DISSERTATION

Secure Communication in Home and Building Automation Systems

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

ao. Univ.-Prof. Dipl.-Ing. Dr. Wolfgang Kastner
Institut für Rechnergestützte Automation
Arbeitsgruppe Automatisierungssysteme

und

Prof. Gianluca Cena, Ph.D.
Italian National Research Council (CNR)
Istituto di Elettronica e di Ingegneria dell'Informazione
e delle Telecomunicazioni
Italy

eingereicht an der

Technischen Universität Wien
Fakultät für Informatik

von

Mag. Dipl.-Ing. Wolfgang Granzer
Mat.Nr.: 0026013

Neubaugasse 6/2
3363 Neufurth

Wien, 15.02.2010

Wolfgang Granzer

Digitally signed by Wolfgang Granzer
DN: cn=Wolfgang Granzer, o=E183-1, ou=TU Vienna, email=wolfgang@granzer.org, c=AT
Date: 2010.02.15 10:51:20 +01'00'

Diese Arbeit ist meinem Großvater Theodor Granzer gewidmet.

Kurzfassung

Systeme der Heim- und Gebäudeautomation (HGA) beschäftigen sich traditionell mit der automatischen Steuerung von Heizungs-, Lüftungs- und Klimatechniksystemen (HLK) sowie mit Gerätschaften aus der Beleuchtungs- und Beschattungstechnik. Dienste aus dem Bereich Safety (d.h. Betriebssicherheit) und Security (d.h. Zugriffsschutz und Zugriffssicherheit) werden fast ausschließlich durch getrennte, anwendungsspezifische Subsysteme realisiert. Eine Integration mit dem zentralen HGA System wird (wenn überhaupt) auf der Managementebene durchgeführt.

Heutzutage ist es wünschenswert, eine Integration von security-kritischen Systemen bereits auf der Feldebene durchzuführen. Diese Erweiterung des Anwendungsbereichs von HGA Systemen bedingt allerdings, dass das darunter liegende Kommunikationssystem zuverlässig und robust gegen böswillige Manipulationen agiert. Eine Analyse von bestehenden Technologien zeigt, dass diese den zusätzlichen Anforderungen nicht gewachsen sind. Der Hauptgrund liegt darin, dass diese Systeme zu einer Zeit entwickelt wurden, in der Security bestenfalls ein Randthema war. Daher vertrauen diese Systeme auf physikalische Isolation und folgen zumeist dem Leitsatz „Sicherheit durch Verschleierung" (Security by Obscurity). Dies ist jedoch für moderne HGA Systeme inakzeptabel, da die Verhinderung des physikalischen Zugriffs zum Netzwerk durch Isolation nicht immer durchführbar ist (z.B. WLANs) und „Security by Obscurity" eine Technik darstellt, die (wenn überhaupt) nur einen zeitlich begrenzten Schutz bietet. Daher ist die Entwicklung eines allumfassenden Security-Konzepts von größter Wichtigkeit für die zukünftige Weiterentwicklung von HGA Systemen.

Diese Dissertation konzentriert sich auf das Bereitstellen von Mechanismen für eine (bezüglich Security) sichere Kommunikation in HGA Netzwerken, d.h., eine Verhinderung von Netzwerkattacken. Basierend auf einer Security-Analyse werden die Anforderungen und Herausforderungen identifiziert. Nach einem Überblick über den Stand der Technik, wird ein generischer Ansatz für eine sichere Kommunikation in HGA Systemen entworfen. Dieser Ansatz basiert auf sicheren Kommunikationsbeziehungen – Kommunikationsentitäten wie Geräte oder Kontrollanwendungen können auf sichere Art und Weise an Kommunikationsbeziehungen teilhaben bzw. diese wieder verlassen. Zusätzlich wird ein Framework präsentiert, welches aufbauend auf einem multi-protocol Stack diesen sicheren Ansatz implementiert. Um die Realisierbarkeit zu belegen, wird einerseits das vorgestellte Sicherheitskonzept einer formalen Evaluation unterzogen und andererseits eine prototypische Implementierung durchgeführt.

Abstract

Home and Building Automation (HBA) systems are traditionally concerned with the control of heating, ventilation, air conditioning, as well as lighting and shading systems. Services from the safety and security domain are typically provided by separated, application specific subsystems. An integration with the core HBA systems is done (if at all) at the management level.

Nowadays, the rising desire to integrate security-critical services even at the field level can be observed. The extension of the application domain of HBA systems therefore demands the underlying communication system to be reliable and robust against malicious manipulations. An analysis of existing technologies, however, exposes that they do not fulfill the additional requirements yet. The main reason is that the systems were developed at a time when security was considered as a side-issue at best. Hence, these systems rely on physical isolation and "Security by Obscurity". This is obviously unacceptable within modern HBA systems since preventing physical access to the network by isolation is not always possible (e.g., WLANs) and "Security by Obscurity" is a technique that (if at all) provides only temporary protection. Thus, the development of a comprehensive security concept is of utmost importance.

This dissertation is focused on providing mechanisms for secure communication in HBA networks thus counteracting network attacks. Based on a security threat analysis, requirements and challenges for secure communication are identified. After an overview of state of the art technologies, a generic approach for securing communication in HBA networks is introduced. This approach uses the concept of secure communication relationships where communication entities like devices or control applications are able to securely join and leave these relationships. Additionally, a framework that implements this security approach based on a multi-protocol stack is described. To prove the feasibility, the proposed security concept is formally evaluated and a prototype implementation is presented.

Danksagung

Zuerst möchte ich mich bei meinem Doktorvater Wolfgang Kastner bedanken. Danke dafür, dass Du mich in die Welt der Wissenschaft eingeführt hast – ohne deine Unterstützung wäre diese Arbeit nicht denkbar gewesen. Wenn ich auf die letzten Jahre zurückblicke, sehe ich in Dir nicht nur meinen Chef und Mentor, sondern ich denke auch an die Freundschaft, die sich über die beruflichen Aktivitäten hinaus entwickelt hat. Danke für Alles!

I would also like to thank Gianluca Cena for acting as second supervisor. Thank you for your comments and valuable ideas that helped me to improve this dissertation. I also appreciate your support especially during the stressful finishing phase.

Weiters möchte ich meinen Kollegen für das exzellente Arbeitsklima am Institut danken. Besonderer Dank gilt hier dem „HBA Core Team", allen voran Fritz Praus, Christian Reinisch und Georg Neugschwandtner. Während der Zeit am Institut lernte ich vor allem die gemeinsamen Diskussionsrunden (sowohl innerhalb als auch außerhalb der Institutsräumlichkeiten) sehr zu schätzen. Ein besonderer Dank gilt dem „Herz der Abteilung" Ruth Fochtner. Außerdem möchte ich mich bei Felix Schuster für seine oft nächtelange Unterstützung in Sachen Systemadministration bedanken. Des weiteren bedanke ich mich bei Daniel Lechner, der mit seiner Diplomarbeit wesentlich am Projekt und an den daraus entstandenen Ergebnissen beteiligt war.

Ein besonderer Dank gilt auch meinen Kollegen und Freunden Albert, Georg, Patrick und Thilo vom Institut für Integrierte Sensorsysteme der Österreichischen Akademie der Wissenschaften. Obwohl meine Zeit an der Akademie nur kurz war, freut es mich besonders, dass wir auch weiterhin sowohl beruflichen als auch privaten Kontakt pflegen.

Der wohl größte Dank gilt meiner Familie, allen voran meiner Frau Michaela. Danke, dass Du mich immer in allen Belangen unterstützt und nie an mir gezweifelt hast. Danke auch dafür, dass Du meine Launen während der letzten heißen Phase nicht nur ertragen, sondern für uns beide erträglich gemacht hast. Meiner Mutter, meinem Vater sowie meinen Großeltern möchte ich für die jahrelange Hilfe und Unterstützung danken.

This work was funded by FWF (Österreichischer Fonds zur Förderung der Wissenschaftlichen Forschung; Austrian Science Foundation) under the project P19673.

Contents

List of Figures

Acronyms

3DES Triple Data Encryption Standard

ACL Access Control List

AES Advanced Encryption Standard

AH Authentication Header

AO Application Object

API Application Programming Interface

ASHRAE American Society of Heating, Refrigerating and Air-Conditioning Engineers

ASL Application Specific Layer

BA Building Automation

BACnet/WS BACnet Web Services

BMS Building Management System

CA Certification Authority

CBC Cipher-Block Chaining

CCM Counter with CBC-MAC

CCTV Closed Circuit Television

CDR Configuration Data Record

CFB Cipher Feedback

COM Component Object Model

CRC Cyclic Redundancy Check

CTR Counter

DALI Digital Addressable Lighting Interface

DCOM Distributed Component Object Model

DDC Direct Digital Control

DES Data Encryption Standard

DH Diffie-Hellman

DLP Discrete Logarithm Problem

DMZ Demilitarized Zone

DoS Denial-of-Service

DoS-RF DoS Risk Factor

DSA Digital Signature Algorithm

ECB Electronic Codebook

ECC Elliptic Curve Cryptography

ECDLP Elliptic Curve Discrete Logarithm Problem

ECDSA Elliptic Curve Digital Signature Algorithm

ECIES Elliptic Curve Integrated Encryption Scheme

EHS European Home System

EIB European Installation Bus

ESP Encapsulating Security Payload

FTP File Transfer Protocol

GCDR Group Configuration Data Record

GCKS Group Controller Key Server

GDOI Group Domain of Interpretation

GSA Group Security Association

GSAKMP Group Secure Association Key Management Protocol

GSTS Group Security Token Set

HA Home Automation

HBA Home and Building Automation

HMAC Keyed-Hash Message Authentication Code

HTTP HyperText Transfer Protocol

HVAC Heating, Ventilation and Air Conditioning

ICD Interconnection Device

ICDR Initial Configuration Data Record

IDS Intrusion Detection System

IETF Internet Engineering Task Force

IFP Integer Factorization Problem

IKE Internet Key Exchange

IKEv2 Internet Key Exchange Version 2

IP Internet Protocol

IPsec Internet Protocol Security

IPv4 Internet Protocol Version 4

IPv6 Internet Protocol Version 6

ISTS Initial Security Token Set

IT Information Technology

IV Initialization Vector

JCI N2 Johnson Controls (Metasys) N2

LAN Local Area Network

LNS LonWorks Network Services

MAC Message Authentication Code

M-Bus Meter-Bus

MD5 Message Digest Algorithm 5

MD Management Device

MIRACL Multiprecision Integer and Rational Arithmetic C Library

NCDR Network Configuration Data Record

NIST National Institute of Standards and Technology

NSL Network Specific Layer

NSTS Network Security Token Set

OAEP Optimal Asymmetric Encryption Padding

oBIX Open Building Information eXchange

OCB Offset Codebook Mode

OFB Output Feedback

OPC A&E OPC Alarms and Events

OPC DA OPC Data Access

OPC HDA OPC Historical Data Access

OPC UA OPC Unified Architecture

PDA Personal Digital Assistant

PIN Personal Identification Number

PKI Public Key Infrastructure

PLC Programmable Logic Controller

PL Powerline

PRBG Pseudo Random Bit Generator

QoS Quality of Service

RF Radio Frequency

ROM Read-Only Memory

RSA Rivest, Shamir, and Adleman

SAC Sensor, Actuator, and Controller

SAD Security Association Database

SAL Security Abstraction Layer

SA Security Association

SBT P1 Siemens Building Technologies (Landis) P1

SCADA Supervisory Control And Data Acquisition

SCDR Session Configuration Data Record

SEIB Secure EIB

SHA-1 Secure Hash Algorithm Family 1

SHA Secure Hash Algorithm

SIG Special Interest Group

SKKE Symmetric-Key Key Establishment

SNEP Secure Network Encryption Protocol

SOAP Simple Object Access Protocol

SPI Security Parameter Index

SSF Single Source FIFO

SSL Secure Socket Layer

SSTS Session Security Token Set

STS Security Token Set

TCP Transmission Control Protocol

TDMA Time Division Multiple Access

TESLA Timed Efficient Stream Loss-Tolerant Authentication

TLS Transport Layer Security

TP Twisted Pair

TVP Time Variant Parameter

UART Universal Asynchronous Receiver/Transmitter

UDP User Datagram Protocol

UML Unified Modeling Language

USB Universal Serial Bus

VDLL Virtual Data Link Layer

VPN Virtual Private Network

WAN Wide Area Network

WLAN Wireless Local Area Network

WS-I Web Services Interoperability

1 Introduction

Building Automation (BA) systems aim at improving control and management of mechanical and electrical systems in buildings – more generally, interaction among all kinds of devices typically found there. The main goal of a BA system is to provide increased comfort while keeping an efficient use of all available resources in mind. Thus, BA systems do not only reduce the overall operational costs for maintenance but also decrease energy consumption and mainly contribute to environmental protection.

The core application area of BA systems is environmental control handled by the traditional building services Heating, Ventilation and Air Conditioning (HVAC) and lighting/shading [1, 2]. Services from other domains are often provided by separated, application specific subsystems. This is especially true for services from the safety domain (e.g., fire alarm systems, social alarms) and security domain (e.g., intrusion alarm systems, access control) [3, 4]. A coupling with the core BA services is done, if at all, at the management level. Here, so called *Building Management Systems (BMSs)* in combination with *Supervisory Control And Data Acquisition (SCADA)* software solutions are used. These systems primarily provide support for management tasks like visualization, monitoring, trending, and alarm handling. To gain a global view of the entire system functionality, interfaces to the subsystems are integrated into the BMS. However, BMSs as well as the used SCADA implementations are far away from providing a fully fledged BA solution. These systems are geared towards providing read-only functionality – interacting with the process under control (e.g., changing an actuator value) is only possible in a very limited way. The main reason is that legal regulation often demands an absence of feedback between services from the safety/security domain and traditional services since the latter cannot meet the given requirements [5].

While this clear separation between traditional and application specific systems is still valid up to now, a tighter integration is desirable. The ultimate goal is to employ a single *all-in-one BA system* which is able to provide the necessary functionality to satisfy the requirements of all kinds of application domains. The promised benefits of all-in-one systems range from lower (life cycle) costs to increased and improved functionality. Using

a comprehensive system resting upon a single technology, it is possible to use equipment in multiple application domains in parallel [6]. Consider, for example, the possibility of sharing data originating from just one sensor. This will reduce, on the one hand, investment and maintenance costs. On the other hand, more complex control functionality can be realized thanks to additional sensor information and actuating variables from other application domains (sensor fusion). A typical example would be an access control system where building functionality is activated (e.g., lighting a corridor or activating elevators) according to the access rights of visiting persons. Furthermore, the information about the current amount of individuals inside a room can be used as additional input parameter for the HVAC system. Opting for one single technology will also facilitate the overall management of the BA system. In particular, configuration and maintenance of an all-in-one BA system will become significantly easier since a multitude of different management solutions can be replaced by a single one. Deployment and operating staff concerned with the BA system has to become familiar with a single technology rather than having to learn the characteristics of each application specific solution.

Extending the application domain of BA systems towards an all-in-one system increases the requirements on the used technology especially if services from the safety and security domain are integrated. While reliability and robustness of the underlying technology are of utmost importance for both domains, the specific requirements of both domains may differ significantly. Security measures protect the system against *intentional* actions that result in damage to the system and as a consequence *may* also harm people. Safety measures reduce the risk of unintentional system states that *do* cause harm people. However, the security of a system also has an impact on the system's safety, since security can be seen as a precondition for safety. Consider, for example, a fire alarm system. Violating the security of a fire alarm system may result in a malfunction that may also harm people. Therefore, a comprehensive security concept is a key prerequisite for all-in-one systems.

The advantages of secure BA systems are manifold. Integrating a security concept widens the application area since services from the security domain can also be served by an all-in-one system. However, incorporating a security concept is also advantageous for traditional services. While security is often not regarded as an issue for these systems, a comprehensive security concept will also protect HVAC and lighting/shading systems against vandalism acts. Consider, for example, a lighting system: while the breakdown of a simple lighting system in a building is not considered as critical, vandalism acts on the lighting system of a hospital (e.g., operating room) may have fatal consequences. Another

example would be the lighting system of large public buildings like a concert hall. A shutdown of the lighting system could lead to panic that may harm people. Obviously, vandalism acts on BA systems may not only violate the safety of people. A malfunction of traditional services may have massive economic impact, too. A company-wide shutdown of the lighting system can be easily compared to a successful attack on the company Web server. The situation is further aggravated in large-scale systems. Consider, for example, a security attack on a national power provider. A shutdown of the power grid may have an economic impact in the amount of billions of Euros.

In the Home Automation (HA) domain, the situation is similar to the one in BA. While energy efficiency and economic benefits are important in both domains, comfort and prestige are the key drivers for HA systems. Therefore, the control of multimedia devices (i.e., "brown goods") and housekeeping appliances (i.e., "white goods") are popular use cases for HA systems. From the technological point of view, a difference between BA and HA systems is the amount of devices that are involved. While the amount of devices in HA systems is significantly lower than in the BA domain, the complexity of HA systems should not be underestimated. Usability, plug-and-play support, and the continuous need for integrating new devices and functionality are of utmost importance in the HA domain. This is also true for services from the security domain. Up to now, security services like access control and intrusion alarm systems have been realized by dedicated subsystems. However, this separation between different services is inefficient especially in the HA domain. Due to the relatively small amount of devices, the separation of subsystems results in an overhead in terms of installation and maintenance costs. Therefore, it is reasonable to replace them by a single all-in-one system that covers the required application areas. To fulfill the demands required by security services, a protection against malicious manipulations will be an essential aspect of future smart homes.

In general, the history of Information Technology (IT) security can serve both as a good and bad example for the future of Home and Building Automation (HBA) security. Important Internet-related protocols were developed for a virtually closed user community. Consequently, security was neglected – services and applications that used these protocols remained unprotected against security attacks. Due to the ubiquitous use of the Internet today, unprotected services and applications became attractive for adversaries. The resulting economic damage through viruses, Trojan horses, and worms is still clearly tangible today. To counteract these threats, protocol extensions and security mechanisms have been developed that are widely used.

As it will be shown in this dissertation, HBA systems are equally prone to security

attacks (as the IT world was years before) because existing technologies lack state of the art security features. Thus, it is mandatory to identify and set up security mechanisms as it has been done in the IT domain. On the road to reach this ambitious goal, it is necessary to secure communication and to provide a secure environment for HBA devices – otherwise adversaries will soon single out unprotected HBA systems as their next target on a large scale.

In the first part of this dissertation, an extensive security threat analysis is presented. While a comprehensive security concept deals with all identified security attacks, this dissertation is focused on counteracting networks attacks i.e., providing mechanisms for secure communication in HBA networks. Therefore, requirements and challenges for a secure communication within HBA networks are identified in Chapter 3. As the use of cryptography is of utmost importance for fulfilling the identified security objectives, an introduction to cryptographic techniques is given in Chapter 4. Since the presented schemes are used throughout the dissertation, a generic formalism for describing cryptographic transformations is also introduced. In the subsequent Chapter 5, an overview of state of the art technologies is presented. This includes available HBA standards as well as related security concepts from other domains that are relevant for HBA systems. Afterwards, a generic approach for securing communication in HBA networks is introduced (cf. Chapter 6). The key component of the framework is a multi-protocol stack. This stack consists of interchangeable data link/physical layer combinations, a common security layer, and an application layer that is place for a generic application layer model. In Chapter 7, the security layer is presented in detail. The provided security services are based on the concept of secure communication relationships. To show the feasibility, an evaluation of the presented security concept is given in the subsequent Chapter 8. Due to the fact that it is not possible to exclusively rely on cryptography for guaranteeing data availability, organizational measures that counteract interruption threats are presented in Chapter 9. The dissertation is closed with a concluding chapter.

2 Security in Home and Building Automation Systems

In literature, a multitude of ambiguous security definitions exist. The two most important ones used as basis in this dissertation are laid down in [7] and [8]. *Security* in homes and buildings can be defined as measures that protect system resources against adversaries that intentionally try to gain unauthorized, malicious access. From a general point of view, security in homes and buildings is concerned with the protection of the building structure against a physical destruction and/or intrusion (*physical home and building security*). A typical example would be the deployment of measures that try to avoid a physical intrusion like breaking doors or window locks. Once unauthorized access to the interior of the building has been gained, the intruder may steal, for example, belongings or confidential information. To guarantee physical security, constructional measures that avoid a physical destruction and/or intrusion are necessary. However, since providing constructional measures is within the scope of architectural design and construction, physical home and building security is not treated in this dissertation.

In homes and buildings where HBA systems are used, security also deals with the protection of the automation system that controls the home and building services. HBA systems where an intentional violation of security causes a malfunction that prevents the system from providing its expected functionality are referred to as *security-critical systems*. Other HBA systems where a violation of security disturbs a system in a way that the system's functionality is limited (e.g., vandalism acts) are called *security-related systems*. Compared to security-critical systems, the basic functionality may persist (graceful degradation). Typical examples of security-critical systems are access control systems, intrusion alarm systems, and Closed Circuit Television (CCTV) systems. Representatives of security-relevant systems are traditional HBA systems like HVAC and lighting/shading systems where the economic impact caused by a malfunction must not be underestimated. Finally, *non security-relevant systems* are HBA systems where, from the users' point of view, security can be neglected since a malfunction has only minimal consequences and

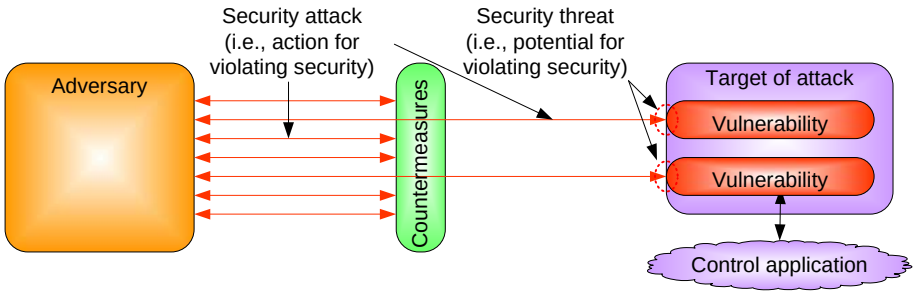


Figure 2.1: Security attacks

the economic damage can be ignored. A typical example of a non security-relevant system would be a home multimedia system that distributes audio/video resources between multimedia equipment.

Obviously, malicious manipulations of HBA systems will always have a minimum economic damage (e.g., attack on an HVAC system) or will at least be cumbersome for the inhabitants due to the decreased comfort (e.g., attack on a home multimedia system). This emphasizes the argument that protecting the functionality of traditional HBA systems even if they only provide non security-relevant services is also desirable.

To be able to integrate security-critical and security-relevant systems into today's HBA systems, an unauthorized, malicious access to the system's functionality has to be avoided. The aim of such an unauthorized, malicious access can be manifold. In the field of HBA, a malicious entity (called *adversary*) may be a human or some piece of malicious software (e.g., Trojan horse, virus, worm) with the intention to gain unauthorized access to the control functionality that interacts with the building environment i.e., applications that control and manage the home and building services. A typical example is the unauthorized access to the control functionality of an access control system that opens and closes a security door. The current action that an adversary performs to gain unauthorized access is generally referred to as a *security attack*. Such a security attack is only possible if the system suffers *vulnerabilities* i.e., flaws and weaknesses that may be exploited by an adversary. The existence of vulnerabilities leads to *security threats* which can be seen as the potential for the violation of security. Note that a security attack is different to a security threat. While a security attack is the action that tries to violate the system's security, a security threat is the potential for a violation of security that may never be utilized [7]. *Security countermeasures* have the aim to counteract security threats and security attacks. The instance of security countermeasures is generally referred to as a *security mechanism*. Figure 2.1 illustrates the relation between these different security terms.

Depending on their purpose, two different classes of security countermeasures are dis-

tinguished. First, the amount of vulnerabilities can be minimized before an adversary may utilize them (*attack prevention*). This can be done by incorporating security in every stage of the system's life cycle especially during design [9]. By using an appropriate security concept, the occurrence of vulnerabilities can be eliminated and thus the potential for violating security is limited. A typical example would be the definition of a *security policy* during system design. A security policy specifies rules and statements that define which system states are allowed and represent secure ones and which system states violate the system's security and thus are insecure ones [10]. Since vulnerabilities can be seen as state transitions from secure to insecure system states, security countermeasures with the aim to prevent security attacks try to avoid such state transitions. Typical examples are organizational countermeasures such as firewalls as well as cryptographic countermeasures like the encryption of confidential information to avoid an unauthorized disclosure. However, designing a perfectly secure system is impossible and so it must be assumed that security vulnerabilities will be discovered during the lifetime.

Second, in cases where a prevention of vulnerabilities and their resulting security threats is not possible with reasonable effort, security countermeasures have to be implemented trying to discover security attacks or systems states that may lead to security attacks (*attack detection*). To minimize the resulting consequences, abnormal system states have to be reported and actions that change the insecure system state back to a secure one again have to be deployed. Initiating these actions can be done automatically or with manual intervention. Consider, for example, an Intrusion Detection System (IDS) that reveals abnormal system behavior, reports it, and tries to prevent a propagation by isolating the source of the attack. Another example is a Message Authentication Code (MAC) that provides the opportunity to detect an unauthorized modification of information.

