after a natural disaster – or too expensive. If the network consists of very small computing devices that are able to sensor their environment we call such a network a sensor network. Here, all sensors collaborate in order to gather information based on their sensing capabilities. The sensor devices are very low-cost and thus extremely resource constrained. However, due to the low cost of these sensors, they can be deployed in the hundreds or thousands in a small area. Sensor networks are in most cases static, but mobile sensors are also conceivable. As for ad-hoc networks, the sensors run self-organized without any external guidance once they are deployed.

## 1.1 Motivation

As ad-hoc and sensor networks become a growing part of our everyday life, they could become a threat if security is not considered carefully before deployment. For instance, consider a possible scenario of the future road traffic. There will be communication between cars, and between cars and roads. Cars will form a wireless network in an ad-hoc manner. They might communicate to each other in order to exchange information about free parking spaces or to warn about road threats. As an example, if there is an obstacle on the highway, a car could warn all following cars. Obviously, this information must be trustworthy and authenticated, and this process has to be done efficiently in real-time. All cars might have an electronic license plate embedded that identifies each car uniquely. If the electronic license plate broadcasts a unique identifier while the car is running, it is possible to identify the car and thus the driver of this car in case of a hit-and-run accident. Hence, it must not be possible to forge an electronic license plate in order to impersonate another driver and car, and it must not be possible to manipulate the electronic license plate of a car. There are several security goals in ad-hoc networks. The provision of authentication is a core requirement for secure and trustworthy communication in ad-hoc networks, and it is the focus of this work. The messages of the car warning all following drivers must be authenticated as well as the broadcasted signal of the electronic license plate, and the electronic license plate must be inseparably bound to the car. Only if there is efficient authentication available in ad-hoc networks, secure protocols and applications can be designed. IT security is the enabler for innovative applications, and provision of authentication is the enabler of security.

The security issues for ad-hoc and sensor networks are different than the ones for fixed traditional networks such as local area networks (LANs) and wide area networks (WANs). While the security requirements are the same, namely availability, confidentiality, integrity, authentication, and non-repudiation, their provision must be approached differently for ad-hoc and sensor networks. This is due to system constraints of mobile devices, frequent topology changes in the network, a missing fixed infrastructure, and the weak physical security of low-power devices. Furthermore, the main security targets differ in ad-hoc networks. For instance, secure routing and secure stimulation of cooperation are crucial issues in ad-hoc networks. The provision of authentication is required to implement secure protocols in ad-hoc networks such that authentication might provide the basis for a secure routing or stimulation scheme. In this thesis we clearly focus on the problem of providing efficient cryptographic authentication. We apply our results to ad-hoc and sensor networks since resource limitations and possibly a lack of central services demand such efficient authentication protocols. Clearly, there are further security goals in ad-hoc networks that are as important as authentication. However, in this thesis we emphasize the importance of authentication, and we consider ad-hoc and sensor networks as well as further security protocols as an application of authentication schemes. The main goal of this thesis is therefore to analyze and design efficient and flexible authentication protocols.

## 1.2 Main Contributions

In this thesis we focus on authentication in ad-hoc and sensor networks. While there are several security issues in such networks as described above, authentication is the core requirement for achieving integrity, confidentiality, and non-repudiation — if you do not know who you are talking to, it does not make sense to establish a secure channel. In the next chapter we start by defining terms and categorizing them. In Chapter 3 we give an overview of authentication schemes and analyze how well they are suited for ad-hoc and sensor networks. While there are several efficient authentication schemes available, it seems to be unclear yet how to perform efficient authentication when there is no secure channel for a key agreement available and when there were no keys distributed before deployment. As this is the case in many real-world applications we analyze this scenario thoroughly. We start by looking at digital signatures for ad-hoc networks in Chapter 4. We discuss which and how well signature schemes are suited to ad-hoc networks, and we give recommendations about their usage. Then we argue that an approach where nodes set up trust associations to their neighborhood by authenticating to them and only rely on their local neighborhood is more efficient than a global approach where nodes use digital signatures to make messages verifiable by all network members. Clearly, digital signatures overstrain the capabilities of sensor nodes such that this analysis is only thought to be for ad-hoc networks. In Chapter 5 we introduce two authentication schemes as a basis for such a local neighborhood approach. The first protocol, which we call Zero-Common Knowledge (ZCK) recognition protocol, is based on a basic form of authentication, namely recognition, that is especially suited to ad-hoc and sensor networks consisting of resource limited devices where there is no external guidance and no supporting infrastructure available. The scheme provides an extremely efficient recognition mechanism by using only symmetric cryptographic primitives, and it is provably secure. The second protocol, called Identity Certified (IC) authentication, provides identification based on certificates in a way that is more efficient than using previously known approaches. However, the identification can only be provided at the cost of some supporting infrastructure. In Chapter 6 we analyze both protocols regarding their resource requirements, including energy consumption, in order to prove their efficiency. In Chapter 7 we provide a solution for another security target, namely component identification. Here, we present a scheme for identifying components as a practical system which uses ad-hoc network mechanisms. Such component identification provides protection against counterfeits in complex systems as well as protection against stolen components and prevents manipulation of the system and its components. For instance, our component identification mechanism can be used to make sure that a car is equipped with an electronic license plate, and that these components were neither manipulated nor stolen. Finally, we conclude with a summary and an outlook in Chapter 8.

# 2 Ad-hoc and Sensor Networks

In the following, we introduce the main aspects of ad-hoc and sensor networks. Both terms are related to the area of pervasive and ubiquitous computing, which describe the situation where all electronic devices are equipped with a microcontroller and take over a crucial part in our everyday life. Devices that have a microcontroller embedded are also called embedded devices. We start by describing the main characteristics of ad-hoc and sensor networks, and then argue why the approach to security in ad-hoc networks is different to that in well known traditional networks. We presented parts of this chapter dealing with security of ad-hoc and sensor networks as well as of embedded systems in [60, 85].

## 2.1 Ad-hoc Networks

An ad-hoc network is made up of static and mobile hosts that communicate mostly via wireless channels and thus do not require any fixed infrastructure. The basic idea behind this model is not recent: as early as the 70's, ad-hoc networks were called *packet radio networks* and were investigated almost exclusively for military applications – PRnet, developed by the American Defense Advanced Research Projects Agency (DARPA), is probably the most famous example [43]. Then, when designing the 802.11 standard for Wireless Local Area Networks (WLAN), the IEEE replaced the term packet radio network by ad-hoc network, hoping to forget the military connotation of the former one. Ad-hoc networks are frequently associated with self-organization, which means that they run solely by the operation of end-users (like the old citizen-band, or CB, voice analog network). Although pure self-organization is not required to form an ad-hoc network, this feature should be understood as a basic requirement for decentralization: Hosts must be capable to enter and leave the network without referring to a central authority. It is important to note that ad-hoc networks would likely neither be a replacement nor an alternative to current and future infrastructure-based networks. Their scope is more to complement the latter in cases where cost, environment, or application constraints require self-organized and infrastructure-less solutions.

Ad-hoc networks include various architectures that range from fully mobile and decentralized radio, access, and routing technologies – the ones being investigated as part of the standardization work carried out by the Internet Engineering Task Force (IETF) [56] – to more infrastructure-dependent standards that include an ad-hoc mode, e.g., IEEE 802.11 [41] and HiperLAN2 [34]. In its original meaning, ad-hoc means that the networks are instantly formed for a specific purpose, and they only exist in this form as long as this service is required. Note that ad-hoc networks are wireless mobile networks since devices are mobile and communicate via wireless links. However, these two terms are not synonym in the sense that there are wireless mobile networks that are no ad-hoc networks. The nodes of an ad-hoc network might establish a one-to-one relationship to another node, a one-to-many, or a many-to-many relationship. In the natural case any device will establish relationships with several devices such that there is a many-to-many relationship.

Figure 2.1 presents a basic ad-hoc network cloud. Device $A$ communicates with device $F$ via the path $(A, B, G, D, E, F)$. The circle around $A$ depicts the area that $A$ can directly reach by radio transmission. We call this area the *neighborhood* of $A$. Note that $A$ could also shorten the path to $F$ by leaving out $B$. However, if $G$ moves a little further away from $A$, then $G$ would leave the neighborhood area of $A$. In many applications, several ad-hoc network clouds might be connected via a fixed infrastructure, e.g., via base stations that are connected to the Internet. Such a case is depicted in Figure 2.2.

There exist mainly two network topologies in ad-hoc networks, namely single-hop and multi-hop networks:

- single-hop network: Each device only communicates with its direct neighborhood such that there is no need for sophisticated routing. If there is a base station available, then each device only communicates with the base station, e.g., as it is done in a cellular phone network. Note that then each device needs to be in the transmission range of the base station. A more sophisticated approach of a single-hop network with a base station allows a device to either communicate to the base station or its direct

Figure 2.1: Ad-hoc network cloud

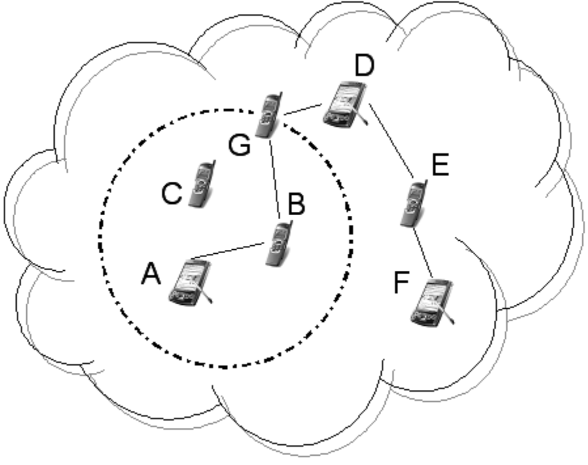
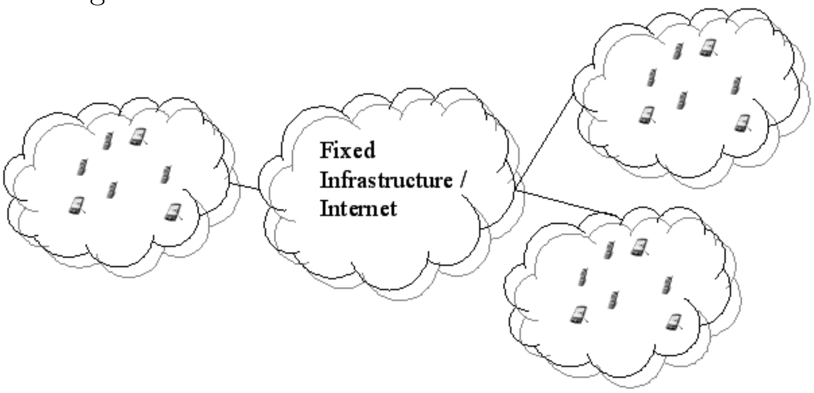


Figure 2.2: Combination of ad-hoc network clouds



neighborhood.

- multi-hop network: Each device can communicate to another device via a path that involves several hops in between. If there is a base station available, the devices do not need to be in the transmission range of the base station. In this case, data packets are forwarded in a multi-hop fashion.

Here are some of the basic features one can expect to find in most ad-hoc networks:

- *Wireless links*: Communication links are wireless to guarantee mobility. Accordingly, their dependability and capacity has to be carefully reviewed.

- *No fixed infrastructure*: Ad-hoc networks act independently from any provider. However, access points to a fixed backbone network might be available in some scenarios.

- *Self organization*: Ad-hoc networks are self-organizing. After the network initialization, the network should maintain itself without any external guidance and without any central services.

- *Cooperation*: Because they do not rely on a fixed infrastructure mobile nodes have to be somehow *cooperative.* This ranges from very simple schemes for short-range networks to highly complex strategies in case of multi-hop wide-area networks.

- *Network topology*: The network topology may be very dynamic, making the links and routes very unstable.

- *Resource limitation*: Ad-hoc networks are formed by mobile low-power devices that have very limited computing power as well as little bandwidth and memory.

- *Battery power*: Power management is an important system design criterion. Hosts have to be power-aware when performing such tasks as routing and mobility management.

- *Selfishness*: Each authority of network devices will be selfish in order to save battery power. Hence, each authority tries to minimize the sum of the energy consumption of all its network devices. In many scenarios an authority is a user that owns one device. However, in a military scenario the authority controls all network devices.

- *Mobility*: Ad-hoc network devices might be highly mobile or mainly static. For instance, consider a cell phone as part of an ad-hoc network that is connected to a driving or parked car.

- *Heterogeneity*: The hosts might be very heterogenous regarding computational resources, mobility, and connectivity.

- *Security*: Finally, security is a critical issue because of the weak connectivity, the constrained devices, and the limited physical protection of the mobile hosts. Security mechanisms should not add much overhead.

Related to ad-hoc networks is the radio frequency identification (RFID) technology [5]. These are simple passive devices that consist of a small memory chip and an antenna. The RFID tag can then be inquired by an external base station. Such RFID tags might be used to replace the barcode on almost any product to make logistics more efficient and cheaper. However, there are several threats introduced by this technology. When RFID tags replace the barcode, there arise serious privacy concerns. There are also plans to embed a tiny RFID tag into bank notes. However, a criminal could then use a simple device to detect who is carrying around a large amount of money [74]. Security in RFID tags is closely related to security in ad-hoc networks. However, today's RFID tags are hardly able to perform any cryptographic computation. Hence, possible solutions at this moment are mostly concerned with privacy issues so we do not focus on the RFID technology in this work. Still, any authentication scheme that runs efficiently on ad-hoc network devices or sensor network devices is a step closer to authentication schemes for RFID tags.

## 2.2 Sensor Networks

Wireless sensor networks (WNS) are a particular class of ad-hoc networks that attract more and more attention both in academia and industry. The sensor nodes themselves are typically of low cost and have a very small form factor. They consist of an application specific sensor, a wireless transceiver, a simple processor, and an energy unit which may be battery driven or which may harvest energy from the environment, e.g., through solar cells. Such sensor nodes are envisioned to be spread out over a geographical area to form a multi-hop network in a self-organizing manner. All sensor nodes cooperate without any external guidance in order to achieve a common goal which would be unattainable for the individual nodes. Most frequently, such WSNs are stationary although also mobile WSNs are possible. Well known examples of sensor networks are the Smart Dust devices of the University of California Berkeley [84], which are expected to be as small as a grain in the future, as well as the Mica Motes [17]. Sensor networks mainly have the following characteristics:

- *Constraints*: The devices (sensors) are low-cost devices and thus are computationally very constrained (4- or 8-bit CPU, or only a state machine).

- *Wireless links*: For interaction purposes, the sensors are equipped with radio frequency communication capabilities.
- *Packet size*: The packet size is very limited (in the range of 100 Bytes or less).
- *Battery power*: The devices are powered by a battery, or by solar energy or other autonomous means. Once the battery is empty, the sensor might be replaced since the battery may be the most valuable part of the device.
- *Mobility*: While the location of sensors is usually static, the sensor network still changes its topology frequently due to the replacement of nodes or the extension of the network.
- *No fixed infrastructure*: Once the sensors are deployed there is no external guidance available anymore. Sensors then collaborate in order to achieve their main purpose.
- *Network topology*: Sensor networks might communicate in a multi-hop fashion with any other node of the network, or they might only communicate with their direct neighborhood in a single hop fashion. We call the first scenario a *global approach* whereas the second one is a *local neighborhood approach.* For instance, the local approach is used in the distributed virtual shared information space (dvSIS) [8] where all information is flooded in the network and sensors decide whether to forward this information. This eliminates the need for a multi-hop routing protocol.
- *homogeneity*: Usually, all deployed sensors are equal, i.e., there is no hierarchy and no external supervision involved. However, sensors might be divided into clusters. Furthermore, often there is a base station (aggregator) that collects and evaluates the aggregated data. The base station is more powerful and might be physically secured. The nodes then establish communication channels to the base station in a multi-hop fashion.
- *Security*: Finally, security must be considered before deployment. The security architecture demands lightweight cryptographic algorithms, and

execution time must not disrupt the sensor's main task. As the energy is strictly limited, transmitted messages should not be extended significantly in size.

## 2.3 Embedded Systems

Ad-hoc networks will be formed mainly by embedded devices. Already today, 98% of all microprocessors produced are used for embedded devices whereas only 2% are used as a CPU of an interactive computer such as a PC [23]. We give an informal definition of an embedded system as a traditional device that is equipped with a microcontroller. Typical properties of an embedded device are the following:

- the device is intended for mainly one purpose such as a washing machine or automobile,
- the device is equipped with a microcontroller,
- the computation is transparent to the user, i.e., the device operates in an intuitive way without any interaction to the user such as a keyboard,
- the device cannot be re-programmed by the user.

These microcontrollers function in a mostly non-interactive way without any interaction to the user as opposed to traditional workstation PCs. For instance, embedded devices include sophisticated coffee machines, driving aid technologies in automobiles such as ABS and ESP, as well as smart phones. There are several security issues in embedded devices that are identical to the security issues of ad-hoc network devices. We will go into more details below.

## 2.4 Potential Applications

As ad-hoc networks have been developed initially for military purposes, potential applications are very often associated with critical situations such as scenarios of battlefields and damaged areas. Other prospective applications are still at an early stage. For instance, let us again consider the vehicle example. Highway toll might be charged without the need to stop the car but just by passing a toll collector sensor. Furthermore, as cars are extremely mobile they might be used to route messages between any two points. For instance, maintenance sensors in a bridge might send information about the bridge's status by using passing cars which forward the packets to the central maintenance station. In case that there is no route available, a car might buffer the message and send it again once it approaches a more connected area. We give a brief overview of further applications in the following list. We believe that in the future ad-hoc networks will be deployed almost anywhere in our everyday life.

- Distributed networks for data collection and device monitoring: sensor networks, home networks, inter-vehicle communications, supply chain management, ...
- Military applications: Communication between soldiers, soldiers monitoring, sensor networks for target detection and identification, ...
- Emergency situations where the existing network infrastructure is not reliable or has been damaged due to geopolitical instability, a natural or man-made disaster, ...
- Provision of wireless connectivity in locations where cellular networks present insufficient coverage, are more expensive, or are wanted to be by-passed (e.g. for privacy reasons).
- Creation of instant and temporary networks for ad-hoc meetings, conferences, or brainstorming.

Sensor networks mainly find applications in the first mentioned field of distributed data collection and device monitoring. Already today, sensor networks are used in battlefield areas to sensor an area and send information via satellite to the main station. In the future, the extremely expensive satellite connection might be replaced by a multi-hop fashion communication network. Sensor networks will also be deployed for the readings of environmental conditions of any kind such as temperature and lightning to detect a forest fire or to adjust climate controls.

## 2.5 Security Issues

The security issues of ad-hoc networks and embedded systems are different to those of traditional computer systems. The latter are well known, and there exist several security solutions such as encryption software, secure communication protocols such as SSL, firewalls, virtual private networks (VPN), intrusion detection systems (IDS), and so on. Furthermore, the network properties of ad-hoc networks restrict possible solutions.

### 2.5.1 Security Goals

We see the following main security goals:

- *Authentication*: Authentication is one of the main security goals in all networks. Also in ad-hoc and sensor networks it is the basis for secure protocols. Authentication can easily be provided if a key agreement or key distribution scheme is available. However, there are authentication schemes available that do not require these. We go into more detail of authentication in the next chapter.

- *Confidentiality*: Confidentiality is another main security goal in all kind of networks. Confidentiality can only be provided if a key agreement or key distribution scheme is available by means of an encryption scheme. We consider key agreement and key distribution schemes in the next chapter, too.

- *Secure routing*: Secure routing is necessary in ad-hoc and sensor networks as a basic security protocol in order to guard against attacks such as malicious routing misdirection. Since there is no fixed infrastructure and due to the mobility, traditional routing schemes are vulnerable to malicious attacks. The design of secure routing protocols is difficult to achieve due to the same reasons and furthermore due to the resource constraints of the network devices. For instance, it is not possible to digitally sign each data packet in order to validate its origin as this is too resource demanding. On the other hand, as a node's neighborhood changes frequently, performing a costly key-agreement scheme in order to use efficient symmetric schemes for authenticating data packets might be too demanding as well. Furthermore, using a public-key infrastructure (PKI) contradicts the idea of a decentralized and self-organized network. As there might always be malicious or compromised nodes in an ad-hoc network, the malfunction of these nodes should not endanger the routing capabilities of the entire network. The situation for sensor networks is slightly different. Here, nodes are mainly static such that routes can be set up initially and extended at the introduction of new sensors.

- *Stimulation of cooperation*: Besides routing the stimulation of cooperation is another main security goal. Since the resources of the network devices are constraint devices tend to be selfish. Hence, some kind of stimulation is required to motivate cooperation in the network. Several approaches work by giving incentives for a successful cooperation whereas other approaches punish selfish behavior. Clearly, such a scheme must be secure against malicious manipulation. Otherwise, devices will be selfish enough to cheat in order to save battery power, especially if their gain is higher than their effort.

The latter two security goals are clearly based on the properties of ad-hoc and sensor networks, namely the lack of a fixed infrastructure and the need for cooperation. We see secure routing and stimulation of cooperation as crucial security protocols in ad-hoc and sensor networks. However, these kind of protocols will in most cases be based on a cryptographic authentication scheme and possibly on an encryption scheme. Hence, in this thesis we focus our main interest on the cryptographical aspects of authentication in ad-hoc and sensor networks as existing solutions are not satisfactory yet. We will consider secure routing and stimulation of cooperation protocols only briefly as an application of the underlying authentication schemes.

### 2.5.2 Restrictions

The properties of ad-hoc networks limit possible solutions to provide authentication and other security goals such that traditional solutions can apply very little here. The constraints that heavily influence possible approaches are as follows:

- *Resource constraints*: Most of the devices are heavily resource constrained. The devices are usually equipped with an 8 or 16-bit CPU and have only little memory and bandwidth. Sensor devices are even more restricted. Furthermore, there is only little battery power available. While the deployed cryptographic algorithms are nearly irrelevant on a desktop PC due to the large available computing resources, the situation for embedded devices as they are used for ad-hoc networks is completely different. Here, it is important to use extremely efficient algorithms both regarding algorithmic efficiency as well as power efficiency. Of particular importance is the fact that the energy consumption caused by CPU execution time as well as radio transmission is considerable relative to resources available. Thus, the running time as well as the transmission overhead should be minimized. Often desktop PCs are faster than embedded devices by a factor of 1000. Hence, asymmetric cryptography should be avoided since it requires very demanding arithmetic operations such as 1024-bit long number operations. As there is sometimes no alternative to using asymmetric cryptography, it should only be used rarely and asymmetric algorithms should be implemented very carefully. This is especially important for applications with real-time requirements where the execution time has to be considered thoroughly. A main goal of our proposed protocol is to approach asymmetric functionality with mainly symmetric methods to minimize the resource requirements.

- *Physical security*: The basic components of security protocols are cryptographic algorithms. Asymmetric and symmetric cryptographic methods are based on the fact that the executing device, e.g., the ad-hoc network node, uses a secret key that cannot be compromised by an adversary. However, the devices used in ad-hoc networks are highly exposed to the user and maintenance staff, both of whom may want to compromise the system. This is even more prevalent for sensor devices as they are deployed by the thousands in untrustworthy environments. To compromise such a system, a reverse engineering attack or a side-channel attack might be applied. Side-channel attacks use information discovered through analysis of the power consumption, time behavior, electric and magnetic field (EMF) radiation of a cryptographic device, and fault introduction to induce knowledge about the secret key. These attacks were introduced in the 1990's, and there are several countermeasures known both in software and hardware. The countermeasure usually obfuscate cryptographic algorithms by performing additional operations. Hence, they use more power and result in higher running times. However, it is unclear today whether it is possible to implement a general countermeasure that makes any side-channel attack impossible, or if new countermeasures need to be introduced once a new attack gets known. Another kind of physical attack are reverse engineering attacks. Here, the structure of a hardware chip is re-engineered in order to obtain any secret key. For instance, the content of a flash memory chip might be analyzed, or the communication bus between CPU and memory might be eavesdropped. Tamper resistant hardware modules deny access to the inside data. However, it is believed that building perfectly tamper resistant hardware is impossible [1, 3]. Furthermore, as devices are intended to be very cheap, only a few countermeasures against physical attacks are implemented. Thus, protocols always have to be designed in such a way that the effort of a successful attack is larger than the gain. For instance, the compromise of one device should only affect this device but must not give any advantage in compromising another device.

- *Limited maintenance*: Security flaws are published almost every day for today's software technology. After a security leak becomes known, software patches are installed, and virus scanners are loaded with a new definition file. However, sensor networks and also ad-hoc networks might not be connected to the Internet and might not provide the functionality for software updates. This is especially pertinent if an update is impossible such as when the devices are not programmable, or when the security functionality is provided in hardware. Thus, careful security engineering during the design is necessary to keep the risk of security flaws as small as possible. The possibility of security leaks must be considered before deployment especially for embedded devices but also for sensor nodes where often standard operating systems such as Embedded Linux are used.

Sensor networks face the same kind of security issues as ad-hoc networks but are even more challenging. Sensor devices are more resource constrained, provide less physical security and are usually easily accessible, and are even harder to maintain. Physical security cannot be established at a high level with today's technology, and it is unclear how a security update can be installed.

## 2.6 Network and Adversary Model

In this thesis our main focus is the cryptographic authentication. For this reason, we assume ad-hoc networks and sensor networks of the most general case when not stated otherwise. Authentication schemes that satisfy such a general scenario also work in more specific scenarios. A main goal of this thesis is to provide efficient authentication schemes for constrained devices where there possibly is no central service available. Ad-hoc networks and sensor networks are the most popular applications for such efficient authentication. Hence, we often consider ad-hoc and sensor networks simultaneously although these networks might differ considerably in their structure and properties. Note that the main security goal is to provide authentication. In several scenarios using the presented schemes it is also easily possible to provide a confidential channel. However, this is not necessarily always the case.

### 2.6.1 Network Model

The physical layer of a wireless network is vulnerable to denial of service attacks such as radio jamming. Such attacks on the physical layer have extensively been studied and countermeasures were proposed such as the spread spectrum mechanism [65]. Since this thesis mainly considers cryptographic solutions on the application layer we do not take into account physical layer attacks here. The same holds for attacks on Medium Access Control protocols.

We assume that devices communicate in a multi-hop fashion to many other devices. Communication is done via wireless links, and the links are symmetric, i.e., the nodes have the same transmission range and the established communication channel is always bidirectional. The network links do not guarantee reliable delivery of packets such that data packets might be dropped, corrupted, reordered, or duplicated. If a device $A$ wants to establish an authenticated channel to a device $B$ usually there are several nodes in between the path from $A$ to $B$. Hence, a key exchange over the insecure channel is only possible by an asymmetric key agreement scheme but not by the direct handover of a key (e.g., by physical contact). There are no channels available between devices besides the digital communication channel. In particular, there are no so-called side-channels such as provided by information exchange of devices' users.

### 2.6.2 Device Model

The devices are heterogenous, and there are no base stations involved. The devices abilities might vary greatly from nodes with little computational power to powerful devices with a large memory. Our goal is to design authentication protocols that work in general such that we always assume that a resource limited device authenticates to another resource limited device. Thus, the considered device class is not able to perform digital signatures frequently, or it is even not able to perform digital signatures at all. The keys are set up in different ways according to the protocol security goals. The keys might be distributed before the deployment of the devices or there might be asymmetric cryptography schemes be used to set up keys. However, in the most general case the devices are not able to perform any asymmetric cryptography and there might be no infrastructure available to pre-distribute keys.

### 2.6.3 Adversarial Model

In general, we assume an active adversary here. An active adversary is able to eavesdrop and manipulate all messages. More precisely, the adversary has full control of the communication channel. He is able to read messages, modify and delay messages or send them multiple time, and inject new messages. The adversary is always computationally polynomial bound.

The main goal of the adversary is the forgery of an authentication. We assume that the adversary aims for an existential forgery in a chosen message scenario where Alice authenticates to Bob. The adversary can choose any message that Alice authenticates. The adversary is successful if Bob accepts a message that was not authenticated by Alice before but was created by the adversary.

As there might be many independent attackers we assume that all adversary instances are controlled by only one powerful adversary. Note that this approach of colluding attackers is stronger than assuming many independent adversaries. The adversary is able to claim multiple identities at the same time and to change identities at will [1]. The adversary has full control over all compromised devices and full access to any data stored in the device including any cryptographic keys. We assume that the adversary is able to overcome any tamper resistant hardware module. However, the adversary is rational such that he will only attack a system and overcome security restrictions in order to gain knowledge of secret key material if the gain is larger than the effort. We do not take into account denial of service attacks on the application layer as almost all authentication schemes are vulnerable to such an attack. Furthermore, if there is any trusted central authority available in the network we do not consider the case in which an adversary compromises this trusted authority since then the entire network is compromised.

## 2.7 Categorization

Finally, we can point out that ad-hoc and sensor networks appear in a variety of settings. These can be distinguished in several dimensions. We classify ad-hoc and sensor networks according to the following categories, and summarize consequences:

- **Network**
  - *Connection quality:* The network channels might be reliable, or packet loss might occur frequently.
  - *Infrastructure:* There are various types of different network topologies. For instance, we call a network without any central services nor any pre-established knowledge a *pure network*. Furthermore, we call a *temporary network* a network where there is no permanent connection to a central service available but a temporary one. This could be a network that is connected to a base station once in a while. There might also be a permanent Internet connection or a connection to some central server available. For instance, there

[1] This attack is known as Sybil attack [21]. The attack can also be exploited to attack routing and resource allocation schemes that can be encountered though [59].

  might be a designated base station that can be reached by most nodes via a multi-hop connection.

  - *Communication strategy:* The devices might use different strategies for communication. They might communicate in a multi-hop fashion with any other node of the network, or they might only communicate with their direct neighborhood in a single-hop fashion.
  - *Device class:* The devices might be homogeneous such that they are equal without any hierarchy. It is also possible that devices are clustered in such a way that there is a more powerful master device for each cluster.
  - *Mobility:* Devices might be static or mobile. In latter case, new trust associations need to be established frequently.

- **Devices**

  - *Tamper resistance:* Protocols might be designed in such a way that they require some kind of tamper resistance. However, such assumption should always be done very carefully since all devices can be broken. Devices might have no kind of tamper resistance build in, or they might implement some limited countermeasures.
  - *Computational power:* The devices abilities regarding its computational power, memory storage, and bandwidth limit possible protocols.
  - *Battery resources:* The battery resources are a valuable good causing selfish behavior of nodes.

- **Applications**

  - *Infrastructure resources:* Applications determine the available infrastructure. This includes everything from a trusted third party (TTP) such as a certificate authority to a permanent or temporary communication channel in order to update devices.
  - *Key distribution:* The key distribution often depends on the available infrastructure. To use pre-distributed key schemes often some infrastructure is required in order to renew or revoke keys. Otherwise, schemes are required to agree on keys at running time.

- *Number of devices:* Networks might consist of a few devices only as in a home setting, or of millions in a battlefield scenario. Furthermore, devices might be added to the network or replaced.
- *Response time:* The application might require a real-time response of the devices, or the response time might not matter at all.
- *Number of authorities:* The application might involve several authorities each having one or some devices, or it might only involve a single or a few authorities each having several devices.

Before choosing or designing security protocols, all properties of the ad-hoc network including the application scope should be considered before appropriate authentication protocols are selected based on the categorization. Table 2.1 gives an overview of the property categories.

Table 2.1: Ad-hoc network categorization

| | | |
|---|---|---|
| Network | connection quality | reliable |
| | | packet loss |
| | infrastructure | pure network |
| | | temporary network |
| | | Internet availability |
| | communication strategy | global approach |
| | | local neighborhood approach |
| | device class | homogenous |
| | | heterogenous |
| | mobility | static |
| | | mobile |
| Devices | tamper resistance | none / some |
| | computational power | able to perform frequent/sporadic/ none PK operations |
| | | able to send certificates |
| | | able to store certificates |
| | battery resources | able to add cryptographic overhead |
| Applications | infrastructure resources | trusted authority |
| | | network infrastructure |
| | key distribution | at running time / pre-distribution |
| | number of devices | little / many |
| | | static / dynamic |
| | response time | real-time / non real-time |
| | number of authorities | one / many |

# 3 Background and Related Work

We now present work related to authentication and security in ad-hoc and sensor networks. We start by giving an overview of authentication and the closely related topic of key-distribution. Then we give a brief overview of embedded systems and security applications. Furthermore, we describe existing technologies in the context of ad-hoc networks. Finally, we give an overview of today's scenarios and analyze how well existing solutions suit to these scenarios.

## 3.1 Authentication

Authentication comes in several flavors, and there are various methods known to provide authentication. There is entity authentication as well as message authentication. Authentication can be provided in a unilateral, mutual (pairwise), or broadcast fashion. We start by stating a definition of these terms provided in [53].

**Definition 3.1.1** Data origin authentication *is a type of authentication whereby a party is corroborated as the (original) source of specified data created at some (typically unspecified) time in the past.*

By definition, data origin authentication includes data integrity.

**Definition 3.1.2** Message authentication *is a term used analogously with data origin authentication. It provides data origin authentication with respect to the original message source (and data integrity, but no uniqueness and timeliness guarantees).*

**Definition 3.1.3** Entity authentication *is the process whereby one party is assured (through acquisition of corroborative evidence) of the identity of a second party involved in a protocol, and that the second has actually participated (i.e., is active at, or immediately prior to, the time the evidence is acquired).*

Entity authentication is also often referred to as *entity identification* or only identification. Message authentication is closely related to entity authentication and recognition. The difference between message authentication and entity authentication is the lack of a timeliness guarantee in the former with respect to when a message was created. This follows from the fact that providing a message authentication gives no guarantee that the message authentication was created immediately prior to sending the message. Thus a major difference between entity authentication and message authentication is that entity authentication involves two parties communicating actively. To provide this requirement an entity authentication scheme has to involve some kind of clock or timeliness.

### 3.1.1 Basic Authentication Schemes

Entity as well as message authentication is provided by public-key schemes as digital signature and by symmetric-key scheme as message authentication code (MAC).

**Asymmetric Authentication:** In a public-key scenario each entity has a certificate $\langle PK \rangle$ issued by a certificate authority (CA) and an assigned public/private key pair. A unilateral message authentication works as presented in Algorithm 3.1. Let $SIG(m, SK)$ be a signature of the message $m$ by the private key $SK$, and $VER(S, m, PK)$ be the verification of the signature $S$ to the message $m$ by the public key $PK$. $VER(S, m, PK)$ is valid if $S$ is the signature of $m$ by the corresponding secret key of $PK$, i.e. if $S = SIG(m, SK)$, and it is invalid otherwise.

**Algorithm 3.1** Public-key message authentication

```
1: $A$ signs a message $m$ as $S := SIG(m, SK)$ and sends $m, S, \langle PK \rangle$ to $B$.
2: $B$ verifies whether $\langle PK \rangle \overset{?}{=} valid$ and whether $VER(S, m, PK) \overset{?}{=} valid$.
```

This can easily be converted into an entity authentication process by introducing a random challenge $r$ and using a challenge-response method as shown in Algorithm 3.2. Here, the challenge $r$ guarantees the timeliness.

**Symmetric Authentication:** In a symmetric-key scheme there is a mutually agreed trusted server $S$ which establishes the trust relationship between $A$ and $B$. Each entity shares a secret key with the server. First, $A$ asks $S$ to establish

**Algorithm 3.2** Public-key entity authentication

```
$B$ sends a random number $r$ to $A$.
$A$ signs the random number and sends $SIG(r, SK), \langle PK \rangle$ to $B$.
$B$ verifies whether $\langle PK \rangle \overset{?}{=} valid$ and whether $VER(S, m, PK) \overset{?}{=} valid.$
```

a relationship to $B$. Then $S$ sends $A$ and $B$ a session key $K$ which these can use for authentication. This is denoted in Algorithm 3.3. Here, $E(m, K)$ is the encryption of a message $m$ by the key $K$, $MAC(m, K)$ is the message authentication code of a message $m$ by a key $K$, and $K_A$ is the shared key between the server $S$ and $A$.

**Algorithm 3.3** Symmetric server message authentication

```
$A$ sends a *hello* message to $S$.
$S$ sends $E(K, K_A)$ to $A$.
$S$ sends $E(K, K_B)$ to $B$.
*For each authentication, do the following:*
$A$ computes $M := MAC(m, K)$ and sends $m, M$ to $B$.
$B$ checks whether $M \overset{?}{=} MAC(m, K)$.
```

As before, this can easily be converted into an entity authentication scheme by introducing a random challenge $r$ as shown in Algorithm 3.4.

**Algorithm 3.4** Symmetric server entity authentication

```
$A$ sends a *hello* message to $S$.
$S$ sends $E(K, A)$ to $A$.
$S$ sends $E(K, B)$ to $B$.
*For each authentication, do the following:*
$B$ sends $r$ to $A$.
$A$ computes $M := MAC(r, K)$ and sends $M$ to $B$.
$B$ checks whether $M = MAC(r, K)$.
```

The public-key scheme provides a signature, whereas the symmetric key scheme provides a key agreement by a central server, and then an authentication process. Hence, in the Algorithms 3.3 and 3.4 Steps $1 - 3$ only need to be performed once for each pair $A$ and $B$ if the entities store the shared key $K$, whereas the remaining steps are performed for each authentication process. Note that pre-distributed key schemes are a special variant of this scheme. Consequently, if all devices share the same key, or if there are pre-distributed keys, we only need to perform Steps $4 - 5$ and $4 - 6$ of above protocols, respectively. This is presented in Algorithms 3.5 and 3.6.

**Algorithm 3.5** Symmetric message authentication

```
1: $A$ computes $M := MAC(m, K)$ and sends $m, M$ to $B$.
2: $B$ checks whether $M \overset{?}{=} MAC(m, K)$.
```

**Algorithm 3.6** Symmetric entity authentication

```
1: $B$ sends $r$ to $A$.
2: $A$ computes $M := MAC(r, K)$ and sends $M$ to $B$.
3: $B$ checks whether $M \overset{?}{=} MAC(r, K)$.
```

**Hybrid Authentication:** In a public-key scenario, an equivalent approach is to use a key agreement scheme such as Diffie-Hellman (DH) followed by a symmetric MAC scheme for each authentication process. In practice, often such hybrid schemes are used as presented in Algorithms 3.7 and 3.8. Here, $a$ and $b$ denote the private keys, and $K_A := a \cdot G$ and $K_B := b \cdot G$ with base element $G$ denote the public keys of $A$ and $B$, respectively. Note that computation of $K$ is done in an abelian additive group such that $K = a \cdot K_B = a(bG) = b(aG) = b \cdot K_A$.

**Algorithm 3.7** Hybrid message authentication

```
1: $A$ sends $B$ its public key $K_A$.
2: $B$ sends $A$ its public key $K_B$.
3: $A$ computes $K := a \cdot K_B$.
4: $B$ computes $K := b \cdot K_A$.
   For each authentication, do the following:
5: $A$ computes $M := MAC(m, K)$ and sends $m, M$ to $B$.
6: $B$ checks whether $M \overset{?}{=} MAC(m, K)$.
```

**Time-Stamp Authentication:** To obtain entity authentication, interaction as proven by some timeliness is required. This might be done by a challenge-response method as described before, or by a time-stamp as presented in Algorithm 3.9. Note that here it is mandatory to include the receiver's identification in the authenticated message to prevent replay attacks. The current time $t$ is part of the authentication tag.

**Zero-Knowledge Proofs:** Another method of providing authentication and digital signatures are zero-knowledge proofs. We describe the Fiat-Shamir identification scheme [25] adapted to our scope. Let $n = pq$ be a modulus with $p$ and $q$ prime. Let $ID$ be a string describing the identity of Alice, and $f$ be a pseudo random function which maps bit-strings to the range $\{0, \ldots, n-1\}$.

**Algorithm 3.8** Hybrid entity authentication

```
$A$ sends $B$ its public key $K_A$.
$B$ sends $A$ its public key $K_B$.
$A$ computes $K := a \cdot K_B$.
$B$ computes $K := b \cdot K_A$.
For each authentication, do the following:
$B$ chooses a random value $r$ and sends it to $A$.
$A$ computes $M := MAC(r, K)$ and sends $M$ to $B$.
$B$ checks whether $M \overset{?}{=} MAC(r, K)$.
```

**Algorithm 3.9** Entity authentication with time-stamp

```
$A$ computes $M := E(t||B, K)$ and sends $M$ to $B$.
$B$ verifies that the time-stamp $t$ is acceptable and that the received identifier is its own.
```

A trusted authority $CA$ which knows $p$ and $q$ issues certificates. Therefore $CA$ computes the square root $id$ of $f(ID)^{-1}$ such that $id^2 = f(ID)^{-1}$. Note that the square root can easily be computed by the knowledge of $p$ and $q$ whereas otherwise its extraction is computationally infeasible if the parameters are chosen carefully. We can assume that $f(ID)^{-1}$ is a square number by attaching some bits to the identification string. Alice now obtains the secret key $id$ as well as the certificate $ID$. In order for Alice to authenticate herself to Bob, she performs Algorithm 3.10. Alice first chooses a random value $r$ and sends $r^2$ to Bob. Bob then challenges Alice by a bit $e$. Alice computes $t := r \cdot id^e$ and sends $t$ to Bob. If Bob sent the challenge $e = 0$, he verifies whether $t^2 \overset{?}{=} T$, otherwise he verifies whether $t^2 \overset{?}{=} T \cdot ID$. This check will always be successful if Alice knows the secret $id$ and computes correctly, since $t^2 = r^2(id^e)^2 = r^2(id^2)^e = r^2 ID^e = T \cdot ID^e$. There is a probability of 1/2 that Alice succeeds in the proof without knowing the secret $id$. For instance, she guesses that Bob will challenge her with $e = 0$. She then chooses a random $r$ and sends $t = r$. Hence, this protocol sequence is repeated $n$ times in order to obtain a sufficient level of security. An interactive proof provides an active communication such that $n = 30$ is supposed to be sufficient for entity authentication. These can be performed in parallel.[1]

The Fiat-Shamir scheme can also provide a digital signature for a message $m$. Here, Alice chooses $n$ random values $r_i$ and computes $T_i := r_i^2$. She then

[1] Note that the parallel version cannot be shown to be a zero-knowledge proof anymore. However, there are no security issues known.

**Algorithm 3.10** Fiat-Shamir authentication

```
for $i = 1$ to $n$ do
    $A$ randomly chooses $r \in \mathbb{Z}_n$ and sends $T := r^2$ to $B$.
    $B$ randomly chooses the challenge $e \in \{0, 1\}$ and sends $e$ to $A$.
    $A$ sends $t := r \cdot id^e$ to $B$.
    $B$ verifies whether $t^2 \cdot ID^e \overset{?}{=} T$.
end for
```

computes $h(m||T_1|| \ldots ||T_n)$ and uses the first $n$ bits as challenges $e_i$. Alice then computes $t_i := r_i \cdot id^{e_i}$ and sends the signature $(m, e_1, \ldots, e_n, t_1, \ldots, t_n)$ to Bob. Bob now verifies the signature by computing $c_i := t_i^2 \cdot ID^{e_i}$ and verifying whether the first $n$ bits of $f(m, c_1, \ldots, c_n)$ equal $(e_1, \ldots, e_n)$. For ad-hoc networks the data overhead of this scheme can be immense even when using efficiency improvements. For instance, only the first $n$ bits of $T$ need to be sent, and several parallel versions can be performed. However, there are several elements of $\mathbb{Z}_n$ sent over the wireless link. As $n$ is in the range of a 1024-bit number, this scheme demands high bandwidth resources. Note that a key-exchange in zero-knowledge proofs is possible [6].

### 3.1.2 Mutual and Broadcast Authentication

Until now we considered unilateral authentication. We now introduce mutual authentication, which we also call pairwise authentication, as well as broadcast authentication. Mutual authentication is an authentication process whereby both involved parties provide a proof of authentication.

**Basic Mutual Authentication:** Mutual authentication can be achieved by performing a unilateral authentication twice. However, in this case there is no logical connection between these. Thus mutual authentication can be achieved as presented in Algorithm 3.11. The number of exchanged messages is reduced compared to doing a unilateral authentication twice. Obviously, mutual authentication can be provided by digital signatures in a very similar way [53].

**Basic Broadcast Authentication:** Broadcast authentication is the process where one entity authenticates messages to several parties. A symmetric scheme requires that keys are distributed pairwise whereas in an asymmetric scheme each entity has a public/private key pair. Hence, a MAC only provides mutual (pairwise) authentication whereas a digital signature also provides broadcast authentication. For instance, an entity is able to sign a

**Algorithm 3.11** Mutual entity authentication

```
1: $B$ sends random $r_B$ to $A$.
2: $A$ computes $M_A := MAC(r_B||r_A, K)$ and sends $M_A, r_A$ to $B$.
3: $B$ checks whether $M_A \overset{?}{=} MAC(r_B||r_A, K)$, computes
   $M_B := MAC(r_A||r_B, K)$ and sends $M_B$ to $A$.
   $A$ checks whether $M_B \overset{?}{=} MAC(r_A||r_B, K)$.
```

message which is then verified by several receivers. Broadcast authentication is often coupled with non-repudiation because an authenticated message can be verified by a third party.

**Asymmetric MAC Broadcast Authentication:** There is a symmetric approach of providing broadcast authentication due to Canetti et al. [11]. They propose an *asymmetric MAC* scheme which is based on multiple symmetric keys and MACs for $n$ receiving parties as presented in Algorithm 3.12. The goal is to achieve the same capabilities as a digital signature with symmetric MACs only. Let $S$ be a set of keys. Each entity then gets a subset of the keys $S_i := \{k_1, \ldots, k_n\} \in S$ at initialization time. The main set and the subsets are chosen in such a way that the probability that each two subsets have at least one key in common is high. If Alice wants to authenticate a message $m$, she computes the MAC over the message by each key of her subset $M_i := MAC(m, k_i)$ for all $k_i \in S_i$, and broadcasts $m, M_1, \ldots, M_n$. Due to the design of the subsets $S_i$, each receiver is able to find at least one key that she shares with Alice in order to verify $m$. This approach provides a computationally efficient broadcast authentication scheme where a key-agreement has to be performed before deployment. However, due to the large message overhead this approach is not easily applicable to ad-hoc networks where bandwidth is very restricted.

**Algorithm 3.12** Asymmetric MAC

```
1: Provide each party with a subset of keys $S_i := \{k_1, \ldots, k_n\} \in S$.
   *For each authentication, do the following:*
2: $A$ computes $M_i := MAC(m, k_i)$ for all $k_i \in S_i$ and broadcasts
   $m, M_1, \ldots, M_n$.
3: Each receiver $B$ determines a key $k_j$ that he shares with $A$ and verifies
   whether $M_j \overset{?}{=} MAC(m, k_j)$.
```

### 3.1.3 Further Authentication Schemes

In the following we present further authentication schemes.

**Lamport's one-time passwords:** We start with Lamport's one-time password scheme [53] that is based on hash-chains. Here, Alice randomly chooses a secret $w$. A hash function $h$ is used to define a sequence of passwords $w, h(w), h(h(w)), \ldots, h^t(w)$ where $h^i(w)$ is the repeated $i$-th iteration of $h$. The password for the $i$-th authentication is defined as $w_i = h^{t-i}(w)$. Then the protocol works as presented in Algorithm 3.13. It should be clear that this scheme does not overcome the need for an authenticated initial key-exchange. By the definition of entity authentication, Lamport's one-time passwords do not provide entity authentication as there is no proof of an active communication included. However, as hash-chains are very popular for authentication schemes in ad-hoc networks, we present the scheme here.

**Algorithm 3.13** Lamport's one-time password

```
Initial Phase: $A$ transfers $w_0$ in an authentic manner to $B$. $B$ stores $w_0$
  and initializes its counter $j \leftarrow 1$.
for each authentication $i = 1$ to $t$ do
    $A$ sends $w_i, i$ to $B$
    $B$ checks whether $i \overset{?}{=} j$ and $h(w_i) \overset{?}{=} w_{i-1}$. If the checks succeed, $B$
      stores $w_i$ and $j \leftarrow j + 1$ for the next session.
end for
```

**TESLA:** TESLA [63] is a protocol for broadcast authentication of messages. As presented before, the asymmetric MAC scheme by Canetti et al. induces a large message overhead. Performing a unilateral message authentication with each receiver as presented in Algorithm 3.7 is inefficient. TESLA follows a different approach by introducing a clock. Here, the sender Alice first generates a hash-chain with temporary keys $k_n, k_{n-1} = h(k_n), \ldots, k_0 = h(k_1)$. First, the final element $k_0$ is broadcasted to all receivers over a secure channel, e.g., by signing it. Then Alice sends messages $m_i$ authenticated by $k_i$ in time interval $t_i$. Such a message is only accepted during the time interval $t_i$ but not later. In the next time interval, Alice opens $k_i$ and the receivers verify $m_i$. The protocol works as presented in Algorithm 3.14. Note that the receiver has to buffer messages before they can be verified. Furthermore, there needs to be time synchronization between the sender and receiver. Otherwise, after a key was opened an attacker could use that key to forge messages. Gong points out that maintaining synchronization of clocks requires a secure authentication scheme [31]. Consequently, basing an authentication (or recognition) scheme on clock synchronization requires another authentication scheme to validate the time. There is also a version called $\mu$TESLA that is part of the SPINS protocol suite for sensor networks [64]. Here, a pre-distributed key is used for the initial authentication of the final hash-chain key $k_0$.

**Algorithm 3.14** TESLA broadcast authentication

```
Initially, $A$ signs $S := SIG(k_0, SK)$ and broadcasts $S$. Each verifier verifies $S$.
for message $m_i$ in time interval $t_i, i = 1$ to $n$ do
    $A$ computes $M_i := MAC(m_i, k_i)$ and broadcasts $M_i, m_i$.
    Each receiver checks whether he received $M_i, m_i$ in time interval $t_i$ and buffers it.
end for
for message $m_i$ in time interval $t_{i+1}, i = 1$ to $n$ do
    $A$ broadcasts $k_i$.
    Each receiver checks whether $M_i \overset{?}{=} MAC(m_i, k_i)$.
end for
```

**Guy Fawkes:** The Guy Fawkes protocol [2] uses key commitments by a hash function $h$ to provide message authentication as presented in Algorithm 3.15. The sender Alice first commits to a key $k_0$ by broadcasting $h(k_0)$. In each authentication step, she commits to a new key $h(k_i)$ and computes the MAC over $m_i$ and $h(k_i)$ under $k_{i-1}$. After broadcasting the MAC, Alice opens $k_{i-1}$ such that all receivers can verify the message $m_i$. The Guy Fawkes protocol does not use a hash-chain but a commitment chain: Each authenticated message includes a commitment to the key used for the next authentication step. The protocol requires that Alice makes the commitment $a_i$ public, and that Alice knows that Bob received $a_i$.

**Algorithm 3.15** Guy Fawkes authentication

```
$A$ chooses a random key $k_0$ and sends $h(k_0)$ to $B$.
for message $m_i, i = 1$ to $n$ do
    $A$ chooses a key $k_i$ and commits to $a_i := MAC(m_i||h(k_i), k_{i-1})$
    $A$ makes $a_i$ public
    $A$ sends $(m_i, h(k_i), k_{i-1})$ to $B$
    $B$ checks whether $a_i \overset{?}{=} MAC(m_i||h(k_i), k_{i-1})$ and whether $k_{i-1}$ is indeed the codeword committed to in the last round
end for
```

In the original scheme the commitment would be published in a newspaper such that the commitment would be stored in a public directory with a time-stamp and could be verified at any time. However, in an ad-hoc network, in most cases there is no such central directory to provide time-stamps, so an explicit acknowledgement of the receipt is necessary for the security of the protocol. This can be provided for signing a bidirectional stream as presented in Algorithm 3.16. Here, Alice and Bob want to sign a stream of messages $m_0, m_1, \ldots, m_n$ and $m'_0, m'_1, \ldots, m'_n$, respectively. Again, let $a_i := MAC(m_{i+1}||h(k_{i+1}), k_i)$ be the commitment to the following message $m_{i+1}$ and $h(k_{i+1})$ under $k_i$. Note that this scheme only requires a secure relay channel for the initial step but no confidential channel.

**Algorithm 3.16** Guy Fawkes bidirectional authentication

```
$A$ chooses a random key $k_0$ and sends $m_0, a_0, h(k_0), SIG(m_0, h(k_0))$ to $B$.
$B$ chooses a random key $k'_0$ and sends $m'_0, b_0, h(k'_0), SIG(m'_0, h(k'_0))$ to $A$.
$A$ sends $h(b_0, k_0)$ to $A$.
$B$ sends $h(a_0, k'_0)$ to $B$.
for messages $m_i, m'_i, i = 1$ to $n$ do
    $A$ chooses a key $k_i$, commits to $a_i$ and sends $m_i, a_i, h(k_i), k_{i-1}$ to $B$.
    $B$ chooses a key $k'_i$, commits to $b_i$ and sends $m'_i, b_i, h(k'_i), k'_{i-1}$ to $A$.
    $A$ sends $h(b_i, k_i)$ to $B$.
    $B$ sends $h(a_i, k'_i)$ to $A$.
end for
```

**Remote User Authentication:** The remote user authentication protocol overcomes this problem [55] as presented in Algorithm 3.17. First, there is an initial set up phase where Alice wishing to authenticate to Bob sends a random string $X$ and a set of MACs $S := \{MAC(X, k_i) \text{ for all } i\}$ keyed by a set of secrets $k_i$. Each time she wishes to authenticate to Bob, a new set of secrets must be generated and MACs of subsets of these keys are passed back and forth. Bob challenges Alice to prove knowledge of a subset of keys. The protocol could be adapted to authenticate messages instead of users. A selection of 17 out of 35 committed values is recommended which leads to a large message overhead. Note that the scheme does only require a reliable relay for the initial message and no confidential channel.

**Algorithm 3.17** Remote User authentication

```
$A$ chooses random keys $k_i, i \in S$ and a random string $X$ and sends $U_i := MAC(X, k_i)$ for all $i$ to $B$.
**for** succeeding authentication phase $j$ **do**
    $A$ chooses random keys $k'_i$ and a random $X'$ and computes
    $U'_i := MAC(X', k'_i)$ for all $i$. She then sends $W_i := MAC(V_i, k_i)$ for all $i$ to $B$.
    $B$ chooses a subset $S' \subset S$ and sends $S'$ to $A$.
    $A$ sends the subset of keys $k_i, i \in S'$ as well as all $U'_i$ and $X'$.
    $B$ now verifies the subset $U_i, i \in S'$ as well as $W_i, i \in S'$.
    $B$ sets $X \leftarrow X'$ and $U_i \leftarrow U'_i$, $A$ furthermore sets $k_i \leftarrow k'_i$.
**end for**
```

### 3.1.4 Overview

At the end of this section we give an overview of the presented authentication protocols in Table 3.1. There are two main columns, requirements as well as provisions of the authentication protocols. Let us first consider requirements. There are the requirements of a secure relay channel, key pre-distribution, a trusted third party (TTP) or public-key infrastructure (PKI), and time synchronization. A '-' means that the protocol does not require the service whereas a 'x' means that it is required. Note that the Guy Fawkes protocol requires a method to provide a reliable acknowledgement. Most protocols require a TTP to provide authentication. In case of digital signature schemes and Zero Knowledge schemes, the TTP is required to issue certificates. The provisions are organized similarly. It is presented whether a protocol provides entity authentication, unilateral and pairwise message authentication, and broadcast message authentication. Finally, it is stated whether protocols can be applied in a self-enforcing manner. Protocols can be used in such a way if there is no active central guidance involved. However, it is up to the higher-level security protocols to implement self-enforcement. A '+' states that a protocol provides this feature, otherwise it does not. Note that all protocols require some timeliness to provide entity authentication. Hence, all entity authentication protocols except the Time Stamp scheme are based on interaction. Table 3.2 states the efficiency of the protocols regarding computation and bandwidth. The entries represent efficiency in ascending order: $\ominus\ominus$, $\ominus$, $\oplus$, and $\oplus\oplus$. A protocol with entries $\oplus\oplus$ has negligible computational cost and induces little bandwidth overhead whereas a protocol with entries $\ominus\ominus$ can only be performed rarely or on powerful devices.

None of the presented protocols is able to provide authentication without a TTP or a key pre-distribution scheme in an efficient way. We believe that this scenario frequently emerges in practical applications. Later on in Chapter 5 we will present two authentication protocols for this scenario.

Table 3.1: Authentication protocols

| | requires | | | | provides | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | message auth. | | |
| | secure relay | key pre-dist. | TTP / PKI | time sync. | entity auth. | unil. / pairw. | broad-cast | self-enforc. |
| Digital Signature | − | − | x | − | + | + | + | + |
| Zero Knowledge | − | − | x | − | + | + | + | + |
| Hybrid | − | − | x | − | + | + | − | + |
| Symmetric Server | − | x | x | − | + | + | − | − |
| MAC | − | x | − | − | + | + | − | + |
| Time Stamp (MAC) | − | x | − | x | + | + | − | + |
| Asymmetric MAC | − | x | x | − | + | + | + | + |
| TESLA ($\mu$TESLA) | − | − (x) | x | x | + | + | + | + |
| Lamport | x | − | − | − | − | + | − | + |
| Guy Fawkes | x$^\dagger$ | − | − | − | − | + | − | + |
| Remote User | x | − | − | − | + | + | − | + |

$^\dagger$ := secure acknowledgement

Table 3.2: Protocol efficiency

| | Low Computations | Low Bandwidth |
|---|---|---|
| Digital Signature | ⊖⊖ | ⊕ |
| Zero Knowledge | ⊖ | ⊖⊖ |
| Hybrid | ⊖ | ⊖ |
| Symmetric Server | ⊕ | ⊕ |
| MAC | ⊕⊕ | ⊕⊕ |
| Time Stamp (MAC) | ⊕⊕ | ⊕⊕ |
| Asymmetric MAC | ⊕ | ⊖⊖ |
| TESLA ($\mu$TESLA) | ⊖(⊕⊕) | ⊕⊕ |
| Lamport | ⊕⊕ | ⊕⊕ |
| Guy Fawkes | ⊕⊕ | ⊕⊕ |
| Remote User | ⊕⊕ | ⊖⊖ |

## 3.2 Key Distribution

Closely related to authentication is key-agreement — if a key can be exchanged authentication can be provided efficiently by a MAC. Key distribution can be performed before deployment of a network or at the actual time when a key is required. We distinguish this as follows:

- *at running time by a*
    - secure channel, and by a
    - distributed public-key infrastructure
- *before deployment by*
    - key pre-distribution

In the following we discuss these three main mechanisms.

### 3.2.1 Secure Channel

If a secure channel is available, a secret key can obviously be exchanged. This can be done in several ways.

**Password-Based Key Agreement:** The work by Asokan and Ginzboorg [4] addresses the scenario of a group of people who want to set up a secure session in a meeting room without any supporting infrastructure. There are three main requirements for such a solution here: (1) only those entities that know the initial password are able to learn the session key; (2) the session key is formed of contributions from all entities such that no entity is able to reduce the key space; and (3) the protocol must not be vulnerable to an attacker who is able to insert messages. The work describes and introduces several password-based key-exchange methods that meet these requirements. The core idea of the protocol is as follows. A weak password is sent to the group members. Each member then contributes part of the key and signs this data by using the weak password. Finally, a secure session key is derived without any central trust authority or supporting infrastructure.

This model works perfectly for small groups. Authentication is done outside of the digital communication system, e.g., the group members authenticate themselves by showing their passports, or authentication is based on common knowledge.[2] The model does not suffice for more complicated environments, though. Groups of people who do not know each other, or pairs of people who want to communicate confidentially without anyone else of the group being able to eavesdrop on the channel, are two examples.

This model that uses a trusted side-channel to agree on a key is also known in slightly different form in Bluetooth. Here, a PIN – essentially a symmetric key – is entered into two devices by a keyboard in order to derive a strong key for a pairwise relationship. This model has similar drawbacks as the first approach.

**Resurrecting Duckling:** The *resurrecting duckling* policy was introduced by Stajano et al. in [75] and extended in [73]. The fundamental authentication problem is solved by a secure transient association between two devices establishing a master-slave relationship. The master is also referred to as mother duck whereas the slave is referred to as duckling. The relationship between mother duck and duckling is secure in the sense that they share a common secret, and transient because the association can be terminated by the master only. A master can always identify its slave in a set of similar devices. A shared key is exchanged via a secure channel, e.g., by physical contact once a device is initialized. This security model can be applied to very large ad-hoc networks such as networks consisting of smart dust devices [84]. A possible scenario is a battlefield of smart dust soldiers (acting as ducklings or siblings) and their general (acting as the mother duck). The mother allows its ducklings to communicate by uploading to each of them a highly flexible policy so that sibling entities become masters or slaves for a short time, enough to perform one transaction. The mother duck gives the ducklings credentials that allow them to authenticate themselves.

The Resurrecting Duckling scheme is an appropriate security model for a well defined hierarchy of trust relationships. It particularly suits inexpensive devices that are not equipped with a display or a powerful processor.

[2] In this case friendship or just knowing each other is considered common knowledge.

### 3.2.2 Distributed Public-Key Infrastructure

Key management for public-key infrastructures (PKI) requires a trusted entity called *Certificate Authority* (CA). The CA issues certificates by binding a public key to a node's identity. The CA should always be available because certificates might be renewed or revoked. Replicating the CA improves availability. However, such a central service runs contrary to the decentralized structure of ad-hoc networks.

**Distributed CA:** Zhou and Haas [96] propose to distribute the CA to a set of nodes by letting them share the key management service, in particular the ability to issue certificates. This is done using *threshold cryptography* [20]. An $(n, t+1)$ threshold cryptography scheme allows $n$ parties to share the ability to perform a cryptographic operation so that any $t+1$ parties can perform this operation jointly whereas it is infeasible for at most $t$ parties to do so. Using such a scheme the private key $k$ of the CA is divided into $n$ shares $(s_1, s_2, \ldots, s_n)$, each share being assigned to a special node. Using this share a set of $t+1$ special nodes is able to generate a valid certificate. As long as $t$ or less special nodes are compromised and behave malicious the service can operate. Even if compromised nodes deliver incorrect data, the service is able to sign certificates. Threshold cryptography can also be applied to well known signature schemes like the *Digital Signature Standard* (DSS) [28]. Another work describing a distributed certification scheme using threshold cryptography was presented by Kong et al. [44] and extended by Luo et al. [50]. To enable ubiquitous services, the certification service is performed by localized sets of nodes. The papers describe threshold functions for key renewal and key revocation but also secret share update. Special attention is spent on networking issues like scalability, mobility, network dynamics, and malicious nodes.

**Self-Organized PKI:** Another approach by Hubaux et al. presents a self-organized PKI [39]. The system replaces the centralized CA by certificate chains. Users issue certificates if they are confident about the owner's identity, i.e., if they believe that a given public key belongs to a given user. Each user stores a list of certificates in its own repository. To obtain the certificate of another entity the requester builds a certificate chain using his repository list and implicitly trusted entities' lists until it finds a path to an entity that has the desired certificate in its repository. It is assumed that the certificate requester trusts each node in the recommendation chain, i.e., trust is inherited via indirect relationships. A significant amount of computing power and time is consumed to obtain a certificate going through the certificate chain. Each node in the chain has to perform public-key operations, first to check the received certificate for authentication (signature verification) and then to sign it before forwarding it (signature generation). This cannot be done in parallel but only one after the other until the certificate went along the entire chain.

Despite its centralized nature, a central CA is preferable for applications with high-security demands. Entities in an ad-hoc network then need to wait until they can connect to a CA in order to perform a secured transaction. To ensure high availability, the CA can be replicated, where the replicated CAs are as secure as the original CA as long as the replication process is not vulnerable to attacks. The private key of the CA does not become weaker because of the replication.

### 3.2.3 Key Pre-Distribution Schemes

**Basic Scheme:** The most basic key pre-distribution scheme is the deployment of only one secret key on all network devices. In that way, all entities can use this master key to generate a new shared key. However, once one device is compromised, the entire network is compromised. Another possibility is to use a PKI as described before which requires some sort of centralized or distributed directory. To overcome this problem, entities could share a key pairwise, i.e., each device would have $N-1$ keys if there are $N$ devices in the network. For large networks, such an approach is infeasible because of the limited memory storage of devices.

**Random Key-Ring Scheme:** There are several proposals to reduce the number of stored keys in order to establish pair-wise keys, e.g., [22, 15]. In the approach by Eschenauer and Gligor [22] each entity is given a random set of keys called a key-ring out of a very large pool of keys. A set of 75 keys drawn out of a pool of 10,000 keys should suffice such that the probability that each two entities share at least one key is larger than 1/2. Then, if Alice and Bob want to agree on a key, they check if they have a key in common in their set of keys. If they do not share a common key, they set up a secure chain via their neighborhood. In the easiest case there is a node in the neighborhood of Alice and Bob that shares a secret key with both Alice and Bob to bridge the gap between Alice and Bob. The scheme provides key revocation and key renewal, incremental addition of further nodes, and it is robust against a set of compromised nodes. If the number of compromised nodes is below a certain threshold, the probability that any other nodes are affected besides the compromised ones is negligible. However, if the number of nodes is very large or if the probability that two nodes share a key should be close to one, the key pool as well as the size of individual key-rings must be increased.

## 3.3 Cryptography on Embedded Systems

As said before, ad-hoc and sensor networks are mainly formed by embedded systems. To design security protocols for such networks, it is important to estimate running times for cryptographic primitives. While symmetric primitives can be implemented very efficiently and have a running time of a few milliseconds even on very constrained devices, asymmetric primitives that are needed for key agreement and digital signatures are by far more demanding. Hence, it is important to get a feeling for the running times of asymmetric primitives. In the following, we present our work in this respect.

Elliptic curve cryptography (ECC) provides efficient asymmetric cryptography and is a good fit for embedded devices as it requires operands that are far shorter than the ones used for RSA. In [87] we presented an ECC implementation over binary curves of a 163-bit NIST curve on a Palm OS device. This device has a 16 MHz CPU with 2 MB of memory. Such a device is a representative of ad-hoc network devices at the lower end. We were able to implement an elliptic curve point multiplication for a random point in 1.5 $s$ and for a fixed point in 0.9 $s$. Thus, an ECDSA [52] signature verification can be done in 2.4 $s$, a Diffie-Hellman key agreement takes 1.5 $s$, and a signature generation takes about 0.9 $s$. Note that a 163-bit curve is considered to be roughly as secure as 1024-bit RSA. This security level is supposed to be secure enough for the next few years, i.e., such a security level can be used for applications that do not require encrypted data to be still confidential in the next decade.

Another work we presented is an ECC implementation for an 8051 processor which is used as a sensor device. The device consists of an 8-bit 8051 CPU, a radio transceiver, and a power supply. We presented in [45] a prototype that provides a Diffie-Hellman key exchange using ECC with a stationary device. The ECC library uses OEFs, that are a special form of extension fields especially fitted to the register size of the hardware. In this case, a curve over $GF(p^{17})$ was used with $p \approx 2^8$ such that the curve supports a key length of $2^{134}$ which provides an appropriate security level for short-term applications. To perform an ECC point multiplication, the device requires around 3 $s$. Hence, a key agreement takes around 3 $s$.