While attack prevention techniques set precautions that avoid security attacks, detecting countermeasures try to discover an abnormal system behavior that (possibly) leads to a security attack. Obviously, the most effective solution is to eliminate vulnerabilities and their resulting security threats a priori. Therefore, prevention shall be preferred whenever possible. However, there are situations where prevention techniques cannot be used due to possibly high realization effort and complexity of the system. Therefore, a hybrid security concept is desirable. In situations, where prevention methods are inapplicable, security attacks shall at least be detected and appropriate counteractions shall be initiated. No matter which kind of countermeasures are used, the right selection and the effectiveness of the chosen countermeasures highly depends on a preceding analysis of possible security threats. Therefore, the remainder of this chapter presents a *security threat analysis* for

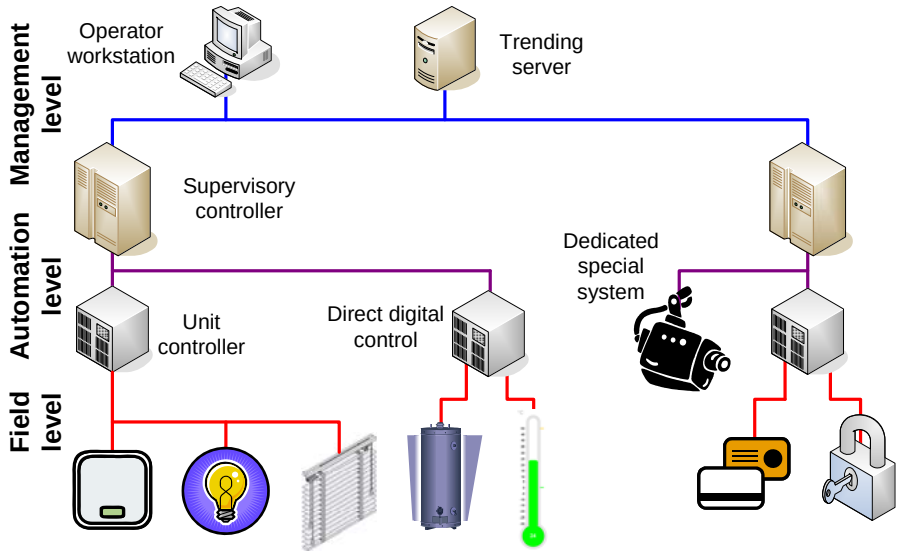


Figure 2.2: HBA three-level functional hierarchy

HBA systems. In the first step, the different system resources that can be a target of an attack as well as possible vulnerabilities of the system resources that an adversary may utilize are determined (cf. Section 2.1). In the second step, the ways how an adversary may try to get unauthorized access to the system are analyzed (cf. Section 2.2).

2.1 Target analysis

In the context of HBA, the adversaries' ultimate goal will always be the access to the control functionality. To identify possible targets, the abstract structure of HBA systems has to be analyzed. In today's HBA systems, the control functionality is organized in a three-level hierarchy [1, 11] (cf. Figure 2.2). Immediate interaction with the environment is associated with the field level: collecting data (measurement, counting, metering) and physically acting upon the process (switching, setting, positioning). The automation level encompasses the various aspects of automatic control – that is, the execution of control loops and sequences, building upon the functionality of the field level. Global configuration and management tasks (such as visualization and accessing trend logs) are considered management level functionality.

Today's HBA systems are implemented as distributed systems where the control functionality is realized by *distributed control applications*. These control applications are spread over different devices that may be interconnected by a so called *HBA network*. The functionality of the control applications is provided by so called *user applications* that are hosted on different devices. Figure 2.3 shows the relation between control appli-

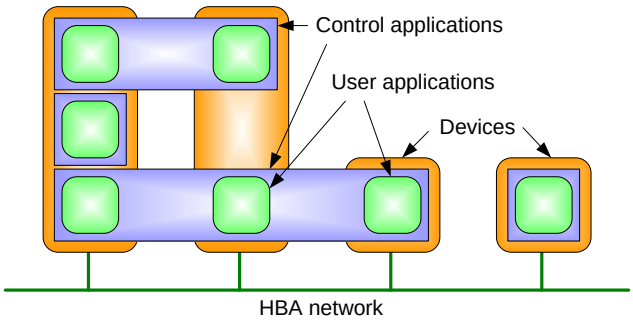


Figure 2.3: Control applications, devices, and user applications

cations, devices, and user applications. While a control application can be spread over one or more devices, a user application that implements part of the control application's functionality is always dedicated to a single device.

As it is naturally the case in distributed systems, there is an inherent need to communicate. On the one hand, the distributed control applications need to exchange *control data* with the intention to influence the environment. Control data can be process data like sensor and actuator values (e.g., a room temperature or a control value of a blind motor) or virtual process data like a setpoint or a control deviation. On the other hand, it is desirable to configure control applications and maintain their behavior. This form of communication is referred to as engineering communication whereas the data that is exchanged is called *engineering data*. Typical examples for the latter are new control application parameter values or a new program code for a user application.

Up to now, a clear separation of the functional levels was used. Sensors and actuators (providing field-level functionality) were connected to controllers via point-to-point interfaces or simple field networks. A dedicated automation network was in place for the exchange of control data between controllers which implemented the automation level (e.g., Direct Digital Control (DDC) stations, unit or supervisory controllers). Management functionality was provided by server stations, which aggregated the automation network traffic and made it accessible over the management network. They also acted as gateways for configuration (and vertical access in general).

Nowadays, HBA networks are implemented following a two-tier model. This is for two reasons. First, processing power has become cheap enough to equip sensors and actuators with network controllers powerful enough for automation level functionality at economically viable prices. Therefore, field devices are able to incorporate functionality once exclusively assumed by dedicated controllers from the automation level. Second, IT and its infrastructure became accepted not only at the management level, but also at the automation level.

Consequently, functionality of the former automation level is split and reassigned either to field devices (e.g., implementing controller functionality) or management devices (e.g., realizing process data monitoring). The result is a two-tier HBA network structure where intelligent field devices are located in multiple network segments called *field networks*. These field networks are in turn interconnected by a common *backbone network* which is also home for management devices that require a global view of the entire network. Additionally, dedicated gateways that provide an interconnection to foreign networks like the Internet or other Wide Area Networks (WANs) may also be located there. The network consisting of the backbone together with the connected field networks is referred to as *control network*. Figure 2.4 shows the resulting two-tier model structure.

At the field level, robustness, flexibility, and cost efficiency are most important. Therefore, fieldbus technologies featuring (whenever possible) free topology are still used there. Typical examples are wired fieldbus technologies that support Twisted Pair (TP) or Powerline (PL) as network media. Furthermore, wireless technologies that use Radio Frequency (RF) as transmission medium are getting more and more important [12].

At the backbone, it is more common to use high-performance network technologies since data from all over the network is concentrated there. Today, a trend towards the use of Internet Protocol (IP) based networks as backbone can be observed. There are various reasons for this. On the one hand, buildings are often already equipped with IP based networks. This is especially true for functional buildings from the BA domain (e.g., office Local Area Networks (LANs)) but also for HA systems since the use of IP networks becomes increasingly popular in our homes (e.g., LANs for Internet and multimedia purposes). Since an HBA system typically demands only moderate response times, it is feasible to share the medium with non control data as long as acceptable performance can be guaranteed. On the other hand, the costs for IP cabling and network interface hardware are rapidly decreasing. Therefore, even small embedded microcontrollers typically found in the HBA domain can be equipped with an Ethernet interface chip.

With this topology in mind, three different device classes can be identified [13]:

- *Sensor, Actuator, and Controllers (SACs)* directly interact with the physical environment, are responsible for data acquisition, and perform control functionality.
- *Interconnection Devices (ICDs)* are used to interconnect different network segments. On the one hand, ICDs like routers are used to interconnect field networks with the backbone. On the other hand, gateways provide connections to foreign networks for remote access (e.g., Web gateway to a WAN).
- *Management Devices (MDs)* are responsible for performing management tasks like

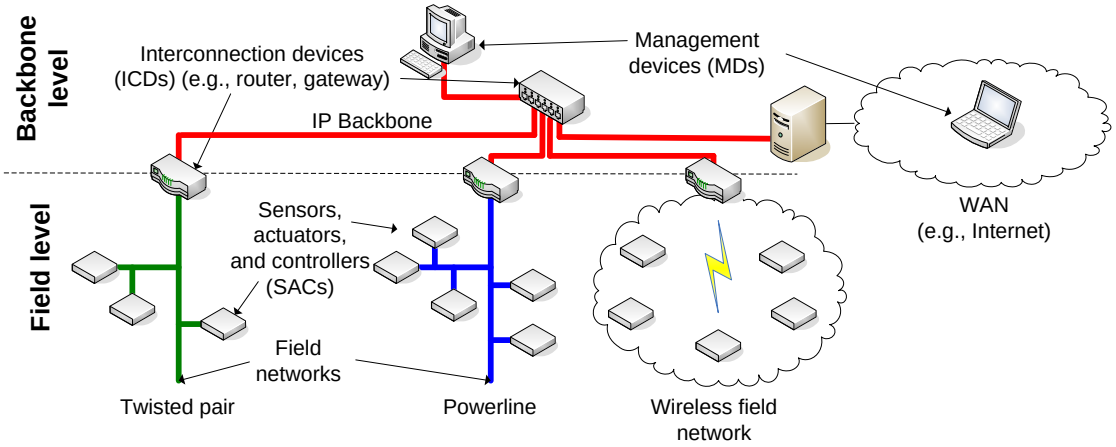


Figure 2.4: HBA control network

configuration (e.g., setting initial configuration parameters, uploading user applications to SACs), maintenance (e.g., changing setpoints, logging, and trending), and operator tasks (e.g., visualization and alarm monitoring).

2.2 Attack analysis

Based on the generic HBA system model mentioned above, five different target attacks can be identified:

- *Field network*: Adversaries with access to a field network segment may try to interfere with the data that is transmitted over the network. This concerns *control data* on the one hand and *engineering data* on the other hand.
- *Backbone*: The attack scenarios at the backbone level are similar to the ones at the field level. The main difference is that since the backbone interconnects the different field network segments, the data that is transmitted across network borders is concentrated at the backbone. As a result, an adversary with access to the backbone has a global view over the whole system.
- *SAC*: Since the control applications are distributed across the different SACs, an adversary may directly access SACs to manipulate the behavior of the hosted user applications. After having gained access to a SAC, an adversary may change configuration parameters (e.g., changing a setpoint), manipulate user applications (e.g., exchange the control logic), or directly access the control data (e.g., setting a new actuator value).
- *ICD*: ICDs are used to interconnect different network segments. Therefore, data that is transmitted across network borders passes through them. A possibility to

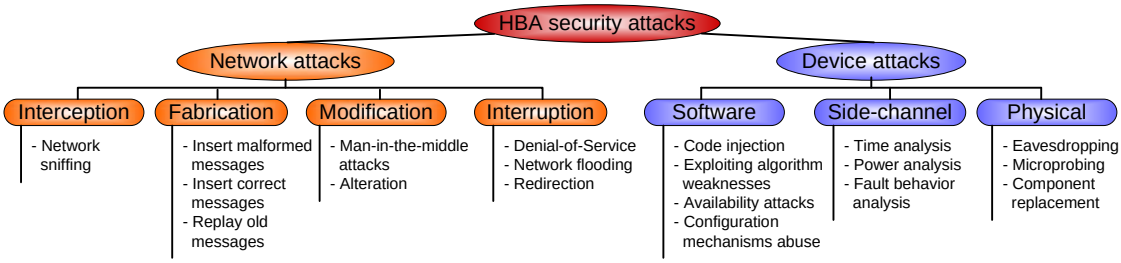


Figure 2.5: Security attacks in HBA systems

interfere with the transmitted data while it is forwarded by the ICD is to gain unauthorized access to the software running on the ICD. After having compromised an ICD, an adversary may have unauthorized access to the forwarded data and so, attack scenarios like the ones depict for the field level are possible. Additionally, an adversary may also change routing of network packets to redirect the network traffic. Furthermore, ICDs may also provide an interconnection to foreign networks. In these cases, adversaries located at public networks (e.g., Internet) may try to use such an ICD (e.g., Web gateway) as access point to the inner HBA network. Using this remote access, the adversary is able to start further security attacks (e.g., attack a SAC or parts of the HBA network).

- *MD*: System engineers and system operators use MDs to perform configuration (e.g., change setpoints) and/or maintenance tasks (e.g., monitor alarm conditions). Therefore, an adversary may try to compromise an MD, impersonate it, and take advantage of the MD's access rights. Using the MD's privileges, an adversary may gain management access to SACs or ICDs and thus similar attack scenarios are possible (e.g., manipulation configuration parameters or malicious interference with the routing table of an ICD).

Summarizing, an adversary has two different opportunities for getting access to control applications: *network attacks* and *device attacks* (cf. Figure 2.5). First, the adversary may attack the network medium to access the exchanged data and thus interfere with the data when it is transmitted. According to [14], network attacks can be divided into 4 classes:

- *Interception attacks*: The adversary tries to gain unauthorized access to confidential data exchanged over the network (e.g., network sniffing).
- *Fabrication attacks*: The adversary tries to insert malicious data (e.g., replay previously sent network messages).
- *Modification attacks*: The adversary tries to manipulate the data while it is transmitted over the network (e.g., change the semantics of network messages).
- *Interruption attacks*: The adversary tries to disturb the communication between the

device and thus makes data unavailable (e.g., Denial-of-Service (DoS) attacks).

While *passive network attacks* like network sniffing try to interfere with the transmitted data without altering it, *active network attacks* modify or affect the data during transmission.

In order to be able to interfere with the data transmitted over the network, the adversary needs access to the network medium. This can be done in two different ways:

- *Medium access*: The adversary gains physical access to the network medium. This can be accomplished more easily when open communication technologies (i.e., RF or PL) are used.
- *Device access*: The adversary can use the network interface of another device (e.g., a compromised SAC or a Web gateway) to get access to the network. Note that the way to gain access to such a foreign network interface is considered as a device attack. The succeeding attack is considered as a network attack.

Second, the adversary may attack a device to access user applications (*device attacks*). These attacks can be classified based on the means used to launch them [15, 16]:

- *Software attacks*: An adversary may try to exploit weaknesses in the software implementation of a device. To gain access to the device's software, the adversary can use the network (e.g., management access via network services) or a local point-to-point management interface (e.g., Universal Serial Bus (USB), interfaces based on EIA-232).
- *Side-channel attacks*: An adversary may observe external (device) parameters which are measurable during operation to collect information about internals.
- *Physical attacks*: An adversary may use physical intrusion or manipulation (e.g., probing of bus lines, replacement of Read-Only Memory (ROM) chips) to interfere with a device.

3 Requirements and Challenges for Secure Communication

A comprehensive security concept for HBA systems implements countermeasures that deal with both network and device attacks. While keeping devices attacks in mind, the remainder of this dissertation is focused on counteracting networks attacks i.e., providing mechanisms for secure communication in HBA networks. However, ongoing research activities in the field of device security for HBA systems are presented in [17, 18].

To be able to analyze security mechanisms of available HBA solutions and to evaluate security concepts from other domains that may come into consideration for the HBA domain, the requirements and challenges have to be identified first. To begin with, the ways how communication is performed in HBA networks have to be analyzed. Communication entities that exchange control as well as engineering data of the same interest are members of a so called *communication relationship*. In the context of HBA, a *communication entity* is a device that hosts user applications which exchange data (e.g., sensor and actuator values that are under control of the user application). From the point of view of this dissertation, a device is the smallest security context which is entirely trusted. Providing security within a device is not considered here since attacks on a device and thus on its hosted user application(s) are regarded as device attacks.

The communication rules within a communication relationship are specified by a certain *communication model*. Depending on the amount of involved communication entities, it is common to distinguish between models that are based on a point-to-point relationship and models that are based on a point-to-multipoint relationship. If there are multiple senders, even a multipoint-to-multipoint relationship can exist (cf. Figure 3.1). A typical model based on a point-to-point relationship is the *client/server model*. Here, the requesting device (client) sends a request to the service provider (server). After having received the request, the server executes the desired action and sends a response or an acknowledgment back to the client.

Two communication models that are based on point-to-multipoint or multipoint-to-

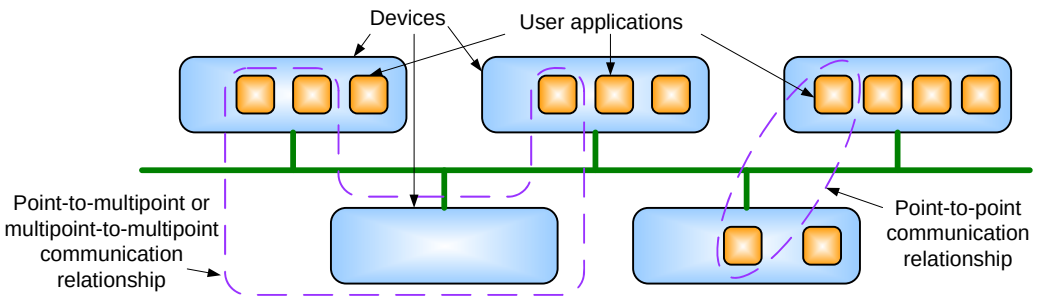


Figure 3.1: Communication in HBA systems

multipoint relationships are the *producer/consumer* and the *publisher/subscriber* model.[1] In a producer/consumer model, one or more senders (called producers) provide data by making it publicly available. Other entities decide on their own whether they are interested in the data or not. If they are interested, they receive and deliver the data to the corresponding user applications (consumers). Otherwise, they ignore it. A set of senders and receivers that are interested in the same data is called a *communication group*. Since each group member is responsible for joining and leaving the group, there is no central instance that manages the group membership (*loose group membership*). Due to the fact that it cannot be guaranteed that only group members are able to communicate within the same group, this model is rather suitable for security-relevant systems than for security-critical ones.

The *publisher/subscriber model* is similar to the above mentioned producer/consumer model. However, each group member has to explicitly join and leave a communication group by sending a (un-)subscription request to a dedicated group coordinator (*strict group membership*). The group coordinator determines whether the requesting entity is allowed to join the particular group or not. Sending and receiving data within a specific communication group is only possible after a successful group join (subscription). When the entity leaves the group, it is no longer able to participate in the communication. Since only group members are able to communicate within the group, this model is suitable for the use in security-critical environments.

While a communication model defines the abstract rules within a communication relationship, it is independent from its realization. Therefore, an abstract communication model needs a communication service that provides the opportunity to send data to the desired receiver(s). The client/server model simply requires a point-to-point communica-

[1] Note that the definition of producer/consumer and publisher/subscriber used in this dissertation may differ from other ones used in literature [19].

tion service. Therefore, a unicast[2] communication service is sufficient. The producer/consumer as well as the publisher/subscriber model need a point-to-multipoint or multipoint-to-multipoint communication service. In principle, *multiple unicast* connections can be used to send data from one sender to multiple receivers. While in a producer/consumer model, this can be achieved by sending data to a public database where each consumer can fetch the previously produced data, in a publisher/subscriber model, a sender can transmit data to each group member separately using a dedicated unicast connection. Alternatively, a central group coordinator can be used to transmit the data to all group members. If a sender wants to publish data to all group members, it sends the data to the group coordinator which forwards it to all subscribers using multiple unicast connections. Obviously, the use of such schemes that simply rest upon multiple unicast connections is inefficient especially in large groups. Another possibility is to use *broadcast* communication. In a producer/consumer model, an entity can decide on its own whether the broadcast message should be processed or not. In a publisher/subscriber model, the broadcast message can, for example, be encoded in a way that only group members are able to handle the message. However, using broadcast is inefficient for networks that consist of many, small groups since each entity has to verify each broadcast message whether it is of particular interest or not. Therefore, support for *multicast* where only the members of the addressed group receive the data is desirable. Nevertheless, since the underlying network technology may not support multicast, multicast for group communication is not mandatory in the field of HBA.

All three communication models can be used for control data as well as engineering data exchange. Therefore, six different *communication types* can be distinguished in the HBA domain:

- *Point-to-point control data communication*: As the name implies, control data is exchanged using the client/server model. For example, a boiler (client) periodically requests the present value of a temperature sensor (server). Another example from the security-critical domain would be an intrusion alarm controller that periodically requests the status of a presence detector.
- *Loose group communication*: Since group membership is not verified here, loose group communication is only applicable for security-relevant environments. A typ-

[2]While the term point-to-point communication service represents an abstract communication relationship, the term unicast is used for the communication service provided by the underlying network technology. Similar, the terms point-to-multipoint and multipoint-to-multipoint are associated with abstract communication groups whereas the terms multicast and broadcast are used for the communication services provided by the underlying network.

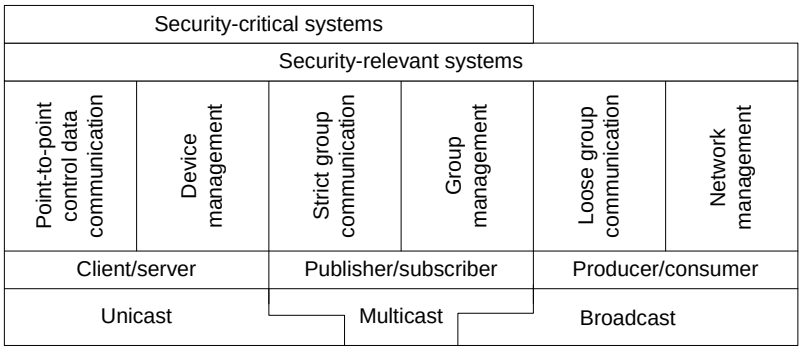


Figure 3.2: Communication services in HBA systems

ical example of this communication type is a lighting system in which multiple light switches are used to control multiple light sources. Another example would be a multimedia system where an audio stream is sent from a source to multiple receivers.

- *Strict group communication*: Strict group communication is advantageous in security-critical environments where only devices with adequate access rights shall be able to send and receive data of particular interest. A typical example would be an access control system where multiple access sensors (e.g., fingerprint reader, card reader) control multiple security doors.
- *Device management*: Device management uses the client/server model. A management client starts a so called management session to the particular device which is to be configured or maintained (management server). After the management client has performed the desired management tasks, the session is terminated. Consider, for example, an operator workstation intending to change configuration parameters of a SAC.
- *Network management*: Network management refers to management tasks that are addressed to all network members. For example, a network coordinator may want to inform all network members about changed routing information. Typically, network management is based on the producer/consumer model with the engineering data being distributed to all network members.
- *Group management*: Group management is used to configure and maintain all devices of a particular group at a given time. Since only the members of the desired group shall be addressed, group management uses the publisher/subscriber model. A typical example is a group coordinator that informs the group members about a new device joining the group.

Figure 3.2 shows the different communication types used in HBA systems and a mapping of them to the corresponding network services. In this dissertation it is assumed that

an underlying HBA network protocol offers at least a unicast and a broadcast service. Multicast services are considered optional. However, if multicast is not supported by the HBA network protocol, either multiple unicasts or a broadcast have to be used instead.

3.1 Functional requirements

To be able to secure communication between all kinds of control applications, all six different *communication types* mentioned above have to be protected. The secure variant of these communication types are *secure point-to-point control data communication*, *secure loose group communication*, *secure strict group communication*, *secure device management*, *secure network management*, and *secure group management*.

For each of these secure communication types, various security objectives have to be guaranteed. In literature, the amount of different security objectives as well as their names and definitions vary. Based on [10, 14, 20, 21, 22], the following *primary security objectives* for a secure data transfer within a secure communication relationship are defined:

- *Entity authentication*: To be able to guarantee a secure communication within a communication relationship, the participating entities must prove their identities first. In other words, they must *authenticate* each other. Guaranteeing entity authentication avoids that a malicious entity impersonates a trustworthy one. While in most cases both the receivers as well as the senders must prove their identities (*mutual entity authentication*), it may be sufficient that only one site i.e., either the receiver site or the sender site within a secure relationship is authenticated (*unilateral entity authentication*). A typical example of a relationship where only unilateral entity authentication is demanded is a sensor that periodically distributes non-confidential sensor data. In such a use case, it is important that the identity of the sender is authenticated – proving the authentication of the receivers is not required. Entity authentication is also referred to as identification.
- *Authorization*: After entity authentication of the members of a secure communication relationship has been proved, it must be verified whether the joining entity has the necessary access rights to join a relationship. If the joining entity has insufficient access rights, participation in this relationship is denied. The authorization process is also known as access control.
- *Secured channel*: After the members of the relationship are authenticated and all of them have the required access rights, the data exchanged within the relationship must be protected in a secure manner. This is done by establishing a so called

secured channel. A secured channel uses non-cryptographic (e.g., physical or organizational measures) and/or cryptographic techniques to protected data against security attacks while it is transmitted over a network. Depending on the requirements of the involved entities, a secured channel guarantees the following security objectives.

- *Data integrity*: Providing data integrity guarantees that the data was not modified by unauthorized entities during transmission. To achieve this, modification attacks have to be prevented. However, if a full prevention is not possible, modification attacks shall at least be detected in order to avoid the use of corrupted data.
- *Data origin authentication*: Data origin authentication (in literature also referred to as data authentication) is a stronger form of data integrity. In addition to the protection of data against unauthorized modification, a receiver can verify the data origin i.e., the data source.
- *Non-reputability*: An even stronger form of data origin authentication is called non-reputability. Here, the verification of the sender can also be done anytime after data transmission by a non-involved third party. Non-reputability is important in systems that are regulated by law. A typical example is an access control system where it may be necessary to prove that a person definitely has opened a security door.
- *Data freshness*: Data freshness guarantees that the transmitted data is recent and that an adversary has not replayed previously sent data. A key feature of guaranteeing data freshness is message ordering which may be difficult especially within communication groups [23]. Depending on the requirements regarding timeliness of the transmitted data, it is common to distinguish between weak and strong data freshness. Weak data freshness guarantees partial message ordering without providing exact timing information of transmitted data. Strong data freshness guarantees message ordering on a request-response pair. While weak data freshness prevents a malicious reordering of data, strong data freshness also allows an estimation of malicious message delays.
- *Data confidentiality*: The disclosure of confidential information must be avoided. It must be guaranteed that only authorized entities have access to it. A typical example of confidential information would be a Personal Identification Number (PIN) code that is entered by a user at a security door. However, data that is exchanged within an HVAC system may also contain confidential

information. Consider, for example, a room temperature sensor. While the actual room temperature seems to be no secret, a low temperature may indicate that the HVAC system is in "vacation mode". Using this knowledge, an adversary may assume that there is nobody present.