Above implementations do not take into account side-channel attacks. Hence, it is to expect that an implementation that implements countermeasures for all known attacks is slower. Symmetric cryptography can be expected to be faster than asymmetric cryptography by a factor of 1000. Asymmetric cryptographic algorithms can be sped up extremely by using a hardware coprocessor. Using such a device, signature operations can be performed in less than 50 $ms$. However, this coprocessor raises the cost of a device such that it will be heavily application dependent whether such a coprocessor is available or not. We do not believe that hardware coprocessors will be implemented in typical future devices used for ad-hoc and sensor networks. Clearly, future devices will become more powerful according to Moore's law. At the same time, devices will become smaller at constant computing power.

## 3.4 Security Protocols

On top of basic cryptographic protocols such as authentication, more complex security protocols can be built. We will give a brief overview of secure routing and secure cooperation based schemes.

**Secure Routing:** Routing is a crucial protocol in ad-hoc networks and has several facets. In an ad-hoc network, each node acts as a router. To route packets from a source to a destination, the nodes depend on each other in forwarding the traffic. Wired network routing protocols do not handle well the frequent topology changes of ad-hoc networks due to the device's mobility. Hence, the design of ad-hoc network routing protocols is a challenging quest. Most routing protocols for ad-hoc networks have been studied in a trusted environment where there are no adversaries and where nodes play fair. In a more realistic setting an adversary might try to disrupt the communication.

Since data forwarding diminishes the battery power the nodes might also be tempted not to participate. Other nodes might act in a malicious way to compromise the network. Thus routing should be robust against malicious, compromised, and selfish nodes. Since we focus on the cryptographic basis to provide authentication we do not consider further security problems in routing schemes besides authentication. Obviously, any secure routing scheme must be based on authenticated data packets to uniquely identify the source of the data packets. A simple approach is to let the sender sign its data packets whereas each intermediate node as well as the receiver verify the signature. However, such an approach is unrealistically due to the limited computational power of devices such that more efficient schemes are needed.

Secure routing schemes have been proposed mainly as extension of to well established routing schemes such as DSR and AODV. The intuitive approach is to incorporate security to an existing ad-hoc network routing protocol. The SEAD (Secure Efficient Ad-hoc Distance vector) routing protocol proposed by Hu et al. [36] is based on the Destination-Sequenced Distance-Vector (DSDV) ad-hoc network routing protocol. The protocol is based on efficient one-way functions only and does not require asymmetric algorithms. SEAD is robust against multiple uncoordinate active attackers such that it provides an efficient and practical routing protocol. However, SEAD adds overhead and latency to the network.

A secure on-demand routing scheme called ARIADNE was also proposed by Hu et al. [38]. ARIADNE is a secure version of the Dynamic Source Routing (DSR) protocol. It discovers routes dynamically when they are needed on demand. ARIADNE is efficient as it only requires symmetric cryptographic primitives for authentication. As underlying authentication the efficient TESLA scheme can be used. ARIADNE prevents attackers to tamper with uncompromised routes consisting of benign nodes only. While ARIADNE assumes security associations between all involved nodes, BISS by Capkun and Hubaux [12] reduces this assumption to requiring only the destination node to have security assocations with all nodes on the route.

Efficient security mechanisms for routing protocols were again presented by Hu et al. [37]. This scheme secures the exchange of routing table entries by using hash tree chains that are similar to hash-chains. This method provides a basis to secure schemes such as secure distance vector and path vector routing protocols. As the methods are based on hash-chains they are very efficient and especially suited to networks of low-power devices.

Papadimitrados and Haas propose a secure route discovery protocol that provides the nodes with accurate connectivity information in a hostile environment [61]. Another approach that does not avoid malicious behavior but is able to detect it was presented by Paul and Westhoff. They propose a mechanism to detect routing misbehavior in DSR based ad-hoc networks and to spread the information of malicious nodes in the network [62].

Clearly, all approaches that only rely on symmetric cryptography require a key pre-distribution scheme. Zhou and Haas [96] as well as Zapata [94] proposed the use of asymmetric cryptographic authentication to secure ad-hoc network routing protocols. However, as we mentioned above such an approach is only suitable if the devices are able to frequently perform demanding asymmetric cryptographic operations.

**Stimulation of Cooperation:** Closely related to routing is the stimulation of nodes in order to induce cooperation, e.g., to participate in the routing process. There are two intuitive approaches here: (1) to detect malicious and selfish nodes in order to punish them, e.g., by spreading this information to the network; and (2) to reward participating nodes. It seems clear that such approaches can only be provided if there is a mechanism available for a trustworthy message exchange. Furthermore, nodes need to build trust associations with their environment to evaluate messages. For instance, if information about a malicious node is spread in the network, all receivers have to decide whether this information is trustworthy since this information might originate from another malicious node. As this is out of scope of this thesis, we point the interested reader to the CONFIDANT project by Buchegger et al. [7].

Ben Salem et al. proposed a charging and rewarding scheme for packet forwarding in multi-hop cellular networks [72]. They present a set of protocols to stimulate cooperation. The solution is efficient since it relies on symmetric cryptography only. Hubaux and Buttyan proposed a scheme for stimulating cooperation in ad-hoc networks [10]. They use virtual currency called nuglets. These nuglets are stored in a security module that is embedded into all devices. When a device sends a packet, the nuglet counter is decreased, whereas it is increased when a device forwards a packet. Nodes are stimulated in the sense that they can only send packets if their counter is larger than zero. Clearly, the security module should be tamper resistant to prevent manipulation of the nuglet counter. Authenticated messages might be used to reduce the requirements of the security module. Then the security module authenticates each message and adjusts the nuglet counter such that the security modules of other devices are able to check whether the message is authentic. This prevents that devices send packets and declare them as forwarded packets such that they get a reward instead of being charged.

Lamparter et al. propose the secure charging protocol (SCP) in so called ad-hoc stub networks [47]. Here, the ad-hoc network cloud is connected to the Internet via a base station. The base station is able to authenticate the nodes either directly or by an authentication server in the Internet. SCP is a motivation based protocol for supporting charging and billing in multi-hop ad-hoc networks. There is a service provider (SP) that provides a reliable connection in the ad-hoc network and that might also provide an Internet service by an access point. Each node that forwards foreign data packets gets a reward whereas the sender and also the receiver are charged by the SP. Note that devices of the ad-hoc network that do not use the base station can still communicate without using the services of the SP. However, since the devices might be selfish, the SP introduces cooperation and thus adds quality of service to the network. Note that the traffic is not routed via the base station but directly from the sender to the receiver. Each sender generates a digital signature for a packet or a bundle of packets, and each intermediate node and the receiver verifying the signature. Furthermore, there are hash-chain operations involved that are negligible compared to public-key operations.

Zhong et al. proposed the SPRITE scheme to reward forwarding nodes and charge sending nodes [95]. They rely on a central authority to collect receipts from the forwarding nodes. These schemes are based on the strategy that a node only routes packets if its packets are forwarded. Note that all these approaches are based on the underlying cryptographic protocols using some mixture of symmetric and asymmetric methods. Furthermore, most of the schemes assume that the hardware cannot be tampered with.

## 3.5 Existing Technology

We now present available technology of wireless networks. We start by giving a brief overview of standards for ad-hoc networks, and then present sensor networks that are ready for deployment. Security can be provided on the application layer as well as the link layer. Solutions on the link layer are transparent to the user and more efficient than on the application layer as they induce less data overhead and can be implemented partly in hardware. On the other hand, security on the application layer is well known from the desktop PC area. For instance, there is PGP available for Palm OS, and modern PDAs run under Linux such that they are able to establish SSL based communication channels. However, as there might be no central trust authority available, e.g., a PGP certificate directory, there might arise security issues here.

**Standards for Wireless Networks:** Bluetooth enables mobile devices to connect to each other or to stationary devices. For instance, Bluetooth is used to connect a head set to a mobile phone. It provides security on the link layer or the application layer. The security on the link layer provides pairwise confidentiality and authentication. A PIN (essentially a symmetric key) is entered into two devices by a keyboard, i.e., a secure external channel is used for the key agreement. This PIN is used to obtain a symmetric key for authentication and encryption. Bluetooth uses sophisticated mechanisms to derive a symmetric key. However, once the entered PIN as well as all communication between the devices is known to an adversary, he is able to compromise the system. This procedure can work for a small number of devices such as for home-entertainment and telecommunication devices. However, the security scheme does not scale well. Once there is a large number of devices, a complete key distribution requires that each of $n$ devices has to store $n-1$ keys. As entering and remembering a large PIN number is inconvenient there is the danger that short PIN numbers are used. If a device does not have a keyboard build in, often fixed PIN numbers are hard-coded which obviously weakens the security level. Furthermore, there are some attacks known to the Bluetooth security algorithm [27] which are, however, infeasible for today's technology.

The IEEE standard 802.11b/g, also often called wireless LAN (WLAN), was developed to enhance local area networks (LAN) by a wireless link. Hence, 802.11 was never thought of as an ad-hoc technology. However, it is flexible enough to realize ad-hoc networks. As Bluetooth, WLAN offers security based on a pairwise trust association on the link layer by a mechanism called wired equivalent privacy (WEP). Here, a 104-bit key is provided to both devices which in turn is used to encrypt the communication data. WEP has been broken and cannot be used to provide a secure communication channel [26]. Thus, for WLAN, confidentiality and data integrity must be provided on the application layer.

**MICA Motes and TinyOS:** The MICA motes are popular sensor nodes developed by the University of California at Berkeley that run under TinyOS. They consist of a low power processor, a radio transceiver, and a battery pack. The current Mica2 Motes are equipped with an 8-bit CPU running at 8 $MHz$ with 128 $KB$ flash memory and 4 $KB$ SRAM [17]. TinyOS has built-in security enabled [80] such that all communication is encrypted and protected against manipulation. Since the available bandwidth, packet size and computing power is very constrained, all motes share one symmetric key for message authentication and encryption. This key is made known to all devices before deployment. Hence, if any device is compromised, the entire network is broken.

**SPINS:** SPINS [64] is a suite of security protocols for sensor networks. It consists of two protocols that use only symmetric primitives, SNEP and $\mu$TESLA. SNEP provides data confidentiality, two-party data authentication, and evidence of data freshness using traditional symmetric key methods. The other protocol, $\mu$TESLA, is a lean version of TESLA for highly constrained environments in order to provide authenticated broadcasts. SPINS assumes a network architecture that consists of a base station and sensors. The base station, which is more powerful than the sensors, shares a secret key with each sensor. Hence sensors can only communicate via the base station to other sensors in a multi-hop manner such that the base station acts as a central server. If a sensor wants to broadcast a message, it needs to do this via the base station. Deploying several base stations introduces some redundancy to diminish the risk of a single point of failure.

## 3.6 Authentication Models

After presenting several approaches for authentication in ad-hoc and sensor networks, it is clear that there is no single model that provides security for all flavors of such networks. Depending on the situation the system designers should choose an appropriate authentication scheme. The main questions to ask here are as follows:

1. Is it possible to use a key pre-distribution scheme, and does it suffice the application constraints?
2. Is it possible to use a distributed PKI?
3. Is there any supporting infrastructure available, e.g., a secure channel or a base station?
4. Is there only a single authority, or are there several authorities involved?

In the following, we list application scenarios and analyze possible approaches for authentication. Of course, the following discussion is based on generalizations, and an actual application in any of the scenarios might have different requirements.

**Military Applications**: Often there will be only one authority that controls all devices, and solutions do not need to be cost efficient such that supporting infrastructure can be provided. Hence, a pre-distribution of keys in the initialization phase can be performed, or a central server might be deployed to realize a PKI. The devices can be powerful enough to perform symmetric and asymmetric cryptographic operations. Since there is only one authority, the major number of devices will collaborate. However, there might be single compromised nodes in the network to which an adversary has full access including all secret key data. Especially as the gain of a successful attack might be high, a small set of compromised nodes must not endanger the security of the entire network. Clearly, this scenario is an authority based approach.

**Home-Entertainment**: Again, there is only one authority, the user of the devices. As the size of the network is usually small, supporting infrastructure is available as an external secure side channel provided by the user. Hence, symmetric cryptography can be used. For instance, a symmetric key can be exchanged by entering a PIN code into the devices as this is done by Bluetooth. More elegant is a key exchange by physical contact at initialization time as proposed by the resurrecting duckling model.

**Meeting**: At a meeting, e.g., at a hotel, a group of people want to establish an ad-hoc network in order to exchange files without depending on any external infrastructure. There is no single authority available here, and furthermore it is not possible to automatically distribute keys at initialization time of the network such that a self-organized approach is appropriate. Hence, a password based key agreement could be used that is based on an external secure side channel, e.g., by interaction of the users. This works fine for a small group. For very large groups more sophisticated approaches are needed, probably including supporting fixed infrastructure.

**Sensor Network**: A simple solution is to deploy a base station that shares a secret key with each sensor device as implemented by SPINS. Whenever new devices are introduced to the network, a new key between node and base station is generated. The base station is then updated with the new key. Sensors are able to agree on a shared key via the base station. However, the base station is a single point of failure and attack. Furthermore, the nodes might be mobile, or the base station might not be available permanently. Clearly, the area of sensor networks induces several different authentication models. Basically, any of the other described authentication models also works in the sensor network setting.

**Ad-hoc Stub Network**: An ad-hoc stub network consists of an ad-hoc network cloud that includes a base station that is connected to the Internet. Devices of the ad-hoc network cloud are not necessarily able to directly reach the base station. However, they are able to reach the base station in a multi-hop fashion. In this scenario, it is possible to apply well known schemes based on Internet services. However, in most cases the devices must be able to perform public-key operations. Then each node holds a unique certificate that binds the node's identity to its public key. The certificate can be issued by the base station or by any CA in the Internet. The base station can verify a device's certificate by relying on certificate chains of other CAs. Also each node of the ad-hoc network cloud can verify a certificate via the base station and its Internet access. Due to the base station also efficient symmetric key schemes are possible. Then the base station hands over a set of secret keys to a node at initialization time. All presented authentication schemes can efficiently be applied to this scenario.

**Single Manufacturer Ad-hoc Network**: The ad-hoc network has no supporting infrastructure but all devices are produced by a single manufacturer or a consortium. Thus, there is an off-line central authority available here. The manufacturer equips all devices with certificates binding a unique identity to a public key. Each node then can identify to another node by proving knowledge of the corresponding secret key. We will present an efficient authentication scheme for this scenario that is able to provide frequent authentication processes in Chapter 5. A key pre-distribution approach is also possible for a single manufacturer ad-hoc network at the cost of high bandwidth resources such as presented in Algorithm 3.12. However, since there is no central online authority available, key revocation has to be considered especially in this scenario [68].

**Pure Ad-hoc Network**: This is the most general case for ad-hoc networks. There are various authorities running several devices, and the devices are heterogenous. The network is fully self-organized such that there is no infrastructure and no central authority, no central trusted third party, no central server, and no secret share dealer even in the initialization phase. Each device owner is its own authority domain. Furthermore, in the most general case there are no secure side channels available to perform a key agreement such as provided by physical contact, but devices are not powerful enough to agree on keys by means of asymmetric schemes. Today it seems that there is no general approach for establishing authentication in pure ad-hoc networks. We will go into more detail in Chapter 5 by introducing a new security classification and an appropriate protocol that we call ZCK recognition protocol that is able to provide a basic level of authentication in such an environment. This protocol is also an appropriate protocol for sensor networks.

To guarantee secure transactions at a very high security level the appropriate choice is to use a public-key system involving the hassle of obtaining certificates, using devices that are able to perform such operations, and finding a reliable connection to the CA in order to check for revoked certificates. Such an approach does not only provide secure authentication but also legal non-repudiation.

Using no security at all is an acceptable approach for a variety of applications. A good illustration is a huge network of cell phones where free calls can be made. Instead of a phone service provider that provides infrastructure in a cellular based fashion, the cell phones are connected in a multi-hop manner without any central infrastructure. Once the network has enough nodes and enough redundant connections, the no-security approach might be the best one since every user benefits from routing other users' data packets so that selfish behavior vanishes. However, using no security at all should be a deliberate decision by the application designers but not a lack of knowledge or effort.

After presenting these scenarios, we conclude that there is the need for more efficient authentication protocols. It seems that only the hybrid scheme as presented in Tables 3.1 and 3.2 is flexible enough to support the single manufacturer as well as the pure ad-hoc network in a somewhat efficient manner. However, in many situations it is not efficient enough. We present new authentication protocols in Chapter 5 for these scenarios. But first, we analyze how well digital signatures are suited to ad-hoc networks beside all drawbacks in the next chapter.

# 4 Signature Schemes in Ad-hoc Networks

In the following, we analyze signature schemes for ad-hoc networks. We focus on standardized signature schemes that are widely deployed and accepted by standard bodies, especially RSA [69] and ECDSA (elliptic curve digital signature algorithm) [52]. Intuitively, one would argue that ECC is more suited to security applications in constrained environments as the operands' length is far shorter. The larger operands' length does not only affect execution time but also bandwidth and memory requirements. However, RSA is known for its extremely fast signature verification whereas ECDSA provides signature generation efficiently. A main goal of this chapter is an analysis of the trade-off between RSA and ECDSA for application in ad-hoc networks, and a recommendation which scenarios are suited to the different signature schemes. Such signature schemes can be based on a distributed PKI or a distributed certificate repository. Note that part of this chapter was presented in [46].

We analyze the digital signature schemes in the context of a security protocol on the transport layer. Such a security protocol might be a secure routing or a secure stimulation of cooperation protocol. We consider such secure protocols on an abstract level with regard to authentication. Hence we argue that each secure protocol between two entities Alice and Bob requires Alice to authenticate to Bob and possibly also to each intermediate node on the path from Alice to Bob. Using digital signatures Alice needs to generate a signature whereas Bob and each intermediate node needs to verify Alice's signature. In the following we consider an abstract security protocol. For instance, such might be a secure routing protocol. However, we do not consider further security issues of routing protocols here that are well researched in literature [36, 38, 37, 61, 62, 96].

We argued that asymmetric cryptography should not be used frequently in ad-hoc networks. Clearly, there are applications for which no pure symmetric cryptographic solution is known yet. Furthermore, many ad-hoc network devices used today, e.g. PDAs, are already in the range of a slow PC, i.e., their

CPU runs at a few 100 $MHz$ so that occasional public-key operations are possible yet no frequent ones. Still, it should be a trend to reduce asymmetric cryptography to a minimum in ad-hoc networks as more and more small and less powerful devices are introduced to ad-hoc networks.

## 4.1 Security Protocols and Digital Signatures

In order to deploy secure protocols that are robust against malicious and selfish users, a method for authenticating data packets is necessary. Without going into detail, security protocols in general usually have the following properties:

1. protocols in a multi-hop network environment require the explicit involvement of more than two entities along the path in a broadcast fashion,

2. the security level of an authentication scheme for messages are rather relaxed concerning their durability, and

3. each entity may be in different roles at different moments.

The first property gives us an argument for using digital signatures in such an environment instead of MACs. The second property states that authenticated messages for most security protocol purposes do not require a security level that is used for highly confidential or integrity-proven messages. The receiver only wants to verify the origin of the message. For our purposes we can limit the life-span of a key to a few days since we do not require a long durability of the authentication scheme. Hence, the key size can be chosen in a way that it resists an attack for a few days only. Note that we only consider the authentication of data packets here and not of its content. If a user wishes to authenticate his message he has to do this on the application layer in a way that there is a long durability guaranteed. The third property states that a node sometimes is in the role of sending, forwarding, or receiving a data packet. Thus a node needs to generate signatures and verify them. Before we design an appropriate model for this scenario, we first consider digital signature schemes in more detail and evaluate their performance.

There are various digital signature schemes known in literature. There are schemes that move the computational load to a powerful server. For ad-hoc networks this approach is not appropriate, though. The NTRU signature scheme NTRUSign might be an appropriate fit. Its security is not clear yet after it was broken in the past and revised [29, 78]. Even, Goldreich and Micali [24] introduced the notion of on-line/off-line signatures. Their scheme is based on integer factorization and has an ordinary and a one-time signature scheme as ingredients. In the off-line phase the ordinary scheme is used to pre-compute the signature of the one-time verification keys which are used to sign the messages during the on-line phase. This results in a signature scheme which is very efficient for both signing and verifying but with a signature size that might be too demanding for ad-hoc networks. Finally, we take a look at signature schemes based on zero-knowledge proofs as presented in Section 3.1. These are computationally efficient but use large operand sizes such as RSA. However, the size of the public-key as well as the signature size is by far larger than in the case of RSA. In [11], Canetti et al. propose an asymmetric MAC scheme which is based on multiple symmetric keys and MACs for $n$ receiving parties. The scheme is efficient but the signature size is very large as described in Section 3.1.

There were many signature schemes proposed in literature that are more efficient than ECDSA and RSA. However, all of them make trade-offs regarding memory and bandwidth requirements, depend on an available powerful server, or are based on further assumptions. Thus for many applications traditional signature schemes are still appropriate. We consider RSA and ECDSA here as they are well established and standardized. The digital signature standard DSS [81] is based on El Gamal such as ECDSA. However, as DSS operates on large operands such as RSA and performs slower than ECDSA, we do not consider DSS. In order to reduce the performance gap of digital signatures compared to MACs we suggest to secure more data, i.e., several packets, with only one signature. In [30] and [93] hash values from several packets are integrated into one that is protected by a digital signature. Hence, to protect $l$ packets only one digital signature has to be generated by the sender, and verified by the receiver and all intermediate nodes. To make this approach robust against burst packet losses, redundant hash information is spread over a sequence of packets. Even in case of packet loss over the wireless channel, the verifying node thus has enough information to start the verify operation. Hash values that require 10-20 bytes are attached to the packets, and not more than two hash values are attached to each packet, such that there is only little data overhead in each packet. Unfortunately, this approach does not take the jitter into account. For real-time traffic it is mandatory to guarantee an almost continuous flow from the source to the destination. Since the schemes have to buffer a bundle of packets either at the source or at the destination, this affects the jitter from the source to the destination. Another approach by Rohatgi [70] prevents jitter and is thus applicable to real-time conversations. It uses the idea of one-time signatures that are only applicable once but perform extremely efficient. This is extended to $k$-time signatures that allow $k$ signatures and still perform efficiently. Adapted to an ad-hoc network, the idea is to sign a $k$-time public key with the ordinary certified key. The receiver and all intermediate nodes can then check if this public-key was signed by a certified key. Now the sender uses his $k$-time secret key to sign the next $k$ packets. Finally, another $k$-time public-key is signed to authenticate the next $k$ packets, and so on. As mentioned above, this scheme does not require buffering of packets such that packets can be signed and verified in real-time. However, the data overhead is high. For each packet, at least 270 bytes are required for the signature of a 1024-bit RSA security level which is more than twice as much as an RSA signature and more than six times as much as an ECDSA signature. It becomes clear that the first approach is suited to reduce data and computational overhead at the cost of jitter, whereas the second approach reduces the computational overhead only.

We now consider RSA and ECDSA in detail. We use RSA with short exponent here, e.g., $e = 3$ or $e = 2^{16} + 1$, to speed up the RSA signature verification. Consider RSA with an $s$-bit modulus and ECDSA with an $r$-bit modulus. Then an RSA signature has a size of $s$ bits whereas ECDSA requires $2r$ bits. As the operand size of ECDSA is far shorter than RSA, an ECDSA signature will be smaller. To verify a signature, a receiver has to obtain the certificate of the sender. A certificate basically consists of the sender's public key, a time and life-span value as well as a unique identifier. This information is signed by the CA. As the sender's public key is the main content we omit the other values for our analysis here. An RSA public key consists of a modulus $m$ and the public exponent $e$. As we said before, we assume short exponents $e$ such that a certificate basically consists of the modulus $m$, which is $s$ bits in size, and the signature of the CA. Hence, an RSA certificate has a size of $2s$

bits. An ECDSA certificate requires a public key of size $r+1$ bits by using point compression as well as a signature of size $2r$ bits such that an ECDSA certificate has a size $3r+1$ bits.

## 4.2 Performance

Before looking at performance numbers, we first consider key sizes. There are several recommendations available for key sizes, e.g., the NIST Key Management Guideline [58]. They incorporate future more powerful and cheaper hardware, and future progress in cryptanalysis. Table 4.1 shows a selection of recommended future RSA and ECDSA key sizes. For instance, data that is to be protected until 2015 requires 1024-bit RSA or 160-bit ECC. Furthermore, the standard for efficient cryptography group (SECG) [76] proposes 512-bit RSA and 112-bit ECDSA for applications with low security requirements. Figure 4.1 compares ECDSA and RSA key sizes that are considered to be equivalent in security.

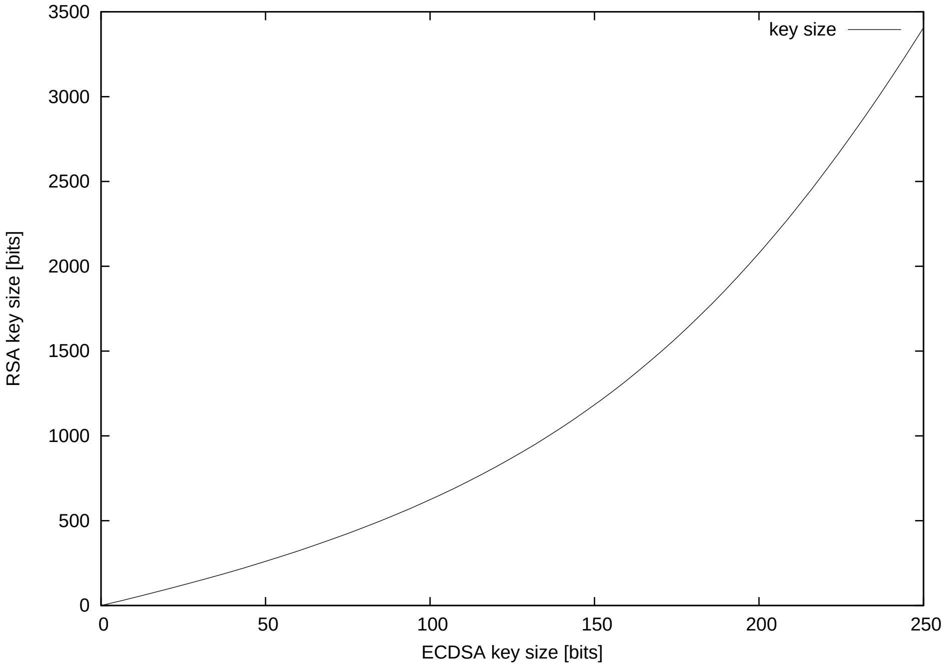
Table 4.1: Key size recommendation for high security level [58, 76]

| Year | RSA key size | ECDSA key size |
|---|---|---|
| - | 512 | 113 |
| present – 2015 | 1024 | 160 |
| 2016 – 2035 | 2048 | 224 |
| 2036 and beyond | 3072 | 256 |

Today 1024-bit RSA is a standardized security level and is considered to be equivalent in its security to 160-bit ECC. There are several comparisons of the running time of the RSA signature scheme and ECDSA available. Depending on the platform and the implementation, ECDSA signature generation is $5-8$ times faster than RSA signature generation, whereas RSA verification is $6-30$ times faster than ECDSA signature generation [71, 92, 32]. However, ECDSA uses shorter operands and has a smaller signature size that saves transmission bandwidth and leads to smaller silicon implementations.

As reference timings for RSA and ECC we use an implementation provided by Riedel in [67]. Here, a high end PDA is used that has a StrongARM CPU at 206 $MHz$. The OpenSSL RSA implementation is used as it is considered to be

Figure 4.1: Comparison of RSA and ECDSA key sizes



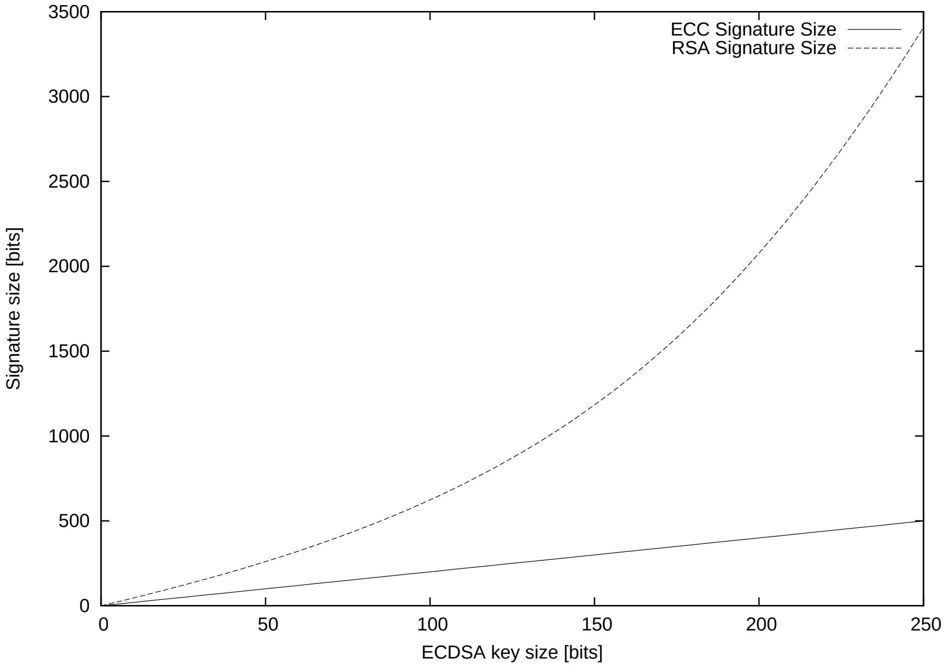
fast, whereas ECC was optimized for the reference platform. Table 4.2 presents the running times. Here, general standardized NIST and SECG [76, 57] elliptic curves are considered according to comparable security levels. Clearly, RSA signature verification performs faster than ECDSA verification, and ECDSA signature generation runs faster than RSA signature generation. However, a 113-bit ECDSA generation is around 5 times faster than RSA whereas for 233-bit the ratio increases to 72. On the other hand, ECDSA signature verification is around 6 times slower than RSA for 113-bit curves whereas the ratio shrinks to a factor of only 2 for a 233-bit curve.

Table 4.2: Execution times for signature operations based on ECDSA and RSA

| Security level [bit] | | Time for signature generation [ms] | | | Time for signature verification [ms] | | |
|---|---|---|---|---|---|---|---|
| ECC | RSA | ECC | RSA | Ratio | ECC | RSA | Ratio |
| 113 | 512 | 2.8 | 13.7 | 4.9 | 7.5 | 1.3 | 5.7 |
| 131 | 704 | 3.8 | 32.4 | 8.5 | 11.5 | 2.5 | 4.6 |
| 163 | 1024 | 5.7 | 78.0 | 13.6 | 17.9 | 4.3 | 4.1 |
| 193 | 1536 | 7.6 | 251.9 | 33.0 | 26.0 | 9.7 | 2.6 |
| 233 | 2240 | 10.1 | 731.8 | 72.0 | 37.3 | 20.4 | 1.8 |

Besides the running time, we also want to consider the data overhead. Thus we need to look at the data volume of signatures and certificates. Figure 4.2 compares the signature size of ECDSA and RSA whereas Figure 4.3 compares the certificate size. The $x$-axis represent the comparable ECDSA key size, i.e., $x = 160$ represents 160-bit ECDSA and 1024-bit RSA. For instance, 163-bit ECDSA signatures have a size of 326 bits whereas the corresponding 1024-bit RSA generates signatures of 1024 bits. ECC outperforms RSA for large key sizes both in running time and signature size. However, as we are looking at signatures with short life-span, we cannot give such a clear statement anymore.

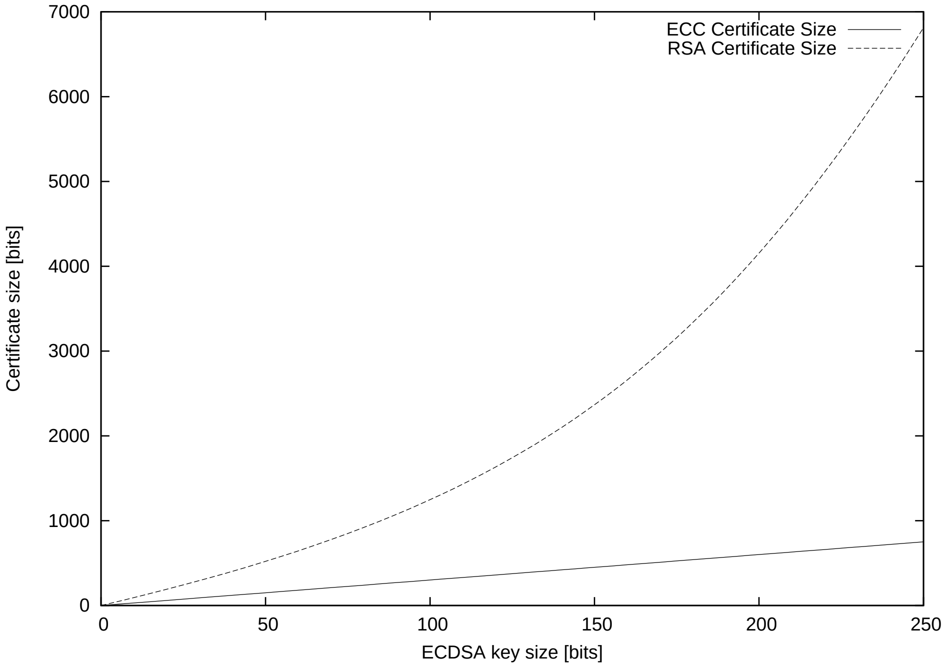
Figure 4.2: Comparison of RSA and ECDSA signature sizes



## 4.3 Digital Signatures for Security Protocols

We now analyze how well ECDSA and RSA are suited for security protocols in ad-hoc networks at the example of an abstract security protocol. Note that the results we obtain for such a model can easily be generalized to many security protocols that require a digital signature.