- *Data availability*: Data availability guarantees that authorized entities have access to the data and that the access is not prevented by adversaries. Data availability thus counteracts interruption threats.

Besides these primary security objectives, several *secondary security objectives* that may be more or less relevant for the HBA domain exist. *Anonymity* guarantees that an adversary is not able to disclose the identity of a communication entity. Furthermore, it is guaranteed that an adversary cannot track an entity and it is not able to derive a behavior pattern. While anonymity is in general a side issue in HBA, it may be important when an entity is directly related to an individual. Anonymity of humans is also known as *privacy*. A typical example would be a social alarm system that monitors the healthiness of elderly people. *Auditability* is concerned with providing the proof what a system has done. Auditability is only possible if several other security objectives like data origin authentication and non-reputability are guaranteed. Auditability is primarily important in systems like alarm or access control systems.

In order to fulfill the requirements of security-critical services, the identified primary security objectives have to be satisfied. However, depending on the type of service, some objectives like data origin authentication, non-reputability, and data confidentiality may be less important. This is especially true for security-relevant services. Secondary security objectives, on the other hand, can be seen as additional objectives that may be required for dedicated services.

3.2 Non-functional requirements and challenges

The characteristics of the requirements of security-critical and security-relevant systems mainly depend on the environment and domain where they are used. For example, IT security mechanisms are tailored to the specifics of the IT world. They cannot be trivially used within the HBA domain. This is also true for closely related domains like industrial automation. The domain specific characteristics of HBA systems lead to the following non-functional requirements and challenges:

- *Low-power embedded devices*: Due to reasons of cost and space efficiency, low-power embedded devices are commonly used in the HBA domain. This is mainly

true for SACs and ICDs, since they are normally equipped with limited system resources. This concerns the amount of available volatile and non-volatile memory, processing power but also the power supply since SACs often rely on bus- or battery-power or are even self-powered (e.g., devices that use energy harvesting techniques). However, since security mechanisms imply an additional computing effort and are computationally intensive (especially cryptographic algorithms), their use must not exceed the available device resources. Therefore, the overhead imposed by these mechanisms needs to be reasonably small. To be suitable for the HBA domain, it is essential to find a balance between a required level of security and available resources ("*good enough security*"). For example, if the non-disclosure of the transmitted data is not strictly necessary, data confidentiality is unnecessary. Furthermore, if non-reputability is not a strict requirement, guaranteeing data origin authentication or even data integrity may be sufficient.

- *Scalability*: For many services in the IT world, the amount of devices within a single communication relationship is relatively small. Therefore, the client/server model is used in most cases. Peer-to-peer communication patterns that are common in the HBA domain are rarely used. Depending on the application, the number of devices is also quite small in the industrial automation domain. In the HBA domain, networks are usually home for a huge number of devices consisting of a few MDs, several ICDs, and thousands of manifold SACs. Thus, scalability of the integrated security mechanisms is of major concern.
- *Non IP field networks*: IT security mechanisms are geared towards different requirements regarding the used network technology. While in the IT world IP based network protocols are dominant, the use of IP networks in HBA networks is reserved to the backbone level. At the field level, non IP field networks are mainly used. As mentioned in Section 2.1, the main reasons for the use of such network are robustness, flexibility, and cost efficiency.
- *Quality of Service (QoS) parameters*: The required QoS parameters of HBA field networks differ from the IT/office world. In the IT/office domain, the data volume to be transferred is commonly high (in the order of mega- or gigabytes) with usually no real time requirements. Control data typically transmitted in HBA networks has a small volume (in the order of some bytes) with perhaps soft real time requirements (e.g., the reaction time of a lighting system). Furthermore, QoS properties like *reliability* and *ordering* of messages have to be considered. While these QoS properties are normally of less concern in the IT/office world, they may be an important issue

in the HBA domain.

- *Untrusted environments*: Up to now, HBA networks were virtually closed networks with, if at all, dial-in connections for remote management. With the introduction of wireless technologies and interconnections to foreign networks (e.g., Web gateways to the Internet), this physical isolation is not longer valid. Moreover, devices often operate in environments where physical access (e.g., an intrusion alarm in a public building) cannot be prevented. Therefore, it has to be assumed that a short time physical access to the HBA network cannot be avoided and thus solely relying on physical security is insufficient for security-critical and security-relevant environments.
- *Long lifetime*: HBA systems have to be kept operable for years or even decades. Since designing a perfectly secure system is impossible, it must be assumed that security vulnerabilities will be discovered during the intended long lifetime. To be able to correct identified flaws in the system design, the possibility to update and maintain the used system components has to be provided. However, since such an update and maintenance mechanism also offers an additional target of security attacks, it has to be protected against unauthorized use.

4 Cryptography

Cryptography is concerned with mathematical techniques to protect information against malicious interference. Besides the use of non-cryptographic techniques like physical security or organizational measures, cryptographic techniques are essential for guaranteeing the security objectives mentioned in Section 3.1. From a general point of view, cryptographic techniques consist of *cryptographic protocols* and *cryptographic algorithms* [21]. A cryptographic protocol is a distributed algorithm that specifies various, correlated actions that are executed on two or more entities. In contrast to a protocol, a cryptographic algorithm (also called cryptographic primitive) consists of actions that are executed on and are related to a single entity. Key components of cryptographic protocols and algorithms are *cryptographic transformations*.

Definition 4.1. Let A and B be a finite set of binary encoded strings with an arbitrary but finite length and let $T_1, \ldots, T_n$ be finite sets of binary encoded strings with fixed lengths. A *cryptographic transformation* is a (n+1)-ary function $f : T_1 \times \ldots \times T_n \times A \rightarrow B$ that is defined by $f(t_1, t_2, \ldots, t_n, a) = b$ where $a \in A$, $b \in B$ and $t_i \in T_i$ for $i = 1, 2, \ldots, n$ and $n \in \mathbb{N}$. The input parameters t_i are called *security tokens*.

From an informal point of view, security tokens can be seen as input parameters for the cryptographic transformation that control the way how the information is transformed. Figure 4.1[1] shows an example how cryptographic transformations are used. The source that wants to securely transmit data via an insecure communication channel (e.g., public network) uses a cryptographic transformation to generate a secured version of the unprotected data. This secured version is transmitted over the channel to the sink which performs another cryptographic transformation to verify the received data and to restore the original version. A pair consisting of a cryptographic transformation at the source site and its corresponding cryptographic transformation at the sink site is called *cryptographic scheme*.

[1] For the rest of this dissertation, Unified Modeling Language (UML) 2.2 [24] is used to model sequence diagrams.

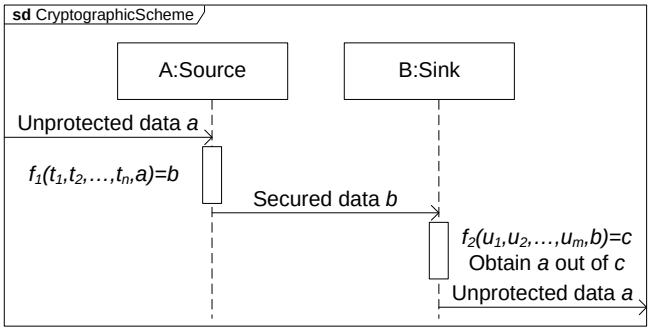


Figure 4.1: Cryptographic scheme

The basic concept of cryptographic schemes is that the involved cryptographic transformations (or at least a subset of them) can only be performed by authorized entities – adversaries shall not be able do that. To achieve this, either the transformations or parts of the input parameters have to be kept secret. Today, almost all modern cryptographic transformations are based on Kerckhoff's principle – while the transformation can be made publicly available, parts of the security tokens must be kept secret [25]. This subset of the security tokens of a cryptographic transformation is referred to as *secret keys*.

Depending on the properties of secret keys, cryptographic schemes are classified into *symmetric* (cf. Section 4.2) and *asymmetric* (cf. Section 4.3) schemes. In addition to cryptographic schemes that require some secret keys, there are cryptographic transformations that need not any secret keys at all. These so called *unkeyed cryptographic transformations* are normally used in combination with other cryptographic schemes (cf. Section 4.1).

4.1 Unkeyed cryptographic transformations

Unkeyed cryptographic transformations have special mathematical properties that are required by other cryptographic schemes. A typical example is a cryptographic hash transformation. Its main purpose is to transform data of arbitrary length into data of fixed length:

Definition 4.2. Let B be a finite set of binary encoded strings with a fixed length l and let h be a cryptographic transformation $h : T_1 \times \ldots \times T_n \times A \rightarrow B$ that is defined by $h(t_1, t_2, \ldots, t_n, a) = b$ where $|b| = l$[2]. The cryptographic transformation h is called *cryptographic hash transformation* if and only if the following conditions are valid:

[2] $|x|$ denotes the length of the binary encoded string x.

- one-way: for almost all pre-specified $b \in B$, it is impossible to find an $a \in A$ such that $h(t_1, t_2, \ldots, t_n, a) = b$.
- weak collision resistance: given a *specific* $a \in A$ it is impossible to find a single $a' \in A$ $(a \neq a')$ such that $h(t_1, t_2, \ldots, t_n, a) = h(t_1, t_2, \ldots, t_n, a')$.
- strong collision resistance: it is impossible to find *any* two arbitrary pairs $a, a' \in A$ $(a \neq a')$ such that $h(t_1, t_2, \ldots, t_n, a) = h(t_1, t_2, \ldots, t_n, a')$.

The output b is called *hash value*.

In some applications, weak collision resistance may be enough and thus strong collision resistance is not necessary. Furthermore, hash transformations exist that guarantee strong collision resistance but do not provide the one-way property. However, most cryptographic hash transformations used in practice satisfy all three conditions. Examples of cryptographic hash transformations are Message Digest Algorithm 5 (MD5) [26] and functions from the Secure Hash Algorithm (SHA) family [27]. However, the use of MD5 shall be avoided in modern implementations since collisions have been found [28].

The quality of many security protocols and algorithms depends on the quality of the used random numbers. Therefore, random numbers and mechanisms to generate them play an important role in cryptography. To generate random numbers in computer systems, a *random bit generator* that produces a sequence of statistically independent and unbiased binary digits is necessary. The key component of these random bit generators is a natural true random source. However, due to the determinism of computer systems, true random bit generators are unsuitable and difficult to implement. Therefore, so called *Pseudo Random Bit Generators (PRBGs)* are used. A PRBG is a deterministic algorithm that uses a truly random input parameter s of length k to produce a binary sequence r of length l ($l >> k$) that "seems to be random". The input parameter s is called seed.

Obviously, the statement "seems to be random" is a critical factor. Informally, the output of a PRBG has to be unpredictable for any entity that does not know the seed. As a result, the output is not truly random – using the same seed a PRBG generates always the same pseudo random bit sequence. To prove the unpredictability of pseudo random bit sequence, the PRBG has to pass statistical tests. A common approach to define cryptographic secure PRBGs is the following [21]:

Definition 4.3. A PRBG is said to be cryptographically secure if and only if it passes the next-bit test i.e., there is no polynomial algorithm that can predict the $(n + 1)$-th bit with a probability greater than 0.5 if the first n bits are known.

Instances of cryptographically secure PRBGs mostly rely on a number-theoretical prob-

lem that is assumed to be intractable. A typical example is a PRBG that is based on the Integer Factorization Problem (IFP) (cf. Section 4.3). However, since these PRBGs are computationally intensive, others based on cryptographic hash functions or block cipher schemes (cf. Section 4.2(a)) can be used as appropriate alternatives. Examples are given in [29].

4.2 Symmetric schemes

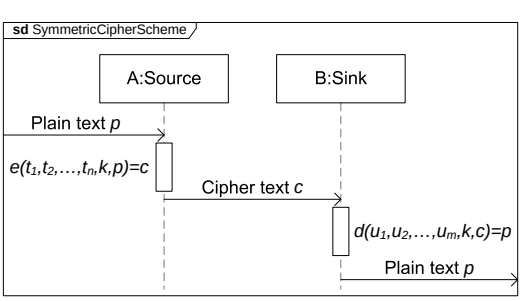
Compared to unkeyed cryptographic transformations, symmetric schemes require input parameters that need to be kept secret. Symmetric schemes can be formally defined as follows:

Definition 4.4. Let f_1 be a cryptographic transformation $f_1 : T_1 \times \ldots \times T_n \times K \times A \rightarrow B$ that is defined by $f_1(t_1, t_2, \ldots, t_n, k_{f_1}, a) = b$ and let f_2 be a cryptographic transformation $f_2 : U_1 \times \ldots \times U_m \times K \times B \rightarrow C$ that is defined by $f_2(u_1, u_2, \ldots, u_m, k_{f_2}, b) = c$. A cryptographic scheme consisting of the pair (f_1,f_2) is called *symmetric* if and only if the following conditions are met:

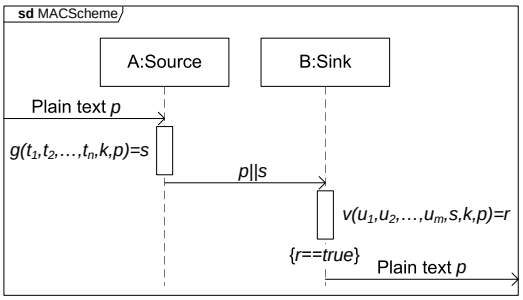
- k_{f_1}, k_{f_2} are only known by the source and the sink respectively,
- it is easy to derive k_{f_1} out of k_{f_2},
- it is easy to derive k_{f_2} out of k_{f_1},
- it is impossible to calculate $f_1(t_1, t_2, \ldots, t_n, k_{f_1}, a)$ for any $a \in A$ without knowing k_{f_1},
- it is impossible to calculate $f_2(u_1, u_2, \ldots, u_m, k_{f_2}, b)$ for any $b \in B$ without knowing k_{f_2},
- it is impossible to derive k_{f_1} or k_{f_2} given zero or more valid a, b pairs and zero or more valid b, c pairs, respectively.

Note that it is required that the keys k_{f_1} and k_{f_2} are only known by the source and the sink. By definition, if one of these two keys gets compromised i.e., an external entity is in possession of it, the scheme is no longer considered as a valid (i.e., secure) symmetric cryptographic scheme. Furthermore, it is not demanded that the secret keys k_{f_1} and k_{f_2} have to be equal. However, in most common symmetric schemes, the secret keys are identical at the source and sink site. This single key is referred to as *shared secret key*.

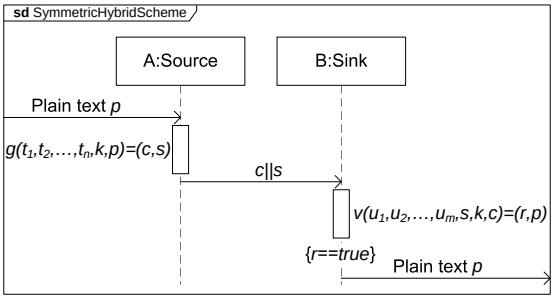
Instances of symmetric schemes can be further divided into *symmetric cipher schemes*, *MAC schemes*, and *symmetric hybrid schemes*. A symmetric cipher scheme protects confidential data in such a way that unauthorized entities are not able to interpret the data's



(a) Symmetric cipher scheme



(b) MAC scheme



(c) Symmetric hybrid scheme

Figure 4.2: Symmetric cryptographic schemes

semantic without proper knowledge of the secret key. Figure 4.2(a) shows how symmetric cipher schemes are applied. The source uses a so called *symmetric encryption transformation* to generate an encrypted version (called cipher text) out of the unprotected data (called plain text) using a shared secret key and other security tokens. The cipher text is then transmitted to the sink which employs the so called *symmetric decryption transformation* to generate the plain text using the same shared secret and other security tokens. This leads to the following definition.

Definition 4.5. Let e be a cryptographic transformation $e : T_1 \times \ldots \times T_n \times K \times A \to A$ that is bijective and defined by $e(t_1, t_2, \ldots, t_n, k, p) = c$ and let d be a cryptographic transformation $d : U_1 \times \ldots \times U_m \times K \times A \to A$ that is bijective and defined by $d(u_1, u_2, \ldots, u_m, k, c) = p$ where (e, d) is a symmetric cryptographic scheme and k is a shared secret key that is only known by the source and the sink. This symmetric cryptographic scheme is called *symmetric cipher scheme* if and only if

$$p = d(u_1, u_2, \ldots, u_m, k, e(t_1, t_2, \ldots, t_n, k, p))$$

for any given $t_1, t_2, \ldots, t_n$, $u_1, u_2, \ldots, u_m$, shared secret key $k \in K$, and for all $p \in A$. p is called *plain text*, c is called *cipher text*, e is called *symmetric encryption transformation*, and d is called the corresponding *symmetric decryption transformation*.

Implementations of symmetric cipher schemes are classified into *block ciphers* and *stream ciphers*. Block ciphers transform a block of plain text with a fixed length into a block of cipher text with the same fixed length. Typical examples are Advanced Encryption Standard (AES) [30], Triple Data Encryption Standard (3DES) [31], Data Encryption Standard (DES) [31], Twofish [32], Blowfish [33], Camellia [34], and SAFER [35]. To encrypt plain texts that are larger than the block size, a *mode of operation* has to be used. This mode of operation defines how subsequent blocks are encrypted with the same key. Most of these modes need an additional input parameter called *Initialization Vector (IV)*. The main aim of the IV is to initialize the encryption of the first block. The used mode of operation is a critical component in symmetric cipher schemes since an insecure mode of operation can break the whole cipher scheme [36]. Therefore, the National Institute of Standards and Technology (NIST) specifies five different modes of operation providing different features [37]: Electronic Codebook (ECB) mode, Cipher-Block Chaining (CBC) mode, Cipher Feedback (CFB) mode, Output Feedback (OFB) mode, and Counter (CTR) mode.

Stream ciphers create a pseudo random generated bit stream (called key stream) that is combined with the plain text. Typical examples are RC4 [38] and modern stream ciphers from the eStream portfolio [39]. Furthermore, it is also possible to use block ciphers to create stream ciphers. A common example is the use of a block cipher like AES in CFB mode.

Finally, MAC schemes are used to secure the data in way that a subsequent modification by an unauthorized entity can be detected. Figure 4.2(b) shows their basic principle. The source uses a so called *MAC generation transformation* to calculate some piece of data with a fixed length (called MAC) out of the data using a shared secret key and other security tokens. The output is transmitted together with the data to the sink where the sink uses a so called *MAC verification transformation* that proves the authenticity of the data using the MAC, the shared secret key, and other security tokens as input.

Definition 4.6. Let B be a finite set of binary encoded strings with a fixed length and let C be a finite set containing the elements $C = \{true, false\}$. Let g be a cryptographic transformation $g : T_1 \times \ldots \times T_n \times K \times A \rightarrow B$ that is defined by $g(t_1, t_2, \ldots, t_n, k, p) = s$ and let v be a cryptographic transformation $v : U_1 \times \ldots \times U_m \times B \times K \times A \rightarrow C$ that is defined by

$$v(u_1, u_2, \ldots, u_m, s, k, p) = \begin{cases} true, & \text{if } g(t_1, t_2, \ldots, t_n, k, p) = s \\ false, & \text{otherwise} \end{cases}$$

where the following conditions are met:

- g is a cryptographic hash transformation,
- (g, v) is a symmetric cryptographic scheme,
- k is a shared secret key that is only known by the source and the sink.

This symmetric cryptographic scheme (g, v) is called *MAC scheme*. Furthermore, g is called *MAC generation transformation*, v is called *MAC verification transformation* and s is called the *MAC* of p.

Instances of MAC schemes are commonly created from other cryptographic transformations like cryptographic hash functions or block ciphers. Keyed-Hash Message Authentication Code (HMAC) [40] is an example of a MAC scheme where cryptographic hash functions are used. Examples of MAC schemes that are based on block ciphers are CBC-MAC where a block cipher like AES in CBC mode [37] is used and CMAC [41] which is a variant of CBC-MAC for plain texts with variable lengths.

A symmetric hybrid scheme[3] provides the functionality of both a cipher and a MAC scheme. Figure 4.2(c) shows the basic principle. The source transforms the unprotected data in a way that neither the semantic of the data can be intercepted nor that it can be modified by an unauthorized adversary while it is transmitted over the network. The sink receives the data and performs the inverse transformation to retrieve the clear text as well as to prove the integrity of the data.

Definition 4.7. Let B be a finite set of binary encoded strings with a fixed length and let C be a finite set containing the elements $C = \{true, false\}$. Let g be a cryptographic transformation $g : T_1 \times \ldots \times T_n \times K \times A \rightarrow A \times B$ that is defined by $g(t_1, t_2, \ldots, t_n, k, p) = (c, s)$ and let v be a cryptographic transformation $v : U_1 \times \ldots \times U_m \times B \times K \times A \rightarrow C \times A$ that is defined by

$$v(u_1, u_2, \ldots, u_m, s, k, c) = \begin{cases} (true, p) & \text{if } g(t_1, t_2, \ldots, t_n, k, p) = (c, s) \\ (false, x) & \text{otherwise where } x \in A \end{cases}$$

where the following conditions are met:

- g is a cryptographic hash transformation,
- (g, v) is a symmetric cryptographic scheme,
- k is a shared secret key that is only known by the source and the sink.

[3] In literature, the term "authenticated encryption schemes" is commonly used for these kinds of cryptographic schemes. However, to avoid confusion with protocols for entity authentication, they are referred to as symmetric hybrid schemes throughout this dissertation.

This symmetric cryptographic scheme (g, v) is called *symmetric hybrid scheme*. Furthermore, g is called *hybrid generation transformation* and v is called the corresponding *hybrid verification transformation*.

Instances of hybrid schemes are normally block ciphers in combination with special modes of operation. Typical examples are Counter with CBC-MAC (CCM) [42], Offset Codebook Mode (OCB) [43], and EAX [44].

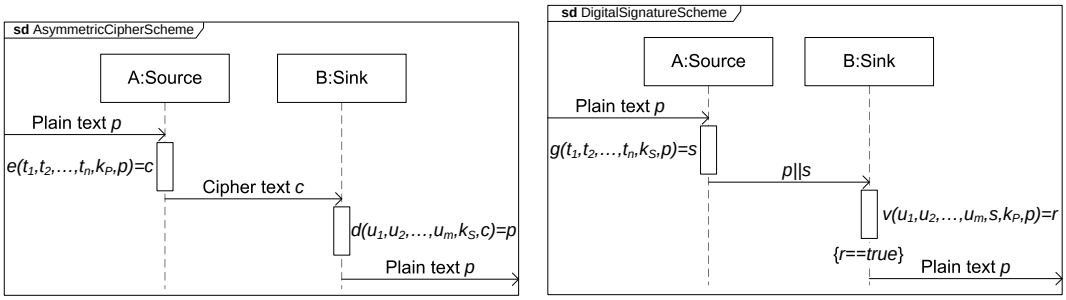
4.3 Asymmetric schemes

In contrast to symmetric schemes, asymmetric schemes are based on so called *key pairs* where one part is made public (*public key*) and the other one is only known by the source or by the sink (*private key*).

Definition 4.8. Let f_1 be a cryptographic transformation $f_1 : T_1 \times \ldots \times T_n \times K \times A \to B$ that is defined by $f_1(t_1, t_2, \ldots, t_n, k_{f_1}, a) = b$ and let f_2 be a cryptographic transformation $f_2 : U_1 \times \ldots \times U_m \times K \times B \to C$ that is defined by $f_2(u_1, u_2, \ldots, u_m, k_{f_2}, b) = c$. A cryptographic scheme consisting of f_1 and f_2 is called *asymmetric* if and only if the following conditions are met:

- if k_{f_1} has been made publicly available (referred to as public key) then
  - k_{f_2} is only known by the sink (referred to as private key),
  - it is impossible to calculate k_{f_2} given k_{f_1},
  - it is impossible to calculate $f_2(u_1, u_2, \ldots, u_n, k_{f_2}, b)$ for any $b \in B$ without knowing k_{f_2},
  - it is impossible to derive k_{f_2} given zero or more valid (b, c) pairs.
- if k_{f_2} has been made publicly available (referred to as public key) then
  - k_{f_1} is only known by the source (referred to as private key),
  - it is impossible to calculate k_{f_1} given k_{f_2},
  - it is impossible to calculate $f_1(t_1, t_2, \ldots, t_n, k_{f_1}, a)$ for any $a \in A$ without knowing k_{f_1},
  - it is impossible to derive k_{f_1} given zero or more valid (a, b) pairs.

Here, it is required that either k_{f_1} or k_{f_2} is kept private. If a private key gets compromised i.e., an external entity knows it, the asymmetric cryptographic scheme is no longer considered as valid (i.e., secure). In contrast to the shared secret keys of symmetric schemes, a private key is only known by a single entity. Therefore, a key pair



(a) Asymmetric cipher scheme (b) Digital signature scheme

Figure 4.3: Asymmetric cryptographic schemes

consisting of the private key and its corresponding public key is always dedicated to one particular entity.

The generation of a key pair that fulfills the requirements of asymmetric cryptographic schemes is based on number-theoretic problems that are assumed to be intractable. Most asymmetric cryptographic schemes are based on the *Integer Factorization Problem (IFP)* (e.g., RSA scheme [45]), the *Discrete Logarithm Problem (DLP)* (e.g., Diffie-Hellman (DH) key generation [46], ElGamal scheme [47]), or the *Elliptic Curve Discrete Logarithm Problem (ECDLP)* (e.g., Elliptic Curve Cryptography (ECC) scheme [48, 49]).

Representatives of asymmetric schemes can further be divided into two classes. *Asymmetric cipher schemes* protect confidential data in a way that unauthorized entities are not able to interpret the data's semantic without proper knowledge of the private key. Figure 4.3(a) shows how cipher schemes based on asymmetric concepts work. The source uses a so called *asymmetric encryption transformation* to generate cipher text out of the plain text using the public key of the sink and other security tokens. The cipher text is then transmitted to the sink which uses the so called *asymmetric decryption transformation* to derive the plain text using its private key and other security tokens.

Definition 4.9. Let e be a cryptographic transformation $e : T_1 \times \ldots \times T_n \times K \times A \to A$ that is bijective and defined by $e(t_1, t_2, \ldots, t_n, k_p, p) = c$ and let d be a cryptographic transformation $d : U_1 \times \ldots \times U_m \times K \times A \to A$ that is bijective and defined by $d(u_1, u_2, \ldots, u_m, k_s, c) = p$ where (e, d) is an asymmetric cryptographic scheme and (k_p, k_s) is the key pair of the sink (i.e., k_s is only known by the sink). The cryptographic scheme (e, d) is called an *asymmetric cipher scheme* if and only if

$$p = d(u_1, u_2, \ldots, u_m, k_s, e(t_1, t_2, \ldots, t_n, k_p, p))$$

for any given $t_1, t_2, \ldots, t_n$, $u_1, u_2, \ldots, u_m$, for a given key pair consisting of a private key $k_s \in K$ and a public key $k_p \in K$, and for all $p \in A$. p is called *plain text*, c is called *cipher*

text, e is called *asymmetric encryption transformation*, and d is called the corresponding *asymmetric decryption transformation*.

Typical examples of asymmetric cipher schemes that are considered as secure are the RSA encryption scheme [45] (based on IFP), the ElGamal encryption scheme [47] (based on DLP), and the Elliptic Curve Integrated Encryption Scheme (ECIES) [50] which is the ECC variant of the ElGamal encryption scheme.

Digital signature schemes, on the other hand, are the asymmetric counterpart to MAC schemes. They protect the data in a way that a modification by an unauthorized entity can be detected. Figure 4.3(b) shows the basic concept of digital signatures. The source uses a so called *signature generation transformation* to calculate some piece of data with a fixed length (called *digital signature*) out of the data using its private key and other security tokens. The output is transmitted together with the data to the sink where the sink uses a so called *signature verification transformation* that proves the authenticity of the data using the public key of the source, the received digital signature, and other security tokens.

Definition 4.10. Let B be a finite set of binary encoded strings with a fixed length and let C be a finite set containing the elements $C = \{true, false\}$. Let g be a cryptographic transformation $g : T_1 \times \ldots \times T_n \times K \times A \to B$ that is defined by $g(t_1, t_2, \ldots, t_n, k_s, p) = s$ and let v be a cryptographic transformation $v : U_1 \times \ldots \times U_m \times B \times K \times A \to C$ that is defined by

$$v(u_1, u_2, \ldots, u_m, s, k_p, p) = \begin{cases} true, & \text{if } g(t_1, t_2, \ldots, t_n, k_s, p) = s \\ false, & \text{otherwise.} \end{cases}$$

where the following conditions are met:

- g is a cryptographic hash transformation,
- (g, v) is an asymmetric cryptographic scheme,
- (k_p, k_s) is the key pair of the sink i.e., k_s is only known by the sink.

This asymmetric cryptographic scheme (g, v) is named *digital signature scheme*. Furthermore, g is called *digital signature generation transformation*, h is called *digital signature verification transformation*, and s is called the *digital signature* of p.

This general digital signature scheme demands that the plain text of the message is also transmitted to the sink. In literature, these kinds of digital signature schemes are called *digital signature schemes with appendix*. Alternatively, there are schemes where the plain text is not required. These are called *digital signature schemes with message recovery* [21]. However, in practice, digital signature schemes with appendix are more common.

Furthermore, it can be shown that a digital signature scheme with message recovery can be converted into a scheme with appendix. Typical examples of digital signature schemes with appendix are RSA signature scheme [45], Digital Signature Algorithm (DSA) [51], and Elliptic Curve Digital Signature Algorithm (ECDSA) [50] which is the ECC variant of DSA. For these digital signature schemes, variants that provide schemes with message recovery also exist.

Compared to symmetric cryptographic schemes, there are no hybrid schemes that are based on asymmetric transformations.

Theorem 4.11. A hybrid cryptographic scheme is always an instance of a symmetric cryptographic scheme.

Proof. Assume to the contrary that a hybrid scheme is an asymmetric scheme. Due to the definition of an asymmetric scheme, at least all security tokens of the cryptographic transformation at the source site are public or at least all security tokens of cryptographic transformation at the sink site are public. Let g be a hybrid transformation function $g : T_1 \times \ldots \times T_n \times A \to A \times B$ that is defined by $g(t_1, t_2, \ldots, t_n, p) = (c, s)$ and let v be a hybrid verification function $v : U_1 \times \ldots \times U_m \times B \times A \to C \times A$ that is defined by

$$v(u_1, u_2, \ldots, u_m, s, c) = \begin{cases} (true, p) & \text{if } g(t_1, t_2, \ldots, t_n, p) = (c, s) \\ (false, x) & \text{otherwise where } x \in A \end{cases}$$

where $p, c \in A$, $s \in B$ and $t_i \in T_i$ for $i = 1, 2, \ldots, n$ and $u_j \in U_j$ for $j = 1, 2, \ldots, m$. However, since it must be impossible to calculate s for any given $p \in A$ without full knowledge of all $t_1, t_2, \ldots, t_m$ and it must be impossible to calculate p for any given $c \in A$ without full knowledge of all $u_1, u_2, \ldots, u_m$, there must be a subset of $\{t_1, t_2, \ldots, t_n\}$ and a subset of $\{u_1, u_2, \ldots, u_m\}$ that has to be kept secret. However, this is contradicting the assumption that either all security tokens at the source site or all security tokens at the sink site are public. □

This conclusion is only true for asymmetric cryptographic schemes that use one single key pair for the secure transformation. Obviously, it is possible to composite two asymmetric cryptographic schemes to construct a cryptographic scheme that provides similar functionality as a hybrid scheme. Such schemes are called *asymmetric combination schemes*.

Definition 4.12. Let B be a finite set of binary encoded strings with a fixed length and let C be a finite set containing the elements $C = \{true, false\}$. Let g be a cry-

ptographic transformation $g : T_1 \times \ldots \times T_n \times K \times K \times A \rightarrow A \times B$ that is defined by $g(t_1, t_2, \ldots, t_n, k_{s1}, k_{p2}, p) = (c, s)$ and let v be a cryptographic transformation $v : U_1 \times \ldots \times U_m \times B \times K \times K \times A \rightarrow C \times A$ that is defined by

$$v(u_1, u_2, \ldots, u_m, s, k_{p1}, k_{s2}, c) = \begin{cases} (true, p) & \text{if } g(t_1, t_2, \ldots, t_n, k_{s1}, k_{p2}, p) = (c, s) \\ (false, x) & \text{otherwise where } x \in A \end{cases}$$

where (g, v) is an *asymmetric cryptographic scheme*. Furthermore, (k_{s1}, k_{p1}) and (k_{s2}, k_{p2}) $((k_{s1}, k_{p1}) \neq (k_{s2}, k_{p2}))$ are key pairs where the source owns (k_{s1}, k_{p1}) and the sink owns (k_{s2}, k_{p2}). This asymmetric cryptographic scheme is called *asymmetric combination scheme*.

Note that combination schemes can also be based on symmetric schemes. Here, two shared secret keys that are not identically are used as input parameters.

4.4 Time variant parameter

A Time Variant Parameter (TVP) is a mathematical data item commonly used in cryptography. Due to the time variant property, TVPs mainly act as input parameters that provide timeliness of information. Therefore, TVPs are essential for guaranteeing data freshness as well as temporal ordering of events.

Definition 4.13. A TVP is a parameter that fulfills the following condition: at any logical point in time, the present value of a TVP is always different to all values at preceding logical points in time.

Note that this definition only demands the notation of a logical clock. Events that trigger such a logical clock can, for example, be the sending or receiving of a message or tick events of a synchronized clock. In literature, TVPs are also called *nonces*[4], unique numbers, or non-repeating values. In practice, different types of TVPs can be used. The most common ones are (pseudo) random numbers, monotonically increasing counters, and timestamps.

4.5 Formal evaluation

The key concept of cryptographic schemes is that they rely on the assumption that it is "impossible" to perform specific mathematical operations. However, in the previous

[4]Nonce is an abbreviation for "number used only once".

definitions, the term “impossible” has been used in a quite informal way.

A more formal definition can be given by introducing models that are used to evaluate the impossibility of the mathematical operations. In [21], five different evaluation models are defined (listed with decreasing level of security):

- *Unconditional security*: It is assumed that the adversary has unlimited computational power. The impossibility of the calculation relies on the missing knowledge of the domain. Since unconditional security offers the best level of security, it is also referred to as perfect security. A typical example is a symmetric scheme that uses one-time keys that are at least as long as the plain text (one-time-pad scheme).
- *Complexity-theoretic security*: Here, it is assumed that adversaries have polynomial resources concerning available time and memory space. Based on this adversary model, a formal proof is given. Complexity-theoretic security provides a very high level of security since polynomial attacks may still be too resource-consuming in real world applications.
- *Provable security*: Using this evaluation model, the formal proof relies on a well-known mathematical problem that is assumed to be intractable. A typical example would the use of number-theoretic problems like the IFP.
- *Computational security*: Considering the best-known algorithms, a cryptographic transformation is called computationally secure if the required level of resources exceeds the available resources of an adversary that uses state of the art technologies. Again, the evaluation often relies on a number-theoretic problem. However, compared to provable security, no formal proof exists.
- *Ad-hoc security*: Ad-hoc security provides the weakest level of security. Here, possible threats are analyzed where adversaries with defined resources are assumed. Based on this analysis, arguments that are contrary to these threats are listed.

5 State of the Art

In this chapter, the most important state of the art technologies are analyzed. At the beginning, protocol standards from the HBA domain with a focus on their support for security are examined. Then, security standards from other domains that are relevant for HBA systems are presented.

5.1 Security in home and building automation standards

Today, many different HBA protocol standards exist (cf. Figure 5.1). The most important open ones are BACnet [52, 53, 54], LonWorks [55, 56, 57], KNX [58, 59], and ZigBee [60]. All of these four standards span more than one application domain and so they can be considered as a reasonable solution for all-in-one systems. The provided security concepts of these four protocols are presented in this chapter.

In addition to these protocols, stand alone solutions that are dedicated to a specific application domain exist. Digital Addressable Lighting Interface (DALI) [61] and Meter-Bus (M-Bus) [62] are primarily established at the field level. While DALI is dedicated to the lighting domain, M-Bus is exclusively used for metering. The main application domain of Modbus [63] is to provide the opportunity for communication between Programmable Logic Controllers (PLCs) and DDCs. As a result, Modbus is primarily established at the automation level. However, since all three standards do not incorporate any security mechanism, they are not further considered here.

Two examples of wireless technologies that are mainly used at the field level are EnOcean [64] and Z-Wave [65]. EnOcean uses energy harvesting techniques which allows devices to work without batteries. Since energy efficiency is a key feature in EnOcean, it does not support any security mechanisms. Z-Wave, on the other hand, provides security features. However, since the protocol and thus its security concept is not publicly available, an analysis of the security mechanisms of Z-Wave is not possible. Furthermore,

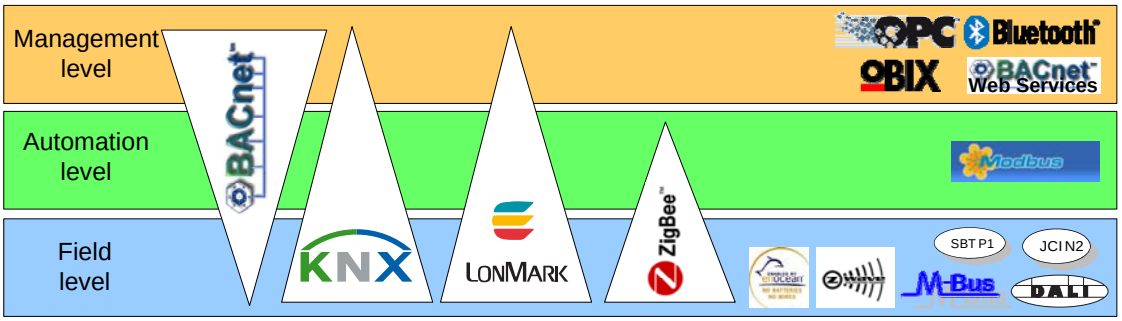


Figure 5.1: HBA standards

numerous special systems for various domains exist. Two typical examples for the HVAC domain are Johnson Controls (Metasys) N2 (JCI N2) and Siemens Building Technologies (Landis) P1 (SBT P1). Again, since these systems are not publicly available, an analysis concerning security is not possible. However, without proof, it is expected to be that they rely in best case on "Security by Obscurity" and thus do not provide a reasonable protection against security attacks.

At the management level, various technologies exist, too. OPC with its former version OPC Data Access (OPC DA) [66] and its new version OPC Unified Architecture (OPC UA) [67] are used to provide interoperability at the management level. While OPC is mainly established in industrial automation, it has also been used in the HBA domain recently. OPC UA incorporates a security concept which will be discussed in Section 5.1.6.

Open Building Information eXchange (oBIX) [68] is an open standard dedicated to the HBA domain. It is also used to provide interoperability of heterogeneous networks. In oBIX, HyperText Transfer Protocol (HTTP) and Simple Object Access Protocol (SOAP) can be used as transmission protocol. In the current oBIX specification, a separate security concept is not specified – supporting security services is left to the implementation. However, the use of standardized concepts like Transport Layer Security (TLS) for HTTP and the security concept for Web Services Interoperability (WS-I) Basic Profile 1.0 [69] for SOAP (which is also based on TLS) are suggested. For more information on TLS see Section 5.2.1.

BACnet Web Services (BACnet/WS) [52] is a generic protocol for accessing data at the management level using Web services. It is part of the current BACnet specification. While the name BACnet/WS may be misleading, BACnet/WS can be used in combination with any other protocols thanks to its generic application model. Therefore, its use is not limited to BACnet networks. BACnet/WS is based on WS-I Basic Profile 1.0 [69] which uses SOAP and HTTP as transmission protocol. To secure data transmission, it relies

on the use of TLS rather than defining its own security concept. However, using TLS to protect BACnet/WS is optional.

Finally, Bluetooth that is well-established in the IT world may also be used in the HBA domain. The main field of application is for management purposes (e.g., an MD connects to an HBA system to perform diagnostic tasks) and user interaction (e.g., a remote control panel using a Bluetooth connection). In contrast to other standards at the management level, Bluetooth is mainly used as simple transmission protocol to access the HBA network rather than as a fully fledged management solution. Bluetooth's security is discussed in Section 5.1.5.

5.1.1 BACnet

In 1987, the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) project committee began with the development of the Building Automation and Control network protocol called BACnet. The main objective was to provide a solution for BA systems of all sizes and types. In 1995, the development was finished and BACnet was published as ANSI/ASHRAE standard 135. Later in 2003, BACnet became ISO 16484-5 standard. Since the first release, the BACnet specification is under continuous development. Extensions to the current BACnet standard are summarized in so called BACnet Addenda. If ASHRAE decides to launch a new standard release, all final addenda are incorporated into the current standard and a new standard version is published. The current version is ANSI/ASHRAE 135-2008 (also called BACnet 2008) [52]. The current ISO standard is ISO 16484-5:2007 (which includes BACnet 2004) [53] and ISO 16484-5:2007/Amd 1:2009 (which includes the differences from BACnet 2004 to BACnet 2008) [54]. Methods of testing for conformance to BACnet have also been standardized by ASHRAE. These conformance tests are specified in ANSI/ASHRAE standard 135.1 and ISO 16484-6. The current version of BACnet conformance tests is laid down in ANSI/ASHRAE 135.1-2007 [70] and ISO 16484-6:2009 [71].

BACnet[1] offers several services which pretend to provide support for data confidentiality, data origin authentication (and thus data integrity), and data freshness as well as an entity authentication service [52, 72]. Authorization is provided on a per-device basis. The security mechanisms are based on DES and a trusted key server which is responsible for generating and distributing session keys. These session keys are used to secure the transmitted data between two network nodes. To establish a secure connection to the key

[1]For the rest of this dissertation, BACnet is used as synonym for the current version BACnet 2008.

server, each node must own a secret key.

BACnet suffers the following general security flaws [72, 73, 74]:

- The generation and distribution of the initial secret keys are not defined in the BACnet standard. These mechanisms are considered as "local matters".
- The freshness of the session keys cannot be guaranteed. During session establishment, the device adds the retrieved session key to the list of valid session keys. The BACnet specification does not force a limitation of the lifetime of session keys. Hence, an adversary can use an old session key to communicate with a particular device.
- The implementation of the key server is not defined by the BACnet standard. Since the key server holds a copy of all secret keys, it is obvious that the key server must be protected against all kinds of malicious attacks. Furthermore, it is only possible to use a single key server which may lead to a single point of failure.
- DES is the only supported encryption algorithm. Since DES uses short keys (56 bit), brute force attacks can be performed to find valid keys [75].

Additionally, [73] and [74] describe the following security flaws of the authentication service: man-in-the-middle attacks, type flaws, parallel interleaving attacks, replay attacks, and implementation dependent flaws.

Due to these security flaws, BACnet does not provide a reasonable security concept for security-critical environments. Therefore, in March 2007, Addendum g [76] was proposed to address the security flaws of the original BACnet security concept. At the time of writing, BACnet Addendum g has finished the 4th public review process and is now waiting for final publication.

The new security concept of BACnet Addendum g completely replaces the old, vulnerable security services of BACnet. It specifies security services that are designed to be applicable to all BACnet media and device types. To protect the transmitted data, symmetric cryptographic schemes are used exclusively. The required shared secret keys have to be distributed to the secure devices in advance or they have to be retrieved from a so called key server during runtime. In BACnet Addendum g, six different key types are distinguished:

- `General-Network-Access` key: This key is shared between all members of a network. It is used, for example, to protect services such as device discovery.
- `User-Authenticated` key: In requests secured with this key, the user identity can be assumed to be properly authenticated. The user authentication has to be performed by an external authentication mechanism (e.g., via a user interface

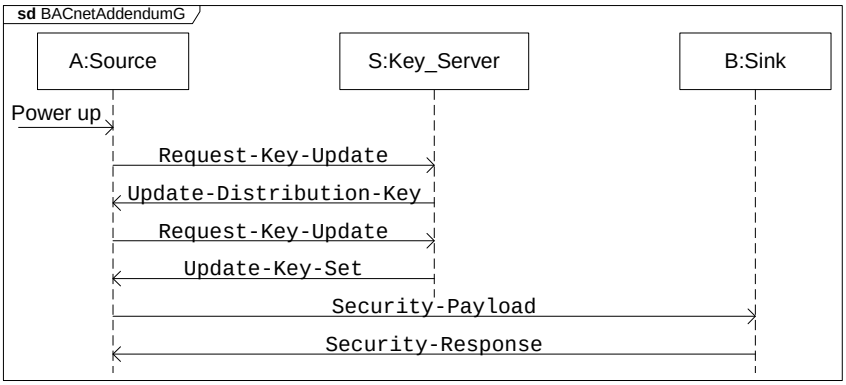


Figure 5.2: Security services in BACnet Addendum g

where the user enters a password). Alternatively, if a device does not have the necessary capabilities (e.g., it does not have a user interface), the user identity can be configured directly at the device (e.g., in its EPROM).

- `Application-Specific` keys: Keys of this type are dedicated to a certain application domain (e.g., HVAC or access control). `Application-Specific` keys are only distributed to a subset of devices that require a higher level of security.
- `Installation` keys: These keys are temporally used for configuration and maintenance purposes. Installation keys are, for example, used by configuration tools to temporally access a BACnet network to perform configuration and maintenance tasks.
- `Distribution` keys: Beside the possibility to use keys that are permanently stored within a device, keys can be retrieved from an online key server during runtime. To securely communicate with this key server, distribution keys are used.
- `Device-Master` keys: They are only used to retrieve `Distribution` keys. Since `Device-Master` keys act as initial secret, their distribution must be done within a physically secured environment.

BACnet Addendum g specifies various secure communication services that are incorporated into the network layer of BACnet. In particular, eight new security services are defined. The `Security-Payload` service is used to encapsulate BACnet network and application layer messages and to securely transmit them over the network. To respond to these secured messages, the `Security-Response` service is available. A `Security-Response` message indicates either the successful retrieval of a secured message or an error condition. The `Challenge-Request` service is used to verify the identity of a device. The device that is challenged has to answer with a `Security-Response` message that contains the result of the challenge. To request

the distribution of the secret keys from the key server, the `Request-Key-Update` service is available. Upon retrieval of a `Request-Key-Update`, the key server responds with an `Update-Key-Set` or with an `Update-Distribution-Key` message which contains the requested key set. These two services can also be used by the key server to force key changes without being requested by a device. Finally, the `Request-Master-Key` and `Set-Master-Key` are used to change the `Device-Master` key. However, since these two services are not secured at all, their use has to be limited to physically secured environments. Figure 5.2 shows an example how these security services can be used. After having powered up, device A requests a `Distribution` key from the key server S by sending a `Request-Key-Update` message (secured with its `Device-Master` key). The key server validates the request and transmits a newly created `Distribution` key to A using the `Update-Distribution-Key` service. Afterwards, A sends another `Request-Key-Update` message to retrieve the current keys. This request is secured using the `Distribution` key retrieved before. After having received the key set from the key server, A is now able to securely communicate with device B using the appropriate key. Note that it is assumed that device B is also in possession of the used key (e.g., `General-Network-Access` or `Application-Specific` key).

Network messages are classified into *plain*, *signed*, and *encrypted* messages. While plain messages are not secured at all, signed messages provide data integrity and data freshness. To guarantee data integrity, HMAC in combination with MD5 or SHA is used. Data freshness is achieved by using a timestamp (32 bit standard UNIX timestamp) in combination with a 32 bit message ID. Encrypted messages are additionally encrypted using AES in CBC mode. Entity authentication is implicitly guaranteed due to the used symmetric algorithms and due to the use of so called device instance numbers. Device instance numbers uniquely identify secure BACnet devices and are assigned to the devices independently of their BACnet addresses since BACnet device addresses may be changed during runtime.

In BACnet Addendum g, BACnet networks are classified according to their so called *network trust level*: `Trusted` networks are physically secured or make use of security services. Devices within these networks fully trust each other. `Non-trusted` networks, on the other hand, are networks where the exchanged messages cannot be inherently trusted. In these networks, neither physical security is guaranteed nor security services are used. Based on these two different network trust levels, four different *network security policies* are specified (listed in increasing order of security policy level):

- `Plain-Non-Trusted`: In these networks, the transmitted data is not secured at all. Neither protocol security services nor mechanisms that provide physical security are used there. Therefore, data exchanged in these networks cannot be trusted.
- `Plain-Trusted`: These networks are physically secured in a way that the use of protocol security is not required. The exchanged data is trusted even if the located devices exchange plain messages.
- `Signed-Trusted`: Messages in `Signed-Trusted` networks are at least signed i.e., data integrity and data freshness is guaranteed. Therefore, the exchange of plain messages is not allowed in these networks.
- `Encrypted-Trusted`: In `Encrypted-Trusted` networks, all messages must be encrypted, too. This means that plain and signed messages are not allowed and therefore, they are silently dropped by the devices.

A network security policy level defines the minimum level of security a message must satisfy on a given network. Messages with a lower security policy level are not allowed within the network. However, each device may decide on its own to require a higher security policy level than the one demanded by the network. This so called *device security level* can be specified on a per-device basis and is used for providing end-to-end security. Furthermore, it is even possible to require a higher security level on a per-service, per-object, or per-property basis. However, the realization of such detailed security policies as well as the associated access control lists are not defined by the standard.

BACnet Addendum g provides a solid base for securing HBA systems. However, there is much room for improvement since the following aspects are missing or left open:

- The distribution of the keys is handled by the key server. The actual distribution to the devices predefines which devices are able to communicate with each other and which devices are excluded from a secure relationship. To distinguish between different secure relationships, the use of the multiple keys (one for each relationship) is necessary. However, the assignment of the keys to the communication relationships is not specified by the standard and thus it is left to the application. This also includes authorization which has to be realized by the application itself.
- Guaranteeing data origin authentication is only possible if a key is limited to two devices. If, for example, the `General-Network-Access` key or an `Application` key that is distributed to multiple devices is used, the sender cannot be identified in a secure manner.
- Due to performance reasons, the distribution of the shared secret keys is based on

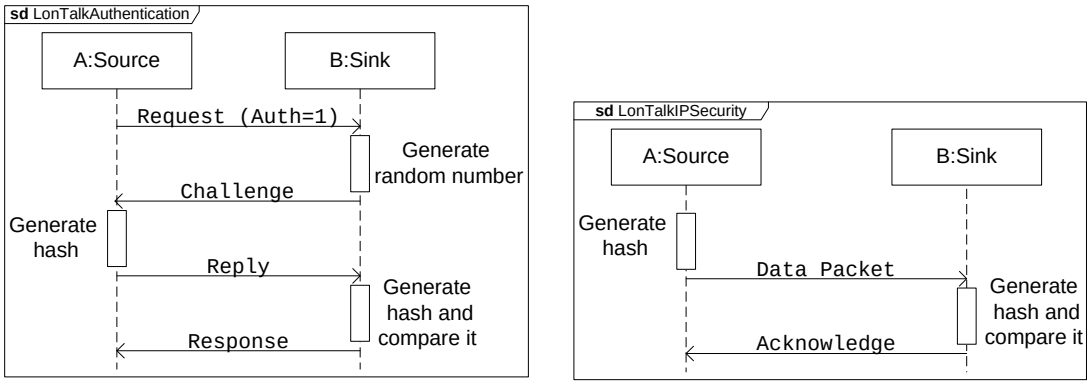
symmetric schemes exclusively. To avoid the use of permanently stored keys, an online key server is required. However, nowadays asymmetric schemes based on ECC are suitable for embedded devices and eliminate the need for a trusted, online key server [77, 78].