Figure 4.3: Comparison of RSA and ECDSA based certificate sizes



### 4.3.1 Security Model

Our model that has the main purpose of analyzing the efficiency of different signature schemes for a secure protocol that runs in the transport layer of an ad-hoc network is simple but powerful. We assume that there is a device that generates a signature, there are $n$ intermediate nodes that verify the signature, and there is a receiver that verifies the signature. A signature is attached to a single data packet or a bundle of data packets. Assume Alice sends a data packet to Bob over a multi-hop route path. We assume that there is already a route available between Alice and Bob, and we do not consider any further protocol steps here to encounter further security issues. Our simple model then looks as follows:

- each message sent by Alice requires her to perform a signature generation
- each forwarded message by an intermediate node requires them to perform a signature verification
- each received message by Bob requires him to perform a signature verification

This should fit to almost any security protocol based on digital signatures. Note that we consider and compare more complex models later.

We consider digital signature schemes on the transport layer of ad-hoc networks. The main goal in applying security is to make the protocol robust against malicious users and selfish users in order to establish a reliable protocol even in an unfriendly network setting. We assume a network as described in Section 2.6.1. The network incorporates an on-line or off-line central authority that issued certificates for each network node. The certificates bind a node's identity to its public key. Each node is able to generate and verify digital signatures, and to verify the certificates. The certificates have a short life-span only, say one day, such that there is no key revocation scheme needed.

We assume an adversary as described in Section 2.6.3. The adversary's goal is to forge a digital signatures. We assume that the applied digital signature scheme imposes no known security leaks. We assume that the adversary uses known algorithms to forge a digital signature such that the adversary has to obtain the secret key corresponding to a node's certificate by its computational resources, i.e., the adversary has polynomial bound computational power. For that purpose the adversary has access to a powerful computer that is not necessarily part of the ad-hoc network. We do not consider any further kind of attack such as a physical attack or an attack where the adversary steals the secret key by another means. Note that the certificate is only used to authenticate control data of the network. It is not used to authenticate or encrypt data on the application layer such as confidential data. Hence, after the certificate expired the secret key has no value anymore. Since we assume a very short life-span of the certificates the adversary has only little time to compromise a node's certificate.

### 4.3.2 Required Level of Security

A typical application scenario of a security protocol is the offering of wireless multi-hop Internet access at public places, for instance a cafe, an airport terminal, or a railway station. Such places have in common that the users do not stay longer than several hours at that place. Consequently, most user certificates as well as private and public keys only need to be valid for a short time, say 24 hours. In case of a longer stay, one could simply enforce a certificate renewal. An attacker may try to get uncharged network access; however, the amount of money involved can be expected to be rather low. Thus, the financial harm of successfully breaking a pair of keys is less than a few 100 Euro. Consequently, one can expect that an attacker will not spend a large amount of money to break a key. Note that breaking a key can only be used to compromise the node that owns the key. No secret information will be revealed though; the attacker may forge only signatures of the compromised node.

So far, the largest RSA Challenge Number that has been factored in RSA Security's Factoring Challenge [71] in December 2003 is a 576-bit number. 576-bit RSA is considered to be slightly more secure than a 113-bit elliptic curve. In November 2002, Certicom announced that the ECC p-109 challenge has been solved using a large network of 10,000 computers within 549 days [14]. By taking into account Moore's law, an elliptic curve "looses" 2 bits of security each 18 months. That is if a curve of at most $r$-bits can be broken today, a curve of $r+2$ bits can be broken in 18 months. We halve this time to take into consideration cryptanalytic improvements[1]. Then we obtain recommended key sizes for ECDSA as presented in Table 4.3. The key sizes of RSA that are considered to be roughly equivalent in their security to ECDSA were obtained by the NIST and SECG guidelines [58, 76]. We believe that today a curve in the range of 130-bit provides a sufficient security level to protect data for a short time interval. Note that these values are almost equivalent with the NIST recommended key sizes of Table 4.1. In the following, we choose 131-bit ECDSA as well as 704-bit RSA as appropriate scheme.

Table 4.3: Recommended key sizes for protocols on the transport layer

| Year | RSA key size | ECDSA key size |
|---|---|---|
| 1999 | 512 | 113 |
| 2006 | 704 | 129 |
| 2015 | 1024 | 153 |
| 2026 | 1536 | 183 |
| 2039 | 2240 | 218 |

[1] Note that we only take into consideration improvements like algorithmic advance but no ground breaking new methods.

### 4.3.3 Performance

When sending a data packet or a bundle of packets, the sender generates a signature over the packet content whereas each intermediate forwarding node as well as the receiver verify the signature. We now estimate the average running time for RSA and ECDSA. Let $t_g$ and $t_v$ be the running time of a signature generation and verification, respectively, and $n$ be the average number of involved intermediate nodes of a single communication process. Then the running time $t$ of a device $X$ on average is given by

$$t(X) = P(X = S)t_g + \sum_{i=1}^{n} P(X = N_i)t_v + P(X = D)t_v \qquad (4.1)$$

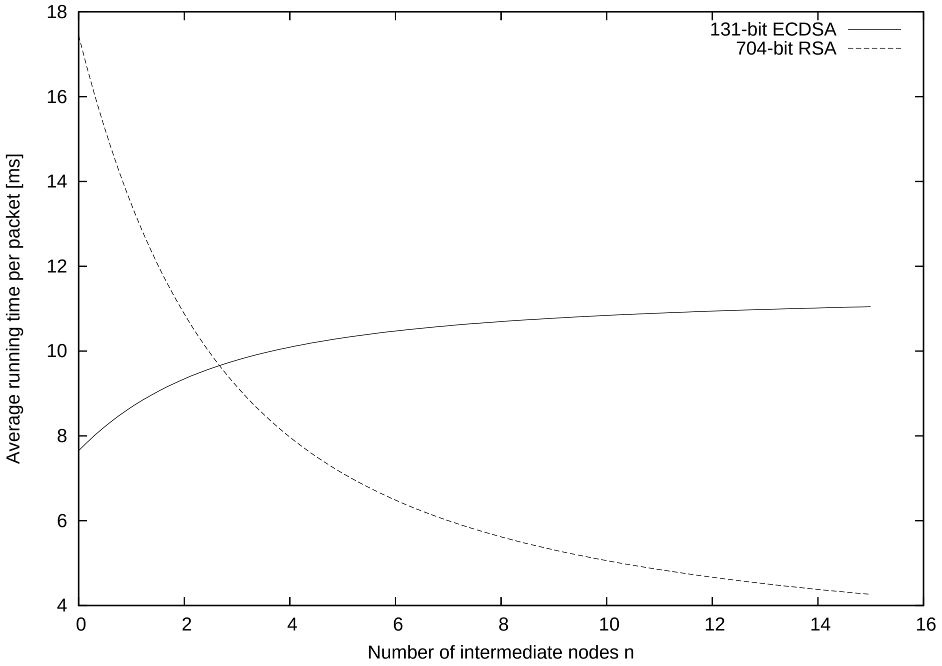
On average, the probability $P(.)$ at which a device $X$ is either in role of $S$, $N_i$ $(i = 1, \ldots, n)$, or in role of $D$ is uniformly distributed such that we obtain

$$t(X) = (t_g + (n + 1)t_v)/(n + 2). \qquad (4.2)$$

The average running time of a device $X$ is presented in Figure 4.4 assuming 704-bit RSA and 131-bit ECDSA with the running times of Table 4.2. When there are more than $n = 2$ intermediate nodes, it is more efficient to use RSA instead of ECDSA regarding computational running time.

We now take a look into the future. We assume that a security level is used that suffices Table 4.3, i.e., today in 2004 a 512-bit RSA is used, in 2010 a 704-bit RSA is used, and in 2020 a 1024-bit RSA is used. Furthermore, we assume that the available hardware becomes more powerful as stated by Moore's law. The execution times of Table 4.2 were measured in 2003 whereas each 1.5 years later the running times half. Hence 15 years later the operations are performed 1024 times faster than stated here. Figure 4.5 presents the trade-off for the number of intermediate nodes $n$ where security protocols using either RSA or ECDSA perform equally fast when looking into the future. The x-axis describes the considered year whereas the y-axis presents the number of intermediate nodes $n$ for which RSA and ECDSA run equally fast. Below the curve, ECDSA performs faster whereas above the curve RSA performs faster. For instance, in 2010 if on average there are more than $n = 2$ forwarding intermediate nodes involved on the path from the sender to the receiver then RSA performs faster

Figure 4.4: Average running time per packet



whereas ECDSA performs faster if less than $n = 2$ intermediate nodes are involved. For $n = 2$, ECDSA and RSA run equally fast. One can see that in the near future the trade-off point is in the range of very low $n$, i.e., in the range of $n = 2$. Until the year 2015 the number of intermediate nodes is in the range of $n = 4$. Later on, the number $n$ rapidly increases to more than 10.

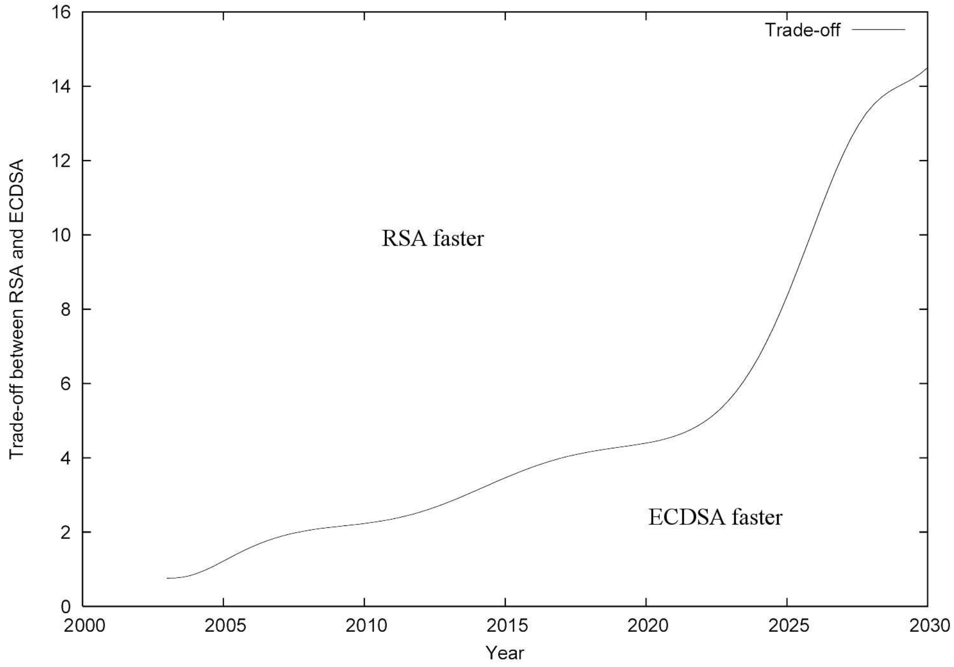
## 4.4 Recommendations

We now generalize our results in order to establish recommendations for the usage of digital signatures in multi-hop ad-hoc networks. As said before, our results are applicable to all kinds of security protocols.

### 4.4.1 Performance

Our earlier discussion showed that there is no clear winner in the RSA vs. ECDSA decision regarding the average running time. As in a multi-hop network it can be expected that on average there are far more signature verification operations performed than signature generations, thus RSA benefits by its efficient signature verification as presented in Table 4.5. However, as the

Figure 4.5: Trade-off between RSA and ECDSA with respect to the number of intermediate nodes $n$



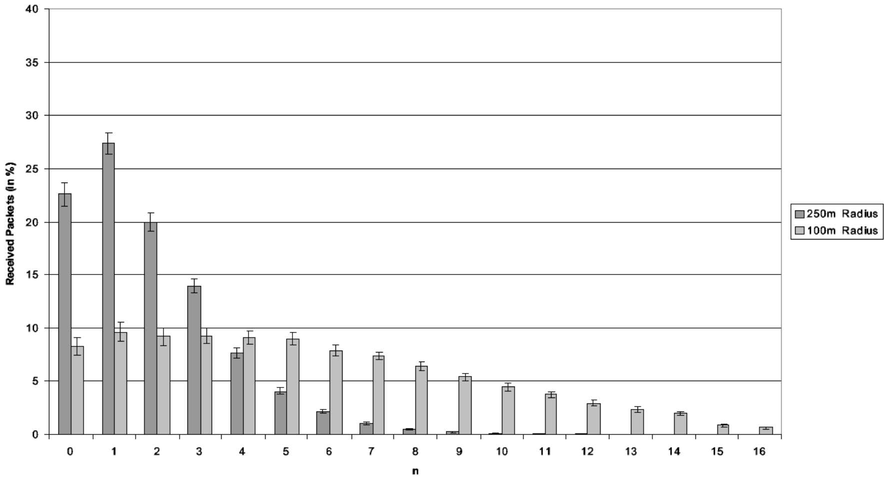
number of average hops $n$ in an ad-hoc network is expected to be fairly low in the range of $n = 5$, the average execution times are close. When considering Figure 4.5 RSA has advantages today whereas ECDSA will outrun RSA in the far future. Obviously, ECDSA has clear advantages regarding the signature size and certificate size. We expect that the available bandwidth and memory size also increases in the future, although not as fast as computational power does, such that the signature size gets less important. Finally we can say that the decision whether to use RSA or ECDSA depends on the application and the network characteristics. If there is a large number of intermediate nodes $n$ than RSA is a good match. Otherwise, due to its advantages in signature and certificate size, ECDSA is the better choice. It is important to note here that ECDSA is not always the better choice for constrained devices.

### 4.4.2 Topology of the Network

In a conventional architecture which is characterized by a single wireless hop between a restricted mobile client and a powerful server the digital signature generation at the client side to authenticate towards the server is the only time-critical and CPU wasting operation. In contrast, in the context of multi-hop ad-hoc networks the operations signature generation and signature verification become coequal operations. Both operations need to be executed by devices that tend to have lower power. For an appropriate choice of a digital signature related to its minimum total execution time $t(X)$ with $X \in \{S, D, N_i\}$, we propose considering the occurring traffic within the ad-hoc network. More precisely, the distribution of the whole network traffic with respect to the required number of intermediate nodes to reach the final destinations needs to be considered.

To estimate the traffic distribution for two exemplary network topologies, we used the ns-2 simulator version 2.1b9 [79]. Nodes move according to the random waypoint model. Initially, 100 nodes are uniformly distributed in a $1000m \cdot 1000m$ area. Subsequently, nodes choose new locations and move with a uniform velocity of 1–2 m/s until they reach their destination. The process is repeated unless the simulation time has elapsed. In our scenario, eight concurrent connections send five packets with 512 bytes per second where the average duration of a connection is 30 sec. The routing protocol is AODV. At the end of the connection a new source destination pair is immediately chosen. To obtain realistic data, each scenario was simulated 120 times with different source and destination pairs. To show that the calculated values are reasonable approximations of the mean values, the 95% confidence interval is depicted in Figure 4.6.

Figure 4.6: Traffic distribution per route length, e.g. with 100m vs. 250m transmission radius



As can be derived from Figure 4.6, in case of a transmission radius of $250m$

, about 90% of all successfully received traffic passes less than five nodes. For such a network topology ECC-based signatures are preferable. In contrast, for a transmission radius of 100m only 41.5% of the arising network traffic is forwarded over less than five intermediate nodes. For such a topology, RSA turns out to be the faster digital signature choice. Summing it up: For the appropriate choice of a digital signature scheme related to its total execution times, we need to understand the traffic distribution per route length within the ad hoc network.

### 4.4.3 Security Relationships

As we already stated there are several security relationships possible that imply the necessary type of authentication. The most basic one is a unilateral and a mutual relationship. Here, a mutual or pairwise authentication scheme is appropriate that is provided by MACs. However, in some scenarios a single entity has relationships with several other entities such that a broadcast authentication scheme is necessary. As of today, only digital signatures are able to provide a broadcast authentication. Here, an entity signs a message that can be verified by each receiver. We suggest to use digital signatures in a careful manner for applications where a broadcast authentication is required. A better approach that is especially suited to sensor networks where digital signatures might be too demanding, is to design security protocols that work with pairwise relationships only. For instance, a device might establish pairwise relationships to its neighborhood, and then base higher level protocols on these. However, in ad-hoc networks of mobile devices with a steadily changing topology, the neighborhood might vary so quickly that establishing security associations to all neighborhood devices might be too demanding as well. Let us call these two approaches global and local neighborhood approach.

We now want to compare these two approaches. The first approach is based on broadcast authentication provided by digital signatures. We assume that on average the sender sends a packet or a bundle of packets secured by a digital signature via $n$ intermediate nodes. Each node keeps a repository of foreign certificates. A sender attaches his certificate to a data packet. However, a sender does not attach its certificate to all packets but on average only to $p$ out of 100 messages if the receiver or the intermediate nodes do not have the certificate in their repository list. As the storage space is limited, devices will delete certificates from their repository once in a while. We use ECDSA with a security level of 160-bits and assume that a certificate has a size of 61 Bytes whereas a signature has a size of 40 Bytes. In the neighborhood approach, each node only communicates with its one-hop neighborhood. A key-agreement based on the elliptic curve Diffie-Hellman (ECDH) is performed with the neighborhood in order to use a MAC later on to secure messages. We assume that a MAC requires 20 Bytes, and that a new key-agreement needs to be performed every $q$ out of 100 messages due to the changing neighborhood cloud. We summarize the scenario as follows:

1. *global scenario:*
    - each message sent requires a signature generation
    - each forwarded and received message requires a signature verification
    - the signature size is 40 Bytes
    - $p\%$ of sent messages require an attached certificate
    - An ECDSA signature generation requires one elliptic curve point multiplication (PM) whereas a signature verification requires two PMs
2. *local neighborhood scenario:*
    - each MAC requires 20 Bytes
    - Generation and verification of a MAC is negligible compared to a public-key operation
    - $q\%$ of the sent, received and forwarded messages require a key-agreement
    - An ECDH key agreement requires a certificate exchange and certificate verification (two PMs) as well as another point multiplication for each involved node.

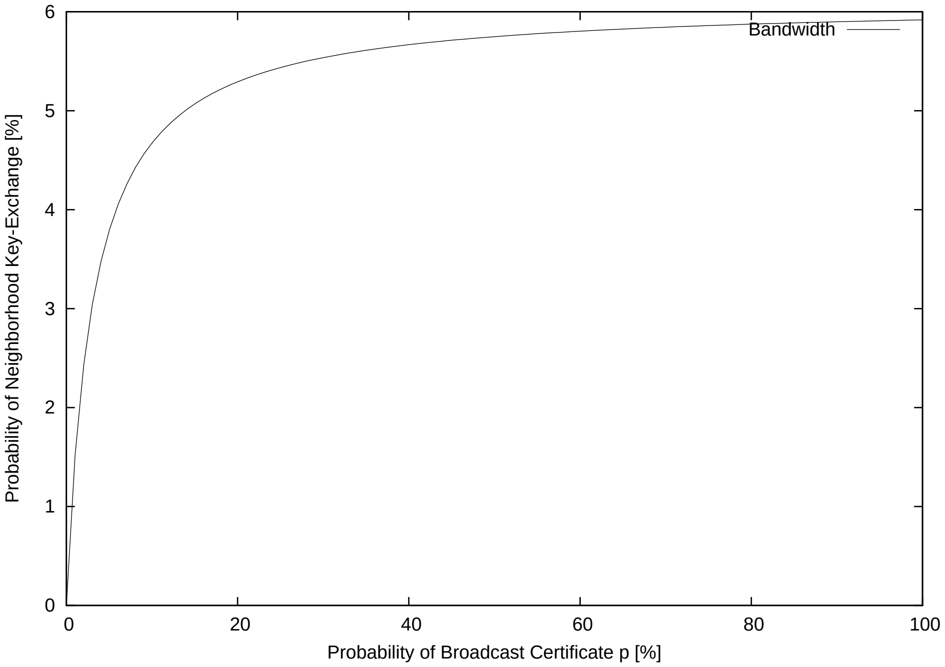
We now investigate the required resources with respect to computational power and bandwidth for these two scenarios. Let $w_B$ and $w_N$ be the required average bandwidth for the broadcast and neighborhood scenario, respectively, and $c_B$ and $c_N$ be the average required computational resources. By applying above assumptions, we obtain the following values:

$$
\begin{aligned}
w_B &= 61/p + 40 \; [Bytes] \\
c_B &= (3 + 2n)/(n + 2) \; [PMs] \\
w_N &= 122/q + 20 \; [Bytes] \\
c_N &= 3/q \; [PMs]
\end{aligned}
$$

We are interested when $w_N < w_B$ and $c_N < c_B$. Obviously, it is $c_N < c_B$ for $q > 2$ [%] such that the local neighborhood approach almost always wins regarding computational complexity. Figure 4.7 depicts the trade-off between the local neighborhood scenario and the global scenario where the required bandwidth is equal. The $x$-axis represents the ratio $p$ of data packets with attached certificate, and the $y$-axis represents the ratio $q$ of messages that require the execution of a key-agreement. The curve presents the trade-off where $w_N = w_B$. For instance, when $p = 20$ [%] and $q = 5$ [%] then both the global and local scenario require the same bandwidth. Above the curve, it is $w_N < w_B$ such that the neighborhood scenario performs more efficient than the global one. For $q > 6$ [%] the bandwidth required by the local neighborhood approach is always smaller.

We presented a simple comparison between the global and the neighborhood scenario. We induce that security protocols should be based on pairwise authentication instead of broadcast authentication. Such schemes are already applied for sensor networks, e.g., by the distributed virtual shared information space (dvSIS) [8]. This approach is almost always better regarding the computational efficiency as well as bandwidth requirements. The bandwidth requirements also affect the energy consumption due to the radio transmission that often is even more power demanding than computations. Hence, in the following chapter we will present an extremely efficient mechanism for establishing pairwise associations that can be used for a local based approach.

Figure 4.7: Trade-off between broadcast and neighborhood approach



# 5 Efficient Authentication in Ad-hoc and Sensor Networks

We are now prepared for proposing two new efficient pairwise authentication schemes for ad-hoc and sensor networks: (1) a protocol that we call Zero Common-Knowledge (ZCK) recognition protocol [1] to provide a basic form of authentication, namely recognition, and (2) a proof of identity at the cost of some external infrastructure that we call *Identity Certified* (IC) authentication protocol. The ZCK protocol is an extremely efficient and provably secure authentication protocol for sensor networks as it uses only hash-chains, whereas IC is an efficient identification protocol closing the gap to asymmetric solutions. We analyze the ZCK and IC protocol with respect to their overhead including operational complexity and energy consumption.

Our new protocols suffice the following environment properties:

- no key pre-distribution
- dynamic network topology
- no trusted third party
- no tamper resistance

The ZCK protocol works in any network environment. As defined in Chapter 2, we call such a network without any supporting infrastructure a pure network. Nodes are possibly tiny and cheap such that demanding arithmetic operations may overstrain their capabilities. For the IC protocol we weaken above limitations to introduce a single manufacturer. Furthermore, we require a loose global time synchronization of devices that are able to perform a public-key operation once in a while. Hence, our protocols provide efficient

[1] Note that ZCK does not relate to zero-knowledge proofs in any kind.

solutions for the Pure Ad-hoc network and the Single Manufacturer scenario as introduced in Chapter 3.

We now explain the main idea of our protocols. Imagine two foreigners that meet in the real world. Usually, people identify each other by showing their passports. Assuming there is nobody else available to ask about the opposite's reputation or passport validity, the best both can do is to establish step-by-step a trust relationship built on their personal experiences. To do so, it is mandatory that these two people recognize each other the next time they meet. In our model we do not assume geographical proximity, however. We call this form of authentication entity recognition.

The above approach aims at sensor and ad-hoc networks without any kind of infrastructure. However, it lacks the possibility of establishing a trust relationship based on the identity (which probably is not possible to provide at all without any infrastructure or previous knowledge). The IC authentication protocol provides identity authentication. We assume that there are some storage resources available and that nodes are loosely time synchronized. We use the term identification and identity certified authentication in the sense that an entity $A$ is able to prove its identity to an entity $B$ assuming that both entities trust some third party, usually the common manufacturer.

We introduced these protocols in [90, 89]. They did, however, contain a security flaw discovered by Lucks et al. [49]. In the same paper, we introduced an improved version that is provably secure and which is the basis of the following protocols. We presented further parts of this chapter in [91] and [33].

## 5.1 Zero Common-Knowledge Recognition

We start by defining a new security classification regarding entity recognition and message recognition and then present our new protocol. For instance, our solution is suited to cooperation and motivation based schemes [47, 10, 95] as well as secure routing methods. These schemes require a large number of authentication steps though. We believe that our ZCK recognition protocol suffices the requirements of these protocols such that they can run efficiently. Our scheme is also applicable for client-server and peer-to-peer relations.

### 5.1.1 Recognition

In a low-power environments, entity authentication might be too costly. Furthermore, in many scenarios it is unnecessary and perhaps even impossible to achieve. We define the following term to refer to a new classification of authentication protocols:

**Definition 5.1.1** Entity recognition *is the process whereby one party after having initially met, can be assured in future recognitions that it is communicating with the same second party involved in a protocol, and that the second party has actually participated (i.e. is active at, or immediately prior to, the time the evidence is acquired).*

Note that entity recognition is a weaker form of entity authentication. Hence, a protocol which performs entity authentication has, implicitly, performed entity recognition. The reverse of this is not true. In addition to entity recognition, we define a process which builds on this.

**Definition 5.1.2** Identity accumulation *is the process whereby one party can obtain a level of trust (through the acquisition of corroborative evidence) such that the party is reasonably certain of the identity of the second party involved in a protocol.*

To perform identity accumulation, entity recognition has to be assured, i.e., a party has to be certain that it is communicating with the same person it thought it was. Identity accumulation used with entity recognition can bring two communicating parties close to — but not fully attain — entity authentication. During the first recognition $A$ cannot be sure that it is communicating with the correct entity $B$, however, the more information that $A$ receives that it expected $B$ to have, the more certain $A$ can be that it is correct. However, identity accumulation is very situational dependent and requires further methods to evaluate trust which we do not focus on here. Similar to entity recognition, Definition 5.1.3 defines message recognition from message authentication.

**Definition 5.1.3** Message recognition *provides data integrity with respect to the original message source and assures the data origin is the same in future recognitions, but does not guarantee uniqueness and timeliness.*

As for entity and message authentication, the difference between message recognition and entity recognition is the lack of a timeliness guarantee in the former with respect to when a message was created. Finally, we define non-repudiation in the context of recognition.

**Definition 5.1.4** Recognition non-repudiation *is the service which prevents an entity from denying a commitment or action chain.*

In our case, this means that an entity $A$ is able to prove to a third party that a number of actions or commitments were taken by the same (probably unknown) entity $B$. The following presents some informal results of entity authentication and recognition.

**Conjecture 5.1.5** *Entity authentication over a network with the no key pre-distribution requires a trusted third-party.*

**Remark 1** *Suppose that there does not exist a trusted third-party. Given that there is no key pre-distribution, an entity $A$ receiving a message from an entity $B$ will have no prior knowledge of the identity of $B$. That is, the source $B$ is anonymous to receiver $A$. Then $B$ will have to send some identification information[2] to $A$. To verify this identification information, $A$, who has no prior knowledge, must ask a third-party $C$ for confirmation. Now $C$ is acting as a trusted third-party, thus giving a contradiction.*

Conjecture 5.1.5 proposes that entity authentication is impossible in environments where there is neither key pre-distribution nor a trusted third-party provided. However, entity recognition combined with identity accumulation can approach the security of entity authentication. Note that entity recognition prevents a man-in-the-middle attack after the first recognition, hence if $C$ was modifying information, $C$ would have to have been in the middle from the beginning (and hence, $A$ has been recognizing $C$, not $B$). A *reliable relay channel* for the first recognition process is a requirement in a formal model. Such a reliable relay channel provides integrity of messages and ensures that all messages are received in order to prevent manipulation or interception of

[2] For example, the identification information could be just the network address in the message source or it could be a secret to be shared.

messages. In many applications, the initial contact and with it the requirement of a reliable relay channel might become meaningless, though. Here, $A$ is never able to distinguish whether the other party is $B$ or $C$. $A$ will accumulate trust based on the interaction with its communication partner such that the initial contact, i.e., the question whether $A$ is talking to $B$ or $C$, becomes irrelevant. For instance, if $A$ is communicating with $C$, and $C$ is able to provide $A$ with all services needed, it is not important for $A$ whether $C$ was a man-in-the-middle at the initial contact that was supposed to happen between $A$ and $B$.

### 5.1.2 Network And Adversarial Model

Our approach of providing recognition makes sense in a network without any central authority, both on-line and off-line. We assume a network, devices, and adversarial model as described in Sections 2.6.1, 2.6.2, and 2.6.3. Furthermore, we assume the following.

The devices communicate in a multi-hop fashion over wireless links. The network links are not guaranteed to be reliable such that data packets might be dropped and manipulated. If a device $A$ wants to establish an authenticated channel to a device $B$ usually there are several nodes in between the path from $A$ to $B$. Hence, a key exchange over the insecure channel is only possible by an asymmetric key agreement scheme but not by the direct handover of a key (e.g., by physical contact). There are no channels available between devices besides the digital communication channel. In particular, there are no side-channels available for a secure key exchange.

The devices are computationally very slow. In most cases, the communication will be performed between two computationally weak devices. However, there might also be communication between computationally weak and computationally powerful devices. In particular, the devices are in general not able to perform any public-key operations. There are no pre-distributed keys available in the network, and also there is no central authority available. As we argued before, it seems reasonable to assume that there is no entity authentication possible in such a network. However, it is possible to provide recognition in such a network, e.g., by using a hybrid approach as presented in Algorithm 3.8 where there are no certificates used but the public key of each entity is exchanged over the insecure channel [3]. However, all known techniques to provide such a solution today require the use of asymmetric cryptography that we disallowed here. Our goal is to provide a protocol that allows for this scenario without the usage of public-key methods. Since we restricted the scenario to low-power devices we do not only meet the properties of an ad-hoc network but also of a sensor network.

We use a standard adversary model. We assume that the adversary has full control over the communication channel between an entity pair Alice and Bob. In particular, he is able to read all messages, modify, delay, and send them twice, and to inject new messages. As we argued before, it seems impossible to provide identification in our scenario. In order to provide recognition, we have to make one exception. For the initial phase we assume that Alice and Bob can exchange a message using a faithfully relay channel. Without some faithfully relayed initial messages, the entire notion of recognition protocols would not make sense. Thus, we assume an initial phase (typically with one message from Alice to Bob, and a second message from Bob to Alice), where the adversary can read messages but relays them faithfully. Note that as we argued before, this assumption does not limit our scenario. At the initial message exchange there cannot be any man-in-the-middle attack since both Alice and Bob do not even know about the identity of the other entity.

The main goal of the adversary is to forge an authentication. We assume that the adversary aims for an existential forgery in a chosen message scenario where Alice authenticates to Bob. The adversary can choose any message that Alice authenticates. The adversary is successful if Bob accepts a message that was not authenticated before by Alice but was created by the adversary.

### 5.1.3 General Recognition Protocols

We now present a simple example to clarify recognition based on a digital signature scheme. Here, $B$ sends his public key $PK$ to $A$. Then $B$ signs his messages which $A$ can verify. Algorithm 5.1 presents entity recognition by introducing a challenge $r$.

**Remarks:**

[3] This is also sometimes called an uncertified authentication

**Algorithm 5.1** General entity recognition protocol

```
B generates $SK/PK$ at random
B sends $PK$ to $A$
Repeat Steps 3 to 5 for each recognition process
$A$ sends random $r$ to $B$
$B$ computes $S := SIG(r, SK)$ and sends $S$ to $A$
$A$ checks if $VER(S, r, PK) \overset{?}{=} valid$
    If 'yes', $A$ accepts, otherwise she rejects
```

- Steps 1–2 need to be performed once for each communication pair $A$ and $B$.
- Steps 3–5 have to be performed for each recognition process.

We consider as main objective of this scheme the capability to ensure that entities are able to re-recognize another entity in order to receive a service they requested. Hence the public key $PK$ always has to be sent together with the offered service, i.e., service and key have to be bound together to avoid a malicious entity injecting his public key into a service that he did not offer at all. Instead of a confidential and secure channel for the initial message (key-exchange), we only require a reliable relay channel for the initial entity recognition (key exchange).

In contrast to PKI scenarios where there is a logical central certificate directory, $A$ has to store $B$'s public key (together with $B$'s $ID$ string) to be able to recognize $B$. After $A$ deletes $B$'s public key from her memory, $A$ is not able to build a connection anymore to a previous relationship with $B$. Note that in many applications a mutual authentication process is required. The above protocol can easily be extended for this case. Obviously, Algorithm 5.1 provides entity recognition as well as recognition non-repudiation. However, devices must be computationally powerful to perform the digital signature operations. We now introduce our new scheme that overcomes these limitations.

### 5.1.4 Zero Common-Knowledge Protocol

Our new protocol for recognition only requires one-way hash functions but no expensive public-key operations. Doing so the scheme is extremely efficient and orders of magnitudes faster than any public-key scheme.