- The use of a single key server introduces a single point of failure. Therefore, a scheme based on multiple key servers is desirable. While the use of multiple key servers is possible, the realization of such a concept is not specified in BACnet Addendum g. Important details like synchronization of key servers and the selection of the key server to be used (especially in case of a faulty key server) are not discussed.
- To support all kinds of applications, the use of different communication models shall be possible. BACnet only provides support for the client/server model – exchanging control data within groups is not supported.
- The new security mechanisms of BACnet Addendum g require the existence of (loosely) synchronized device clocks. Otherwise, data freshness cannot be guaranteed since the used mechanisms rely on timestamps.
- Finally, mechanisms to protect against interruption attacks (e.g., DoS attacks) are not supported. Therefore, data availability cannot be guaranteed.

5.1.2 LonWorks

LonWorks was developed by Echelon Corp. The LonWorks system consists of the LonTalk communication protocol, a specific microcontroller called Neuron Chip, and a network management tool called LonWorks Network Services (LNS). In 1999, LonTalk has become the formal standard ANSI/EIA-709. The current version is ANSI/EIA-709 Rev. B [55]. In 2004, the use of IP as tunneling medium for the LonTalk protocol has been specified as ANSI/EIA-852 [79]. Additionally, in 2005, LonTalk has also been approved as European standard EN 14908 [56]. Finally, the release of LonTalk as ISO/IEC 14908 [57] is currently under way.

LonTalk provides a rudimentary security concept based on a four step challenge-response protocol. During this protocol, the identity of the sender is verified. Furthermore, it pretends to guarantee data integrity and freshness. Figure 5.3(a) shows the different steps: a sender which desires to secure a request sets the so called authentication bit of the corresponding message. After having received this request, all receivers reply with a 64 bit random number. The sender receives these random numbers and individually calculates a 64 bit hash value over the content of the message and the random number



(a) Four step challenge-response protocol (b) IP tunneling security mechanism

Figure 5.3: LonTalk security mechanisms

using a shared secret key. These hash values are sent back to the receivers where the same calculation is performed and compared with the previously received value.

In [72, 80], the following security flaws are described:

- Disclosure of confidential data cannot be avoided, since the data is transmitted in clear text.
- The used challenge-response protocol only supports the verification of the sender's identity. The identity of the receiver cannot be checked. Therefore, mutual entity authentication is not guaranteed.
- The usage of the authentication protocol is restricted to acknowledged unicast and multicast. If a broadcast or an unacknowledged transmission mode is used, the identity of the sender cannot be verified.
- It is not possible to establish communication sessions. Thus, it is always necessary to transmit four messages for secured requests, even if a sender transmits multiple data messages to the same receiver(s) in sequence.
- Using authenticated multicast, each receiver generates its own random number and sends it to the sender. As a result, the sender must respond to all receivers with an individual calculated hash value. If a LonTalk communication group contains n members, the sender must calculate $n - 1$ hash values.
- The authentication protocol is vulnerable to DoS attacks. To start a flooding attack, an adversary sends a number of messages with the authentication bit set. For each message, the receiver will generate a random number and calculate the necessary hash value. Since this is time-consuming, this will result in a DoS attack.
- The length of the used shared secret keys is limited to 48 bits which is too short to avoid brute force attacks.
- Only the data portion of the application layer is used as input for the hash calcula-

tion. Headers from other layers including the address information are not protected.

- The LonTalk protocol does not provide a mechanism to distribute the secret keys in a secure manner. Hence, key distribution has to be performed in a physically secured environment to prevent interception.
- Each device can only use one authentication key. This means that all devices that want to communicate with each other must share the same secret key. As a result, data origin authentication cannot be guaranteed in networks with more than two members.
- Authorization is not supported since the same key is used for all LonTalk services and devices.
- The challenge-response protocol can only be initiated by the sender. A receiver does not have the opportunity to demand secured requests.
- There are no countermeasures that avoid interruption attacks. Thus, data availability cannot be guaranteed.

In addition to the basic challenge-response protocol, the IP tunneling scheme of LonTalk defines its own security mechanism (cf. Figure 5.3(b)). It uses MD5 together with a shared secret to calculate a hash value. This hash value is sent together with the message to the intended receiver(s). After having received a secured message, the receiver calculates its own hash value using the same shared secret and compares it with the received one. If both values are equal, the message is accepted – otherwise it is discarded. Note that the four step challenge-response mechanism mentioned above is not used here.

However, due to the following reasons, the used mechanism is insecure, too:

- MD5 is not collision resistant and thus it should not be used anymore.
- Data confidentiality is not provided since the message is transmitted in clear.
- Data freshness is not guaranteed since message replays cannot be detected due to the absence of a nonce (e.g., random number).
- The management and distribution of the used shared secrets are not defined. Furthermore, the length of the shared secret is not specified – it is only demanded that it has at least 128 degrees of freedom (e.g., 16 byte binary key). The exact length has to be defined by the application.
- The security mechanism is only optional. Therefore, a receiver is not able to force the use of secured messages.
- While it is possible to use different shared keys for different devices, using multiple keys is not specified by the standard. Therefore, authorization is not supported in a native way.

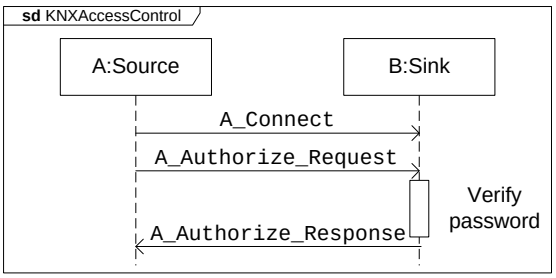


Figure 5.4: Access control mechanism of KNX

- Data availability is not guaranteed since interruption attacks cannot be avoided.

5.1.3 KNX

The European Installation Bus (EIB) was developed as a fieldbus for electrical installations in homes and buildings. Until 2002, the corresponding EIB specification was maintained by the EIB Association. In 2002, EIB was merged with Batibus and European Home System (EHS) and the KNX standard was defined. Additionally, the Konnex Association was formed which is responsible for the maintenance of the KNX specification as well as for promotional activities and the certification of test labs and training centers. In 2003, KNX has been approved as European Standard EN 50090 [81]. In 2006, KNX has also been released as ISO/IEC 14543-3 [58]. The current standard is KNX Standard Version 2.0 [59].

KNX only provides a basic access protection scheme based on clear text passwords (cf. Figure 5.4). Up to 255 different access levels can be defined, each of them is associated with a different (otherwise unspecified) set of privileges. Access level 0 has the highest privilege and access level 255 is the lowest one. For each of these access levels, a 4 byte password can be specified. This scheme is only available for engineering communication. Control data exchange remains insecure.

Since this access protection is very rudimentary, KNX does not provide the necessary mechanisms to guarantee a secure environment. Furthermore, it suffers the following general security flaws [80]:

- Data confidentiality, data integrity, data origin authentication, and data freshness are not guaranteed at all.
- Mutual entity authentication is not provided since the identity of the receiver is not verified.
- The passwords are transmitted in clear text. If an adversary has access to the network, the adversary can simply intercept and retrieve the transmitted password.

- KNX does not support mechanisms to manage, generate, and distribute passwords in a secure manner. Therefore, the passwords must be specified manually. It is up to the system administrator to guarantee that this configuration is performed in a physically secured environment.
- To configure and maintain a KNX network, a single management tool called ETS is used. However, it does not make full use of the access protection mechanism (e.g., only one password is pertained for the whole installation). Hence, the rudimentary access protection scheme cannot be fully used.
- The access protection mechanism cannot be applied to control data communication. An unauthorized use of these services cannot be avoided.
- Parallel connections are not supported: if a device has established a connection with a particular device, all other connection requests are ignored. An adversary can use this restriction to perform a DoS attack.
- As the source address of a transmitted message can be spoofed very easily, an adversary can simply inject malicious messages without knowing the password.
- Data availability is not guaranteed since interruption attacks cannot be avoided.

To be able to use IP networks for KNX installations, KNXnet/IP has been introduced. In the corresponding specification [59], some rudimentary security guidelines are presented in addition to the access protection mechanism mentioned above. These guidelines are based on network isolation (e.g., use of firewalls or KNXnet/IP only Intranets) and on "Security by Obscurity" (e.g., use of non-standard IP addresses, rely on the missing expertise of an attacker). Since preventing physical access to the network by isolation is not always possible (e.g., Wireless Local Area Networks (WLANs)) and "Security by Obscurity" is a technique that (if at all) provides only temporary protection, neither the access protection mechanism nor the security guidelines provide an effective protection.

In addition to the rudimentary security concept of KNX, there are non-standardized security extensions for KNX available. The most important one is called Secure EIB (SEIB) [82]. SEIB provides data confidentiality, integrity, and freshness for control data communication. It is based on the Secure Network Encryption Protocol (SNEP) protocol which is part of the Secure Protocol for Sensor Networks called SPINS [22]. SEIB uses AES to encrypt the content of group messages and a 32 bit Cyclic Redundancy Check (CRC) checksum to detect unauthorized modifications. To protect the communication against replay attacks, a 128 bit counter is used. However, a number of problems still remain unsolved. Since SEIB is only available for control data communication, engineering communication remains unprotected. Furthermore, the used key management is very rudimentary. There

are no mechanisms to change secret keys and thus, compromised keys cannot be revoked during runtime.

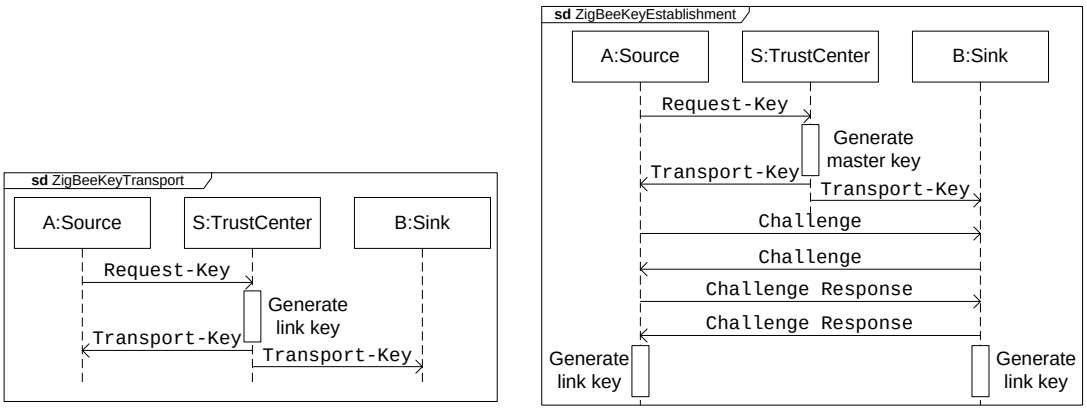
5.1.4 ZigBee

In 2003, IEEE 802.15.4 has been released as an open standard for wireless communication in sensor and actuator networks. IEEE 802.15.4 specifies a flexible and powerful protocol stack that is suitable for low cost and low power embedded devices. The current version is IEEE 802.15.4-2006 [83] which is fully backward-compatible to the original standard also known as IEEE 802.15.4-2003. In addition to the core specification, two additional physical layers specified as IEEE 802.15.4a-2007 are also available [84].

IEEE 802.15.4-2006 only specifies the data link layer and the physical layer. Therefore, it is mainly used as base protocol for other wireless standards like WirelessHART [85], 6LoWPAN [86], and ZigBee. However, IEEE 802.15.4-2006 already implements a basic security concept. This concept provides 8 different security levels which can individually be chosen on a per-device basis. To protect the transmitted data, AES in combination with CCM* (a minor variant of CCM mode) is available – in contrast to IEEE 802.15.4-2003 where AES in CBC mode is used. Depending on the chosen security level, AES in combination with CCM* guarantees data integrity, data freshness and/or data confidentiality.

While the security concept of IEEE 802.15.4-2006 provides reasonable mechanisms to guarantee a secured channel, the management of the required shared secret keys is not defined by the standard – it is left up to the higher protocol layers. This includes the generation and distribution of the shared secret keys as well as their alteration during runtime. Therefore, other security objectives like data origin authentication, entity authentication as well as authorization can only be guaranteed in combination with an adequate key management.

ZigBee is the most well-known protocol that builds upon IEEE 802.15.4. ZigBee uses the data link layer of IEEE 802.15.4 and enhances the available features by specifying an application layer and a network layer. Additionally, new services that provide the opportunity for multi-hop routing and advanced security services have been added. The first ZigBee specification was released in 2004 by an industry group called ZigBee Alliance. In 2006, a new ZigBee specification has been released which is not backward compatible to the original specification of 2004. The current valid specification is ZigBee 2007 [60] which is fully backward compatible to ZigBee 2006. Note that the core specification of ZigBee only defines the network layer and the application layer. Instances of the applica-



(a) Key transport (b) Key establishment

Figure 5.5: ZigBee security mechanisms

tion model referred to as ZigBee profiles are defined by separate specifications. Typical examples are the ZigBee Home Automation Public Application Profile [87], the ZigBee Smart Energy Public Application Profile [88], as well as the non-public ZigBee Building Automation Application Profile.

Furthermore, it is important to note that the current specification ZigBee 2007 is based on IEEE 802.15.4-2003. Although, while ZigBee uses the transmission services of the data link layer of IEEE 802.15.4-2003, it defines its own security architecture that is independent from IEEE 802.15.4. Thus, the security services provided by IEEE 802.15.4-2003 are entirely not used.

The security concept of ZigBee is exclusively based on symmetric cryptographic schemes. In particular, AES in combination with CCM* is used. Entity authentication as well as data origin authentication, freshness, and confidentiality are provided at the network and/or application layer. Additionally, ZigBee provides services for management and distribution of the required shared secret keys. Depending on their use, ZigBee distinguishes three different key types. *Link keys* are shared between two devices. They are used to secure communication between them. *Network keys* provide security across the whole network segment. Finally, so called *master keys* are optionally available. Master keys are only required during the establishment of link keys.

Beside the possibility to manually install shared secret keys in advance, it is possible to retrieve secrets during runtime. This runtime distribution of shared secret keys is handled by a single entity called *Trust Center*. To exchange secret keys, three different distribution methods are available in ZigBee:

- *Pre-installation*: Here, the keys are uploaded to the devices before runtime. The exact method how pre-installation is performed is not defined by the ZigBee speci-

fication. Pre-installation can, for example, be done by the device manufacturers or by a proprietary management tool. The trust center is not involved in this distribution method.

- *Key-transport*: Using key-transport, the trust center sends the keys directly to the devices using a dedicated communication service. Key-transport is used to distribute the actual network key during the device joining process and to distribute link keys during runtime. Figure 5.5(a) shows an example how key-transport can be performed to distribute a link key. To retrieve a link key that is shared between two devices, the initiating device sends a `Request-Key` message to the trust center. The trust center generates a new link key and distributes it to both devices using a `Transport-Key` message. The message is secured with the trust center link key that is shared between the trust center and the corresponding devices.
- *Key-establishment*: Key-establishment is only available for link keys. In contrast to key-transport, both devices are involved in the key generation process. The key-establishment is performed using the so called Symmetric-Key Key Establishment (SKKE) protocol. Figure 5.5(b) shows the principle of this protocol. To start the key-establishment process, the initiating device sends a `Request-Key` message to the trust center. If the trust center is configured to use key-establishment instead of key-transport, it generates a master key which is distributed to both devices using the `Transport-Key` service. After having received the master key, the devices start the SKKE protocol. First, each device generates a random challenge that is sent to the other device. Using this challenge and the previously retrieved master key, each device calculates a challenge response which is sent to the other device. After having retrieved the challenge response, both devices verify it. If it is valid, a link key is calculated out of both challenges which can later be used to secure the communication between the two devices.

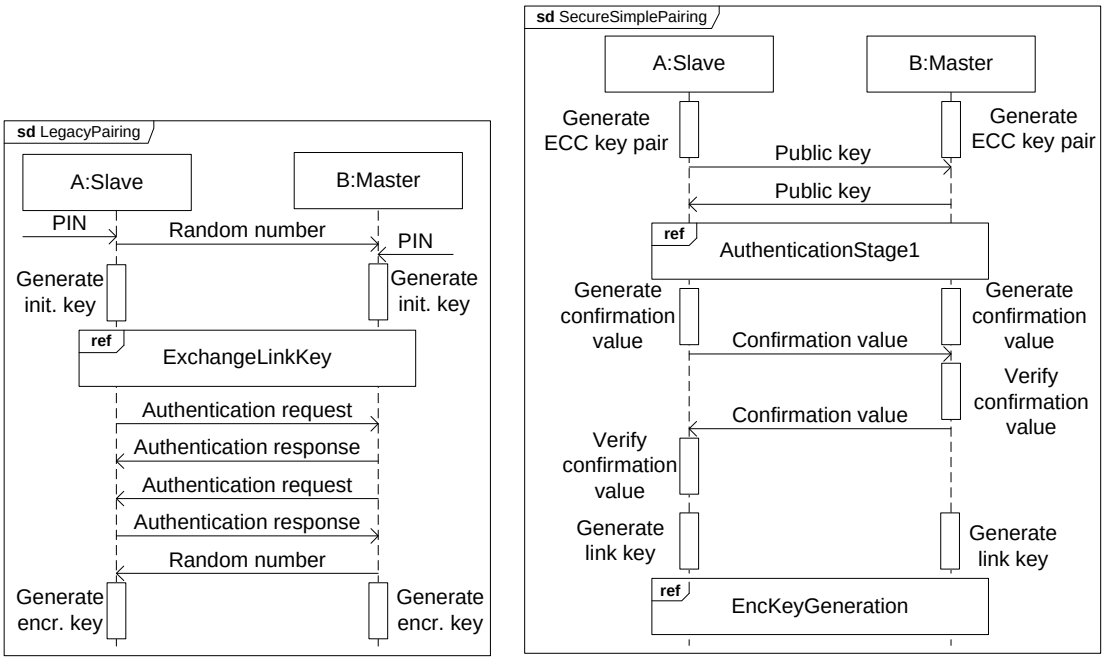
To be able to securely retrieve network, master, or link keys from a trust center, the requesting device must share a link or master key with the trust center. These initial trust center keys can either be pre-installed or distributed using unsecured key-transport messages. However, in the latter case, a malicious interference has to be avoided. Therefore, the exchange of insecure key-transport messages has to be done in a physically secure environment.

The security concept of ZigBee provides a solid base for securing HBA systems. However, there is ample room for improvement since the following aspects are missing or left open:

- Key management is handled by a single trust center which may result in a single point of failure. Furthermore, in wide-range networks, multiple hops may be necessary to reach the trust center. Therefore, a security concept based on multiple trust centers is desirable.
- As mentioned above, the security services provided by IEEE 802.15.4 are not used by ZigBee. As a result, the data link header is not secured since ZigBee only protects the network and/or application layer parts of the messages. Furthermore, data link layer services like sending beacon frames and associate requests are not secured. As a result, security threats that are dedicated to the data link header or to data link services cannot be avoided (e.g., re-routing of network traffic, sending of malicious beacons).
- The smallest security context in ZigBee is a device. Using different secret keys for different user applications on a single device is not possible. Therefore, access control is only provided on a per-device basis.
- While ZigBee defines a multicast communication service, it is not clear how group communication is secured in ZigBee. It seems that the only possibility is to use the network key. However, a secure separation between different multicast groups is not possible if the network key is used. Furthermore, data origin authentication cannot be guaranteed. Link keys cannot be used to secure multicast communication, since link keys can only be shared between two devices.
- Interruption threats are not considered in ZigBee. Especially the joining procedure is vulnerable to DoS attacks. The first part of the joining process (i.e., address assignment, synchronization with ZigBee coordinator) is not secured since entity authentication is only provided afterwards.

5.1.5 Bluetooth

In 1998, the Bluetooth Special Interest Group (SIG) was formed. Its main purpose was to find a wireless alternative for traditional cabling like replacements for wired EIA-232 connections. In 1999, the Bluetooth specification 1.0 was published. Since this first release, the Bluetooth specification was under continuous development. In 2002, Bluetooth was approved as IEEE standard 802.15.1. After various sub-releases, the current version Bluetooth 3.0 has been released in 2009 [89]. The latest IEEE 802.15.1 standard release which incorporates Bluetooth 1.2 was in 2005 [90]. However, the IEEE society decided to not release any further Bluetooth versions as IEEE standard.



(a) Legacy pairing (b) Secure simple pairing

Figure 5.6: Bluetooth security

Since the initial idea of Bluetooth was to find a replacement for cabling to peripheral devices, the main application area of Bluetooth is to provide short range connectivity. The list of applications ranges from wireless control of multimedia devices (e.g., mobile headsets), to remote control devices (e.g., wireless keyboards), and to provide connectivity for peripherals like mobile phones and Personal Digital Assistant (PDA) devices.

The security concept of Bluetooth provides support for entity authentication as well as data confidentiality, freshness, and data origin authentication. Communication between Bluetooth devices is based on the client/server model – each Bluetooth network (called piconet) consists of a Bluetooth master and several Bluetooth slaves. To secure the communication between master and slaves, a secured channel has to be initialized. This initialization procedure is called *pairing* (cf. Figure 5.6(a)). The pairing procedure starts with the generation of a so called initialization key. This initialization key is calculated out of a random number, a PIN code, and the Bluetooth address of the remote device (i.e., the device where the PIN code has to be re-entered). The PIN code can be up to 16 octets long and may be factory-fixed if the device does not provide a user interface.

After having calculated the initialization key, a so called link key has to be generated. Depending on the involved devices, three different key types are available. The first one is called *combination key*. A combination key is generated in a way that both devices that want to establish a secured channel equally contribute to the key generation process.

A *unit key* is created by a single device and thus the other communication party is not able to influence the key generation process. Compared to combination keys, unit keys are rarely changed and are mainly used for devices with limited memory capabilities. However, from a security point of view, combination keys shall be preferred to unit keys. In addition to these keys that are shared between two communication parties, a so called *master key* is available. A master key temporally replaces the currently used link key and is used by a Bluetooth master to send secured data to multiple slaves at the same time. Communication is secured using the master key can only be initiated by the Bluetooth master.

After having agreed on a common link key, the Bluetooth devices authenticate each other. The authentication protocol uses a challenge-response scheme. Suppose, device A wants to authenticate device B. First, A generates a random number and sends it to B. B calculates a challenge response using the retrieved random number, its own Bluetooth address, and the shared link key. Then, it sends the challenge response back to A which performs the same operation and compares the calculated result with the retrieved one. If they are equal, the identity of device B has been verified. Because this challenge-response scheme only authenticates one participant, the device that was authenticated can also start the authentication protocol to ensure mutual authentication.

Finally, an encryption key that is used to provide data confidentiality is generated. This encryption key is calculated from the previously exchanged link key, a random number (that is sent to the other device), and a so called ciphering offset number. If the master key acts as input parameter, the Bluetooth address of the master is used as ciphering offset number. Otherwise, an additional output parameter from the authentication process is taken. The length of the encryption key may be up to 16 octets. Note that encryption in Bluetooth is optional.

From a security point of view, this pairing mechanism is vulnerable to brute-force attacks. If someone guesses the used PIN code (which is possible if the chosen PIN length is too short), the adversary is able to establish a secure session since the PIN code may directly be used as input parameter for the generation of the initialization key. Therefore, another pairing mechanism called *secure simple pairing* was introduced in Bluetooth version 2.1 (cf. Figure 5.6(b)). Due to the used mechanisms that are based on ECC, secure simple pairing is more secure than legacy pairing. In the first step, both devices generate an ECC key pair where the public keys are mutually exchanged. Afterwards, two authentication stages follow. In authentication stage 1, the exchanged public keys are authenticated. To achieve this, three different protocols are available. The first one is called

numeric comparison protocol. Here, a 6-digit commitment value that is associated with the public keys is displayed. The user has to verify the displayed values on both devices by comparing it. If the two values are equal, the authentication was successful. The second protocol, called *out of band protocol*, uses an out of band communication channel to verify the authenticity of the exchanged public keys. The third protocol, called *passkey entry protocol*, is based on a passkey that a user has to input into both devices. Using this passkey, the authenticity of the exchanged public keys is verified.

The following authentication stage 2 confirms that both devices have successfully finished authentication stage 1. This is done by exchanging and verifying a confirmation value that is calculated out of the values from authentication stage 1. If the validation was successful, both devices calculate a link key using previously exchanged confirmation values. The final step that is used to generate an encryption key is identical to the one used in legacy pairing.

The main drawback of Bluetooth is that its security concept has been developed for the use in domains other than HBA. To provide reasonable security, the security concept of Bluetooth demands that at least one device is equipped with a user interface and/or the opportunity for user inputs. Otherwise, factory-fixed PIN codes have to be used if authentication is based on legacy pairing. In case of simple secure pairing, an out of band mechanism has to be provided during authentication stage 1. Alternatively, authentication stage 1 can be skipped which is referred to as "Just Works" mechanism. However, this possibility shall not be used in security-critical environments since it provides no protection against man-in-the-middle attacks. Since SACs and ICDs typically used in the HBA domain are embedded devices without a user interface, Bluetooth and its security concept are of limited use for the field level. However, while it cannot fully replace traditional HBA network technologies, Bluetooth can be used to provide wireless connectivity for MDs. Typical examples are remote control panels and smart phones for user interaction as well as operator devices like PDAs and notebooks for configuration and maintenance purposes.

Others reasons for the limited use of Bluetooth's security concept and general remarks are:

- As mentioned above, key management in Bluetooth is based on user interaction using PIN codes and passkeys. Key management without user intervention is not possible in a secure manner.
- Since legacy pairing is vulnerable to brute-force attacks if short PIN codes are used, secure simple pairing shall be preferred.

- Bluetooth is limited to client/server communication where a single master is able to communicate with different slaves. While a master is able to reach multiple slaves at the same time, slaves are only able to communicate with their masters. Therefore, peer-to-peer communication where each device can communicate with all other devices is not possible in Bluetooth.
- The number of devices is very limited in a single piconet. To overcome this limitation, a slave can be member of more than one piconet. However, even if a device is a member of multiple piconets, it is only able to communicate with the responsible masters.
- The large protocol overhead of Bluetooth results in a protocol stack size which may exceed the limited resources of low power embedded devices.
- Using the power saving features of Bluetooth, a battery lifetime of days or at most weeks is possible. These power saving features are not suitable for wireless HBA networks where a battery lifetime of years is aimed.

5.1.6 OPC UA security

The OPC foundation is a nonprofit organization that provides and maintains specifications for data exchange in industrial automation systems. The most prominent specification that has been published by the OPC foundation is OPC DA. While the first version of OPC DA was released in 1996, the most recent one is version 3 [66]. OPC DA provides services for reading, writing, and monitoring of control data. It is mainly used at the management level (e.g., for MDs like visualization clients or operator workstations). Like many other OPC specifications (e.g., OPC Alarms and Events (OPC A&E), OPC Historical Data Access (OPC HDA)), OPC DA is based on Microsoft's technologies Component Object Model (COM) and Distributed Component Object Model (DCOM). As a result, the use of implementations that are based on these specifications is limited to Windows systems. To overcome this limitation, OPC UA has been introduced [91]. OPC UA is a platform-independent, service-oriented architecture that uses Web Services or a Transmission Control Protocol (TCP) based protocol for data exchange. While initially designed for the industrial automation domain, OPC specifications and especially OPC UA are also becoming more and more important in the HBA domain. The current version of OPC UA is shown in [67] and will soon be published as IEC 62541 [92].

OPC UA is completely based on the client/server model – the communication is performed in sessions. To gain access to control data, an OPC client establishes a connection

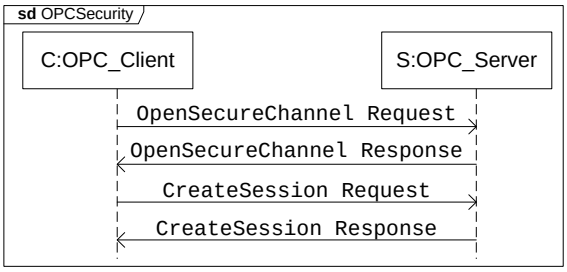


Figure 5.7: Security handshake in OPC UA

to one or more OPC servers. Since OPC devices are mainly located at the management level where the network may be shared with other application domains, a security model has been introduced in OPC UA. To secure the communication, a secured channel is set up during session establishment. This secured channel is provided by the OPC communication layer which is located between the transport layer and the application layer. The secured channel uses cryptographic techniques to guarantee data confidentiality, freshness, and data origin authentication. Entity authentication is provided during session establishment with the help of certificates. Authorization i.e., access control to the control data and services, is left up to the application.

In OPC UA, OPC devices can choose one of two different transport protocols for communication. The first one is based on WS-I Basic Profile 1.1 [93] where SOAP and HTTP are used. The second one is a binary TCP based protocol. To set up a secured channel, a generic channel establishment protocol is introduced which is applicable to both transport protocols (cf. Figure 5.7). To start a session, the OPC client sends an `OpenSecureChannel` request message to the server. This request consists of a security token and the client's certificate. The request is secured using asymmetric cryptographic schemes. After having received the request, the server responds with an `OpenSecureChannel` response message that is again secured using asymmetric schemes. Afterwards, the client sends a `CreateSession` request message to the server. This request is secured using symmetric schemes where the required keys are derived from the security token that has been sent to the server. To finish the establishment of the secured channel, the server responds with a `CreateSession` response message which is again secured using symmetric algorithms and the previously received security token. After the secured channel has been established, further messages that are secured with the same symmetric schemes can be exchanged.

This generic security protocol is mapped to the transport protocol that is used by the individual OPC devices. In case of Web Service based communication, the Web Service security concept defined in [94, 95, 96] is mapped onto the generic security handshake.

Note that this is contrary to BACnet/WS and oBIX where security is based on TLS. If OPC's binary transport protocol is chosen, a native instance of the generic security handshake that has been specified by OPC UA is used.

In OPC UA, security policies define which security mechanisms have to be used by the devices. Security policies are derived from security profiles that specify the algorithms for providing the secured channel and the protocols for managing the used shared secret keys. The selection of security policies is made by the system administrator that sets up the OPC devices. Security policies can be specified on a per-client basis i.e., an OPC server can use different policies for different OPC clients.

While it is possible to add new or vendor-specific security profiles, three different security profiles are currently defined by the OPC UA specification. `SecurityPolicy - None` does not use any cryptographic algorithms to secure the communication channel. Therefore, it shall only be used for environments with low security requirements (if at all). `SecurityPolicy - Basic128Rsa15` is applicable to environments with medium or high demands regarding security. It uses HMAC in combination with SHA, AES with a key length of 128 bits, and RSA with padding scheme version 1.5. Finally, `SecurityPolicy - Basic256` is available for highly security-critical environments. Instead of RSA with padding scheme version 1.5, this profile uses RSA with Optimal Asymmetric Encryption Padding (OAEP). Furthermore, `SecurityPolicy - Basic256` demands a key length of 256 bits for AES encryption.

While the security model of OPC UA provides reasonable protection, the following features are missing:

- OPC UA is dedicated to client/server communication and therefore securing the communication within groups is not supported.
- The management of the used certificates is not defined by OPC UA. Therefore, some kind of Certification Authority (CA) that maintains and distributes the certificates is required.
- OPC UA is dedicated to the management level where high-performance devices are located. Therefore, the security concept is inapplicable to embedded devices since the used algorithms exceed the available resources. For example, RSA encryption is too resource-consuming to run on low power embedded devices.
- OPC clients must already be in possession of the server certificates. Otherwise, the handshake cannot be initiated by the client. However, a retrieval of certificates during runtime is not specified.
- The security concept of OPC UA does not care for interruption threats. Therefore,

data availability cannot be guaranteed.

5.2 Security from the IT domain

Due to the widespread use of the Internet, security has been a major research field in the IT world for years. Therefore, many security mechanisms for IP based networks are available where each of them is suitable for a certain application field. In this section, a small subset of available, state of the art mechanisms that are suitable for HBA systems are presented.

5.2.1 Secure communication protocols

This section contains a subset of typical communication protocols that protect the transmitted data against malicious interference. The most well-known ones are Internet Protocol Security (IPsec) and Transport Layer Security (TLS).

IPsec

IPsec [97] is a security extension to the IP protocol and thus operates on the network layer. IPsec is a part of Internet Protocol Version 6 (IPv6) but since Internet Protocol Version 4 (IPv4) is still the predominantly used network protocol in the IT world, it has also been ported to extend IPv4.

The basic concept of IPsec is the use of so called Security Associations (SAs). An SA specifies the security services (e.g., used cryptographic schemes) and their input parameters that are used to protect the messages between a sender and a receiver.[2] Since an SA is dedicated to a uni-directional connection between the sender and the receiver, a pair of SAs is necessary to protect bi-directional communication. The parameters of each SA are stored in the Security Association Database (SAD). To uniquely identify an SA within the database, the so called Security Parameter Index (SPI) is used. Optionally, the identities of an SA can be extended by the destination and/or source IP address of the connection.

An SA either supports transport or tunnel mode. Transport mode is used to provide end-to-end security between two entities by protecting the data payload and parts of the IP header. In tunnel mode, the whole IP packet is encapsulated into the payload of a separate IP tunneling packet. Tunnel mode is mainly used to secure a tunnel by two security gateways.

[2]The term SA can be compared to the notation of a secured channel used throughout this dissertation.

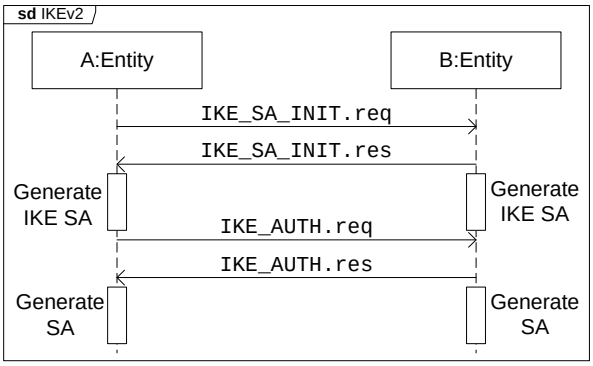


Figure 5.8: The IKEv2 protocol

To protect the transmitted data within an SA, two different security protocols are available. The IP Authentication Header (AH) protocol [98] uses cryptographic schemes to guarantee data origin authentication and thus data integrity. Optionally, data freshness can also be provided. The second protocol is called Encapsulating Security Payload (ESP) protocol. In addition to the security objectives guaranteed by AH, ESP also provides data confidentiality by encrypting it. In principle, an encryption-only mode of ESP is also available. From a security point of view, the use of encryption-only mode shall be avoided since it is vulnerable among others to modification attacks. Both protocols are flexible regarding the used cryptographic algorithms. While in principle any algorithm can be selected, [99] defines a set of mandatory (e.g., HMAC in combination with Secure Hash Algorithm Family 1 (SHA-1), 3DES in CBC mode) and recommended algorithms (e.g., AES in CCM mode).

As an alternative to manual secret key distribution, the Internet Key Exchange (IKE) protocol can be used. In the current version of IPsec, IKEv2 is used [100]. IKEv2 provides mutual entity authentication and authorization by using a request-response protocol. Depending on the application scenario, this protocol consists of various steps. To set up an SA for IPsec, four messages are sufficient (cf. Figure 5.8). First, the initiating entity A sends an `IKE_SA_INIT.req` message to the other entity B. This request consists of a list of cryptographic schemes that the entity supports, a DH value, and a random number. B responds with an `IKE_SA_INIT.res` that contains the chosen algorithm, the corresponding DH value, and another random value. Afterwards, both entities generate a so called IKE SA using the DH operation. After having generated the IKE SA, A sends an `IKE_AUTH.req` that contains an authentication payload that proves the identity of the sender as well as the authenticity of the first message. B verifies the authentication payload and responds with an `IKE_AUTH.res` message that also contains an authentication payload. After A has verified the authentication payload received from B, both entities

finish the IKE protocol by generating an SA out of the DH value and the exchanged random values.

The authentication payload is calculated using asymmetric algorithms like RSA or DSA. Alternatively, pre-shared secret keys can also be used. The use of ECC may also be possible but its usage has not been standardized yet.

An SA is dedicated to a unicast connection i.e., it is shared between a single sender and a single receiver. Therefore, SAs cannot be used to secure multicast communication in a native way. To overcome this limitation, an optional multicast extension for IPsec has been specified [101]. In this extension, multicast communication is protected using so called Group Security Associations (GSAs) [102]. GSAs can be retrieved from special entities called Group Controller Key Servers (GCKSs). These GCKSs are responsible for managing the group membership as well as their associated by GSAs. However, the following details are left open in this extension:

- Guaranteeing data origin authentication is only possible if the GSA consists of separate SAs for each sender. Additionally, the standard algorithms to protect the transmitted data cannot be used. Instead, asymmetric algorithms or special variants of symmetric algorithms (e.g., Timed Efficient Stream Loss-Tolerant Authentication (TESLA) [103]) have to be used. Otherwise, the identity of the sender cannot be uniquely identified. However, the choice of the specific protocol is left up to the implementation.
- Since the use of the IKE protocol is not possible, a communication protocol has to be defined which provides the opportunity to securely retrieve GSAs from the GCKSs. While the use of Group Domain of Interpretation (GDOI) [104] or Group Secure Association Key Management Protocol (GSAKMP) [105] is suggested, their use is not mandatory.
- As it is possible to use multiple GCKSs, the realization of such a concept is not specified. Important details like synchronization of GCKSs and the selection of GCKS to be used are left open.

Additionally, IPsec suffers the following limitations:

- To generate the required IKE SA, a DH calculation is necessary. In general, standard DH calculations are resource-consuming and thus they are inapplicable on embedded devices. To be able to use IKE on embedded devices, an ECC variant of the DH algorithms has to be used. However, the standardization of ECC for IKE has not been finished yet.
- The authentication payload that is generated during the `IKE_AUTH` handshake re-

quires trust-worthy certificates or shared secret keys. However, the distribution of these certificates or shared secret keys is not defined by IPsec.

- Due to its nature, IPsec is dedicated to the use for the IP protocol. Using it within non-IP networks (e.g., in field networks of HBA systems) requires major changes in the current IPsec protocol.
- Counteracting interrupt threats (e.g., DoS attacks) is not considered in IPsec.

TLS

Secure Socket Layer (SSL) and its successor TLS are both protocols for securing communication between two entities. While SSL was originally developed by Netscape, TLS has been released by the Internet Engineering Task Force (IETF) as a minor variant of the last version of SSL (SSL 3.0). The current version is TLS 1.2 [106].

TLS is an application independent protocol that is implemented as middleware between the transport layer and the application layer. The most well-known usage example of TLS is its use between TCP and HTTP to protect Web Services. To secure the transmitted data against malicious interference, TLS provides a secured channel that guarantees data origin authentication, data freshness, and data confidentiality. The protection of the secured channel is based on symmetric algorithms which are negotiated between the entities. Typical examples are 3DES or AES in CBC mode as well as HMAC in combination with MD5 or SHA.

The establishment of the secured channel is done by the so called TLS handshake protocol. During this handshake, the cryptographic schemes used by the handshake protocol as well as by the secured channel are negotiated. Additionally, the shared secret keys that are required later by the secured channel are exchanged. Furthermore, the entities are authenticated during the handshake. While the verification of the identity of the receiver is mandatory (unilateral entity authentication), providing mutual entity authentication i.e., proving the identity of the sender too, is optional in TLS.

The handshake protocol consists of different steps that may vary according to the authentication requirements and the chosen cryptographic schemes. The full handshake protocol for providing mutual entity authentication is shown in Figure 5.9. The handshake is initiated by A by sending a `ClientHello` message. B responds with a `ServerHello` message. Both messages are used to negotiate attributes like the used protocol version and the chosen cryptographic schemes. Furthermore, random values to avoid replay attacks are also exchanged. Afterwards, B continues by sending its certificate (`Certificate` message) as well as input parameters for the shared secret calculation (`ServerKey`

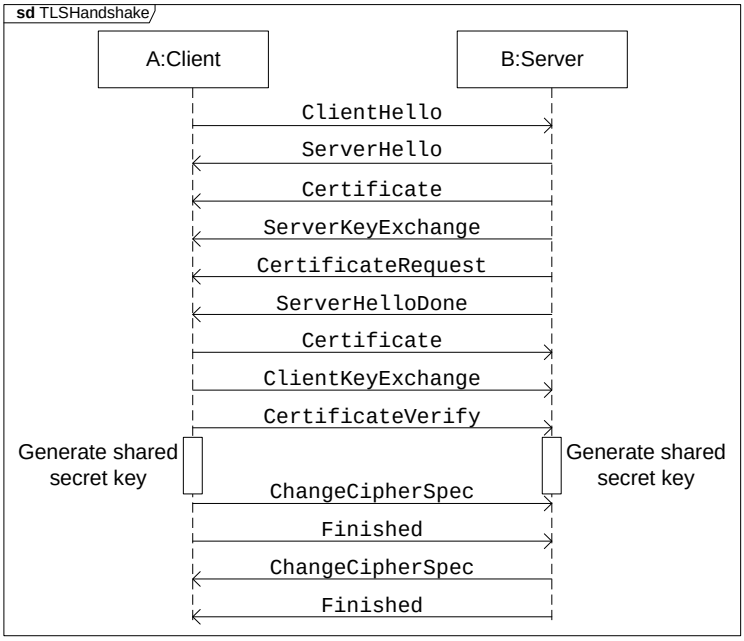


Figure 5.9: The full handshake in TLS

`Exchange` message). Furthermore, the certificate of A is requested to provide mutual entity authentication (`CertificateRequest` message). Finally, B finishes the hello phase by sending a `ServerHelloDone` message. After having received the `ServerHelloDone` message, A responds by sending its certificate (`Certificate` message) as well as input parameters for the shared secret calculation (`ClientKey Exchange` message). To prove its identity, the client also sends a `Certificate Verify` message. Using exchanged input parameters and random values, both entities calculate the shared secret keys for establishing the secured channel. To finish the handshake, both entities send a `ChangeCipherSpec` and a `Finished` message.

During the initial handshake, asymmetric algorithms are used. These algorithms are used to protect the exchanged messages as well as to generate the shared secrets. Again, the chosen algorithms can be negotiated by the entities. Typical examples are Rivest, Shamir, and Adleman (RSA) for protecting the message content as well as DH for secret generation. Recently, the use of ECC algorithms is also possible in TLS. Introducing ECC allows to implement TLS on embedded devices. For embedded devices, [77] shows an implementation of a complete secure Web server using TLS in combination with ECC.

While TLS is very flexible with respect to the used cryptographic algorithms and thus a solid base for securing unicast communication, it completely lacks multicast support – there is no way to use TLS to secure communication in groups. Additionally, the following limitations can be identified:

- TLS assumes that there is a working Public Key Infrastructure (PKI) that manages the required certificates. Details about the implementation of the PKI are not mentioned in the specification of TLS.
- To satisfy a broad range of applications, the handshake protocol of TLS has been designed in a generic way. Therefore, the handshake may consist of several communication steps that may result in an overhead for a dedicated application domain.
- TLS supports features for providing backward compatibility to older versions. However, since these fallback mechanisms may lead to vulnerabilities that may be utilized by adversaries, they should be disabled by the application.
- TLS supports the use of anonymous authentication that neither verifies the identity of the sender nor the one of the receiver. Since this mode is vulnerable to man-in-the-middle attacks, its usage should be avoided.
- As stated in the TLS specification, interruption threats cannot be avoided.

5.2.2 Organizational countermeasures

In addition to secure communication protocols that protect the transmitted data, various security concepts exist that use organizational mechanisms to counteract security threats and attacks. Typical examples are secure Virtual Private Networks (VPNs) and firewalls that try to separate unprotected traffic from protected one.

Secure VPN

A VPN is a logical network that is built upon a foreign network called host network. A VPN is transparent to the connected devices. Most VPNs use the underlying host network as simple transport medium – the exchanged messages that are encoded using the logical network protocol are encapsulated into messages using the host network protocol. If the encapsulation mechanism uses a secure communication protocol that provides a secured channel (e.g., TLS or IPsec), the resulting VPN is called a secure VPN.

A popular secure VPN implementation for IP based networks is OpenVPN [107]. In OpenVPN, each device opens a secure unicast connection to a centralized server where the whole network traffic to and from the device is tunneled through. To secure the tunneling connection, TLS is used. OpenVPN provides several methods to ensure authentication: a pre-shared symmetric key, a user name and password combination, a TLS certificate, or a combination of these methods. OpenVPN encapsulates the encrypted VPN packets into untouched TCP or UDP packets of the host network.

The main drawback of OpenVPN is the need for a centralized server that has to route the whole traffic. Therefore, the server has to manage one connection for each connected client. For multicast communication, this concept is clearly inappropriate. Each multicast message would have to be sent to the server where it is distributed to the clients. So the server has to decapsulate each multicast message once and encapsulate it again several times for all other clients. This results in a high demand on bandwidth, memory (for storing the different secret keys), and computational power at the server. Thus, the server represents a bottleneck and a single point of failure for the whole network. Furthermore, the required resources would exceed the capabilities of an embedded device. Therefore, an OpenVPN solution is of limited use for embedded networks.

Firewalls

A firewall is a network entity that protects a trusted network, host, or service against unauthorized access [14]. Firewalls originate from the IT world where they are mainly used to separate public foreign networks (e.g., Internet) from trusted inner networks (e.g., LANs). However, with the integration of gateways to other foreign networks, firewalls are also relevant for the HBA domain [108].

To provide a protection against unauthorized access, the firewall inspects the incoming and outgoing network traffic to decide whether the traffic is allowed or not. The decision about allowing or denying network traffic is made based on the firewall's security policy. Such a security policy commonly consists of a default policy and set of application specific rules that specify exceptions to the default policy. Consider, for example, a management interface to an automation controller. The default policy to that interface may be set to "DENY" while a specific rule may also be added that allows the system operator's management workstation to access the interface.

A security policy is normally predefined according to the system's policy. However, it may also be necessary to dynamically change the rule set of a firewall. For example, if an IDS identifies a malicious host within a network, the IDS may add a specific rule to the rule set of the firewall that explicitly drops all traffic originating from the identified malicious host (dynamic blacklist).

Depending on the capabilities provided by the firewall, three different types can be distinguished. A *packet filtering firewall* is the simplest form of firewall. It uses parts of the header information to decide whether a packet shall be accepted or dropped. For example, to filter IP traffic, the address information (IP address, User Datagram Protocol (UDP)/TCP port number) as well as the encapsulated application protocol type (e.g.,

HTTP, File Transfer Protocol (FTP)) may be considered for identifying valid traffic. However, details about the data portion of the forwarded packets (e.g., distinguish between different HTTP methods) as well as the state of connection (e.g., has the connection already been closed?) are not considered.

A *stateful filtering firewall* additionally maintains the state of a connection. Using this extra information, the firewall is able to detect illegal connection states and so it is possible to specify a more advanced rule set. For example, to avoid unsolicited connections with a protected device, the firewall only permits client packets after a dedicated connection has been established to the server.

Finally, *application proxies* also inspect the data portion of network packets. To fully analyze the effects of incoming and outgoing network packets, an application proxy simulates the behavior of the entire application. Therefore, a proxy acts as a man-in-the-middle: to the outside network, a proxy behaves like the destination device – to the inside network, the proxy acts as the request origin and vice versa. Sophisticated application proxies are completely transparent to the involved communication parties. A typical example is a Web gateway that is located between the management clients at the outside network and SACs located within the inside network. The Web gateway acts as an application proxy for Web gateway traffic. It receives and inspects the incoming management traffic to identify malicious traffic. If malicious traffic is identified, it is simply dropped – otherwise the data packets are forwarded to the management server located inside the network. If used properly, neither the management clients from the outside network nor the SACs from the inside notice the existence of the Web gateway.

Due to the importance of firewalls, they have to be reliable and robust against security attacks especially. To minimize vulnerabilities, firewalls are often isolated, stand-alone devices that are kept as simple as possible. Furthermore, access to firewalls is only permitted to users with special administrator privileges (if at all). From a security point of view, using a single firewall that acts as a single wall of protection may not be sufficient. If an adversary is able to bypass the firewall, the adversary has full access to the entire network. Therefore, it is more appropriate to separate the network into several zones where each zone is protected with a dedicated firewall and a dedicated security policy (*defense in depth*). Besides a separation between an outside and an inside network, the inner network is further divided into a local network that is home for the different client workstations and a so called Demilitarized Zone (DMZ). The firewall between the outside network and the DMZ is responsible for filtering the incoming network traffic that is intended for the servers as well as to the client workstations. However, to further protect the clients, a

second firewall is located at the boundary between the DMZ and the local network. This firewall is able to additionally filter traffic which is irrelevant for the client workstations (e.g., HTTP traffic to the Web server). The main advantage of this second firewall is that if a server within the DMZ gets compromised, the second firewall acts as an additional wall of protection between the client workstations and the compromised server. Finally, to further secure the client workstations against compromised clients, each client may have its own personal firewall which provides an additional layer of protection.

6 Security Enhanced Building Automation Network

As shown in Chapter 5, available solutions do not satisfy all requirements identified in Chapter 3. While some technologies provide a solid base for security-critical applications (e.g., ZigBee, BACnet), others are not able to fulfill the domain-specific challenges. However, there are even communication standards where security is still completely neglected. Therefore, a new generic approach to secure communication and thus counteract network attacks is introduced in this chapter.

As a generic solution, it has to be applicable to HBA systems of all sizes and types. To fulfill the requirements of such an all-in-one solution that is also suitable for security-critical applications, the presented approach is based on a modular, plugin-based, multi-protocol communication stack (cf. Figure 6.1). Its main feature is the support for communication services that guarantee end-to-end security on a per-device level. The stack is partitioned into three layers. The *Network Specific Layer (NSL)* provides low-level communication services that are used to transmit messages over the native network medium. To be able to reuse an already existing network infrastructure, any data link/physical layer combination can be used. On top of the NSL, the so called *Security Abstraction Layer (SAL)* is located. The SAL is the key component within the stack. It abstracts the communication services of the NSL, enhances them with security, QoS, and routing/-naming features, and offers generic secure communication services to the above located *Application Specific Layer (ASL)*. The ASL implements the functionality of a common application layer and provides an interface to user applications. As all layers operate on plugins, easy extension is supported.