Figure 5.1: Zero Common-Knowledge message recognition protocol

| **Transmitting:** | **Processing:** |
|---|---|
| For key exchange, do Steps 1-2 only once: | |
| 1. $A \rightarrow B : a_n$ | $B : \mathcal{P}(a_n)$ |
| 2. $B \rightarrow A : b_n$ | $A : \mathcal{P}(b_n)$ |
| For each recognition process, repeat Steps 3-7: | |
| 3. | $A$ knows $b_i$ |
| | $B$ knows $a_j$ |
| 4. $A \rightarrow B : m, M := MAC(m, a_{j-1})$ | |
| 5. $B \rightarrow A : b_{i-1}$ | $A : h(b_{i-1}) \overset{?}{=} b_i$ |
| 6. $A \rightarrow B : a_{j-1}$ | $B : h(a_{j-1}) \overset{?}{=} a_j, MAC(m, a_{j-1}) \overset{?}{=} M$ |
| 7. | $A : \mathcal{P}(b_{i-1})$ |
| | $B : \mathcal{P}(a_{j-1})$ |

Consider the case where Alice wants to send authenticated messages to Bob. We define a hash-chain, which is also known as Lamport's hash-chain [53], as $x_{i+1} = h(x_i)$ with $x_0$ being the anchor and $h$ being an unkeyed one-way hash function. Alice chooses randomly an anchor $a_0$ and computes the final element $a_n$ of the hash-chain. We call $a_n$ the public key and $a_0$ the secret key of Alice. Bob is doing likewise to obtain $b_0$ and $b_n$. We use a chain value as key to generate an authenticated message by a MAC. The core idea of the protocol is as follows: first exchange a value $a_i$ which the receiver will tie together with some experience. Then prove knowledge of the pre-image of $a_i$, i.e. $a_j$ with $j < i$, in order to authenticate by establishing a relationship to $a_i$ and the past experience. In order to repeat the authentication step arbitrarily many times, a hash-chain based on a one-way hash function is used. The protocol works as illustrated in Figure 5.1. Here, $\mathcal{P}(.)$ represents the storage of some data. If any comparison $\overset{?}{=}$ fails, the protocol flow is interrupted. The protocol can then only be resumed at the same position.

**Remarks:**

- Steps 1–2 ensure the exchange of public-keys which is done only once per pair $A$ and $B$. Note that if the network settings does not provide a reliable relay channel at initialization time the keys' integrity could be ensured by using a digital signature. For instance, then Alice would send $SIG(a_n)$ to Bob.
- Steps 3–7 are done for each recognition.
- For each recognition of each communication pair we assume that $A$ stores

$B$'s key $b_i$ and that $B$ stores $A$'s key $a_j$ (Step 3). Hence, after the initial key exchange $A$ stores $b_n$ and $B$ stores $a_n$. After each successful recognition, $A$ and $B$ replace $a_j$ and $b_i$ by $a_{j-1}$ and $b_{i-1}$, respectively. Furthermore, $A$ stores the private key $a_0$ that she created especially for Bob, and Bob is doing likewise.

- The exchanged message is guaranteed to be fresh since both parties are involved actively.
- The message sent in Step 4 can be read but is not authenticated at this moment. Messages can only be checked after the keys were opened in Step 6, i.e., there is a message buffer required.
- If $A$ opens a key which $B$ does not receive due to network faults, $A$ can send the key again without endangering the security of the scheme. The same holds for keys opened by $B$.
- A man-in-the-middle attack at initialization time is possible. However, once the public keys $a_n$ and $b_n$ were exchanged, messages cannot be forged.
- The scheme is not resistant to denial-of-service attacks. A malicious entity can try to overflow the message buffer. Thus, an implementation must take care of an appropriately sized message buffer.
- The scheme is provably secure if its building blocks $h$ and $MAC$ are secure [49]. In particular, our scheme is secure in the model as outlined in Section 5.1.2.
- When a key $a_0$ is compromised, only the security of the communication channel between Alice and Bob is affected.
- To save memory and to avoid extensive computations, short hash-chains can be used. These can be renewed by transmitting a new final element as an authenticated message in order to keep the trust association.
- The public key is only sent once. A message recognition requires three messages, each of them a $t$-bit string. Using an efficient hash-chain algorithm as presented in [16], an element of the hash-chain can be computed by $1/2\ \log_2(n)$ hash iterations with storage of $\log_2(n)$ hash-chain elements. In the following, we let $n = 100$ which should suffice most applications. Then there is storage required for 7 elements each $t$-bits in size, and 4 hash iterations are needed to obtain an element of the chain.

Note that the execution of a hash function has very low running time compared to any asymmetric operation.

- For each communication pair of $A$ with any $B$, she needs to store $B$'s public key $b_i$ of $t$ bits. Thus altogether, for each communication partner there is storage needed of 7 hash-chain elements as well as the public key resulting in $t$ Bytes.

For a better understanding, we illustrate our protocol again in Figure 5.2. Here, the initial key exchange as well as two recognition processes are depicted. First, Alice and Bob exchange keys $b_n$ and $a_n$. In the first recognition, Alice authenticates message $m$ to Bob by key $a_{n-1}$. After a successful message recognition, Alice and Bob store keys $b_{n-1}$ and $a_{n-1}$, respectively. At any time later on, if Alice wishes to authenticate another message $m'$ to Bob she computes the MAC over $m'$ by key $a_{n-2}$, and follows the ZCK recognition protocol. After the initial key exchange, any number of recognition processes can be performed between Alice and Bob.

Figure 5.2: Zero Common-Knowledge recognition protocol example

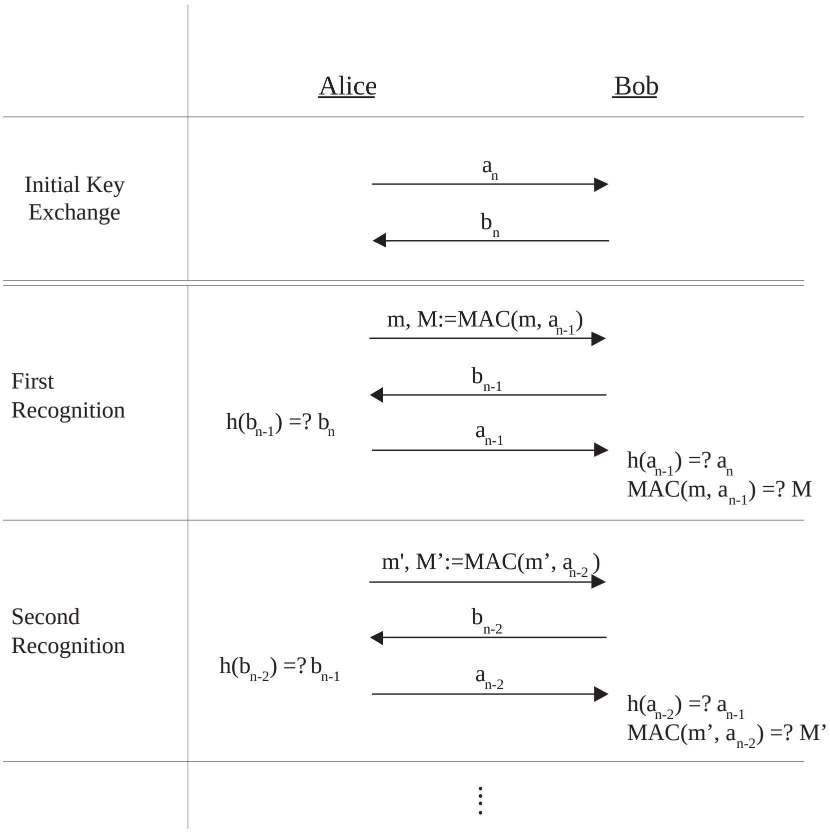


Figure 5.3: Zero Common-Knowledge entity recognition protocol

| **Transmitting:** | **Processing:** |
|---|---|
| `For key exchange, do Steps 1-2 only once:` | |
| 1. $A \to B : a_n$ | $B : \mathcal{P}(a_n)$ |
| 2. $B \to A : b_n$ | $A : \mathcal{P}(b_n)$ |
| `For each recognition process, repeat Steps 3-8:` | |
| 3. | $A$ knows $b_i$<br>$B$ knows $a_j$ |
| 4. $A$ randomly chooses $r$ | |
| 5. $A \to B : M := MAC(r, a_{j-1}), r$ | |
| 6. $B \to A : b_{i-1}$ | $A : h(b_{i-1}) \overset{?}{=} b_i$ |
| 7. $A \to B : a_{j-1}$ | $B : h(a_{j-1}) \overset{?}{=} a_j, MAC(r, a_{j-1}) \overset{?}{=} M$ |
| 8. | $A : \mathcal{P}(b_{i-1})$<br>$B : \mathcal{P}(a_{j-1})$ |

Our scheme also provides entity recognition as presented in Figure 5.3. Note that timeliness is provided by the active involvement of Alice and Bob. Contrary to usual entity authentication schemes such as presented in Algorithm 3.2 and Algorithm 3.9, we neither require a challenge nor a time-stamp to introduce timeliness. Hence we save both the initial challenge message and the global time synchronization.

Let $t$ be the output size of the MAC and the hash function. It is widely believed that computing a collision to a given message (target collision resistance) in a one-way hash function that maps strings to $t = 80$ bits is approximately as hard as factoring an RSA modulus of 1024-bits [58]. We believe that target collision resistance is sufficient for our application whereas collision resistance (finding any pair of colliding messages) requires twice the number of bits. An attack which finds any pre-image of an opened key $a_i$ has a complexity of around $2^t$. Hence we assume $t = 80$ in the following. We believe that an average hash-chain of length $n = 100$ should meet the lifespan demands of most security associations.

The above scheme provides unilateral recognition. A mutual recognition can also be provided as presented in Figure 5.4. The number of exchanged messages increases to four. Here, Alice must start each round of the authentication.

Figure 5.4: Zero Common-Knowledge mutual message recognition protocol

| **Transmitting:** | **Processing:** |
|---|---|
| `For key exchange, do Steps 1-2 only once:` | |
| 1. $A \rightarrow B : a_n$ | $B : \mathcal{P}(a_n)$ |
| 2. $B \rightarrow A : b_n$ | $A : \mathcal{P}(b_n)$ |
| `For each recognition process, repeat Steps 3-8:` | |
| 3. | $A$ knows $b_i$ |
| | $B$ knows $a_j$ |
| 4. $A \rightarrow B : m_A, M_A := MAC(m_A, a_{j-1})$ | |
| 5. $B \rightarrow A : m_B, M_B := MAC(m_B, b_{i-2}), b_{i-1}$ | $A : h(b_{i-1}) \overset{?}{=} b_i$ |
| 6. $A \rightarrow B : a_{j-1}$ | $B : h(a_{j-1}) \overset{?}{=} a_j, MAC(m_A, a_{j-1}) \overset{?}{=} M_A$ |
| 7. $B \rightarrow A : b_{i-2}$ | $A : h(b_{i-2}) \overset{?}{=} b_{i-1}, MAC(m_B, b_{i-2}) \overset{?}{=} M_B$ |
| 8. | $A : \mathcal{P}(b_{i-2})$ |
| | $B : \mathcal{P}(a_{j-1})$ |

## 5.2 Identity Certified Authentication

We now extend our previous scheme to provide identification. We assume that the devices are able to perform a signature verification which is reasonable in the case of RSA with short exponent. However, the devices do not need to perform signature verifications frequently, but only once in a while. We further assume that devices are loosely time synchronized to a global time. Our goal here is to extend the ZCK recognition protocol in such a way that it provides identification to entities that do not know each other nor had any contact in the past. This can be seen similar as the exchange of a certificate in the public-key scenario. Once the identity proof was performed the ZCK recognition protocol can be used which ensures that this level of trust in the other's identity is maintained.[4] Thus an exchange of keys that can be used for the ZCK recognition protocol for future identification or message authentication processes is included in our new scheme. For each communication pair $A$ and $B$, the certificate exchange only needs to be performed once for proving identity whereas later on only the ZCK recognition protocol needs to be executed for identification based on the identity proven level of trust.

[4] The ZCK recognition scheme improves or maintains the level of trust between two entities. Since the proof of identity is the highest trust level we can achieve here, in this case the ZCK recognition maintains this level.

### 5.2.1 Network and Adversarial Model

The goal in our setting is now not only to provide recognition but to provide identification at the cost of further infrastructure. Hence we change the underlying model of Section 5.1.2 used in the recognition setting to suffice the new security goal.

The devices communicate in a multi-hop fashion over a wireless link. The network links are not guaranteed to be reliable. A key exchange is not possible via physical contact or a side-channel. The devices have moderate computing power. In most cases, the communication will be performed between two homogeneous devices. However, there might also be communication between devices with moderate and high computing power. The devices are not able to perform frequent public-key operations. However, they are able to perform an asymmetric operation once in a while, in particular an RSA signature verification where a short exponent is used. There are no pre-distributed keys available in the network. However, there is an off-line certificate authority available that issues certificates for each device. The certificates bind a device's public key to its identity. We do not consider key revocation here as it would require a central directory [5]. Once a device is deployed there is no central authority available anymore. Such a scenario can easily be provided by using a standard hybrid scheme as presented in Algorithm 3.8 where there are certificates exchanged. However, in our setting we assume that frequent public-key operations are not possible. In order to still be able to design a solution that fulfills our requirements we furthermore assume more infrastructure. In particular, we assume that all devices are loosely time synchronized. In particular, devices are synchronized at deployment and do not need any further synchronization methods to stay synchronized. Since we only require that devices are synchronized in the order of a few seconds this assumption is reasonable with today's technology, even in low-cost devices.

As before, we use a standard adversarial model. However, this time we do not make any assumptions about the initial contact between Alice and Bob. We assume that the adversary has full control over the communication channel at any time. The main goal of the adversary is to forge an authentication. In our setting this induces that the main goal of the adversary is the impersonation

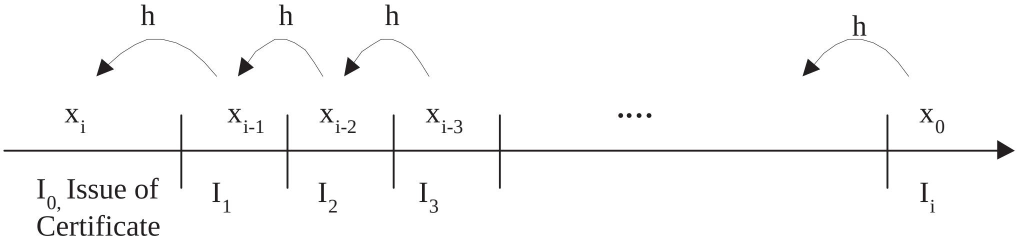
[5] A simple approach to address this issue is to use certificates of a short life-span.

of Alice. The adversary is successful if Bob believes him to be Alice.

### 5.2.2 Outline

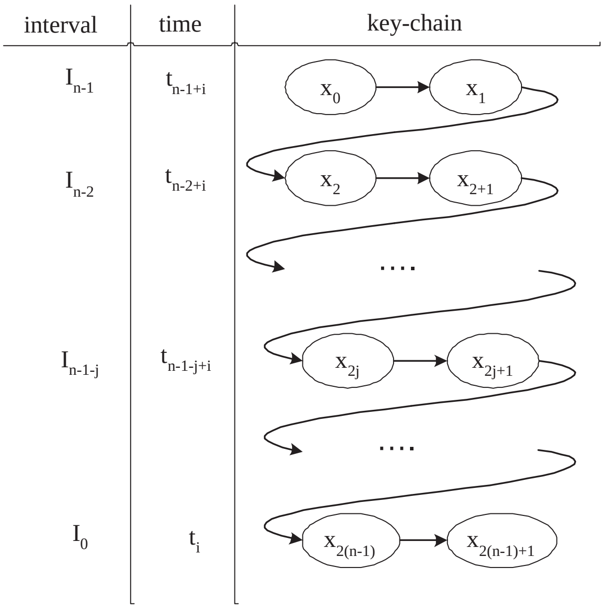
The idea of the IC protocol is to divide time into intervals and let a set of keys of a hash-chain only be valid for one time interval, similar to the idea of TESLA [64]. Figure 5.5 presents the main idea. A certificate that includes the final hash-chain element $x_i$ is issued at time interval $I_0$. The element $x_{i-1}$ is then valid for time interval $I_1$, $x_{i-2}$ is valid for interval $I_2$, and so on. If Alice wants to identify to Bob during time interval $I_3$, she sends her certificate to Bob and proves knowledge of $x_{i-3}$. If Bob now tries to use the obtained values to impersonate Alice to John, time progresses and John will require a proof of knowledge of $x_{i-4}$.

Figure 5.5: Identity Certified authentication example



Let us now go into more detail. Let $a_{i+1} = h(a_i)$ be a hash-chain, $a_0$ be the anchor of the chain and $a_{2(n-1)+1}$ be the final element of the hash-chain. We let two keys of the chain be valid in a time interval. Let $t_i$ be a point in time which we assume to be the starting time, $L$ be the length of the time intervals, and $I_{j-1}$ be the $j$-th time interval. We assume that time is divided into time intervals of length $L$, i.e., there are time values $t_i$, $t_{i+1}$, $t_{i+2}$, and so on, such that the time difference between $t_j$ and $t_{j+1}$ is $L$, and the time difference between $t_j$ and $t_{j+m}$ is $mL$. The $k$-th time interval $I_{k-1}$ lasts from $t_{i+k-1}$ to $t_{i+k}$. Let $d_s$ be the time difference measured in interval lengths $L$ between the current time $t_c$ and the time $t_s$, i.e., $d_s = c - s$. Assume the starting point is $t_i$ and the keys valid during the corresponding interval $I_0$ are $a_{2(n-1)}$ and $a_{2(n-1)+1}$ such that there are keys for $n$ time intervals in the hash-chain. This fact is depicted in Figure 5.6. Then $a_{2(n-1)}$ and $a_{2(n-1)+1}$ are keys for the first time interval $I_0$ between time $t_i$ and $t_{i+1}$, $a_{2j}$ and $a_{2j+1}$ are keys valid for the time interval $I_{n-1-j}$, and $a_0$ and $a_1$ are the keys valid in

Figure 5.6: Time intervals for hash-chains



the final time interval $I_{n-1}$. Furthermore, the interval difference between the current time $t_c$ and starting time $t_i$ is $d_i = c - i$ such that the current time interval is $I_{d_i}$, and the keys valid for this interval are $A_c = a_{2(n-1-d_i)}$ as well as $A_{c+1} = a_{2(n-1-d_i)+1}$.

Assume Alice wants to identify herself to Bob. Then Alice holds a secret anchor $a_0$ and the public value $PK_A = a_{2(n-1)+1}$. Bob likewise holds $b_0$ and $PK_B = b_{2(n-1)+1}$. The public values are signed by the manufacturer $MF$ at some time interval $t_u$ and $t_v$, respectively, such that Alice and Bob obtain certificates $\langle t_u, PK_A \rangle$ and $\langle t_v, PK_B \rangle$. Now, if Alice wants to prove her identity to Bob she sends Bob her certificate. Assume Bob determines the current time as $t_j^B$ and Alice as $t_i^A$. These two timings might be different since we do not require a strict time synchronization. Then Alice and Bob determine the time difference between the current time they measure and the certificate time so that they obtain the interval differences $d_u^A = i - u$, $d_v^A = i - v$, $d_v^B = j - v$, and $d_u^B = j - u$. Alice then opens the key $A_{c+1} = a_{2(n-1-d_u^A)+1}$. Bob now checks whether $h^{2d_u^B \pm \{0,1\}}(A_{c+1}) = PK_A$ where $h^i(x)$ is the $i$-th iteration of the hash function. We allow that Bob's determined time $t_j^B$ differs from Alice's determined time $t_i^A$ by one interval length $L$, i.e., Alice's and Bob's determined times may each differ from the actual global time by $L$.[6] Hence, Bob will

[6] Note that the required level of time synchronization only depends on the size of the time

tolerate that he got a chain element that is one position behind or before the element he expected. We express this as a correction term '$\pm\{0,1\}$' in the formula.

We now want to reduce the risk that Bob uses key values he just obtained to impersonate Alice (or vice-versa). To prevent such an attack where Bob obtains the hash-chain value for some time interval and then uses it to negotiate with a third entity, we introduce a random seed. Alice and Bob select a random seed $r_A$ and $r_B$. Both Alice and Bob have a set of pre-computed certificates in the form $\langle t_A, r, PK_A\rangle$ and $\langle t_B, r, PK_B\rangle$ for all possible $r$ which they obtain from the manufacturer. Now Bob can only impersonate Alice to a third party John if John uses for some time interval the same random seed as Alice. Since this is an interactive proof system we expect that a small set for the seeds $r$ suffices, e.g., $r$ being an 8-bit number. In that case each entity has to store $2^8$ pre-computed certificates. If we would set the time interval to be one minute, Bob had to ask on average $2^7$ entities in two minutes to find an entity using the same random seed as Alice. Further possibilities to increase the security level are given below.

## 5.3 Identity Certified Authentication Protocol

Assume Alice wants to authenticate to Bob and starts the authentication process. The mutual ZCK protocol of Figure 5.4 is used as basis here in order to pairwise exchange a certified hash-chain. The IC protocol is illustrated in Figure 5.7. The core idea here is that for each time interval there is a different key. One has to prove knowledge of the appropriate key for a given time interval. However, a third party cannot use this captured key for an impersonation attack since time moves forward and the captured key is not valid anymore. The protocol has several phases. First there is the certificate exchange (Steps 0–3). Then Alice and Bob determine the difference between current time and issuing time of the certificates (Step 4), and perform the ZCK key exchange (Steps 5–6). Now Alice and Bob prove knowledge of a previous hash-chain element by opening keys (Steps 7–8). Finally, if the knowledge proof was successful, Alice and Bob store the exchanged ZCK key for further

intervals.

identification processes performed by the ZCK protocol. For a better understanding, we explain the phases in more detail.

**Certificate exchange:** (Steps 0–3)
$A$ sends a *hello* message to $B$ which responses by sending a random seed $r_B$ to $A$. $A$ now chooses a seed $r_A$ and sends it as well as the pre-computed certificate $\langle t_u, id_A, r_B, PK^A\rangle_{MF}$ with $PK^A = x^A_{2(n_A-1)+1}$ to $B$. $B$ verifies the signature of the manufacturer, and subsequently sends the pre-computed certificate $\langle t_v, id_B, r_A, PK^B\rangle_{MF}$ with $PK^B = x^B_{2(n_B-1)+1}$ to $A$. $A$ verifies the signature of the manufacturer.

**Compute time differences:** (Step 4)
Assume $A$ determines its current time $t^A_i$ with $d^A_u = i - u$ and $d^A_v = i - v$, and $B$ determines its current time $t^B_j$ with $d^B_v = j - v$ and $d^B_u = j - u$.

**ZCK-key exchange:** (Steps 5–6)
$A$ chooses a hash-chain with final element $y^A_{n'_A}$ and sends an authenticated session key for the ZCK recognition protocol with time key $k^A_c = x^A_{2(n_A-1-d^A_u)}$ as $(y^A_{n'_A})_{k^A_c}$ to $B$. Then $B$ chooses a hash-chain with final element $y^B_{n'_B}$ and sends an authenticated session key for the ZCK recognition protocol with time key $k^B_{c+1} = x^B_{2(n_B-1-d^B_v)}$ as $(y^B_{n'_B})_{k^B_c}$ to $A$.

**Knowledge proof:** (Steps 6–8)
$B$ starts opening keys by sending $k^B_{c+1}$ as part of the last message in Step 5. $A$ then checks whether $h^{2d^A_v \pm \{0,1\}}(k^B_{c+1})$ is equal to $PK^B$. Then $A$ opens her key by sending $k^A_c$ to $B$. Now $B$ checks if $PK^A$ is equal to $h^{2d^B_u+1\pm\{0,1\}}(k^A_c)$. Finally $B$ opens another key to be checked by $A$.

**Store ZCK-key:** (Step 9)
If all checks hold, Alice and Bob store the ZCK-key. Subsequently, both parties use the certified session keys for message authentication as described in the ZCK recognition protocol. If any check fails, $A$ and $B$ stop the execution and restart it from the beginning.

We observe the following characteristics of the IC protocol:

- Once a session key of a hash-chain is exchanged and authenticated, this key is used as described in the basic ZCK protocol, i.e., it is used for all further identification and message authentication procedures for the entity pair Alice and Bob.

Figure 5.7: Identity Certified authentication protocol

| **Transmitting:** | **Processing:** |
|---|---|
| Certificate Exchange: | |
| 0. $A \rightarrow B : hello$ | |
| 1. $B \rightarrow A : r_B$ | |
| 2. $A \rightarrow B : r_A, \langle t_u, id_A, r_B, PK_A \rangle$ | $B$: verify certificate |
| 3. $B \rightarrow A : \langle t_v, id_B, r_A, PK_B \rangle$ | $A$: verify certificate |
| Compute Time Differences: | |
| 4. | $A$: determine $t_i^A$ with $d_u^A = i - u$, $d_v^A = i - v$ |
| | $B$: determine $t_j^B$ with $d_v^B = j - v$, $d_u^B = j - u$ |
| Key Exchange: | |
| 5. $A \rightarrow B : a'_n, MAC(a'_n, A_c)$ | |
| 6. $B \rightarrow A : b'_n, MAC(b'_n, B_c), B_{c+1}$ | $A$: $h^{2d_v^A \pm \{0,1\}}(B_{c+1}) \overset{?}{=} PK_B$ |
| Knowledge Proof: | |
| 7. $A \rightarrow B : A_c$ | $B$: $h^{2d_u^B + 1 \pm \{0,1\}}(A_c) \overset{?}{=} PK_A$ |
| 8. $B \rightarrow A : B_c$ | $A$: $h(B_c) \overset{?}{=} B_{c+1}$ |
| Store ZCK-Key: | |
| 9. | $A$: $\mathcal{P}(b'_n)$ |
| | $B$: $\mathcal{P}(a'_n)$ |

- For the public-key certificate an arbitrary signature scheme might be used. We assume an RSA like scheme because it performs signature verification very efficiently.

- The scheme is vulnerable to a denial-of-service attack by sending invalid certificates. To the author's knowledge there is no solution known when using asymmetric cryptography for this problem in general, though.

### 5.3.1 Security

If we set the interval length $L$ to be infinitely small and the bit size of the challenge $r$ to an appropriate security level, it is clear that this scheme is as secure as the ZCK protocol. However, the loose time synchronization as well as a small security parameter $l_r$ for $r$ introduce weaknesses. We see the following two attacks. An adversary Mallory starts an authentication process with Alice to obtain her certificate and a key of the hash-chain for the seed $r_A$. He then uses these values for Bob to obtain his certificate and one of Bob's keys for the seed $r_B$. Now he uses Bob's key and certificate against Alice who has to use the same seed $r_A$ again such that the adversary obtains the second key for the current time interval. Mallory is then able to impersonate Alice against

Bob in the current time interval. In the second attack, Mallory eavesdrops an authentication process between Alice and Bob to obtain a certificate and the keys for the current time interval. He then tries to impersonate Alice against a third entity John with these values in that same time interval. In both cases Mallory uses the fact that he knows keys that are valid in the current time interval. Hence he has at most two time intervals to impersonate Alice because of the time tolerance we allow. However, in both cases Mallory has to find an entity that uses the same random seed $r_A$ as Alice since the keys he obtained are only valid for this seed. It becomes clear that by increasing the bit-size of the seed $r$ and shortening the time-intervals the security level can be set arbitrarily high.

For a higher security level it is possible to perform several authentication processes in parallel. In such a case, all messages of the parallel processes are sent at once such so number of exchanged messages does not increase whereas the data, computation, and storage overhead increases linearly. Note that the used certificates need to be distinguishable, i.e., for two parallel processes there need to be two distinguishable sets of certificates for each $r$. For instance, when performing three processes in parallel, an adversary Mallory has two minutes to find an entity requiring the proper nonce which on average takes him $2^{23} = 8388608$ authentication processes. Then there have to be three distinguishable sets of certificates. Hence there is memory space required for $p \cdot 2^r$ certificates, where $p$ is the number of parallel instances.

### 5.3.2 Certificate Lifespan

The certificate lifetimes depend on the application and the available infrastructure. If certificates can be renewed once in a while, the lifetime can be chosen quite short. In many applications a temporary network will be available to support certificate renewal. Other applications may require that the certificate lifetime exceeds the device's lifetime. In the following, we consider a sensor network scenario because of our intention to analyze the energy consumption which is of special interest in sensor networks. We assume that the IC hash-chain has $n = 1000$ elements at a time interval length of five minutes such that the certificate life-time is more than three days. Using the methods of [16], such a hash-chain can be computed with $1/2\ \log_2(n)$ computations at the storage of $\log_2(n)$ elements, i.e., if 10 elements are stored, each element can be recovered by 5 hash function iterations. The verifier has to iterate the hash-chain to the public-key, i.e., on average he has to run $n/2$ iterations. Since an element has 10 Byte in size, there is storage needed of 100 Byte for each hash-chain. Furthermore, each 1024-bit RSA certificate requires memory of $128 + 10 = 138$ Byte. Thus, using a certificate set of $2^8$, there is memory space required of $256 \cdot (138 + 100)$ Byte $\approx 60$ $KB$. When using the IC protocol for a static sensor network, it is reasonable to limit the number of successful IC authentications to one per time interval. Thus an adversary Mallory has less than 5 minutes time left to impersonate Alice when there are $2^8$ different challenges $r$, i.e., Mallory's probability of a successful impersonation is $2^{-7}$ which for some applications might be sufficient for an interactive proof system in the considered network class. However, it is not appropriate for applications such as financial transactions of significant value. Clearly, if there is more memory space and computational power available, e.g., in a more powerful device such as a sensor aggregator or a more powerful device in an ad-hoc network, there could be used 6 parallel sessions with a certificate set of $r = 5$. Thus the complexity of an attack would increase to $2^{30}$ which is considered to be a high security level for interactive proof systems. In ad-hoc networks with more powerful devices the certificate life-time can be chosen far higher by lengthen the hash-chains. Furthermore, recent research by M. Fischlin suggests to include further information with the public key to reduce the effort for the verifier [54]. Hence, we believe that chain lengths of $n = 1,000,000$ can be chosen to support a certificate lifetime of several years in ad-hoc networks.

In the next chapter we analyze the efficiency of our protocols. Therefore we assume parameters for the IC protocol suited for a lifetime of several days using a certificate set of $r = 8$, i.e., there is a memory space requirement of 60 $KB$. On low-power devices with highly constrained memory resources we could instead use $r = 6$ resulting in 14 $KB$ of required memory space at a lower security level.

# 6 Efficiency Analysis of Authentication Protocols

We now analyze the efficiency of our protocols from the previous chapter and compare it to the hybrid authentication protocol (Algorithm 3.7) which is a standard authentication protocol. Efficiency can be evaluated regarding the energy consumption as well as computational complexity. We start by considering the energy consumption of the authentication protocols, then analyze their efficiency, and finally compare them. Parts of this chapter were presented in [91].

## 6.1 Energy Consumption

We introduce a simple energy model that suffices for our purpose. We hereby consider the energy consumption that results of radio transmission and processor activity due to the cryptographic overhead. A more sophisticated energy model was introduced by Carman et al. [13]. Actual comparisons of the energy consumption of standard cryptographic algorithms can be found in [66, 35].

### 6.1.1 Device Architecture

For the subsequent estimation of the arising energy cost we consider a highly constrained homogeneous network consisting of devices equipped with an 8-bit CPU, and small memory and bandwidth capacity. For our analysis, we use the Mica Motes of Crossbow [17] as reference platform. The Mica2 has 128 $KB$ programmable flash memory, 4 $KB$ SRAM, and runs at 8 $MHz$. Hence, it is already a relatively high powered sensor device. A less costly sensor might have a 4-bit CPU at less than 4 $MHz$, and might thus be an order of magnitude slower. The packet size is 36 Byte of raw data, and 28 Byte of data without header at 38.4 Kbaud and a transmission range of at most $150m$. The Mica Motes run under TinyOS [80].

Asymmetric cryptographic primitives, e.g. digital signatures, require quite some execution time and thus cause much energy consumption. In some cases, asymmetric methods might be too demanding at all. A 134-bit ECC operation on an 8-bit 8051 processor which is roughly comparable to our considered hardware platform and our security level takes 2998 $ms$ for a point multiplication [45]. Thus, the execution time for a signature verification is about $6s$ (two point multiplications). We can estimate that a 1024-bit RSA signature verification is more than 10 times faster than an ECC verification at a similar security level [92]. Thus we expect an RSA verification to run in $0.5s$ [48]. For the subsequent estimation of the protocols' energy consumption we chose RC5 for both, computing the MAC and the hash-chain since it is already part of TinySec, the security module of TinyOS. An additional scheme would cost additional storage space. On the Mica Motes an RC5 based hash function $\mathcal{H}$ has a running time of 2.22 $ms$ and a $CBC - MAC$ based on RC5 takes 4.18 $ms$ [19].

### 6.1.2 Energy Model

There are two operation modes of sensor nodes which largely reduce the battery charge, namely processor activities and the transmission of data. The number of CPU operations for the IC and the ZCK protocol is fixed and so is the resulting battery consumption for processing. However, the energy cost for data transmissions is variable for both protocols. They increase linearly with the packet size, and proportionally to the square of the transmitting distance. In particular, the latter means that with a varying node density in the network, the energy cost for transmission will significantly vary.

Our subsequent estimation of the total energy consumption for the ZCK protocol and the IC protocol is based on the following assumptions. All values are derived of the Mica2 Motes specification [17]. At a speed of 38.4 Kbaud, the transmission of one packet of 36 Byte takes about 10 $ms$. The Motes are powered by 2 AA batteries which supply around 2200 $mAh$ [51]. For computation, we assume that the energy consumption requires a current consumption of 8000 $nA$ such that computation of one millisecond costs 0.0022 $nAh$. Receiving a data packet also has a current of 8000 $nA$, i.e., receiving of a packet which takes $10ms$ costs 0.022 $nAh$. Transmitting at maximum power requires a current of 25 $mA$, i.e., transmitting a packet which takes $10ms$ costs 0.07 $nAh$.

The Mica Motes specification gives a maximum distance of 150 $m$. As this value is for outdoors in a perfect environment, we assume a distance of $50m$ here at maximum transmission power. We now give an overview of our assumptions:

**Processing Energy $E_P$**

- the energy consumption $E_P$ for processing a basic operation $\zeta$ is linear to its execution time $t_\zeta$:
- $E_P(t_\zeta) := t_\zeta \cdot E_P(1)$.

**Transmitting Energy $E_T$**

- Let $p(\eta)$ be the number of packets required to send a message $\eta$. The energy consumption $E_T$ for transmitting a message $\eta$ for a given distance $d$ is linear to the number of packets $p(\eta)$. Furthermore, the energy consumption $E_T$ for transmitting a given message $\eta$ is quadratic to its transmission distance $d$:
- $E_T(d, \eta) := d^2 \cdot p(\eta) \cdot E_T(1, 1)$.

**Receiving Energy $E_R$**

- the energy consumption $E_R$ for receiving a message $\eta$ is linear to the number of packets $p(\eta)$:
- $E_R(p(\eta)) = p(\eta) \cdot E_R(1)$.

Note that for the device architecture introduced before it holds that

- $E_P(r\ ms) = 0.0022 \cdot r\ nAh$,
- $E_T(d\ m, p(\eta)\ packets) = 0.07\ (d/50)^2 \cdot p(\eta)\ nAh$, and
- $E_R(p(\eta)) = p(\eta) \cdot 0.022\ nAh$

We are aware that the Mica Motes do not allow dynamic adjustment of the transmission power. However, as future motes might allow this we include this possibility in our model. A node's energy consumption in idle, sleep or receive mode is negligible compared to transmission and processing energy. In the reminder, we use the following values:

Table 6.1: Energy map of the ZCK protocol per protocol step and per involved entity

| | Alice | $N_i(\forall i = 1, \cdots, n)$ | Bob |
|---|---|---|---|
| 1. | $100E_P(\mathcal{H}) + E_T(d, \mathcal{H})$ | $E_R(\mathcal{H}) + E_T(d, \mathcal{H})$ | $E_R(\mathcal{H}) + E_P(\mathcal{P})$ |
| 2. | $E_R(\mathcal{H}) + E_P(\mathcal{P})$ | $E_R(\mathcal{H}) + E_T(d, \mathcal{H})$ | $100E_P(\mathcal{H}) + E_T(d, \mathcal{H})$ |
| 3. | - | - | - |
| 4.‡ | $4E_P(\mathcal{H})$ | - | - |
| 5.‡ | $E_R(\mathcal{H}) + E_P(\mathcal{H})$ | $E_R(\mathcal{H}) + E_T(d, \mathcal{H})$ | $4E_P(\mathcal{H}) + E_T(d, \mathcal{H})$ |
| 6. | $E_T(d, \mathcal{H})$ | $E_R(\mathcal{H}) + E_T(d, \mathcal{H})$ | $E_R + E_P(\mathcal{H})$ |
| 7. | $E_P(\mathcal{P})$ | - | $E_P(\mathcal{P})$ |

‡ := message exchange

| **operations** $\zeta$ | $t_\zeta$ | $E_P(\zeta)$ |
|---|---|---|
| $VER_{RSA}$ | $0.5s$ | $1.1\ nAh$ |
| $\mathcal{H}$ | $2.22ms$ | $0.0049\ nAh$ |
| $\mathcal{P}$ | negligible | - |

| **messages** $\eta$ | size (Bytes) | $p(\eta)$ | $E_T(d\ m, \eta)$ | $E_R(\eta)$ |
|---|---|---|---|---|
| $\mathcal{H}$ | 10 | 1 | $0.07 \cdot (d/50)^2\ nAh$ | $0.022\ nAh$ |
| $cert_{RSA}$ | 138 | 5 | $0.35 \cdot (d/50)^2\ nAh$ | $0.11\ nAh$ |
| $r$ | 1 | 1 | $0.07 \cdot (d/50)^2\ nAh$ | $0.022\ nAh$ |

Clearly, our model assumes an ideal environment. One may argue that our model simplifies unrealistically. Nevertheless, we believe that it is well suited as a basic metric to decide under which circumstances protocols like the ZCK and the IC protocol are applicable to sensor networks.

### 6.1.3 Energy Map of the ZCK Protocol

We now list the energy consumption of the ZCK protocol of Figure 5.1 per each protocol step and per each involved sensor node. We are using the notations of the previous section. Steps 1 and 2 are done only once and include the computation of the public keys $a_n$ and $b_n$, respectively. Steps 4 and 5 include the authentication of the message for which the right key of the hash-chain needs to be computed. For hash-chains of length $n = 100$, there are 4 iterations of the hash function necessary.

In Table 6.1 we present a node's total energy consumption when running the ZCK protocol. We only consider the energy overhead here and not the energy consumption for normal messages or the MAC as this needs to be transmitted in every scheme to obtain a basic security level. Obviously, the energy consumption depends on a node's role within the authentication process. If a node is in the role of Alice, Bob or an intermediate node, its energy consumption is $E_A$, $E_B$, or $E_N$. For the key exchange, the following energy cost is necessary only initially (Steps 1-2):

$$\begin{aligned} E'_A = E'_B &= E_R(\mathcal{H}) + E_T(d,\mathcal{H}) + E_P(\mathcal{P}) + 100E_P(\mathcal{H}) && (6.1)\\ &= 0.506 + 0.07(d/50)^2\ nAh \\ E'_N &= 2E_R(\mathcal{H}) + 2E_T(d,\mathcal{H}) && (6.2)\\ &= 0.044 + 0.14(d/50)^2\ nAh \end{aligned}$$

For each recognition process, there is the following energy necessary (Steps 3-7):

$$\begin{aligned} E_A = E_B &= E_R(\mathcal{H}) + E_T(d,\mathcal{H}) + E_P(\mathcal{P}) + 5E_P(\mathcal{H}) && (6.3)\\ &= 0.0462 + 0.07(d/50)^2\ nAh \\ E_N &= 2E_R(\mathcal{H}) + 2E_T(d,\mathcal{H}) && (6.4)\\ &= 0.044 + 0.14(d/50)^2\ nAh \end{aligned}$$

Let $x$ be the number of message authentication processes that are performed between a pair Alice and Bob. Thus the total energy can be computed as $\hat{E} = E' + x \cdot E$. Hence we obtain:

$$\begin{aligned} \hat{E}_A = \hat{E}_B &= (x+1)E_R(\mathcal{H}) + (x+1)E_T(d,\mathcal{H}) + (x+1)E_P(\mathcal{P}) && (6.5)\\ &\quad +(100+5x)E_P(\mathcal{H}) \\ &= x\ 0.0462 + 0.506 + (x+1)\ 0.07(d/50)^2\ nAh \\ \hat{E}_N &= 2(x+1)E_R(\mathcal{H}) + 2(x+1)E_T(d,\mathcal{H}) && (6.6)\\ &= (x+1)\ 0.044 + (x+1)\ 0.14(d/50)^2\ nAh \end{aligned}$$

Table 6.2: Energy map of the IC protocol per protocol step and per involved entity

| | Alice | $N_i(\forall i = 1, \cdots, n)$ | Bob |
|---|---|---|---|
| 1. | $E_R(r)$ | $E_T(d,r) + E_R(r)$ | $E_T(d,r)$ |
| 2. | $E_T(d, r + cert_{RSA})$ | $E_T(d, r + cert_{RSA})+$<br>$E_R(r + cert_{RSA})$ | $E_R(r + cert_{RSA})+$<br>$E_P(VER_{RSA})$ |
| 3. | $E_R(cert_{RSA})+$<br>$E_P(VER_{RSA})$ | $E_T(d, cert_{RSA})+$<br>$E_R(cert_{RSA})$ | $E_T(d, cert_{RSA})$ |
| 4. | - | - | - |
| 5. | $E_T(d, 2\mathcal{H}) + 110E_P(\mathcal{H})$ | $E_T(d, 2\mathcal{H}) + E_R(2\mathcal{H})$ | $E_R(2\mathcal{H})$ |
| 6. | $E_R(3\mathcal{H}) + 500E_P(\mathcal{H})$ | $E_T(d, 3\mathcal{H}) + E_R(3\mathcal{H})$ | $E_T(d, 3\mathcal{H}) + 110E_P(\mathcal{H})$ |
| 7. | $E_T(\mathcal{H})$ | $E_T(d, \mathcal{H}) + E_R(\mathcal{H})$ | $E_R(d, \mathcal{H}) + 500E_P(\mathcal{H})$ |
| 8. | $E_R(\mathcal{H}) + E_P(\mathcal{H})$ | $E_T(d, \mathcal{H}) + E_R(\mathcal{H})$ | $E_T(d, \mathcal{H})$ |
| 9. | $E_P(\mathcal{P})$ | - | $E_P(\mathcal{P})$ |

### 6.1.4 Energy Map of the IC protocol

We now derive the energy consumption of the IC protocol of Figure 5.7 as presented in Table 6.2. Again, the energy that is required by Alice, Bob and intermediate nodes is divided into the different columns. The pre-computation that is done by a workstation is not part of this analysis. We assume that the hash-chains have $n = 1000$ elements. On average there are 10 hash iterations computed by Alice and 500 by Bob to verify the public key (Steps 6 and 7). Furthermore, we assume that the ZCK hash-chain, which is created and transmitted in Steps 5 and 6, has a length of $n' = 100$ elements such that altogether $500 + 100 + 10 = 610$ hash iterations are performed both by Alice and Bob in Steps 5–7. Note that the 100-element chain is used for the subsequent ZCK protocol whereas the 1000-element chain is used for the IC only. When using the ZCK protocol afterwards, the initial phase of the ZCK protocol does not need to be redone.

From this energy map we derive the following equations which represent the sensor node's energy consumption of the IC protocol for the different roles.

$$\begin{aligned} E''_A &= E_T(d, r + cert_{RSA} + 3\mathcal{H}) + E_R(r + cert_{RSA} + 4\mathcal{H}) + \\ &\quad E_P(611\mathcal{H} + VER_{RSA} + \mathcal{P}) \\ &= 0.63\ (d/50)^2 + 4.3072\ nAh \end{aligned} \tag{6.7}$$

$$E''_N = E_T(d, 7\mathcal{H} + 2r + 2cert_{RSA}) + E_R(7\mathcal{H} + 2r + 2cert_{RSA}) \tag{6.8}$$

$$
\begin{aligned}
&= 1.33\ (d/50)^2 + 0.418\ nAh \\
E''_B &= E_T(d, r + cert_{RSA} + 4\mathcal{H}) + E_R(r + cert_{RSA} + 3\mathcal{H}) + \qquad (6.9) \\
&\quad E_P(610\mathcal{H} + VER_{RSA} + \mathcal{P})) \\
&= 0.7\ (d/50)^2 + 4.2804\ nAh
\end{aligned}
$$

After the identification phase and the exchange of a ZCK key, the ZCK protocol can be used for message authentication. Again, let $x$ be the number of message authentications that are performed afterwards. Then we can compute the total energy as $\hat{E} = E'' + x \cdot E$ by combining Equations (6.3) and (6.4) with Equations (6.7) to (6.9).

$$
\begin{aligned}
\hat{E}_A &= E_T(d, r + cert_{RSA} + (3+x)\mathcal{H}) + E_R(r + cert_{RSA} + (4+x)\mathcal{H}) + \qquad (6.10) \\
&\quad E_P((111 + 5x)\mathcal{H} + VER_{RSA} + (1+x)\mathcal{P}) \\
&= (x\ 0.07 + 0.63)\ (d/50)^2 + x\ 0.0462 + 4.3072\ nAh \\
\hat{E}_N &= E_T(d, (7+2x)\mathcal{H} + 2r + 2cert_{RSA}) + E_R((7+2x)\mathcal{H} + 2r + 2cert_{RSA}) \qquad (6.11) \\
&= (x\ 0.14 + 1.33)\ (d/50)^2 + x\ 0.044 + 0.418\ nAh \\
\hat{E}_B &= E_T(d, r + cert_{RSA} + (4+x)\mathcal{H}) + E_R(r + cert_{RSA} + (3+x)\mathcal{H}) + \qquad (6.12) \\
&\quad E_P((110 + 5x)\mathcal{H} + VER_{RSA} + (1+x)\mathcal{P})) \\
&= (x\ 0.07 + 0.7)\ (d/50)^2 + x\ 0.0462 + 4.2804\ nAh
\end{aligned}
$$

### 6.1.5 Energy Map of Traditional Protocols

With respect to the following comparisons we now want to present an energy consideration of alternative traditional protocols using hybrid methods as presented in Algorithm 3.7. We presented the ZCK and the IC protocol to provide recognition and identification, respectively. Alternatively, one can use a single shared key for all devices which does not cause any overhead. However, then all the devices had to be under the control of the same authority, and the entire network depends on the security of each of the devices, i.e., if one devices gets broken or tampered with, all the devices are insecure. A pairwise pre-distribution of symmetric keys overcomes some of these problems, but it does not allow the network to be extended and thus is limited to only a few authorities. Thus we consider the following two protocols which are comparable to the ZCK and IC protocol.

(1) *Uncertified DH (UDH)*: uncertified Diffie-Hellman key exchange with subsequent symmetric authentication ↔ ZCK

(2) *Certified DH (CDH)*: certified Diffie-Hellman key exchange with subsequent symmetric authentication ↔ IC + ZCK

The first one, an uncertified DH key exchange, is used to exchange a symmetric key which in the following is used to authenticate messages or entities by a symmetric MAC scheme. The second one, a certified DH key exchange, implements a key exchange using certificates to provide identification of the participating parties. Both schemes work as presented in Algorithm 3.7. However, in UDH a public key is first computed and then exchanged whereas in CDH the public key is exchanged as part of a certificate. The UDH protocol provides recognition and is equivalent to using the ZCK protocol, and the CDH protocol provides the functionality of the IC+ZCK protocol, i.e., using IC for a ZCK key exchange and then using ZCK for message authentication. To derive the energy consumption, we extend our previous assumptions. We assume that elliptic curve cryptography (ECC) is used for the key exchange (ECDH) as it is far more efficient for this task than RSA. An ECC certificate $cert_{ECC}$ consists of an ECDSA signature of 40 Byte and a public key $PK_{ECC}$ of 21 Byte when using point compression such that 3 packets are required to transmit an ECC certificate. A point multiplication $OP_{ECC}$, which is the core operation on an elliptic curve, takes $3s$ as described before, such that a signature generation $SIG_{ECC}$ takes $3s$ and a signature verification $VER_{ECC}$ takes $6s$. Altogether we obtain:

| **operations** $\zeta$ | $t_\zeta$ | $E_P(\zeta)$ |
|---|---|---|
| $OP_{ECC}$ | $3s$ | 6.6 $nAh$ |
| $SIG_{ECC}$ | $3s$ | 6.6 $nAh$ |
| $VER_{ECC}$ | $6s$ | 13.2 $nAh$ |

| **messages** $\eta$ | size (Bytes) | $p(\eta)$ | $E_T(d\ ft, \eta)$ | $E_R(\eta)$ |
|---|---|---|---|---|
| $cert_{ECC}$ | 61 | 3 | $0.21 \cdot (d/50)^2$ $nAh$ | 0.066 $nAh$ |
| $PK_{ECC}$ | 21 | 1 | $0.07 \cdot (d/50)^2$ $nAh$ | 0.022 $nAh$ |

We now derive the energy equations for the UDH protocol. To be able to act anonymously such as it is provided by the ZCK protocol, i.e., to change identity, we cannot assume that the public key is pre-computed but needs to be computed once in a while. Here, for the ECDH each party performs an ECC operation $OP_{ECC}$ to obtain a public key, submits this value and then performs another ECC operation to obtain the shared secret. Hence we obtain the energy consumption of the protocol flow for each of the involved entities.

$$\begin{aligned} E_A = E_B &= 2\ E_P(OP_{ECC}) + E_T(PK_{ECC}) + E_R(PK_{ECC}) \\ &= 0.07\ (d/50)^2 + 13.222\ nAh \\ E_N &= 2\ E_T(d, PK_{ECC}) + 2\ E_R(PK_{ECC}) \\ &= 0.14\ (d/50)^2 + 0.044\ nAh \end{aligned}$$

If the CDH protocol is used, the certificate also needs to be checked, thus requiring another signature verification on each side. Contrary to the uncertified version, here the public key is pre-computed as it is part of the certificate. Thus, the signature step is omitted, but another verification step for the certificate is included. Therefore we can derive the energy consumption of the protocol flow for each of the involved entities as follows:

$$\begin{aligned} E_A = E_B &= E_T(cert_{ECC}) + E_R(cert_{ECC}) + E_P(OP_{ECC} + VER_{ECC}) \\ &= 0.21\ (d/50)^2 + 19.866\ nAh \\ E_N &= 2\ E_T(d, cert_{ECC}) + 2\ E_R(cert_{ECC}) \\ &= 0.42\ (d/50)^2 + 0.132\ nAh \end{aligned}$$

Since we only consider the overhead caused in addition to a basic security level by using a MAC when using UDH or CDH, there is no additional overhead for the following message authentications. Hence for the total energy it is $\hat{E} = E$ in this case.

## 6.2 Efficiency Analysis

We now consider the efficiency of our new protocols. First, we give a summary of the ZCK and IC protocol properties following Tables 3.1 and 3.2. Tables 6.3 and 6.4 clearly show that our protocols provide efficient authentication. We are now prepared to compare the ZCK and IC protocol in more detail to the hybrid schemes CDH and UDH, respectively. In the following, we first compare the operational complexity, and then we analyze how the ZCK and IC protocol compare to UDH and CDH regarding the energy consumption.

Table 6.3: ZCK and IC properties

| | requires | | | | provides | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | entity | message recogn./auth. | | |
| | secure relay | key pre-dist. | Single Manuf. | time sync. | recogn./ auth. | unil. / pairw. | broad-cast | self-enforc. |
| ZCK | x | - | - | - | + / - | + / - | - / - | + |
| IC | - | - | x | x | + / + | + / + | - / - | + |

† := secure acknowledgement

Table 6.4: ZCK and IC efficiency

| | Low Computations | Low Bandwidth |
|---|---|---|
| ZCK | ⊕⊕ | ⊕ |
| IC | ⊕ | ⊖ |

## 6.3 Comparison of the Schemes

We now want to compare the ZCK and IC protocol to other protocols. In Chapter 3 we considered several authentication protocols. Obviously, the scenario we are considering here can also be provided by the Guy Fawkes and the TESLA protocol that use similar approaches.

We first compare the ZCK to the Guy Fawkes protocol. Both protocols require a reliable relay channel for the initial phase. ZCK is based on hash-chains whereas the Guy Fawkes protocol is based on commitment schemes based on hash values. Both the protocols are extremely efficient. Comparing the case of mutual message authentication, the protocols have the following complexity as presented in Table 6.5.

Table 6.5: Comparison between ZCK and Guy Fawkes protocol

| | ZCK | Guy Fawkes |
|---|---|---|
| exchanged messages | 4 | 4 |
| exchanged bytes | 50 | 80 |
| computational effort | - | - |

One can see that both the schemes require negligible computations. However, the Guy Fawkes protocol requires more exchanged bytes. Furthermore, the Guy Fawkes protocol is more complex. If Alice only wants to authenticate a single message $m_0$ she needs to perform two iterations of the Guy Fawkes protocol since the key for the authenticated message is opened in the next iteration. If the ZCK protocol is used a single protocol run is sufficient to authenticate a single message.

Furthermore, we want to compare the ZCK and IC protocol to TESLA [64]. Both approaches are based on Lamport's hash-chains. The IC protocol even couples hash-chains with with a loose time synchronization mechanism as does TESLA. However, there are the following major differences between TESLA and our protocols:

- *ZCK protocol:* The ZCK protocol does not require any time synchronization whereas TESLA does. TESLA is based on time intervals in order to enlarge the number of messages that can be authenticated by only one signature. TESLA is providing a mechanism that even allows to provide message authentication for a broadcast stream. On the other side the ZCK protocol uses hash-chains in order to provide authentication by interaction without any requirement for a time synchronization. The difference becomes apparently when applying the ZCK and TESLA protocol in order to authenticate a single message. Then TESLA degenerates to authentication by a digital signature (or in the case of $\mu$TESLA by a pre-distributed shared secret) whereas the ZCK protocol does not require public-key algorithms or a pre-distributed shared secret.
- *IC protocol:* The IC protocol is similar to the TESLA protocol. However, in contrast to TESLA our IC protocol applies hash-chains in combination with pre-computations [1] at initialization time in order to minimize the

[1] Note that these pre-computations do not induce the pre-deployment of secret shared keys

Table 6.6: Operational complexity of ZCK and UDH

| | | ZCK | UDH |
|---|---|---|---|
| init | public-key size (Bytes) | 10 | 21 |
| | exchanged messages | 2 | 2 |
| | exchanged bytes | 20 | 42 |
| | computational effort | - | 4 PK Op. |
| authentication | exchanged messages | 3 | 1 |
| | exchanged bytes | 30 | 10 |
| | computational effort | - | - |
| | code size | 35 KB | - |

required computation later on. Hence, the hash-chain is used in order to agree on a key that is later on used as key in the ZCK protocol without actually computing it but by leading it back to a pre-computed certificate value. Again, consider the case of a single message authentication. As said before, TESLA degenerates to the basic signature scheme where the sender generates a digital signature of the message whereas the receiver verifies the signature. In contrast, the IC protocol only requires the verification step in order to authenticate a message.

### 6.3.1 Operational Complexity

We now consider the operational complexity of above protocols for message recognition and message authentication, respectively. We assume the same parameters as above. Table 6.6 gives an overview of the complexity of the ZCK and UDH protocols. It is distinguished in the initialization phase and the recognition phase. Note that contrary to the previous considerations here the transmission of the MAC for each recognition is also considered. We implemented the ZCK protocol on the Mica2 Motes. A preliminary version has a code size of around 35 $KB$. Since a hash function has negligible complexity compared to a public key operation we omit it here. Note that for both the ZCK and the UDH protocol a full distribution of all the public keys requires $n(n-1)$ keys as there is no central directory of public keys as there is for a public-key infrastructure.

We now present the operational complexity of the IC and the CDH in Ta-

but computational pre-computations.

Table 6.7: Operational complexity of IC and CDH

| | | IC | CDH |
|---|---|---|---|
| init | certificate size (bytes) | 138 | 61 |
| | exchanged messages | 7 | 2 |
| | exchanged bytes | 348 | 122 |
| | computational effort | 2 PK Op. | 4 PK Op. |
| authentication | exchanged messages | 3 | 1 |
| | exchanged bytes | 30 | 10 |
| | computational effort | - | - |
| | code size | 45 KB | - |

ble 6.7. Note that the IC protocol needs quite some memory storage for precomputation. An implementation of the IC protocol is to be expected at around further 10 $KB$ for the RSA verification such that we estimate a code size of 45 $KB$. Again, we distinguished the initialization phase and the authentication phase. The number of exchanged bytes for the IC can be computed as $2 \cdot 138$ Byte for the certificates, further 70 Byte for the hash and MAC values as well as 2 Byte for the seeds, such that we obtain 348 Byte altogether.

### 6.3.2 Device Lifetime

We define the lifetime of a device as the duration of time the node works with a given power supply (e.g. battery) once it is deployed until the energy resources are exhausted and the device needs to be re-charged or replaced. For simplicity we are considering a flat and homogeneous network, i.e., each node has the same hardware characteristics and there is no clustering structure within the network. Note that for the energy efficiency of the ZCK and the IC protocol, a flat homogeneous network is the most challenging architecture since here the total initial energy load of the network is most restricted. We now consider the lifetime of a network when using our new protocols. Let $n$ be the average number of forwarding nodes, $x$ be the average number of message authentication processes between each pair of parties that established a communication channel, and $p$ be the average number of channels that a party establishes to other parties. Then the average energy consumption of an involved node $\widetilde{E}$ can be computed as follows:

$$\widetilde{E} = p(\hat{E}_A + n \cdot \hat{E}_N + \hat{E}_B)/(n+2) \tag{6.13}$$

For the ZCK protocol, by applying Equations (6.5) and (6.6) we obtain

$$\begin{aligned} \widetilde{E}_{ZCK} &= p/(n+2)\ (n(x+1)\ 0.14(d/50)^2 + n(x+1)\ 0.044 + \\ &\quad (x+1)\ 0.14(d/50)^2 + x\ 0.0924 + 1.012)\ [nAh] \end{aligned} \tag{6.14}$$

Note that for simplicity we do not consider additional costs resulting from retransmission over the wireless. Let $D$ be the average distance between Alice and Bob over several hops, and $d$ be the distance of a node to its neighbor node, i.e., $D = (n+1)d$. For a given $D$, $\widetilde{E}$ will obviously be minimal for small $d$ as this goes into Equation (6.14) with square complexity. The battery of a sensor node has a capacity of around $2200mAh$ of energy. For instance, if $D = 50m$, $d = 10m$, $n = 4$, and $x = 1000$, a sensor node can establish trust associations to $p > 40,000$ different parties before the energy is exhausted. If $D = 100m$, $d = 50m$, $n = 1$, and $x = 1000$ this decreases to around $p = 15,000$.

We now consider the lifetime of a sensor network when using the IC protocol to establish identified channels and afterwards ZCK to authenticate messages. The average energy consumption of an involved node can be computed by Equations (6.10) to (6.12) as follows:

$$\begin{aligned} \widetilde{E}_{IC} &= p(\hat{E}_A + n \cdot \hat{E}_N + \hat{E}_B)/(n+2) \\ &= p/(n+2)\ (((n+1)0.14x + (n+1)1.33)(d/50)^2 + \\ &\quad nx\ 0.044 + n\ 0.418 + x\ 0.0924 + 8.5876)\ [nAh] \end{aligned} \tag{6.15}$$

Since the IC protocol requires more computational resources than the ZCK protocol, we assume that a node only establishes identified relationships to its neighborhood. Thus we pick a random node Alice and consider the neighborhood of this node. All nodes that are reachable by at most $l$ hops are in the $l$-hop neighborhood of our node. We denote by $c$ the connectivity of nodes, i.e., the number of nodes that a node can directly reach without any intermediate nodes. The number $p(l,c)$ of $l$-hop neighbors in a $c$-connected network for $3 \leq c \leq 6$ is then computed as

$$p(l,c) = c\sum_{i=1}^{l+1} i. \tag{6.16}$$

Next, we need to know how many intermediate nodes are on average involved on the route from our node to a node of its neighborhood. All nodes that are reachable by one hop do not involve any intermediate nodes. All nodes that are reachable by two hops involve one intermediate node, and so on. Thus altogether the number of intermediate nodes $n(l,c)$ involved in pairwise authentication in a $c$-connected network within an $l$-hop neighborhood of our node Alice can be computed as follows for $3 \leq c \leq 6$:

$$n(l,c) = (c\sum_{i=1}^{l+1}(i-1)i)/p(l,c). \tag{6.17}$$

Note that with respect to their position nodes need to be counted multiple times to take into account that their involvement in multiple authentications with varying communication partners from the $l$-hop neighborhood.

Thus, we can compute the total average energy of a node within its neighborhood by plugging in values $p$ and $n$ of Equations (6.16) and (6.17) into (6.13). For instance, consider a homogeneous network that is 4-connected with $d = 20m$. When using the IC protocol to establish a identified channel to the 1-hop neighborhood, i.e. to $p(1,4) = 12$ nodes, and on average there are $x = 1000$ messages exchanged, then $n(1,4) = 2/3$. By inserting these values into Equation (6.15) we obtain an energy consumption of $\widetilde{E} = 757\ nAh$ which is less than 0.04% of the total battery capacity of 2200 $mAh$.

Clearly, the energy consumption of the CDH and UDH protocols also depends on the overall distance $D$ and the distances $d$ between each two nodes. As argued above, the energy consumption is minimal for a small distance $d$ as it goes into the energy consumption with a square complexity. The average total energy can easily be obtained for the UDH protocol as

$$\widetilde{E}_{UDH} = p/(n+2)\ ((n+1)0.14(d/50)^2 + n\ 0.044 + 26.444)\ [nAh] \qquad (6.18)$$

and for the CDH as

$$\widetilde{E}_{CDH} = p/(n+2)\ ((n+1)0.42(d/50)^2 + n\ 0.132 + 39.732)\ [nAh] \qquad (6.19)$$

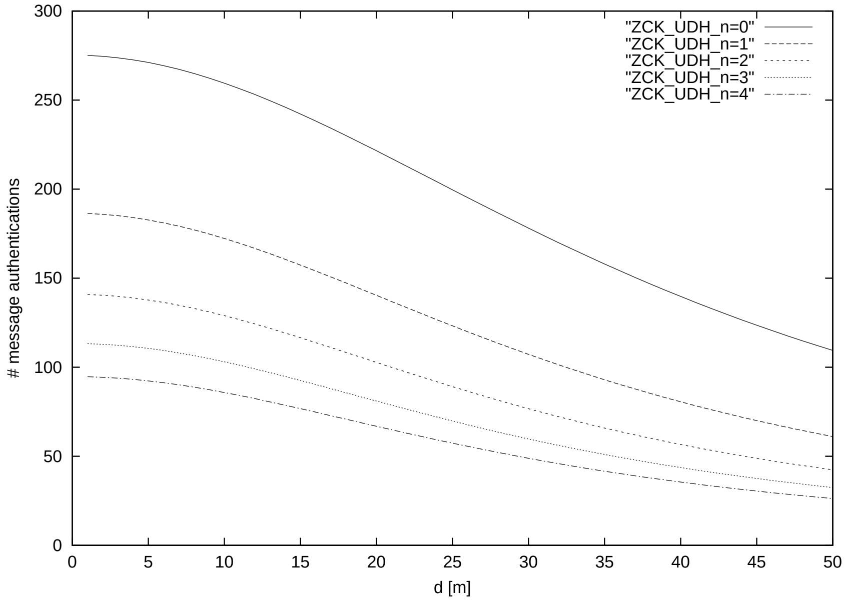
### 6.3.3 Comparison of the Schemes

Finally, we want to compare our new schemes to the hybrid ones. We first emphasize the benefits of the ZCK protocol: (1) it is computationally extremely efficient; (2) it is provably secure; (3) it works between two low power devices; and (4) it allows a party easily to start fresh (anonymity). First, it seems clear that for recognition our ZCK protocol is computationally far more efficient than any asymmetric scheme, and it is nearly as efficient as a message authentication scheme using a MAC. Hence, for time critical applications, or when there is the need of establishing new channels often, ZCK should be the first choice. If the number of intermediate nodes is small, ZCK is also from an energy perspective a good solution. If identification is needed, IC performs well in some scenarios. If the network is reliable and packets do not get lost frequently the large certificate size only has moderate impact. Furthermore, if very low-power devices are used, e.g., sensors that are equipped with a 4-bit CPU at less than 4 MHz, IC might be the only possibility to perform an identification since an ECC key agreement is too demanding for these devices. Again, if there are time critical applications, it might be a good idea to use IC since it is operationally more efficient than an ECC solution.

We now want to compare the energy consumption in more detail. Thus we compare the energy consumption for ZCK to the UDH, and IC+ZCK to the CDH. Namely, we compare Equation (6.14) to (6.18) and Equation (6.15) to (6.19). Hence we are interested when

$$ZCK \leftrightarrow UDH : n(x+1)\ 0.14(d/50)^2 + n(x+1)\ 0.044 + (x+1)\ 0.14(d/50)^2$$

Figure 6.1: Trade-off between the ZCK and UDH protocol



$$+ \; x \; 0.0924 + 1.012 < (n+1)0.14(d/50)^2 + n \; 0.044 + 26.444$$

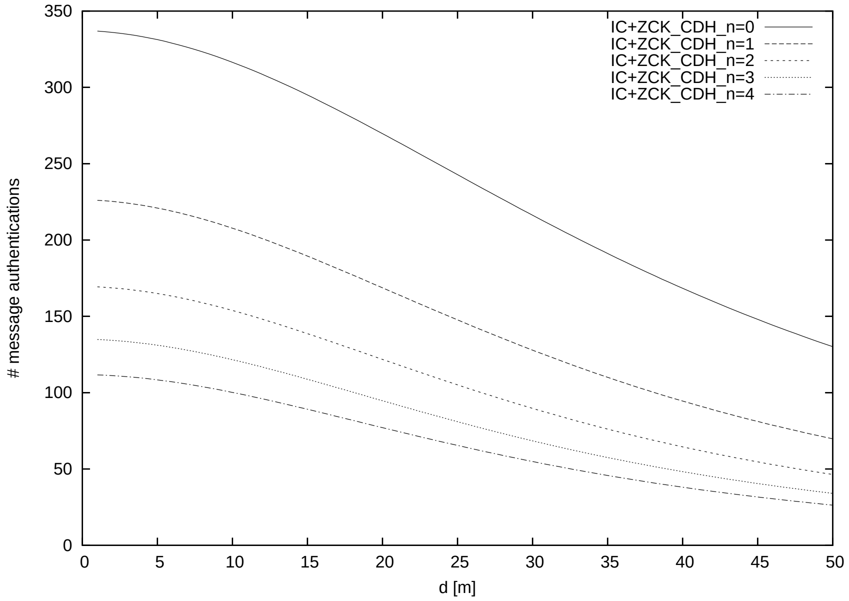
and

$$IC + ZCK \leftrightarrow CDH : ((n+1)0.14x + (n+1)1.33)(d/50)^2 + nx \; 0.044$$
$$+ \; n \; 0.418 + x \; 0.0924 + 8.5876 < ((n+1)0.42(d/50)^2) + n \; 0.132 + 39.732$$

Figure 6.1 illustrates the trade-off between the ZCK and the UDH protocol. There are five graphs for fixed $n = 0 \ldots 4$ that determine the trade-off of messages $x$ for different distances $d$, i.e. the number of messages $x$ where the ZCK and UDH protocols are equivalent regarding their energy consumption. Below the curve, the ZCK protocol requires less energy than the UDH protocol.