6.1 Network specific layer

The NSL corresponds to layers 1 and 2 of the OSI reference model. It provides basic access to the underlying network medium and supports low-level communication services

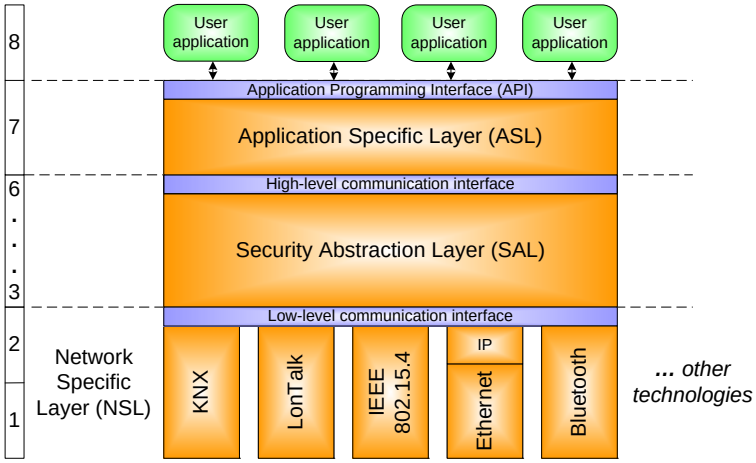


Figure 6.1: Multi-protocol stack

for sending and receiving messages on a single hop basis. Concerning the current state of the art, many different HBA standards supporting various network media are available. Each of them offers significant advantages regarding their physical characteristics. While Ethernet based networks provide high bandwidth that is necessary for backbone networks, fieldbuses based on TP or PL are advantageous at the field level since they support free topology. Wireless technologies, with all their benefits and challenges, are also getting more and more important. Available technologies also differ at the data link layer. For example, some of them offer native support for multicast (e.g., KNX, LonTalk) which can be used to efficiently exchange data within communication groups. Others, in turn, provide medium access mechanisms based on Time Division Multiple Access (TDMA) schemes. A typical example is IEEE 802.15.4.

As a result, choosing a single existing data link/physical layer combination or even developing a new one is not desirable. Therefore, the presented solution does not demand the use of a single data link/physical layer combination – in principle, it is possible to reuse any existing network technology. To provide an abstraction of the underlying data link communication services, so called *data link plugins* are introduced. A data link plugin is dedicated to a specific data link/physical layer combination and is located between the NSL and the SAL (cf. Figure 6.2). Data link plugins are geared towards sensible (re-)use of already existing data link primitives. This means that each plugin chooses the services that fit best for the communication services required by the SAL. A typical example would be the reuse of the existing multicast communication services of KNX and LonWorks. In contrast to that, the use of unsuited protocol features (e.g., the insecure security mechanisms of KNX or LonWorks) can be blocked by the plugin. Furthermore, it is even possible that a device implements more than one data link/physical layer com-

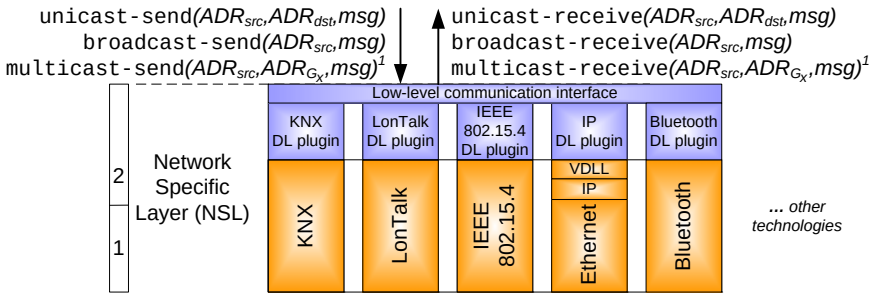


Figure 6.2: Low-level communication interface between NSL and SAL

bination and so, devices are able to have several interfaces to heterogeneous networks.

In addition to native data link plugins, it shall also be possible to use a higher OSI layer (i.e., layer 3 or above) as data link layer. To achieve this, a so called *Virtual Data Link Layer (VDLL)* has to be included. A VDLL provides the opportunity to integrate a higher protocol layer as native data link layer. A typical example would be the use of IP as data link layer for the SAL. This concept is similar to BACnet/IP where UDP is a possible network option for BACnet internetworks [52].

To be able to use a broad range of network technologies, the demands on the underlying data link/physical layer combinations shall be reduced to a minimum. Therefore, only an unconfirmed unicast and broadcast communication service are considered as mandatory – native data link layer support for multicast is optional. Figure 6.2 shows the resulting *low-level communication interface* that is located between the NSL and the SAL. The data link services `unicast-send` and `unicast-receive` are used to send and receive local unicast messages. The basic structure of a unicast message is as follows:

$$ADR_{src}||ADR_{dst}||msg$$

ADR_{src} and ADR_{dst} denote the source and destination addresses and msg the message that has to be transmitted.

To send and receive local broadcast messages i.e., messages to all network members located within the same network segment, the services `broadcast-send` and `broadcast-receive` are available. Broadcast messages have the following format:

$$ADR_{src}||msg$$

Finally, if provided by the underlying data link/physical layer combination, the services `multicast-send` and `multicast-receive` are used for multicast communication. A multicast message has the following structure:

$$ADR_{src}||ADR_{G_x}||msg$$

[1]Only available if the data link layer supports native multicast.

The parameter ADR_{G_x} specifies the data link address of the multicast group.

6.2 Security abstraction layer

The SAL corresponds to the OSI layers 3 to 6 and serves two objectives. First, it is responsible for providing a *global naming* and *routing* scheme that allows a communication across heterogeneous network segments. To achieve this, an HBA network model is assumed. As shown in Figure 2.4 in Section 2.1, a common HBA network is divided into different network segments. While HBA networks normally consist of several field networks that are interconnected by a common backbone, it is supposed that the network segments are arranged in a tree-structure of finite but arbitrary depth. Furthermore, it is assumed that the structure is static. This means that the network topology (i.e., the network cabling as well as the routers) is not changing during runtime. For the HBA domain, this assumption is valid since the physical building structure is also changed rarely. Furthermore, a heterogeneous network structure is possible i.e., the used data link/physical layer protocols may vary.

Due to the heterogeneous nature of this network model, a global addressing scheme based on global *SAL addresses* is mandatory. Using SAL addresses, the addressing schemes of the different data link layers can be abstracted and a routing across network segment borders is possible. The SAL addressing scheme works as follows. Each network segment has a dedicated network address N_x that is unique within the whole HBA network. Between these network segments, routers are located. Since the network topology is static, it is assumed that routers are configured accordingly. This means that each router knows the network addresses of its connected network segments and has sufficient routing information to find the next hop to any network segment. The configuration of the routers is done a priori during deployment. Using a routing protocol that allows a change of routing information during runtime is possible but out of scope of this dissertation.

Each device has a unique ID (denoted as ID_x) which acts as SAL address. In practice, this ID can be a serial number or a well-defined human-readable name. The SAL maps the device's ID to the so called device address (denoted as ADR_x) and the network address of the segment (denoted as N_x). A device address has to be unique within the network segment and is identical to the address used by the corresponding data link layer.

Additionally, devices can be arranged in so called *communication groups*. Each device can be member of one or more communication groups. Since communication groups are not limited to a dedicated network segment, they are defined within the scope of the

whole network. Each group has a dedicated group ID (denoted as ID_{G_x}) which acts as SAL address. Again, this ID is used by the user applications to uniquely identify a group within an HBA network. If the underlying network technology provides a native multicast service at the data link layer, the group's ID is mapped to the group address(es) used by the data link layer(s) (denoted as ADR_{G_x}). Otherwise, the SAL maps the request to multiple unicasts or to a global broadcast instead.

The second objective of the SAL is to provide generic secure communication services to the ASL. The security concept of the SAL is based on the concept of *secure communication relationships*. A device can be member of one or more secure communication relationships. Depending on the number of members, three different types of secure communication relationships are distinguished. A *network relationship* (denoted as N_x) consists of all members of a network segment. Relationships that consist of only two members are referred to as *session relationships* or *sessions*. A session is denoted as S_{XY} where the devices X and Y are the two members of the relationship. Finally, relationships that consist of three or more members being located across the entire network are referred to as *communication groups*. A communication group is denoted as G_x where ID_{G_x} holds the unique ID within the network.

Based on these three different types of communication relationships, all six communication types typically found in the HBA domain (cf. Chapter 3) can be supported by the SAL.

- *Secure point-to-point control data communication*: The control data is exchanged in sessions using secured unicast. A typical example would be a window contact that indicates an intrusion to an alarm controller.
- *Secure loose group communication*: Here, the control data exchange is performed using secured broadcast. The secured control data is transmitted to all network segment members using local broadcast. Each network member may decide on its own whether it is interested in the data or not. To reach all members of the entire HBA network and not only the members located at the same segment, a global broadcast has to be used.
- *Secure strict group communication*: Control data exchange is done by communication groups using secured multicast[2]. Compared to broadcast communication, the membership is controlled during the subscription process by a dedicated membership coordinator. Therefore, secure strict group communication is advantageous for security-critical applications. Consider, for example, a fingerprint reader that sends

[2]Here, the term multicast denotes the logical multicast services provided by the SAL.

a message to open multiple access doors.

- *Secure device management*: Secure unicast is also used to securely perform device management tasks. A typical example would be an MD that opens a management session to a SAC to change configuration parameters. The most important difference to secure point-to-point control data communication is the lifetime of a session. While a session for device management is only valid until the management client finishes its task, a session for control data communication may be valid for a longer period. This approach is advantageous since control data communication may occur more frequently and over a longer time period.
- *Secure network management*: Here, an MD is able to simultaneously perform configuration or maintenance of all members of the network segment in a secure manner. A typical example would be the distribution of new routing information.
- *Secure group management*: A typical example for secure group management would be a group coordinator that announces some change within the group membership.

To protect the communication services against security attacks, a secured channel is necessary. As mentioned in Section 3.1, the basis for providing a secured channel is the use of cryptographic schemes. However, cryptographic schemes are computationally intensive. Since embedded devices with limited system resources (processing power, persistent and volatile memory, power consumption, and network bandwidth) are commonly used in the HBA domain, the realization of a secured channel must not exceed the available device resources. Depending on the purpose and content of the exchanged data, it is not always necessary to guarantee all possible security objectives. For example, if the non-disclosure of the transmitted data is not a strict requirement, guaranteeing data origin authentication and freshness may be sufficient. In general, only those cryptographic schemes that are absolutely necessary to satisfy the security demands of the application shall be implemented. This property is denoted by the term “good enough security”. According to the security objectives that are guaranteed by a secure communication service, the following *security levels* are distinguished:

- *Raw*: Raw communication services are not secured at all. In order to avoid an unauthorized manipulation of data, the communication channel must be secured physically.
- *Protected*: Protected communication services are services where only data integrity is guaranteed. Data freshness is not provided.
- *Trusted*: A communication service that is classified as trusted is secured in a way that data integrity and data freshness are provided.

- *Confidential*: Trusted communication services where the exchanged data is additionally encrypted are called confidential communication services. It is guaranteed that only authorized devices are able to read the clear text version of confidential data. The ultimate goal of confidential communication services is to provide semantic security. The term "semantic security" refers to the strongest level of security where an adversary is not able to gain any information about the clear text, even if being in possession of many encrypted versions of the same clear text [109].

For trusted and confidential communication services, guaranteeing data origin authentication instead of data integrity is optionally possible.

Depending on the requirements of the control applications, these underlying communication services must also guarantee different *QoS properties* related to security. The most important one is *reliability* [23].

Definition 6.1. A unicast service is said to be reliable if and only if the following properties are guaranteed:

- *Integrity*: Every message is received as it was previously sent i.e., there is no corruption of a message while it is transmitted.
- *No duplicates*: Every message is received at most once.
- *Liveness*: Every message is received at least once.

While integrity is required by most applications in the HBA domain, preventing duplicates and guaranteeing liveness may be optional. For example, control applications that receive absolute values of control data (e.g., the present room temperature) do not care about duplicates since receiving the same value twice does not influence their proper functionality. If a control application receives absolute values at regular intervals, missing a few values may be tolerable and liveness may also be optional. However, applications that send relative state changes (e.g., increase the set point of the room temperature by five degrees) demand a reliable communication service that guarantees liveness as well as the absence of duplicates. Otherwise, irregular system states may result.

These reliability properties for unicast communication services can be easily extended to general communication relationships consisting of multiple members.

Definition 6.2. A communication service of a communication relationship is said to be reliable if and only if the following properties are guaranteed:

- *Integrity*: Every message is received as it was previously sent i.e., there is no corruption of a message while it is transmitted. This property is identical to unicast.

- *No duplicates*: Every message is received at most once by *all* non-faulty[3] relationship members.
- *Liveness*: Every message sent by a non-faulty relationship member is received at least once by *all* non-faulty relationship members. If a message is sent by a faulty relationship member, either all other non-faulty relationship members receive it or none of them.

In addition to reliability, the *ordering* of the messages may also be a requirement. In the general case of multiple sources and sinks, three kinds of ordering are distinguished [23]:

- *Single Source FIFO (SSF) ordering*: For all messages m_i, m_j and all devices d_k, d_l, if d_k sends m_i before m_j then d_l does not receive m_j before m_i.
- *Causal ordering*: For all messages m_i, m_j and each device d_k, if m_i *"happens before"* m_j then d_k does not receive m_j before m_i where "happens before" is defined as the following relation [111]:
  - If a and b are two events (i.e., receiving or sending a message) within the same device and a comes before b, then a "happens before" b.
  - If a is the sending event of message m by a device and b is the receiving event of the same message m by another device, then a "happens before" b.
  - If a "happens before" b and b "happens before" c, then a "happens before" c.
- *Total ordering*: For all messages m_i, m_j and all devices d_k, d_l, if d_k receives m_i before m_j then d_l does not receive m_j before m_i.

The kind of ordering depends on the requirements of the application. For example, if a single sensor periodically sends its status value, SSF ordering may be sufficient. However, in other environments where multiple sensors provide data to multiple controllers or actuators, causal ordering or total ordering may be necessary. Note that only causal ordering implies SSF ordering but none of the other ordering properties implies any other.

The communication services provided by the SAL are accessible through the so called *high-level communication interface* (cf. Figure 6.3). To establish a session for secure point-to-point control data communication or device management, the `session-start` primitive has to be used. ID_{dst} denotes the ID of the destination device, $Security$ the required security level, and $Reliability$ as well as $Ordering$ the demanded QoS properties. Message exchange during a session is done by the `session-send` and `session-`

[3]Within this dissertation, a non-faulty member is an entity that is capable of performing its specified function [110].

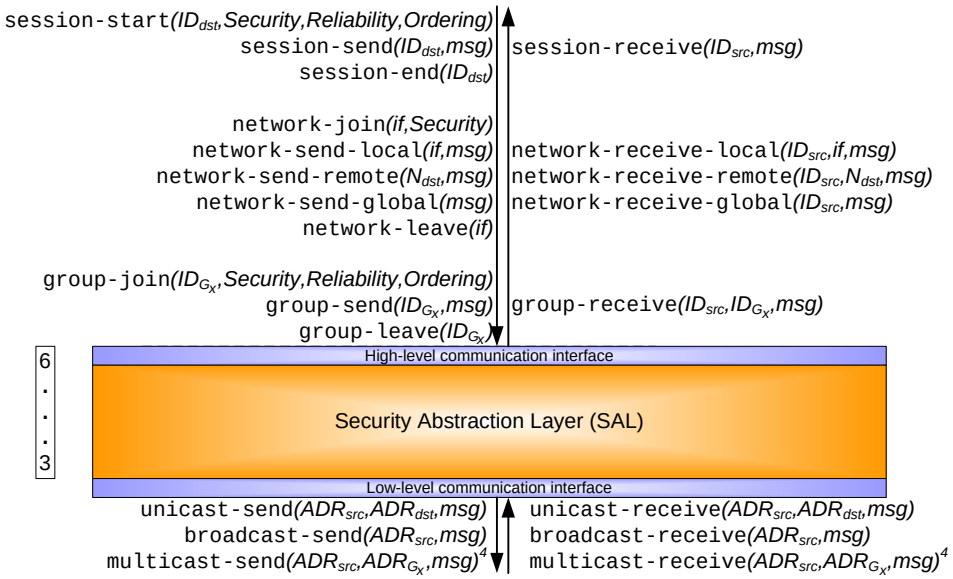


Figure 6.3: High-level communication interface between SAL and ASL

`receive` services. To terminate a session again, the `session-end` service is available.

To join a network, the `network-join` service is present. Since it is possible that a device has more than one network interface, the parameter if specifies the interface where the join has to be performed. The parameter $Security$ denotes the level of the required security level. To exchange messages within network relationships, three different kinds of services are available. The `network-send-local` and `network-receive-local` services are dedicated for local broadcast i.e., sending messages to all members of the network segment where the device is connected to. The parameter if specifies the target interface. The `network-send-remote` and `network-receive-remote` primitives are used to send and receive messages that need to be forwarded to a remote network segment. The address of the destination network segment is specified using the parameter N_{dst}. Finally, the `network-send-global` and `network-receive-global` services are available for addressing all members within the entire HBA network. To leave the network where the device is connected to, the `network-leave` service is provided. As it will be shown later in this dissertation, QoS parameters cannot be specified for network relationships.

Using the `group-join` service, a device is able to participate in the communication of a group. ID_{G_x} specifies the ID of the communication group, $Security$ the required security level, and $Reliability$ as well as $Ordering$ the QoS properties demanded by the ASL. Sending and receiving of group messages are possible using the `group-send`

[4]Only available if the data link layer supports native multicast.

and `group-receive` services. To leave a communication group again, the `group-leave` service is at hand.

A detailed description of the functionality of the SAL is given in Chapter 7.

6.3 Application specific layer

The ASL corresponds to the application layer of the OSI reference model. It makes use of the communication services provided by the underlying high-level communication interface and offers an Application Programming Interface (API) to the hosted user application(s). The main aim of the ASL is to completely hide the complexity of the underlying communication system. For user application engineers, it shall be possible to focus on the implementation of the functionality of the desired control applications – that is the collection of input information, performing control functionality, and interacting with the environment by setting output values. However, user application engineers should not need to bother with communication details. Managing communication relationships and dealing with data exchange between the user applications shall be left to the communication stack.

To provide such a high-level approach, the ASL is based on the concept of *data points*. They provide an abstract encapsulation of the control data that is under control by the user applications. The ASL is responsible for the management of these data points. All data points of a device are represented as so called *Application Objects (AOs)* stored in a generic *application object database*.

To access these AOs, two different ways exist (cf. Figure 6.4). First, the user applications must be able to manipulate the AOs of interest. The access to the application object database shall only be possible through a well-defined API. Second, since control applications are typically of distributed nature, remote devices must also be able to access remote AOs via the network. This is achieved by associating AOs of one device with (multiple) AOs hosted on remote devices. These associations are also referred to as *bindings*. If two AOs located at two remote devices are bound with each other, changing the value of the AO at one site also changes the value of the corresponding AO at the remote site. Using this scheme, user applications can take full advantage of control data that is distributed across the entire HBA network. The necessary binding information is stored within a so called *binding table* that is under control of the ASL. Note that associations between AOs are not restricted to one-to-one relations. One-to-many or even many-to-many bindings may also be possible.

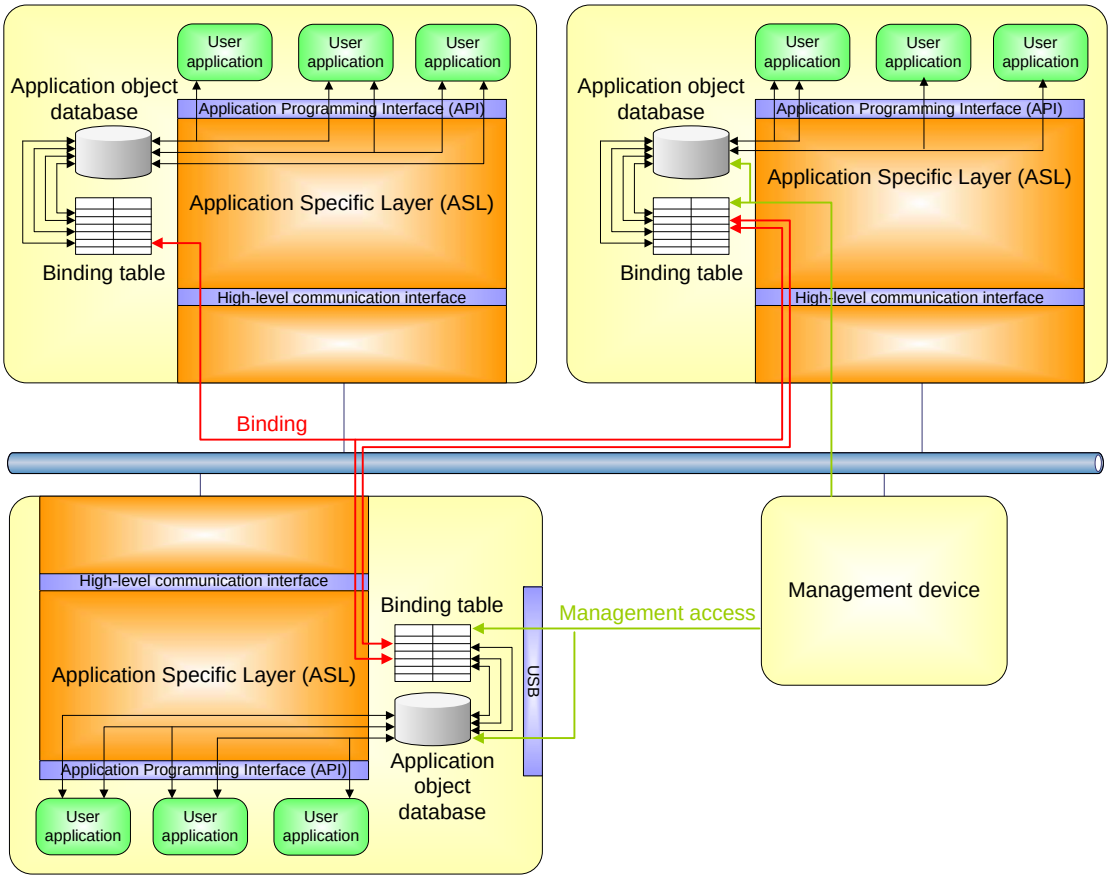


Figure 6.4: Application Specific Layer

To be able to perform a reasonable binding between remote AOs, the structure and semantics of the associated data points have to be specified, too. This concerns the *data point type*, the corresponding *representation* (i.e., the encoding of the data points' values within network messages and their interpretation), and *meta-data* that is associated with the data point. Typical examples of meta-data among others are engineering units, upper and lower bounds, and most important the required *security*, *reliability*, and *ordering level*. It must be possible to specify the minimum security and QoS levels that the remote devices must fulfill in order to be allowed to bind to remote AOs.

Beside the ability to change the value of AOs, their management is also of great importance. This concerns creation, changing, and removal of AOs as well as of the corresponding binding entries. Performing these configuration and maintenance tasks shall be possible by two ways. First, the API shall provide user applications the opportunity to dynamically manage AOs and their associated bindings during runtime. Second, it shall also be possible to access the AOs using management tools. Management access can be provided via the network or via a dedicated local interface. Obviously, to avoid malicious misuse, the management access must be protected accordingly.

7 Security Abstraction Layer

Figure 7.1 shows the internal structure of the SAL. The communication stack is divided into three sublayers. The *routing/naming sublayer* implements the generic routing/naming scheme of the SAL. The *security sublayer* enhances the low-level communication services with security features and provides services for secured unicast, secured broadcast, and secured multicast. On top of the security sublayer, the *reliability/ordering sublayer* is located. It is responsible for guaranteeing the QoS properties that are requested by the ASL. To be able to fulfill all communication objectives, the communication stack is supported by the *management entity*. It is responsible for managing the membership to the different secure communication relationships. The corresponding meta-data that is associated with the relationships is stored in the *security database*. The remainder of this chapter explains the different components of the SAL in more detail. Priority is given to the regular operation of the services. All failure handling will be discussed in Chapter 8 Evaluation.

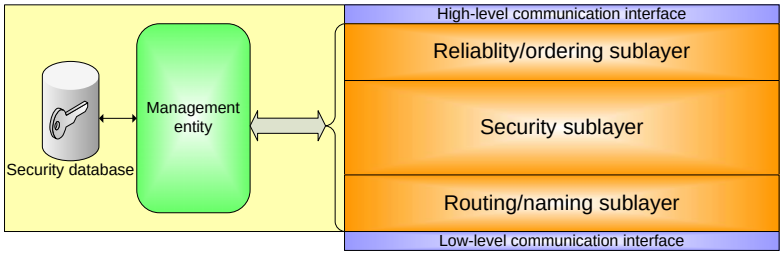


Figure 7.1: Security Abstraction Layer (SAL)

7.1 Routing/naming sublayer

Figure 7.2 shows the communication services provided by the routing/naming sublayer. Due to the (possible) heterogeneity of the underlying networks, two different types of addresses are used: generic SAL addresses and native data link layer addresses. The routing/naming sublayer has the aim to translate SAL addresses into the addresses used

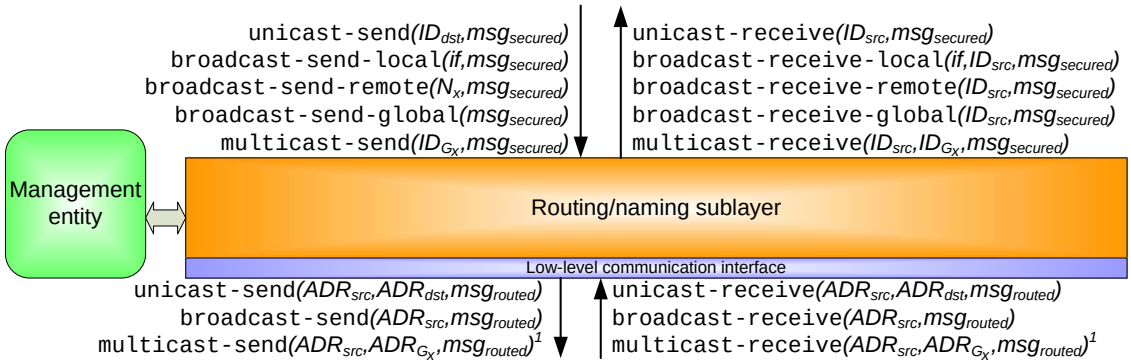


Figure 7.2: Routing/naming sublayer

by the chosen data link layer and vice versa. Furthermore, it is responsible for providing routing across heterogeneous network segments.