Figure 6.2 illustrates the trade-off between the IC+ZCK and the CDH protocol in the same manner. For instance, let us consider the first case for a scenario where $d = 20\ m$ and $n = 4$. If on average there are at most $x = 66$ message authentications performed, ZCK requires less energy then the UDH, otherwise it requires more energy. For a network where $d = 50\ m$ and $n = 0$, ZCK is more energy efficient for $x \leq 109$. Now let us compare the IC+ZCK

Figure 6.2: Trade-off between IC and CDH



to the CDH. Then for $d = 20m$ and $n = 4$, IC+ZCK is more energy efficient for $x \leq 77$, and for $d = 50\ m$ and $n = 0$ it is for $x \leq 131$.

It now becomes clear that the ZCK protocol and also the IC protocol are very energy efficient for a small average number of hops $n$ in the network and a small number of message exchanges $x$ per communication pair. Hence, we foresee applications in highly mobile networks. Furthermore, when computational efficiency is of importance, e.g. in time critical applications, our protocols are by far more efficient than any solution using intensive asymmetric cryptography. On the other hand our ZCK protocol requires caching of a message before it can be verified. Thus the scheme is vulnerable to a message overflow attack. Due to the increased number of messages compared to other schemes our ZCK as well as our IC protocol might be less powerful in unreliable networks. If data packets get lost frequently, the number of re-sent messages will decrease network performance.

We believe that our schemes close the gap of symmetric protocols to asymmetric protocols. We provided efficient solutions for several scenarios when there is no TTP nor key pre-distribution available. In particular, the ZCK protocol provides mutual recognition solutions for Pure Ad-hoc networks whereas the IC is suited to Single Manufacturer networks. When used with trust accumulation schemes, ZCK is able to provide almost the functionality of entity authentication and message authentication. We believe that this the optimum to achieve in a Pure Ad-hoc network. Our schemes become especially powerful in combination with a local neighborhood approach of higher-level security protocols.

# 7 Component Identification

After considering several issues related to authentication in ad-hoc networks and presenting new authentication protocols, we now present an application of ad-hoc network protocol techniques in order to provide component identification in complex systems. Our component identification provides theft protection as well as protection of counterfeits and manipulation. Our solution might be applied to systems where such security goals are crucial and an enabler for safety, e.g., in automobiles and airplanes. We present our protocol in the context of automobiles because new technologies are quickly introduced to this industry sector. Parts of this chapter were presented in [86, 88].

## 7.1 Introduction

As the ad-hoc network technology matures it also becomes increasingly interesting for the vehicle industry. Ad-hoc networks of cars could considerably make driving a car more comfortably and more safe. Several scenarios and protocols regarding security of smart vehicles were introduced by Hubaux et al. in [40]. For instance, cars will communicate to exchange information about free parking spaces and about threats. A basic enabler for such a technology is a unique identifier for each car. This would be the electronic counterpart to today's license plate. If this electronic license plate broadcasts its identification while the car is running, this information could be used by toll systems, by police, or in case of a hit-and-run accident. A core requirement is that the hardware module that implements the electronic license plate cannot be manipulated nor faked or stolen in order to install it in another car. The same requirements hold for all security related components of a car. As a car is entirely accessible by its owner, the approach of ensuring such security goals is not obvious. The straightforward idea is to have a tamper resistant hardware module that implements all security mechanisms. For instance, this security module would implement the electronic license plate in order to broadcast a unique identification string over a built-in radio. However, an adversary could just steal the security module and build it into his car for impersonation, or he could remove the security module from his car. Thus, there needs to be a bond between components and the vehicle. Now, if an adversary manipulates or removes any security related part of the vehicle, the vehicle could stop working. Even more, if an adversary manipulates any device that is related to safety, e.g. the airbag or brake system, the vehicle will stop working. We propose a scheme for providing component identification in order to bind security and safety related parts of a vehicle to a central security module. Only if there is any such mechanism implemented in a car, security of ad-hoc networks between cars can be implemented in a way that suffices legal requirements. Otherwise, if a car owner is able to manipulate his car in a way that it broadcasts malicious messages, traffic might be affected or endangered. For instance, if a car driver sends messages about a non-existing threat, drivers of following cars might initiate an emergency brake and endanger their life. Hence, our scheme is an enabler for security of ad-hoc networks between cars. On the other hand, all components form an ad-hoc network by itself. New components can be added and replaced, and components are provided with a secure communication channel.

To the author's knowledge there is no system yet that provides component identification using cryptographic methods. Traditional methods use tags, e.g. holographic stickers, that are supposed to be unforgeable. However, as it can be seen in the airplane industry, such tags can easily be bought on the black market [9]. A watermark can nicely be embedded into a component by a DNA fingerprint. A unique special DNA sequence is composed by means of modern biological mechanisms. Then the DNA is mixed with ink which in turn is printed onto the component. A special pen filled with the counterpart to the DNA sequence is then used to validate the component. More about seals and stickers can be found at [82]. Another way of providing a basic protection against counterfeits and theft is ensured by mechanical countermeasures. For instance, there are car radios of special size such that they only fit into cars of a single manufacturer. However, the owner cannot keep his radio after buying a new car so that this solution is uncomfortably. Ensuring component integrity of cars was already considered by automotive companies [18, 83]. They propose to load the vehicle identification number into each component.

System integrity is ensured by checking whether all components use the same vehicle identification number and whether all components are still in the car. An authorized mechanic performs any changes at the car. If a component is installed or demounted by the authorized mechanic, on-line access is required. Clearly, there is a considerable threat due to non-trustworthy mechanics.

In the following we present a scheme to bind crucial components to a car. We use component identification to secure all components of a car against cloning (faked parts), manipulation, and theft. In particular, our scheme protects the car owner of bogus parts that endanger the operation of the car, and it minimizes the financial damage of suppliers. This damage is estimated to be as high as 250 billion Euros worldwide [42]. Our first protocol is based on a central security module which we call High Security Module (HSM). An HSM is a tamper resistant security module that is able to perform cryptographic operations. The key data is stored inside of the HSM. Trying to compromise the HSM in order to get a key will destroy the HSM and delete its memory such that the keys are lost. Since building such a module is hard to achieve and costly the second protocol we present gets rid of the HSM by distributing the task of the HSM to all components. Our solution is very general and easy to apply. There is no need to adjust components before they are installed, and all processes can be executed automatically. Furthermore, our scheme does not only provide a secure environment for the owner of a car but for all involved parties such as the manufacturer and the mechanics. Our solution is applicable to cars but also to other transportation vehicles such as trains and airplanes. Furthermore, our work is applicable to other systems which consist of components that need protection against cloning, manipulation, and theft.

### 7.1.1 Assumptions

For the remaining we assume the following.

1. A simple computing tag is attached to each crucial component in such a way that removing the tag destroys the component.

2. The HSM as well as the computing tags are able to perform cryptographic operations.

3. There is a temporary connection available between the vehicle and a central server.

Assumption (3) can be omitted if there is no connection available. Protection against cloned parts is then not possible, though. We present approaches where both an HSM is used and where it can be omitted. Also we present approaches where asymmetric cryptographic methods can be replaced by symmetric methods such that low-cost RFID tags can be attached to the components. We now present our assumptions in more detail:

- The component tags communicate over a wireless communication channel. All components are able to communicate to each other component either by a single-hop or multi-hop communication channel. Furthermore, all component tags are able to communicate to the car's board computer.

- To perform cryptographic operations and to transmit data packets the component tags need quite some energy. Hence, we assume that they are equipped with a battery having a life span of several years that outlasts the lifetime of the component, or that the component is directly powered by the component that by itself is connected to the car's power supply network. We also present a solution for passive RFID tags that do not require a battery.

- There is a trusted third authority that issues certificates, e.g., a government institution or a car manufacturer.

- The components or a central security module are able to take actions in case that a protocol step fails. For instance, the central security module might share a key with crucial car components. It then orders these components to turn off in case of a failure.

- If any check in the protocol fails, an alarm is issued. For instance, the engine stops working, or an alarm message is broadcasted.

- Each component is equipped with a small CPU tag. This tag, for instance a smart card or RFID chip, is able to perform basic cryptographic operations. It is advantageous to bind the module to the component in such a way that removing or tampering with the module destroys the component as well as the cryptographic secrets. For instance, a tag might be embedded at the time when an engine is molded.

- Each component holds a unique $ID$ as well as a cryptographic secret $SK$. $ID$ comprises a unique identification string as well as further information such as the manufacturing date and the component's quality class.
- The cryptographic mechanisms are well chosen and implemented. The component tags are well embedded to the components such that it is not possible to measure any side-channel information such as power supply or EMF radiation. Depending on the used technology there might be sophisticated side-channel attacks possible in order to recover the secret data of a component tag, though.

### 7.1.2 Adversarial Model

For the attacker we assume the following. In short, we assume that the adversary has full control over the communication channel. Hence, the adversary can

- read all messages sent from any component,
- modify messages, delay them or send them multiple times, and
- send messages generated by himself to any component.

Furthermore, an adversary has full physical access to the car and its components. Using sophisticated technologies the adversary might be able to recover secret key data of the component tags. Since the lifetime of components and tags will be a decade or so, the secret key material might be exposed to such attacks after a while. The adversary has full control over all components, and he can program faked component tags.

The backend system is not vulnerable to any attacks. We assume that there are no insider attacks, that central servers and directories are not vulnerable and cannot be read or manipulated by non-authorized entities.

The adversary has several aims:

1. to a given car install a component that is not an original one

2. modify or replace a component in any car by a component that is not an original one

3. steal a component of any car and install it to another one

4. to any car install a component that is not an original one

The aims (1) and (2) can be seen as the attack of a single entity, e.g., the owner of a car who wants to install a faked component to his car. Aims (3) and (4) might be seen as the objectives of organized groups that want to distributed stolen, faked, or cloned components.

## 7.2 Asymmetric Component Identification

The main goal is to bind components to a vehicle. We first present a solution based on an HSM using asymmetric cryptography. The main threat for a car arises if components are stolen, manipulated, or cloned. We define *cloning* as the act of rebuilding a component up to a perfect copy, in some cases even including secret cryptographic keys. Hence, our scheme provides piracy protection, system protection, and theft protection. *Piracy protection* provides the ability to identify original parts and the possibility to detect counterfeits and cloned components. *System protection* provides system integrity by monitoring systems for unauthorized changes, and *theft protection* prevents the use of stolen components in another system. We are considering a system in which all components hold authentic data which can be verified via a data bus whereby all claimants and at least one verifier are connected to the bus. There are three phases that we consider: (1) The installation of a component into a car, (2) the running system, and (3) the demounting of a component out of the car. We describe our security goals by an example. Only original parts can be built into the car. Every time the ignition key is turned the system integrity is checked, and only in case of an unaltered system the engine starts. A display in the dash board shows the status of all present system components. This could be useful, e.g, to prove that the mechanics of a car garage used original parts only. There might also be a button to manually start the check such that the owner can prove integrity to a third party. For instance, the owner can prove to police that a proper electronic license plate is built in.

We assume the following vehicle environment as above. Furthermore, we assume the following:

- There is an HSM inside of each car that is considered to be tamper resistant.
- The component tag is able to perform asymmetric operations.
- The HSM performs cryptographic operations and stores secret key data, and it is able to take actions in case of malicious behavior of components. The HSM might be a component itself. For instance, the role of the HSM might be executed by the board computer. The HSM is able to take actions in case of a malicious behavior of a component. For instance, the HSM might share a special secret key with the engine in order to send the engine a command to turn itself off.
- Each component holds a certificate $\langle PK, ID \rangle$ as well as the corresponding secret key $SK$. $ID$ comprises a unique identification string as well as further information such as the manufacturing date and the component's quality class.
- The HSM of a vehicle holds a list $\mathcal{UL}$ of all components built into the vehicle. This list is regularly synchronized with a global list $\mathcal{GL}$ of all components. The synchronization is performed in a secure manner to avoid any manipulation. Each component can be uniquely identified.
- There might be a global certificate revocation list ($\mathcal{CRL}$) with all components that were revoked. A component might be revoked if it was stolen, or if it is known that it was cloned.

In the context of sensor networks we argued that tamper resistant hardware modules are hard to produce and costly. In the car environment, we believe that a tamper resistant device such as an HSM is possible, however. The cost for the HSM can be high but it is still low relative to the car's cost. Then a sophisticated HSM can be used such that the effort to compromise the HSM is extremely high and the benefit low. For the tags, we expect that they can be embedded into the components by means of mechanical engineering in such a way that removing them requires that much effort that it exceeds any gain.

All identifications are performed in a challenge-response manner. After $A$ presents a certificate, $B$ needs to check whether $A$ has knowledge of the corresponding secret key $SK$. Algorithm 7.1 provides this check.

**Algorithm 7.1** Challenge-response identification

```
$B \rightarrow A : r_B$
$A \rightarrow B : r_A, S := SIG(r_A||r_B||B, SK)$
$B$ checks whether $VER(S, PK) \overset{?}{=} 'valid'$
```

The check of a symmetric key $K$ is performed in a similar way by using a MAC. Algorithm 7.2 provides this check in a mutual way. Here, $A$ checks whether $B$ knows the secret key $K$ and also $B$ checks whether $A$ knows the secret key.

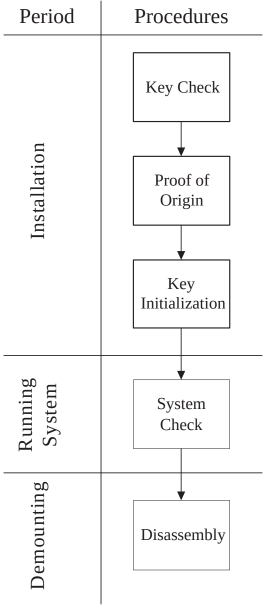
**Algorithm 7.2** Mutual challenge-response check of a symmetric key

```
$B \rightarrow A : r_B$
$A \rightarrow B : r_A, M_A := MAC(r_A||r_B||B, K)$
$B$ checks whether $MAC(r_A||r_B||B, K) \overset{?}{=} M_A$
$B \rightarrow A : M_B := MAC(r_B||r_A||A, K)$
$A$ checks whether $MAC(r_B||r_A||A, K) \overset{?}{=} M_B$
```

Note that in general all messages of components are encrypted and authenticated by a commonly shared key $K$.

As already mentioned there are three phases to consider when talking about component identification which we will discuss in detail. The main idea is to have an initial component, e.g., the HSM that is imprinted with a secret key $K$. Each component holds a certificate. If a component is added to the vehicle, the certificate is checked. Then the component obtains the secret vehicle key $K$ which is checked while the car is running in order to prevent manipulations after the installation. Finally, our solution allows a controlled demounting of a component in order to distinguish stolen components from properly demounted ones. The life cycle of a component is depicted in Figure 7.1.

Figure 7.1: Life cycle of a component



## 7.2.1 Initialization of a Car

At initialization time, the HSM is installed into the car as first component. Further components might be installed into the car at initialization time. If the HSM requires means of disabling the car or issuing an alarm the HSM exchanges secret keys with crucial components. For instance, the HSM might agree on a key with the car's engine and the car's dashboard, e.g. by using the Diffie-Hellman key agreement based on the HSM's and components' certificates such that it is able to disable the car or to display warning messages. Note that for this purpose a separate key is shared between the HSM and the component that is only used for this purpose.

The HSM now randomly chooses a key $K$ that becomes the vehicle key. Assuming that the car is assembled in a trustworthy environment such as a manufacturer plant the vehicle key $K$ is now distributed to all installed components. Note that here the key $K$ is not encrypted. If there is no trustworthy environment available then for each installed component the installation procedure can be executed as described below.

If the component tags are equipped with computationally strong CPUs a traditional key management for the formation of ad-hoc networks can be used. For instance, a group key-agreement scheme based on the components' certificates could be used for a higher security level at higher computational costs [77]. For the devices we envision here we believe such a scheme to be too resource demanding, though.

### 7.2.2 Installation of a Component

A new component is installed by adding it to an already existing set of components that form the car. Components are added stepwise. Once a new component is added to the car the installation phase is executed. It consists of the following steps:

- *key check:* check whether the component is already part of the vehicle, e.g., after the component was disassembled for repairing it.
- *proof of origin:* check whether the component has a valid certificate.
- *key initialization:* providing a valid component with the vehicle key $K$.

The key check runs as presented in Figure 7.2. First, the newly installed component $C$ provides its unique $ID$. The HSM checks whether $ID \in \mathcal{UL}$. This holds if $C$ was part of this car before. If so, the HSM checks if the new component knows $K$ by a challenge-response authentication. Otherwise, if the component is new, the proof of origin check is performed.

During the proof of origin, the HSM checks whether the new component provides a valid certificate. The HSM performs the steps as presented in Algorithm 7.3 and depicted in Figure 7.3 for a new component $C$.

**Algorithm 7.3** Proof of origin

```
1: The HSM verifies the certificate $\langle PK, ID \rangle$ and checks whether $C$ knows
   the secret key $SK$ by a challenge-response method.
2: It checks the $\mathcal{CRL}$ for $\langle PK \rangle$.
3: It puts $C$ on $\mathcal{UL}$ with a preliminary flag.
4: After $\mathcal{UL}$ was synchronized with $\mathcal{GL}$ the preliminary flag of $C$ is deleted.
   If $C$ was already on $\mathcal{GL}$, an alarm is raised.
```

In the last two steps, $C$ is put on $\mathcal{UL}$ in a preliminary manner. If $C$ is on the global built-in list $\mathcal{GL}$ then it is already built into another vehicle which indicates that this component was cloned or stolen.

Figure 7.2: Key check

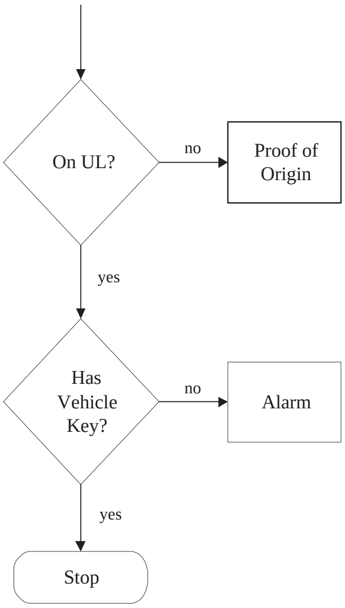


Figure 7.3: Proof of origin

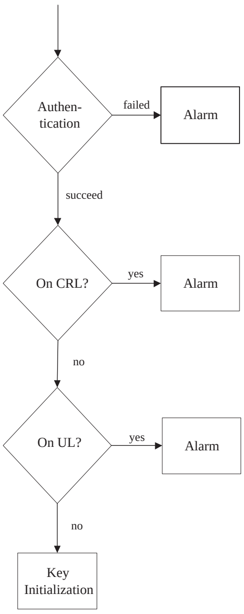
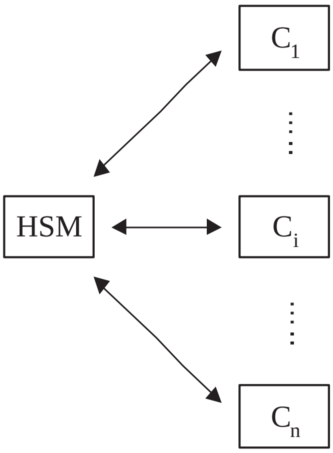


Figure 7.4: System check with one verifier



Finally, after all checks were performed successfully, $C$ is provided with the vehicle key $K$ during the key initialization. For that reason, the HSM sends $E(K, PK)$, which is the encryption of $K$ by the key $PK$, to $C$.

### 7.2.3 The Running Vehicle

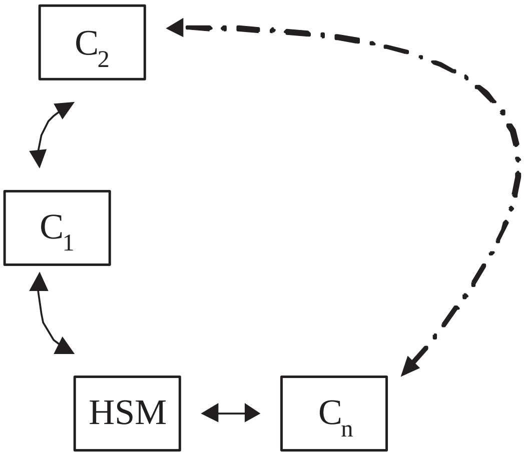
We ensured that all components that are installed into the car were verified. As trustworthy components they were initialized with the vehicle key $K$. However, after they are installed they might be manipulated, exchanged, or removed. Thus, we execute the system check in order to verify whether all components know the vehicle key $K$. The system check can be executed several times:

- every time the car is started
- periodically, e.g., every 30 minutes
- initiated manually, e.g., to prove system integrity to a policeman

There are basically two methods to execute the system check. In the first version, which is depicted in Figure 7.4, the HSM challenges all components one after the other by a challenge-response method to check the components' knowledge of $K$. Clearly, only components that are listed in $\mathcal{UL}$ are involved.

In the second method, which is depicted in Figure 7.5, each component of the vehicle checks another component. The HSM starts by challenging another

Figure 7.5: System check with cyclic verifier



component, which in turn challenges another component, and so on, until the security module is challenged by a component such that the circle is closed. Here, each component holds the list $\mathcal{UL}$ of all installed components, and all components have to agree on an order of being challenged. If any check fails, the challenging component raises an alarm by sending a message to the HSM. The HSM then takes appropriate actions. Note that if the communication bus supports parallel communication, the system check can be performed in parallel by all components. Then all components send their challenge and respond to a challenge in parallel. Obviously, it is easy to prevent an attacker from jamming the communication channel in order to delete the alarm message by letting each component either send a positive or negative alarm message. By using a challenge-response method, replay attacks are prevented.

Clearly the system checks are vulnerable to compromised components. A component might always fake an authentication since all components use the same shared key $K$. Furthermore, in the cyclic version a component might "short-cut" an authentication process by skipping components in the cycle. However, we assume here that all components that have knowledge of the system key $K$ play fair. If $K$ is known the system can be considered to be broken. The check can also be performed based on the component's certificates such as described in Algorithm 7.1. Such a check is not vulnerable to an insider attack where the insider has knowledge of the system key $K$. However, such a check based on asymmetric cryptography is in many cases too demanding for the considered device class.

### 7.2.4 Demounting of a Component

The demounting procedure has to be performed once a component is removed of a vehicle in order to install it into another vehicle. If a component is removed and then installed again into the same car, nothing needs to be done. Hence, the demounting procedure is performed to be able to distinguish a legal demounting to a theft of a component. The demounting of a component $C$ works as presented in Algorithm 7.4.

**Algorithm 7.4** Demounting

```
1: The HSM checks whether $C$ knows the secret key $K$ by a challenge-response
   method.
2: Then the HSM deletes $C$ of the $\mathcal{UL}$.
3: At the next synchronization of $\mathcal{UL}$, $C$ is also deleted of the global list $\mathcal{GL}$.
```

To remove $C$ in a controlled way, it first has to prove that it knows the secret vehicle key $K$. Then $C$ is deleted of the system and of the built-in lists $\mathcal{UL}$ and $\mathcal{GL}$ such that it appears like a new component. Components which are bound to a specific car and thus must not be demounted can be marked as such in their certificate, the $\mathcal{UL}$, and the $\mathcal{GL}$ by a flag. The HSM will then not allow these components to be demounted of the vehicle.

In cases where there is no $\mathcal{GL}$, a component stores a flag in its memory which indicates whether the component is installed in a car. When the component is demounted, the flag is deleted. Components can only be installed into a car if the flag is not set. Such a solution is also preferable in cases where the synchronization of $\mathcal{UL}$ and $\mathcal{GL}$ is only performed rarely.

## 7.3 Distributed Component Identification

In above approach we assumed that there is an HSM in each vehicle. As the HSM is a central point of failure, it probably will be the first point of attack. Thus, the HSM is assumed to be tamper resistant. More realistic and cheaper is a tamper evident module. Here, a manipulation of the device cannot be avoided but detected. However, manipulation cannot be detected in an automatic way but only by individual inspection. We now extend above scheme such that the role of the HSM is distributed to all components, meaning to all component tags. As before, a tag is embedded to each component. The tags must be able to verify certificates and perform public-key operations.

We now introduce supporting procedures for our system in order to distribute the HSM's task.

**Distribution of $\mathcal{UL}$:** As said before, the HSM holds a list $\mathcal{UL}$ of built-in components. This list is now distributed to all components. In the easiest case, each component holds a copy of the $\mathcal{UL}$. Once a component changes its $\mathcal{UL}$, it broadcasts the new version to all other components. To ensure integrity of the broadcast and in order to prevent replay attacks, we perform Algorithm 7.5. Here, $i$ is an incremented counter.

**Algorithm 7.5** List distribution

```
A component $C$ increments $i$ and computes
    $U := [E := E(\mathcal{UL}||i, K), M := MAC(E, K)]$.
$C$ broadcasts $U$.
A receiver $R$ only accepts $U$ if $M$ verifies correctly and if $i$ is larger than its stored counter $i'$
$R$ stores $\mathcal{UL}$ and updates $i' \leftarrow i$.
```

**Selection of a Random Component:** Furthermore, we need a mechanism to choose a component randomly out of all vehicle components. As said before, we assume that all components that know the vehicle key $K$ are trustworthy. First, all components have to choose a random number $x$. This works as presented in Algorithm 7.6.

**Algorithm 7.6** Selection of a random number

```
Each component $C_i$ chooses a random number $r_i$ and broadcasts $E := E(r_i||ID_i, K)$ as well as $M := MAC(E, K)$.
All components check whether they got input from all components and verify the correctness.
Each component computes $x := h(\sum_i r_i)$.
```

Clearly, this scheme is vulnerable to a malicious component that knows the vehicle key $K$ because such a component could wait for its output until it received input of all other components. It is not vulnerable to an external attack though. An unconditionally secure version of choosing a random number is based on commitment schemes which works as presented in Algorithm 7.7. Assume that $m$ is an RSA modulus, and $e$ and $d$ are the RSA public and private key, respectively. Then a simple commitment function that commits to a value $x$ under a key $d$ is $O := \mathit{commit}(x, d) = x^e \bmod m$. Intuitively, the committed value $x$ cannot be changed since $O$ is given, but it can easily checked once $d$ is published.

**Algorithm 7.7** Selection of a random number by commitments

```
1: Each component $C_i$ chooses a random number $r_i$ and a random key $K_i$,
   commits to it as $O := \mathit{commit}(r_i || ID_i, K_i)$ by key $K_i$, and broadcasts $O$.
2: After all components broadcasted their commitment, each component
   opens its commitment by broadcasting $r_i$ and $K_i$.
3: All components verify the commitments and compute $x := h(\sum_i r_i)$.
```

Now, all components can choose a random component as presented in Algorithm 7.8. All components know which component acts as verifier, and only accept a new $\mathcal{UL}$ of the selected component.

**Algorithm 7.8** Selection of a random component

```
1: All components together select a random number $x$.
2: Each component $C_i$ computes $v_i := h(ID_i || x)$ where $h$ is a hash function.
3: The component $C_i$ with the smallest value $v_i$ acts as verifier. As the IDs
   are publicly known, all components can verify the selection.
```

**Raising an Alarm:** Finally, we need an efficient scheme for raising an alarm in a distributed manner. This can be done as follows. Each component that performs a check has to broadcast its result, i.e., a positive or a negative result message. If the result is negative, each component takes action. In the easiest case, a component stops its operation. In order to prevent that an adversary jams the communication channel to prevent broadcasts of negative results, we introduce a time-out of result messages. If a time-out occurs, a component assumes a negative result and takes action. To prevent failures, the time-out value can be chosen very large, or a component can be asked to send its result again.

After designing these auxiliary protocols, we now consider the life cycle phases of a component. These follow in a straightforward manner by combining the phases installation (Section 7.2.2), running system (Section 7.2.3), and demounting (Section 7.2.4) with the auxiliary protocols above.

**Installation**: For the installation of a component $C$, there has to be a component acting as verifier $S$ instead of the HSM. This component is chosen randomly out of all vehicle components using Algorithm 7.8. Now, $S$

performs the installation procedure, updates $\mathcal{UL}$, and broadcasts it by using Algorithm 7.5.

**Running System**: For the system check, a random component is chosen by Algorithm 7.8 to initialize the check. If any check fails, a distributed alarm is raised as described above. If the cyclic system check is performed, all components perform the check in parallel.

**Demounting**: For the demounting of a component $C$, a random component $S$ is chosen by Algorithm 7.8 to act as verifier. Now, $S$ performs the demounting operation, updates $\mathcal{UL}$, and broadcasts it by using Algorithm 7.5.

## 7.4 Symmetric Component Identification

After presenting an approach based on asymmetric cryptography we now present an approach based on symmetric cryptography only. Such an approach is especially useful if we assume that the security tags are resource limited. In particular, the power as well as computational power might be limited. In the optimal case, RFID (Radio Frequency Identification) tags are used to implement the security tags. These do not require a battery but are powered by the radio waves such that the security tag's lifespan is not limited by its power resources. However, such low-power devices only have extremely limited computational resources just enough to perform basic symmetric cryptographic methods.

For this approach we modify our assumptions as follows:

1. There is an HSM in each car. The HSM acts as a gateway between component tags and a central server $S$.
2. The HSM is temporarily connected to a central server $S$.
3. The HSM and $S$ share a symmetric key $K_{HSM}$.
4. A security tag, e.g. an RFID, is embedded into each component.
5. The security tags as well as the HSM are able to perform basic symmetric cryptography such as computing a MAC but no asymmetric cryptography.

6. Each component $C_i$ is equipped with a symmetric key $K_i$. The central server $S$ holds a list $\mathcal{KL}$ of all symmetric keys $K_i$.

7. Each car holds a system key $K$.

Hence, all challenge-response authentication processes are performed by a symmetric scheme as described in Algorithm 7.2. We base the symmetric approach on an HSM inside of each automobile. The HSM is temporarily connected to a central server such that data can be exchanged whenever an on-line connection is available. The HSM only needs to perform symmetric cryptography but no asymmetric one. Note that the HSM mainly acts as gateway between the car's components and the central server such that the HSM can be a dedicated component tag. Hence, this approach is based on low-cost RFID tags only and does not require any expensive devices. We now consider the life cycle of a component.

### 7.4.1 Installation of a New Component

As before, a new component is installed to an already existing set of components that form the car. Again, we run through the following phases:

- *key check*
- *proof of origin*
- *key initialization*

The key check to check whether the new component already was part of the car in the past is executed as presented in Section 7.2.2. The proof of origin and key initialization need to be adapted though. During the proof of origin, the HSM first sends a challenge $r$ to the new component $C_i$ to be signed by the new component. $C_i$ computes the MAC over $r$ by its secret key $K_i$ and sends back $M := MAC(r, K_i)$ to the HSM. The HSM accepts $C_i$ in a preliminary manner and waits for the next available on-line connection to the server. The HSM then forwards $M$ to the central server together with $r$ and the system key $K$. The server has a list of all components' secret keys $K_i$. If it can successfully verify the new component's response, it sends the new component the encrypted system key. The server $S$ also checks whether $C_i$ was already build into another car and thus is a clone. If any check fails in the algorithm, an alarm is issued and countermeasures are taken as described above. These steps are presented in Algorithm 7.9.

**Algorithm 7.9** Proof of origin for the symmetric case

```
The HSM sends a challenge $r$ to $C_i$.
$C_i$ computes $M := MAC(r, K_i)$ and sends $M$ to HSM.
The HSM puts $C_i$ on $\mathcal{UL}$ with a preliminary flag.
After a connection between the HSM and $S$ is established, the HSM sends
  $Enc(M||r||K, K_{HSM})$ to the server $S$.
$S$ checks whether $C_i$ is on $\mathcal{GL}$.
If the check is negative, $S$ obtains $K_i$ for $C_i$ from $\mathcal{KL}$ and verifies whether
  $MAC(r, K_i) \overset{?}{=} M$.
If the verification is successful, $S$ computes $E := Enc(K, K_i)$, puts $C_i$ on
  $\mathcal{GL}$, and sends $E$ to HSM.
The HSM forwards $E$ to $C_i$ and deletes the preliminary flag.
$C$ and the HSM now perform a challenge-response authentication to prove
  knowledge of $K$.
```

### 7.4.2 Running System and Demounting of a Component

The system integrity of the running system is performed based on a symmetric cryptographic scheme as presented in Section 7.2.3. Furthermore, the demounting of a component is performed in the same manner as presented in Section 7.2.4. Hence, no adaption of above algorithms is necessary for the running system and the demounting phase.

## 7.5 Features and Enhancements

Our security scheme provides piracy protection, system protection, as well as theft protection. We now consider features we can implement additionally.

**Policies and Priorities:** Each component certificate can include a string about its properties and limitations. For instance, this information might include a flag whether the component is allowed to be removed of a given car, or if a replacement must be installed. Furthermore, it might include what action has to be taken if the component does not act in an appropriate way or if it is missing. For instance, the other components might stop operating if the airbag is missing, or they might display a warning message if the car radio is missing. This additional information is stored in the $\mathcal{UL}$ shared by all components.

**Controller Units:** If the component is already equipped with a microcontroller, the security tag can be combined with the component. For instance, consider a mileage counter. These are often manipulated in order to increase the value of used cars. The manipulation can be prevented by restricting access to the mileage counter to those components that know the vehicle key $K$. Here, it is possible to lower the requirement for a tamper resistant device to a trusted computing platform at the same security level. Such trusted computing solutions are able to protect the microprocessor against manipulation and thus provide a sufficient level of security.

**Component History:** This feature provides the opportunity to distinguish between new and used components. We present two solutions. First, a component holds an initial key which can only be used once. Each new component holds a certificate $\mathcal{Z} := \langle PK, ID \rangle$ as well as another certificate $\langle PK_{new}, \mathcal{Z} \rangle$ and the secret key $SK_{new}$. When a component is installed for the first time, it proves knowledge of $SK_{new}$. The security tag then deletes $SK_{new}$ from its memory once the component was successfully installed. If the same component is installed into another car, it cannot prove knowledge of $SK_{new}$ anymore. The second variant is based on a central directory of all components. This directory records all information about the component including installation and demounting date. It is straightforward to implement this directory based on $\mathcal{UL}$ and $\mathcal{GL}$. Clearly, this solution only works if there is a communication channel available to synchronize $\mathcal{UL}$ and $\mathcal{GL}$.

**Key Update:** Another feature deals with the update of the system key $K$ to increase the system's security. The system key $K$ should be updated periodically and under certain circumstances, e.g., if an unauthorized component is detected in the system. Another reason for updating $K$ could be a missing component. Only symmetric encryption is used for the key's update. The new system key $K_{t+1}$ is simply distributed by encryption with the old system key $K_t$ after all malicious components were removed from the car. This is shown in Algorithm 7.10.

Alternatively, all components might be verified again. Hence, a new sys-

**Algorithm 7.10** Key update

```
Choose a random component $S$ out of all components, or set $S = HSM$
$S$ randomly chooses $K_{t+1}$
$S$ broadcasts $E := E(K_{t+1}, K_t)$ and $M := MAC(E, K_{t+1})$
```

tem key $K'$ is randomly chosen and then all components have to execute the installation procedure. Also a group key-agreement scheme based on the components' certificates can be used for a higher security level at higher computational costs [77].

**Key Initialization:** The key initialization can be performed in two ways: (1) there is a single component that founds the system to which new components are added step by step; or (2) the system is completely assembled and then all components are initialized at once. We now provide an example for the latter scenario. After the assembly of all parts, e.g., after a car is completely assembled a security engineer inserts a smart card to start the initialization. The smart card generates a random vehicle key $K$ and distributes this key to all installed components. There is a smart card for each car such that each car is initialized with another key. Alternatively, the HSM or a crucial component like the engine generate a random key at first ignition and distribute this key to all car components via the bus.

**Key Hierarchies:** In some applications there will be several groups of different interest and authorizations. There is the owner, the mechanics of independent and authorized garages, the suppliers of components, and the car manufacturer. The different levels of authorization can be implemented by the use of different keys to enable different levels of rights. Key hierarchies provide methods to implement complex and interwoven relationships in large systems. For instance, each owner is provided with a smart card that allows him to replace parts of the car that are not relevant for safety. Whenever the owner wants to install or dismount a component he has to attach the smart card to the car. Whenever the owner gives the car to a garage for fixing it, the mechanics use another smart card for replacing components including safety relevant ones. Clearly, there must be no master-card with omnipotent authorization. Even when using the smart card with the highest authorization level, the car performs all security checks and raises an alarm if necessary.

**Subgroups:** We could divide all components into subgroups providing different levels of security. Each subgroup holds its own system key. The higher the security level and safety relevance of a group, the better the protection of the assigned key, for example implemented by tamper resistant features. If a key is compromised, only the key of one subgroup is compromised. Each subgroup performs the system check independent of the other subgroups. Communication among subgroups requires additional keys that are provided at initialization time.

## 7.6 Security

We based the security of our scheme on the tamper resistance of the HSM as well as the security tags. If these assumptions hold, our system is obviously secure since all involved parties play fair. However, what happens if a key is compromised?

If an adversary Mallory compromises the vehicle key $K$, he could equip counterfeits with $K$ and install them into the car. However, this attack will only work if the counterfeits are listed on the list of built-in components $\mathcal{UL}$ that is stored by the HSM or distributed to all components. Hence, Mallory has to compromise the HSM which is assumed to be especially protected. In a distributed solution, Mallory could compromise a component and force a broadcast of the modified $\mathcal{UL}$ list. However, the broadcast will only be accepted by the remaining components after an installation or demounting procedure, and the new $\mathcal{UL}$ must origin from the component that was randomly selected as verifier for the installation or demounting procedure. If the random selection is based on commitments, Mallory can only add a new component to $\mathcal{UL}$ if his compromised component is chosen as verifier.

If Mallory is able to gain knowledge of a certificate $\langle PK, ID\rangle$ as well as the secret key $SK$, he equips counterfeits with this certificate. He then installs his counterfeits in a car to identify successfully. However, Mallory could not use the same certificate twice for the same car since the HSM would detect a collision. Otherwise, this attack is basically a cloning attack. It can only be avoided when there is a global directory of all built-in lists $\mathcal{GL}$ available as described before.

Mallory might replace the public key of the certificate issuer $PK_{CA}$ with his own public key $PK_M$. Then he is able to issue certificates for his counterfeits.

These perform the proof of origin successfully. However, Mallory had to replace the public key of the HSM. In a distributed solution, he had to replace the public key of components. As described before, Mallory cannot influence which component acts as verifier though.

Obviously, if $SK_{CA}$ is compromised, the security of the entire system is endangered. However, there are several techniques known how to securely store the secret key of the certificate authority.

## 7.7 Conclusions

We presented a scheme for implementing component identification in the automotive context based on cryptographic authentication schemes. We first presented a solution using a central high security module, and then we distributed the central task to security tags. Furthermore, we presented an approach of using symmetric cryptographic methods only that can even be used with passive RFID tags attached to the components. We introduced several features such as key hierarchies and policies to make our system more flexible.

Our solution is mainly based on the tamper resistance of special security hardware modules. In cases where the component has a mechanic main function, e.g. an engine, it is to be researched how secure hardware modules can be embedded into the component. If the component has mainly an electronic function such as a dashboard instrument, the security tag can take over the role of the component. In such cases, trusted computing mechanisms are able to provide tamper resistance. Since more and more parts of a car depend on electronics, using such a solution will provide the required means for our component identification scheme.

# 8 Summary and Outlook

In this thesis we analyzed and proposed authentication schemes for ad-hoc and sensor networks. We started by giving an overview of existing schemes and classified these. We then considered signature schemes and analyzed how well they are suited to ad-hoc networks. We argued that in ad-hoc and especially in sensor networks, a local neighborhood approach is always more efficient than a global approach. In a local neighborhood approach, nodes establish relationships to their direct neighbors in order to fulfil their task whereas in a global approach there are central directories such that nodes can securely communicate with each other node. The local neighborhood approach is provided by pairwise authentication schemes. The local approach improves efficiency substantially compared to a global approach. The global approach is supported by broadcast authentication schemes such as provided by digital signatures. However, the local approach limits the capability of protocols. We proposed two new extremely efficient pairwise authentication schemes that benefit of the local approach's efficiency. The first scheme is especially suited to sensor networks whereas the second protocol is more powerful but also requires more computational power as well as infrastructure such that it is appropriate for ad-hoc networks. Finally, we presented a scheme for providing component identification based on ad-hoc mechanisms. Such component identification provides systems with theft protection, protection against counterfeits, as well as protection against manipulation. Our component identification can be used for complex systems such as cars or airplanes.

At the end of this thesis, we point out some thoughts and open problems for future research:

- We believe that it should be a main goal of future authentication protocols to provide asymmetric functionality with symmetric methods. Furthermore, authentication protocols for ad-hoc networks should also be provably secure. We believe that our ZCK recognition and IC authentication protocol which are extremely efficient and also provably secure are a good start in this direction.

- It seems today that all authentication protocols are limited by their requirements regarding computational efficiency and especially supporting infrastructure. It would be valuable to have a more formal analysis about these limitations, i.e., to obtain a maximum set of scenarios that are supported by today's protocols, and a maximum set of scenarios that can theoretically be supported.

- Regarding the limitations of today's protocols, it would be very valuable to design and implement a protocol that works in the following environment: the nodes are highly mobile and have computational resources to perform public-key operations once in a while. However, once the devices are deployed there is no supporting infrastructure available anymore besides the devices itself. As this scenario fits many real-world applications a proof of concept would be important.

# Bibliography

[1] R. Anderson. Protecting embedded systems — the next ten years. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Workshop on Cryptographic Hardware and Embedded Systems — CHES 2001*, volume LNCS 2162, pages 1–2. Springer-Verlag, 2001. Invited Talk.

[2] R. Anderson, F. Bergadano, B. Crispo, J.-H. Lee, C. Manifavas, and R. Needham. A new family of authentication protocols. In *ACM Operating Systems Review*, 1998.

[3] R. Anderson and M. Kuhn. Tamper Resistance - a Cautionary Note. In *Second Usenix Workshop on Electronic Commerce*, pages 1–11, November 1996.

[4] N. Asokan and P. Ginzboorg. Key agreement in ad-hoc networks. *Computer Communications*, 23, 2000.

[5] Association for Automatic Identification and Mobility. Webpage, 2004. `http://www.rfid.org`.

[6] J. Brandt, I. D. rd, P. Landrock, and T. Pedersen. Zero-Knowledge Authentication Scheme with Secret Key Exchange. *Journal of Cryptology*, 1998.

[7] S. Buchegger and J.-Y. L. Boudec. Performance analysis of the CONFIDANT protocol: Cooperation of nodes — fairness in dynamic ad-hoc networks. In *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*. IEEE, June 2002.

[8] C. Buschmann, S. Fischer, N. Luttenberger, and F. Reuter. Middleware for swarm-like collections of devices. *IEEE Pervasive Computing Magazine*, 2(4), 2003.

[9] Business Week, W. Stern. Warning! Bogus parts have turned up commercial jets. Where's the FAA?, June 1996.

[10] L. Buttyán and J.-P. Hubaux. Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks. Technical report dsc/2001/001, Swiss Federal Institute of Technology – Lausanne, Department of Communication Systems, 2001.

[11] R. Canetti, J. Garay, G. Itkis, D. Miccianicio, M. Naor, and B. Pinkas. Multicast security: A taxonomie and some efficient constructions. In *Proceedings of IEEE INFOCOM '99*, New York, USA, March 1999.

[12] S. Capkun and J.-P. Hubaux. Biss: building secure routing out of an incomplete set of security associations. In *Proceedings of the 2003 ACM workshop on Wireless security*, pages 21–29. ACM Press, 2003.

[13] D. W. Carman, P. S. Kruus, and B. J. Matt. Constraints and Approaches for Distributed Sensor Network Security. Technical Report 00-010, NAI Labs, 2000.

[14] Certicom Corp. Certicom ECC Challenge, since 1997. Available at `http://www.certicom.com/resources/ecc\_chall/challenges.html`.

[15] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2003.

[16] D. Coppersmith and M. Jakobsson. Almost optimal hash sequence traversal. In *Proceedings of the Fourth Conference on Financial Cryptography (FC '02)*. Springer-Verlag, 2002.

[17] Crossbow. Webpage, 2004. `http://www.xbow.com`.

[18] Daimler Chrysler. Ausbausicherung für elektronische Komponenten bei einem Fahrzeug. Offenbarungsschrift, DE 100 21 811 A1, November 2001.

[19] J. Deng, R. Han, and S. Mishra. Security support for in-network processing in wireless sensor networks. In *Proceedings of 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'03)*, pages 83–93, October 2003.

[20] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Advances in Cryptology – Crypto '89*, volume LNCS 435. Springer-Verlag, 1990.

[21] J. R. Douceur. The sybil attack. In *First International Workshop on Peer-to-Peer Systems*, pages 251 – 260, 2002.

[22] L. Eschenauer and V. Gligor. A key management scheme for distributed sensor networks. In *Proceedings on ACM CCS 2002*, 2002.

[23] D. Estrin, R. Govindan, and J. Heidemann. Embedding the Internet. *Communications of the ACM*, 43(5):39–41, May 2000.

[24] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9:35–67, 1996.

[25] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings on Advances in Cryptology–CRYPTO '86*. Springer-Verlag, 1986.

[26] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of RC4. In *Selected Areas in Cryptography (SAC) 2001*, volume LNCS 2259, 2001.

[27] S. R. Fluhrer and S. Lucks. Analysis of the E0 Encryption System. In *Proceedings of the 2001 ACM Symposium on Applied Computing (SAC)*, 2001.

[28] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In *Advances in Cryptology – EUROCRYPT '96*, volume LNCS 1070, Berlin, Germany, 1996. Springer-Verlag.

[29] C. Gentry, J. Jonsson, J. Stern, and M. Szydlo. Cryptanalysis of the NTRU Signature Scheme (NSS). In *Advances in Cryptology – EUROCRYPT '01*. Springer-Verlag, 2001.

[30] P. Golle and N. Modadugu. Authenticating streamed data in the presence of random packet loss. In *ISOC Network and Distributed System Security Symposium (2001)*, pages 13–22, 2001.

[31] L. Gong. Variations on the themes of message freshness and replay. In *Proceedings of the IEEE Computer Security Foundations Workshop VI*, pages 131–136, Franconia, New Hampshire, June 1993.

[32] V. Gupta, S. Gupta, and D. Stebila. Performance analysis of elliptic curve cryptography for ssl. In *Proceedings of the 2002 ACM Workshop on Wireless Security*, 2002.

[33] J. Hammell, A. Weimerskirch, J. Girao, and D. Westhoff. Recognition in a low-power environment. In *Proceedings of the 2nd International Workshop on Wireless Ad Hoc Networking (WWAN 2005)*, 2005.

[34] HiperLAN2 Global Forum. Webpage, 2004. `http://www.hiperlan2.com`.

[35] A. Hodjat and I. Verbauwhede. The energy cost of secrets in ad-hoc networks. In *IEEE Circuits and Systems workshop on wireless communications and networking*, September 2002.

[36] Y.-C. Hu, D. B. Johnson, and A. Perrig. Sead: Secure efficient distance vector routing in mobile wireless ad hoc networks. In *Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '02)*, pages 3–13, June 2002.

[37] Y.-C. Hu, A. Perrig, and D. Johnson. Efficient security mechanisms for routing protocols. In *Proceedings of the Tenth Annual Network and Distributed System Security Symposium (NDSS 2003)*, 2003.

[38] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In *Eighth Annual International Conference on Mobile Computing and Networking (MobiCom 2002)*, September 2002.

[39] J.-P. Hubaux, L. Buttyán, and S. Čapkun. The quest for security in mobile ad hoc networks. In *ACM Symposium on Mobile Ad Hoc Networking and Computing – MobiHOC 2001*, 2001.

[40] J.-P. Hubaux, S. Čapkun, and J.Luo. Security and Privacy of Smart Vehicles. *to appear in IEEE Security & Privacy*, 2004.

[41] IEEE Computer Society LAN/MAN Standards Committee. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - IEEE 802.11, 1997.

[42] IHK Stuttgart. Wirtschaftsrecht: IHK setzt sich aktiv für die Bekämpfung von Produkt- und Markenpiraterie ein - Jährlich 250 Milliarden

Euro Schaden durch Produktfälschung weltweit. World Wide Web, 2004. http://www.stuttgart.ihk24.de.

[43] J. Jubin and J. Tornow. The DARPA Packet Radio Network Protocols. *Proceedings of IEEE (Special Issue on Packet Radio Networks)*, 75:21–32, January 1987.

[44] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *International Conference on Network Protocols (ICNP)*, pages 251–260, 2001.

[45] S. Kumar, M. Girimondo, A. Weimerskirch, C. Paar, A. Patel, and A. Wander. Embedded End-to-End Wireless Security with ECDH Key Exchange. In *IEEE Midwest Symposium on Circuits and Systems*, December 2003.

[46] B. Lamparter, C. Paar, A. Weimerskirch, and D. Westhoff. On digital signatures in ad hoc networks. *Wiley Journal European Transactions on Telecommunications*, September 2005. Special Issue on Self-Organization in Mobile Networking.

[47] B. Lamparter, K. Paul, and D. Westhoff. Charging support for ad hoc stub networks. *Journal of Computer Communication*, 26:1504–1514, August 2003. Special Issue on Internet Pricing and Charging: Algorithms, Technology and Applications.

[48] J. López and R. Dahab. Performance of elliptic curve cryptosystems. Technical Report IC-00-08, ICUniCamp, May 2000.

[49] S. Lucks, E. Zenner, A. Weimerskirch, and D. Westhoff. Efficient entity recognition for low-cost devices. *Technical Report*, 2004.

[50] H. Luo, P. Zefros, J. Kong, S. Lu, and L. Zhang. Self-securing ad hoc wireless networks. In *Seventh IEEE Symposium on Computers and Communications (ISCC '02)*, 2002.

[51] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, USA, September 2002.

[52] A. Menezes and D. Johnson. The elliptic curve digitial signature algorithm (ECDSA). Technical Report CORR 99-34, Department of C & O, University of Waterloo, Ontario, Canada, August 1999.

[53] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, Florida, USA, 1997.

[54] M.Fischlin. Fast verification of hash chains. In *RSA Security Cryptographer's Track 2004*, volume LNCS 2964, pages 339–352. Springer-Verlag, 2004.

[55] C. Mitchell. Remote user authentication using public information. In *Cryptography and Coding*, 2003.

[56] Mobile Ad-hoc Networks (MANET). Webpage, 2003. `http://www.ietf.org/html.charters/manet-charter.html\`.

[57] National Institute of Standard and Technology. Recommended elliptic curves for federal government use, May 1999. Available at `http://csrc.nist.gov/encryption`.

[58] National Institute of Standard and Technology. *Special Publication 800-57, Recommendation for Key Management.* Federal Information Processing Standards, National Bureau of Standards, U.S. Department of Commerce, January 2003.

[59] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the third international symposium on Information processing in sensor networks*, pages 259 – 268, 2004.

[60] C. Paar, J. Pelzl, K. Schramm, A. Weimerskirch, and T. Wollinger. Eingebettete Sicherheit: State-of-the-art. In *D-A-CH Security 2004*, March 2004.

[61] P. Papadimitrados and Z. Haas. Secure routing for mobile ad hoc networks. In *SCS Communication Networks and Distributed Systems Modelling and Simulation Conference (CNDS 2002)*, 2002.

[62] K. Paul and D. Westhoff. Context aware detection of selfish nodes in dsr based ad-hoc networks. In *IEEE GLOBECOM 2002*, 2002.

[63] A. Perrig, R. Canetti, D. Tygar, and D. Song. The TESLA Broadcast Authentication Protocol. Technical Report 2, RSA Laboratories, 2002.

[64] A. Perrig, R. Szewczyk, V. Wen, D. Cullar, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *Proceedings of MOBICOM 2001*, 2001.

[65] R. L. Pickholtz, D. L. Schilling, and L. B. Milstein. Theory of spread-spectrum communications - a tutorial. *IEEE Transactions on Communications*, pages 855–884, May 1982.

[66] N. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha. Analyzing the energy consumption of security protocols. In *IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, August 2003.

[67] I. Riedel. Security in ad-hoc networks: Protocols and elliptic curve cryptography on an embedded platform. Diploma Thesis, March 2003. Ruhr-University Bochum, Communication Security Group.

[68] R. Rivest. Can we eliminate revocation lists? In *Financial Cryptography – Second International Conference*, pages 178–183. Springer-Verlag, 1998.

[69] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[70] P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet. In *6th ACM Conference on Computer and Communication Security*, pages 93–100, November 1999.

[71] RSA Laboratories. Factoring Challenge. Available at `http://www.rsasecurity.com/rsalabs/node.asp?id=2093`.

[72] N. B. Salem, L. Buttyan, J.-P. Hubaux, and M. Jakobsson. A Charging and Rewarding Scheme for Packet Forwarding in Multi-hop Cellular Networks. In *ACM/SIGMOBILE 4th International Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, June 2003.

[73] F. Stajano. The resurrecting duckling – what next? In *The 8th International Workshop on Security Protocols*, volume LNCS 2133. Springer-Verlag, 2000.

[74] F. Stajano. *Security for Ubiquitous Computing.* John Wiley and Sons, Feb. 2002.

[75] F. Stajano and R. Anderson. The resurrecting duckling: Security issues in ad-hoc wireless networks. In *The 7th International Workshop on Security Protocols*, volume LNCS 1796, Cambridge, UK, April 19–21 1999. Springer-Verlag.

[76] Standards for Efficient Cryptography Group. SEC2: Recommended Elliptic Curve Domain Parameters. Working draft, version 0.7, SECG, September 2000.

[77] M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: A New Approach to Group Key Agreement. In *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS'98)*, pages 380–387, Amsterdam, 1998. IEEE Computer Society Press.

[78] M. Szydlo. Hypercubic Lattice Reduction and Analysis of GGH and NTRUSign Signatures. In *Advances in Cryptology – EUROCRYPT '03*. Springer-Verlag, 2003.

[79] The Network Simulator NS-2, 2003. Available: http://www.ietf.org/html.charters/manet-charter.html.

[80] TinyOS Project. TinyOS: a component-based OS for the networked sensor regime. Available at `http://webs.cs.berkeley.edu/tos/download.html`.

[81] U.S. Department of Commerce/National Institute of Standard and Technology. *Digital Signature Standard (DSS)*, January 27, 2000. Available at `http://csrc.nist.gov/encryption`.

[82] Verpackungsrundschau. Webpage, 2004. `http://www.verpackungsrundschau.de`.

[83] Volkswagen AG. Kraftfahrzeug mit einer Vielzahl von Bauteilen. Offenbarungsschrift, DE 100 01 986 A1, July 2001.

[84] B. Warneke, M. Last, B. Leibowitz, and K. Pister. Smart dust: Communicating with a cubic-millimeter computer. *IEEE Computer Magazine*, pages 44–51, January 2001.

[85] A. Weimerskirch. Authentikation in Ad-hoc und Sensor Netwerken. In *GUUG Frühjahrsfachtagung 2004*, March 2004.

[86] A. Weimerskirch, K. Hoeper, C. Paar, and M. Wolf. Component identification: Enabler for secure networks of complex systems. In *Proceedings of Applied Cryptography an Network Security 2005 (ACNS 2005)*, 2005.

[87] A. Weimerskirch, C. Paar, and S. C. Shantz. Elliptic Curve Cryptography on a Palm OS Device. In V. Varadharajan and Y. Mu, editors, *The 6th Australasian Conference on Information Security and Privacy — ACISP 2001*, volume LNCS 2119, pages 502–513, Berlin, 2001. Springer-Verlag.

[88] A. Weimerskirch, C. Paar, and M. Wolf. Cryptographic component identification, enabler for secure inter-vehicular networks. In *Proceedings of the IEEE 62nd Semiannual Vehicular Technology Conference (VTC 2005 Fall)*. IEEE Computer Society Press, 2005.

[89] A. Weimerskirch and D. Westhoff. Identity certified authentication for ad-hoc networks. In *ACM Workshop on Security of Ad Hoc and Sensor Networks in conjunction with the Tenth ACM SIGSAC Conference on Computer and Communications Security (ACM SASN'03)*, October 2003.

[90] A. Weimerskirch and D. Westhoff. Zero Common-Knowledge Authentication for Pervasive Networks. In *Selected Areas in Cryptography - SAC, 2003*, 2003.

[91] A. Weimerskirch, D. Westhoff, S. Lucks, and E. Zenner. Efficient Pairwise Authentication Protocols for Sensor and Ad-hoc Networks. *Sensor Network Operations*, 2005.

[92] M. Wiener. Performance comparison of public-key cryptosystems. Technical report, RSA Laboratories, 1998.

[93] C. Wong and S. Lam. Digital signatures for flow and multicasts. In *Proceedings of IEEE ICNP'98*, October 1998.

[94] M. G. Zapata. Secure ad hoc on-demand distance vector (saodv) routing. In *INTERNET DRAFT*, 2001.

[95] S. Zhong, Y. R. Yang, and J. Chen. Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-hoc Networks. In *Proceedings of IEEE INFOCOM '03*, March 2003.

[96] L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.

# Curriculum Vitae

**André Weimerskirch**
aweimerskirch@web.de

## Education

*2001-2004* *Ruhr-University Bochum, Germany*
*Ph.D. (Dr.-Ing.) in Computer Science*
Thesis: Authentication in Ad-hoc and Sensor Networks
Advisor: Prof. Christof Paar

*1999-2001* *Worcester Polytechnic Institute, MA, USA*
*MS in Computer Science*
Thesis: The Application of the Mordell-Weil Group to Cryptographic Systems
Advisor: Prof. Christof Paar

*1995-1999* *Darmstadt University of Technology, Germany*
*Bachelor (Vordiplom) in Computer Science/Business Administration*
*Bachelor (Vordiplom) in Mathematics*

## Experience

*since 2004* *escrypt GmbH, Bochum, Germany*
*CTO*

*2001 - 2004* *Ruhr-University Bochum, Germany*
*Research Fellow*

*2003* *Aarhus University, Denmark*
*Researcher, Marie Curie European Union fellowship*

| | |
|---|---|
| *2002* | *Sun Microsystems Laboratories, Mountain View, CA, USA*<br>*Researcher, internship* |
| *2002* | *Josteit, Herten & Partner, Dusseldorf, Germany*<br>*IT-Consultant, part-time position* |
| *2001* | *Accenture Technology Labs, Sophia-Antipolis, France*<br>*Researcher, internship* |
| *2000 - 2001* | *Worcester Polytechnic Institute, MA, USA*<br>*Research Assistant* |
| *2000* | *Philips Research, Briarcliff Manor, NY, USA*<br>*Researcher, internship* |
| *1997 - 1999* | *Deutsche Post AG, Darmstadt, Germany*<br>*Web developer, part-time position* |

## Awards

- European Union Marie Curie Fellowship, 2003
- e-fellows.net fellowship 2001–2004
- Scholarship of the German Academic Exchange Service (DAAD), 1999–2000
- Best Pre-Diploma out of 77 students at Darmstadt University of Technology in the year 1997/98.

## Publications

### Journal Papers

- Bernd Lamparter, Christof Paar, André Weimerskirch, and Dirk Westhoff, "On Digital Signatures in Ad Hoc Networks", Wiley Journal European Transactions on Telecommunications, Special Issue on Self-Organization in Mobile Networking, September 2005.

### Book Chapters

- André Weimerskirch, Dirk Westhoff, Stefan Lucks, and Erik Zenner, "Efficient Pairwise Authentication Protocols for Sensor and Ad-hoc Networks", Sensor Network Operations, IEEE Press, 2004.

- André Weimerskirch, "Fixed-base exponentiation", Encyclopedia of Cryptography and Security, 2004.

- André Weimerskirch, "Fixed-exponent exponentiation", Encyclopedia of Cryptography and Security, 2004.

- André Weimerskirch, "Karatsuba algorithm", Encyclopedia of Cryptography and Security, 2004.

### Conference Papers

- André Weimerskirch, Christof Paar, and Marko Wolf, "Cryptographic Component Identification: Enabler for Secure Inter-vehicular Networks", 62nd IEEE Vehicular Technology Conference, September 25-28, 2005, Dallas, TX, USA.

- André Weimerskirch, Katrin Höper, Christof Paar, and Marko Wolf, "Component Identification: Enabler for Secure Networks of Complex Systems", Applied Cryptography and Network Security (ACNS) 2005, June 7-10, 2005, New York City, NY, USA.

- Marko Wolf, André Weimerskirch, and Christof Paar, "Digital Rights Management als Enabling Technology im Automobil", Sicherheit 2005: Sicherheit - Schutz und Zuverlässigkeit, Regensburg, April 5-8, 2005.

- Jonathan Hammell, André Weimerskirch, Joao Girao, and Dirk Westhoff, "Recognition in a Low-Power Environment", WWAN 2005, The 25th IEEE International Conference on Distributed Computing Systems (ICDCS-2005), Columbus, Ohio, USA, June 6-9, 2005.

- Ulrich Kaiser, Christof Paar, Dörte Rappe, Werner Schindler, André Weimerskirch, and Thomas Wollinger, "Kriterien für die Auswahl der kryptographischen Algorithmen bei Low-Cost-RFID-Systemen", D-A-CH Security 2005, Darmstadt University of Technology, 2005.

- Marko Wolf, André Weimerskirch, and Christof Paar, "Security in Automotive Bus Systems", escar 2004 - Embedded Security in Cars Workshop, Bochum, 10.-11. November, 2004.

- André Weimerskirch, Marko Wolf, and Christof Paar, "Komponentenidentifikation: Voraussetzung für IT-Sicherheit im Automobil", Automotive - Safety & Security 2004, Stuttgart, 6.-7. October, 2004.

- Marko Wolf, André Weimerskirch, and Christof Paar, "Sicherheit in automobilen Bussystemen", Automotive - Safety & Security 2004, Stuttgart, 6.-7. October, 2004.

- André Weimerskirch, "Authentikation in Ad-hoc und Sensornetzwerken", GUUG-Frühjahrsfachgespräch 2004, Ruhr-Universität Bochum, 9.-12. März, 2004.

- Christof Paar, Jan Pelzl, Kai Schramm, André Weimerskirch and Thomas Wollinger, "Eingebettete Sicherheit: State-of-the-art", D-A-CH Security 2004, University of Basel, March 30-31, 2004.

- Sandeep Kumar, Marco Girimondo, André Weimerskirch, Christof Paar, Arun Patel, and Arvinderpal S.Wander, "Embedded End-to-End Wireless Security with ECDH Key Exchange", 46th IEEE Midwest Symposium On Circuits and Systems, December 27-30, 2003, Cairo, Egypt.

- André Weimerskirch and Dirk Westhoff, "Identity Certified Authentication for Ad-hoc Networks", 2003 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03), October 31, 2003.

- André Weimerskirch and Dirk Westhoff, "Zero Common-Knowledge Authentication for Pervasive Networks", Selected Areas in Cryptography - SAC, August 14-15, 2003.

- André Weimerskirch, Douglas Stebila, and Sheueling Chang Shantz, "Generic $GF(2^m)$ Arithmetic in Software and its Application to ECC", The Eighth Australasian Conference on Information Security and Privacy (ACISP 2003), 9-11 July 2003, Wollongong, Australia.

- Olivier Pelletier, André Weimerskirch, "Algorithmic Self-Assembly of DNA Tiles and its Application to Cryptanalysis", The Genetic and Evoluationary Computation Conference 2002 (GECCO 2002), July 9-13, 2002, New York City, USA.

- André Weimerskirch and Gilles Thonet, "A Distributed Light-Weight Authentication Model for Ad-hoc Networks", The 4th International Conference on Information Security and Cryptology (ICISC 2001), December 6-7, 2001, Seoul, South Korea.

- André Weimerskirch, Christof Paar, and Sheueling Chang Shantz, "Elliptic Curve Cryptography on a Palm OS Device", The 6th Australasian Conference on Information Security and Privacy (ACISP 2001), July 11-13 2001, Sydney, Australia.

**Technical Reports**

- Stefan Lucks, Erik Zenner, André Weimerskirch, and Dirk Westhoff, "Efficient Entity Recognition for Low-cost Devices", 2004.

- André Weimerskirch and Christof Paar, "Generalizations of the Karatsuba Algorithm for Efficient Implementations", 2003.

**Patents**

- André Weimerskirch, "Method and apparatus for preventing noise from influencing a random number generator based on flip-flop metastability", United States Patent Application 20030101205.

- André Weimerskirch, "Method and apparatus to prevent the unauthorized copying of digital information", United States Patent Application 20030088775.

- André Weimerskirch, "Apparatus and methods for attacking a screening algorithm based on partitioning of content", United States Patent Application 20020152172.

# Index

$\mu$TESLA, 45
802.11, 6, 44

ad-hoc, 5
ad-hoc network, 5
adversarial model, 18, 72, 80
AODV, 41
ARIADNE, 41
asymmetric authentication, 24
asymmetric component identification, 112
asymmetric MAC authentication, 29
authentication
- asymmetric, 24
- asymmetric MAC, 29
- broadcast, 28
- data origin, 23
- entity, 23
- Guy Fawkes, 31
- hybrid, 26
- message, 23
- mutual, 28
- overview, 33
- remote user, 32
- symmetric, 24
- TESLA, 30
- time-stamp, 26

authentication models, 46

BISS, 41
Bluetooth, 44
broadcast authentication, 28

challenge-response
- identification, 114
- symmetric, 114

charging
- SCP, 43
- SPRITE, 43

component identification, 107
- asymmetric, 112
- distributed, 120
- symmetric, 123

CONFIDANT, 42
cooperation
- CONFIDANT, 42
- nuglets, 42

data origin authentication, 23
device model, 18
digital signatures, 51
distributed CA, 37
distributed component identification, 120
distributed PKI, 37
DSR, 41

ECC, 39, 54
ECDSA, 54
elliptic curve cryptography (ECC), 39, 54

embedded cryptography, 39
embedded devices, 11
embedded systems, 11, 39
energy consumption, 88
energy model, 89
entity authentication, 23
entity identification, 24
entity recognition, 70

Fiat-Shamir identification, 26

Guy Fawkes, 31

hybrid authentication, 26

IC, 79
- energy map, 93
- protocol, 83

identification, 24
identity accumulation, 70
identity certified (IC) authentication, 79

key agreement
- password-based, 35
- resurrecting duckling, 36

key distribution, 35
key pre-distribution, 38

Lamport's one-time passwords, 30

message authentication, 23
message recognition, 70
MICA Motes, 45
model
- adversarial, 18, 72, 80
- authentication, 46
- device, 18
- energy, 89
- network, 17, 72, 80

mutual authentication, 28

network
- ad-hoc, 5
- multi-hop, 6
- sensor, 9
- single-hop, 6

network model, 17, 72, 80
network topology, 62

password-based key agreement, 35
PGP, 44
PKI
- distributed, 37
- distributed CA, 37
- self-organized, 37

random key-ring, 38
recognition non-repudiation, 71
recognition protocols, 73
remote user authentication, 32
resurrecting duckling, 36
routing
- AODV, 41
- ARIADNE, 41
- BISS, 41
- DSR, 41
- SEAD, 41

RSA, 39, 54

SCP, 43
SEAD, 41
secure routing, 40
security relationships, 64
self-organized PKI, 37
sensor network, 9
SNEP, 45

SPINS, 45
SPRITE, 43
symmetric authentication, 24
symmetric component identification, 123

TESLA, 30
time-stamp authentication, 26
TinyOS, 45

WEP, 45
WLAN, 44

ZCK
 energy map, 91
 protocol, 74
 recognition, 69
zero common-knowledge (ZCK), 74
zero-knowledge proof, 26