Depending on the destination, a unicast message has the following format:

$$msg_{routed} = \begin{cases} msg_{secured} & \text{for direct unicast} \\ N_A||ADR_A||N_B||ADR_B||msg_{secured} & \text{for forwarded unicast} \end{cases}$$

where "direct unicast" refers to a unicast message where the destination is located at the same network segment and "forwarded unicast" denotes a unicast message where the receiver is located at a remote network segment. Forwarded unicast messages, thus, have additional address information fields. N_A and ADR_A denote the network and device address of the original source. N_B and ADR_B correspond to the network and device address of the final receiver.

To send unicast messages, the `unicast-send` service is available. The destination device is specified by its ID. For direct unicasts, the routing/naming sublayer translates the destination ID (denoted as ID_{dst}) to the local data link address ADR_{dst} and invokes the low-level `unicast-send` service where the own local data link address serves as source address ADR_{src}. If the destination device is located in a remote network, the destination ID is mapped to N_B and ADR_B where N_B denotes the remote network address and ADR_B the address of the destination device within the remote network. Additionally, the own network and local data link address are set to N_A and ADR_A respectively. Afterwards, the message is forwarded to the next hop router by using the low-level `unicast-send` service. Here, the data link address of the next router is used as ADR_{dst} and the own local data link layer address is specified as source address ADR_{src}. The information for performing the necessary address mapping and for finding the next router is provided by the management entity. It is also within the management entity's responsibility to collect the necessary information and to keep it up-to-date.

[1] Only available if the data link layer supports native multicast.

If a unicast message is received by the data link layer, the routing/naming sublayer first verifies the destination address ADR_{dst}. If it matches its own, the message is accepted. If the incoming unicast message is a direct unicast, the source address of the incoming message is translated to the ID of the sender and the message is forwarded to the next higher layer. If the incoming message is a forwarded unicast and the receiving device is a router, the router evaluates N_B and ADR_B to determine the next hop. If the router is directly connected to the destination, the message is simply forwarded using the low-level `unicast-send` service where ADR_{dst} is set to ADR_B and ADR_{src} is set to the routers data link layer address. Otherwise, if the router is not connected to the destination device, the message is forwarded whereas ADR_{dst} is set the data link address of the next hop router. If the receiving device is the final destination (regardless whether it is a router or not), N_A and ADR_A are mapped to the sender's ID and the message is forwarded to the next higher layer. Again, the management entity assists in mapping the addresses.

A broadcast message has the following structure:

$$msg_{routed} = \begin{cases} msg_{secured} & \text{for local broadcast} \\ N_A||ADR_A||N_B||msg_{secured} & \text{for remote broadcast} \\ N_A||ADR_A||msg_{secured} & \text{for global broadcast} \end{cases}$$

Depending on the destination network, three different broadcast services are distinguished. Local broadcasts are used to send messages to all members of the network segment where the device is connected to. To reach all members of a remote network segment, remote broadcast services are available. Here, three additional addressing fields have to be specified. N_A and ADR_A denote the network and device address of the original source and N_B specifies the address of the destination network. Global broadcasts are used to send messages to all members of the entire HBA network. Again, N_A and ADR_A identify the network and device address of the original source.

The `broadcast-send-local` is used to send local broadcast messages. To perform local broadcasts, the low-level `broadcast-send` service is used where the own local data link layer address acts as source address ADR_{src}. The `broadcast-send-remote` service is used for remote broadcasts. Here, the request is sent by using the low-level `broadcast-send` service where N_B specifies the destination network. For N_A and ADR_A, the own network and device address are used. To reach the remote network segment, the next hop router has to forward the broadcast message. The `broadcast-send-global` service is available to send global broadcast messages. Again, the message is sent using the low-level `broadcast-send` service. To reach all other network segments, the routers have to forward the broadcast message to all connected network

segments. Again, N_A and ADR_A are set to the sender's network and device address.

If a broadcast message is received, the routing/naming sublayer determines the broadcast type. If it is a local broadcast message, the source address is mapped to the corresponding ID and the `broadcast-receive-local` service is invoked. If the message is a remote broadcast, the routing/naming sublayer verifies N_B. If N_B matches the network address where the device is connected to (regardless whether the device is a router or not), it maps N_A and ADR_A to the corresponding ID and forwards the message to the next higher layer using the `broadcast-receive-remote` service. If N_B does not match the own network address but the device is the next hop router, it forwards the message to the corresponding network segment using the low-level `broadcast-send` service. Otherwise, the message is discarded. If the message is a global broadcast and if the device is a router, it forwards the message to all connected network segments using the low-level `broadcast-send` service. In any case (regardless whether the device is a router or not), it converts N_A and ADR_A to the corresponding ID and invokes the `broadcast-receive-global` service primitive.

The structure of a multicast message depends on the underlying data link layer:

$$msg_{routed} = \begin{cases} msg_{secured} & \text{if native multicast is supported} \\ N_A||ADR_A||N_B||ADR_B||ID_{G_x}||msg_{secured} & \text{if multiple unicasts are used} \\ N_A||ADR_A||ID_{G_x}||msg_{secured} & \text{if global broadcast is used} \end{cases}$$

If the underlying data link layer supports multicast, the sending and receiving services of the routing/naming sublayer are mapped to the corresponding services of the data link layer. To send a multicast message, the low-level `multicast-send` service is used where the group ID is mapped to the corresponding data link group address ADR_{G_x}. ADR_{src} is set to the local data link layer address. The translation of the group ID to the data link group address is done by the management entity. If low-level multicast services are not available, they have to be simulated by using multiple unicasts or a global broadcast. While the former one is more advantageous for small groups, the latter one is advisable for large groups. If multiple unicasts are used, the necessary membership information has to be provided by the management entity – the mapping of the different address fields is identical to the one used for forwarded unicasts. If global broadcast is utilized, the multicast message is simply encapsulated into a global broadcast message. In both cases, the group ID (denoted as ID_{G_x}) is included, too. This helps the routing/naming sublayer to distinguish between simulated multicast messages and ordinary forwarded unicast or global broadcast messages.

How a multicast message is received depends on the underlying data link layer, too. If

native multicast is supported, the source address as well as the data link group address are mapped to the corresponding IDs and the `multicast-receive` service primitive is invoked. If multiple unicasts or global broadcast are used, the management entity verifies ID_{G_x}. If the device is member of the corresponding group, N_A and ADR_A are mapped to the sender's ID and the message is forwarded to the next higher layer by using the `multicast-receive` primitive. Otherwise, the message is discarded.

7.2 Security sublayer

The main aim of the security sublayer is to enhance the communication services with security features. Figure 7.3 shows the secure communication services that are provided to the next higher sublayer. All communication services have two parameters. The parameter `Level` specifies the basic security level that has to be guaranteed. A value of `Raw` denotes that the message needs not to be secured at all. If `Level` is set to `Protected`, data integrity has to be guaranteed by appending a MAC or a digital signature to the message. A security level of `Trusted` denotes that measures with the aim to guarantee data freshness are enabled. If `Level` is set to `Confidential`, the data must be encrypted before it is transmitted over the network. The parameter `TEXT` denotes a binary string that is an input parameter for the MAC or digital signature calculation. For security levels `Trusted` or `Confidential`, `TEXT` has to fulfill the properties of a so called Time Variant Parameter (TVP). A TVP guarantees that each request is unique. As a result, it provides data freshness. Corresponding to the chosen parameters `Level` and `TEXT`, the following *message security classes* are distinguished:

- *Raw messages* (denoted as $(m)_R$): `Level=Raw`, `TEXT` is ignored.
- *Protected messages* (denoted as $(m)_{P,SEED}$): `Level=Protected`, `TEXT` can be set to the fixed string $SEED$.
- *Trusted messages* (denoted as $(m)_{T,TVP}$): `Level=Trusted`, `TVP` has to fulfill the properties of a TVP.
- *Confidential messages* (denoted as $(m)_{C,TVP}$): `Level=Confidential`, `TVP` has to fulfill the properties of a TVP.

Trusted and confidential messages that guarantee data origin authentication instead of data integrity are denoted as $(m)'_T$ and $(m)'_C$, respectively. As shown later in the chapter, the used secret keys must fulfill special requirements to guarantee data origin authentication.

To guarantee the required security objectives, *symmetric* or *asymmetric* cryptographic schemes are used. The required input parameters of the involved cryptographic transfor-

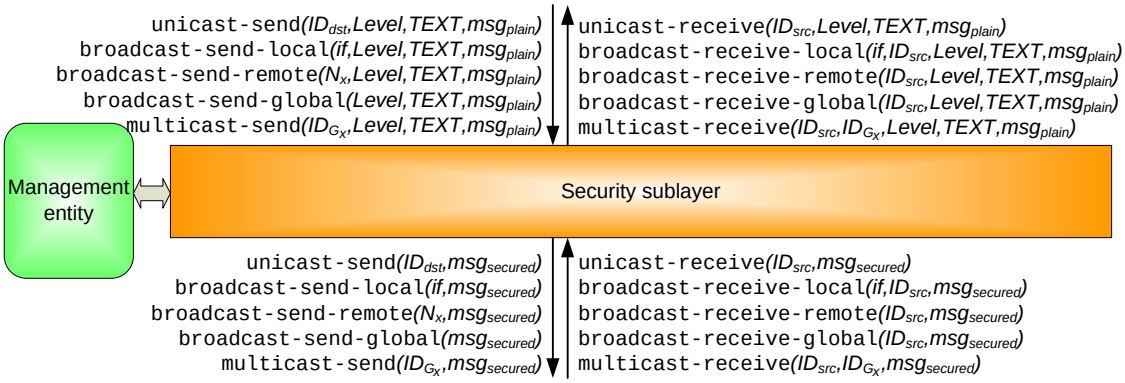


Figure 7.3: Security sublayer

mations are provided by the so called *Security Token Set (STS)*. An STS consists of public and secret input parameters (cf. Chapter 4) and is always dedicated to a single communication relationship. From a security point of view, this assignment has various advantages. First, it guarantees a logical separation between the different communication relationships – if the secret part of an STS is compromised only a single communication relationship is affected. Second, the number of security token uses is reduced which makes brute-force attacks more expensive. Third, it is guaranteed that only authorized devices i.e., members of the relationship can process secured messages. Members from other relationships are not able to do that. The management of the used STSs is done by the management entity. It is responsible for generation, distribution, and retrieval. The STSs are stored in the security database which is only accessible by the management entity.

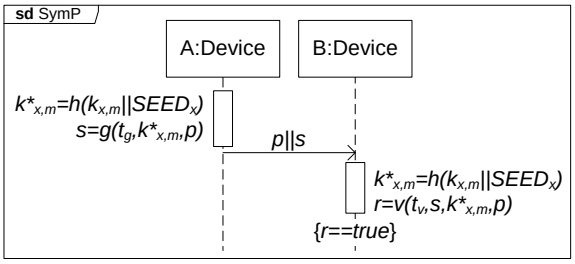
The content of the STS depends on the cryptographic transformations. If symmetric transformations are used, the STS that is dedicated to a secure communication relationship is identical for all members of the relationship. The STS for a secure communication relationship x consists of the following items:

$$STS_{x,P} = STS_{x,T} = t_g||t_v||k_{x,m}$$

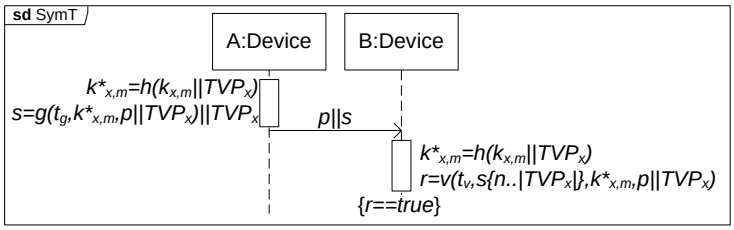
$$STS_{x,C} = \begin{cases} t_{hg}||t_{hv}||k_{x,h} & \text{if a hybrid symmetric scheme is used} \\ t_g||t_v||k_{x,m}||t_e||t_d||k_{x,c} & \text{otherwise} \end{cases}$$

where $STS_{x,P}$ denotes the STS for generating protected messages, $STS_{x,T}$ the STS for generating trusted messages, and $STS_{x,C}$ the STS for generating confidential messages. t_g, t_v[2] denote the public security tokens of the used MAC generation and verification transformation, t_e, t_d the public security tokens of the used symmetric encryption and decryption transformation, and t_{hg}, t_{hv} the public security tokens of the used hybrid generation and verification transformation. Depending on the chosen security level, $k_{x,m}$,

[2]For the rest of this dissertation, t_x is used as an abbreviation for the public security tokens that are denoted as $t_1, t_2, \ldots, t_n$ in Chapter 4.



(a) Symmetric protected message



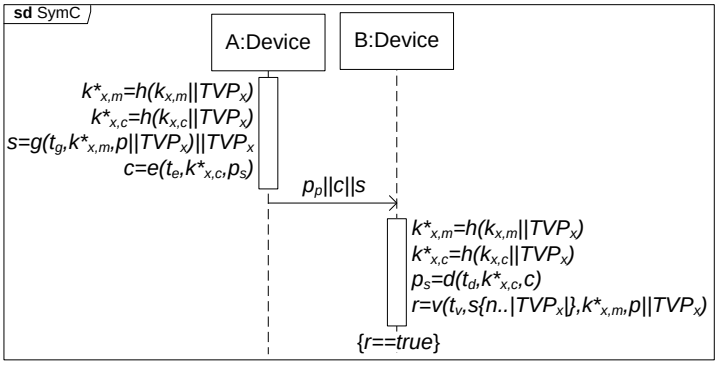
(b) Symmetric trusted message

Figure 7.4: Securing messages using symmetric cryptographic schemes

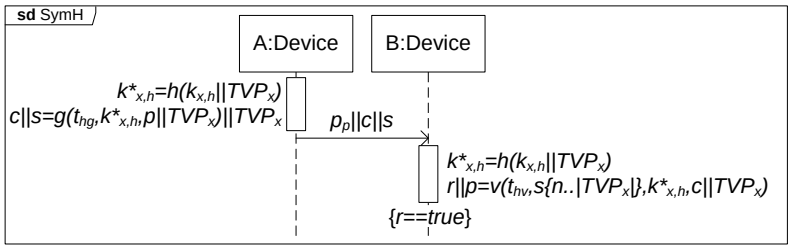
$k_{x,h}$, and $k_{x,c}$ represent the symmetric secret keys that are shared between the members of a communication relationship.

Figure 7.4(a) shows how a protected message is generated. First, a so called *dynamic shared secret key* (denoted as $k^*_{x,m}$) is generated. Dynamic keys have the aim to avoid the direct use of shared secret keys since shared secret keys are vulnerable points. If a secret key gets disclosed (e.g., due to a brute force attack), the affected key has to be replaced at all devices that share this key. A dynamic key, however, is derived from a cryptographic hash transformation and calculated as $k^*_{x,m} = h(k_{x,m}||SEED_x)$ where $SEED_x$ represents the predefined fixed string that is retrieved from the higher sublayer. Due to the one-way property of the cryptographic hash transformation, compromising a dynamic shared secret key $k^*_{x,m}$ does not disclose the corresponding shared secret key $k_{x,m}$ that is used to generate it. By changing the parameter $SEED_x$, a new dynamic key that is still secure can be generated even if the shared secret key $k_{x,m}$ remains unchanged. After the dynamic shared secret key has been generated, the MAC (denoted as s) is calculated by using the MAC generation transformation g. The public security token t_g that is included in the STS and the dynamic shared secret key $k^*_{x,m}$ are used as input parameters. The resulting MAC s is transmitted together with the plain message p to the receiver which performs the MAC verification procedure.

In Figure 7.4(b), the generation of trusted messages is presented. The main difference to the generation of protected messages is that a TVP (denoted as TVP_x) is used as parameter `TEXT` instead of a fixed string. Due to the properties of a TVP, replayed messages



(c) Symmetric confidential message



(d) Symmetric confidential message using a hybrid scheme

Figure 7.4: Securing messages using symmetric cryptographic schemes

are detected and thus data freshness is provided. In practice, a monotonically increasing counter or a timestamp can be used as TVP. However, since an attentively chosen TVP helps to support reliability and ordering of messages, providing an appropriate TVP is left to the higher sublayers. The TVP is included in the dynamic key calculation (i.e., $k^*_{x,m} = h(k_{x,m}||TVP_x)$). The resulting MAC value (denoted as s) consists of the output of the MAC generation transformation and the provided TVP value. Afterwards, s is transmitted together with the plain message p to the receiver which performs the MAC verification procedure. First, the TVP is extracted from the retrieved message[3]. Then, the dynamic shared secret key is calculated. Using this key, the receiver verifies the received MAC. If it is correct, it accepts the retrieved plain text p and delivers it together with the received TVP to the next higher sublayer. Thus, only the MAC is verified by the security sublayer. Proving the correctness of the TVP and thus guaranteeing data freshness is left to the next higher sublayer since the TVP is reused for guaranteeing reliability and a defined ordering of the messages.

[3] $s\{n \ldots |TVP_x|\}$ denotes the substring between position n (i.e., the most significant bit) and position $|TVP_x|$ of the binary string s where $|TVP_x|$ represents the length of the binary string TVP_x. In other words, the first $|TVP_x|$ bits of s are cut.

In Figure 7.4(c), the generation and verification of confidential messages are shown. In addition to calculate the MAC s, the part[4] of the plain text message p that has to be kept confidential (denoted as p_s) is additionally encrypted before it is transmitted over the network. The resulting MAC s, the cipher text c as well as the part of the plain text message that has not been encrypted (denoted as p_p) is transmitted to the receiver where the cipher text is decrypted and the correctness of the MAC is verified. From a security point of view, it is recommended to use different, independent keys for encryption/decryption and for MAC generation/verification. Otherwise, compromising the key of one algorithm will also break the other. Furthermore, it is necessary that the MAC scheme does not disclose the transmitted data. MAC generation transformations that satisfy the conditions described in Chapter 4 are sufficient of this purpose. Figure 7.4(d) shows an alternative solution where a hybrid cryptographic scheme is used instead of a separate scheme for encryption/decryption and MAC generation/verification.

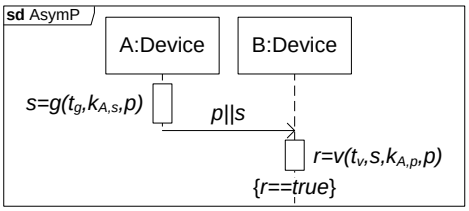
As mentioned before, data origin authentication can be guaranteed instead of data integrity. Providing data origin authentication differs from data integrity in the distribution of the shared secret keys. According to the definition of data origin authentication, the receiver of the message must be able to uniquely prove the identity of the sender. This is in contrast to data integrity where the unique identification of the sender is not a requirement. Using shared secret keys, data origin authentication can only be guaranteed if the used secret key is shared between exactly two devices. Within a group of devices, special protocols are necessary. An example of such a protocol is μTesla which is based on a delayed disclosure of secret keys [22]. Another possibility is to use a trusted third party that provides a proof for the data origin of a message.

As an alternative to symmetric cryptographic schemes, asymmetric ones can be used to generate and verify secured messages. While the STSs for symmetric transformations are identical for all members of the relationship, each device has a different STS for asymmetric transformations. An asymmetric STS of relationship x that is dedicated to device A consists of the following items:

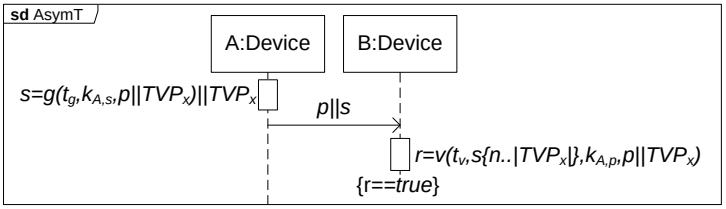
$$STS_{x,A,P} = STS_{x,A,T} = t_g||t_v||(k_{A,p}, k_{A,s})||k_{B,p}||k_{C,p}|| \dots$$
$$STS_{x,A,C} = t_g||t_v||t_e||t_d||(k_{A,p}, k_{A,s})||k_{B,p}||k_{C,p}|| \dots$$

where $STS_{x,A,P}$ denotes A's STS for generating protected messages, $STS_{x,A,T}$ A's STS for generating trusted messages, and $STS_{x,A,C}$ A's STS for generating confidential messages. t_g, t_v denote the public security tokens of the used signature generation and ver-

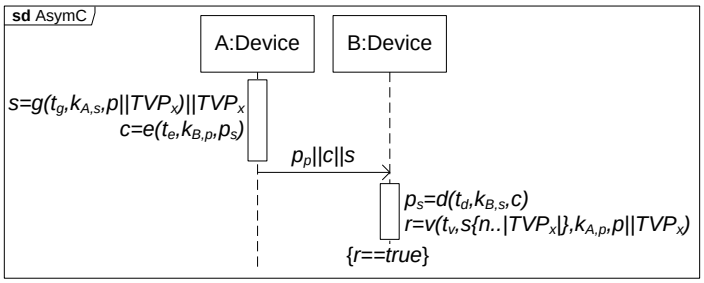
[4] As it will be shown in Chapter 8 Evaluation, it is sufficient that the user data is encrypted. The routing information has to be available in clear since encrypting it is disadvantageous for heterogeneous routing.



(a) Asymmetric protected message



(b) Asymmetric trusted message



(c) Asymmetric confidential message

Figure 7.5: Securing messages using asymmetric cryptographic schemes

ification transformations and t_e, t_d denote the public security tokens of the used asymmetric encryption and decryption transformations. $(k_{A,p}, k_{A,s})$ denotes the public/private key pair of device A where $k_{A,p}$ denotes the public key and $k_{A,s}$ the corresponding private key. Furthermore, A must be in possession of the public keys of all remaining relationship members (denoted as $k_{B,p}||k_{C,p}||\ldots$).

Figure 7.5 shows how asymmetric cryptographic schemes are used to generate and verify secured messages. The main difference is that dynamic keys are not used for asymmetric schemes. This is due to the fact that public/private key pairs cannot be generated dynamically since they are asymmetric. If a device dynamically generates a new private key, the public key also changes. However, since it is impossible to generate a public key without knowing the private key, the other devices that need to know the public key are not able to update it without retrieving it from a trusted third party.

In contrast to symmetric schemes, securing messages using asymmetric transformations always provides data origin authentication since digital signatures are generated with the sender's private key. Since the owner of the private key is the only device that

knows it, the corresponding public key is uniquely bound to that device. As a result, the data origin of the message that is signed with a private key can be identified using the corresponding public key.

Due to the characteristics of the different communication services types, the security tokens of the STSs are different. Therefore, the following three types of STSs are distinguished. *Network Security Token Sets (NSTSs)* are responsible for *securing broadcast communication*, *Session Security Token Sets (SSTSs)* for *securing unicast communication*, and *Group Security Token Sets (GSTSs)* for *securing multicast communication*.

7.2.1 Secured broadcast

To secure local broadcast messages i.e., messages sent to all members of the network segment, the NSTS is used as input parameter for the cryptographic transformations. Due to the properties of symmetric transformations, the NSTS for the symmetric broadcast variant is identical for all network members. The NSTS that is used to generate secured broadcasts using symmetric schemes consists of the following items:

$$NSTS_{N_x,P} = NSTS_{N_x,T} = t_g||t_v||k_{N_x,m}$$

$$NSTS_{N_x,C} = \begin{cases} t_{hg}||t_{hv}||k_{N_x,h} & \text{if a hybrid symmetric scheme is used} \\ t_g||t_v||k_{N_x,m}||t_e||t_d||k_{N_x,c} & \text{otherwise} \end{cases}$$

where $k_{N_x,m}$, $k_{N_x,h}$, and $k_{N_x,c}$ denote the secret keys that are shared between all network members. Since all devices use the same shared secret key for securing local broadcast messages, data origin authentication is not provided.

Using asymmetric schemes for secured broadcasts is only reasonable for protected and trusted messages. To decrypt confidential messages secured with asymmetric algorithms, the private key has to be used. To provide the opportunity that all network members are able to decrypt confidential messages, they must be in possession of the decryption key. As a result, the private key would have to be shared among the network members. However, this is contrary to the definition of asymmetric algorithms since a private key must only be known by one device. Therefore, confidential broadcasts are only reasonable in combination with symmetric schemes (i.e., $NSTS_{N_x,C}$ does not exist if asymmetric schemes are used). The NSTS of device A connected to network segment N_x that is used to generate secured broadcasts using asymmetric transformations consists of the following items:

$$NSTS_{N_x,P} = NSTS_{N_x,T'} = NSTS_{N_x,T} = t_g||t_v||(k_{A,p}, k_{A,s})||k_{B,p}||k_{C,p}||\ldots$$

where $(k_{A,p}, k_{A,s})$ denotes the public/private key pair of A and $k_{B,p}||k_{C,p}||\ldots$ the public keys of all other network members. Since asymmetric algorithms provide data origin authentication, $NSTS_{N_x,T'}$ is equal to $NSTS_{N_x,T}$.

Regardless whether symmetric or asymmetric schemes are used, a secured broadcast message is of the following format:

$$(msg_{secured})_{x,TEXT} = (ID_A||N_B||msg_{plain})_{x,TEXT}$$

where $x \in \{P, T, C\}$ for symmetric schemes and $x \in \{P, T, T'\}$ for asymmetric schemes. ID_A denotes the ID of the sender, N_B the address of the destination network address[5], and msg_{plain} the data that has to be transmitted by the upper sublayer. If security level `Confidential` is chosen, msg_{plain} is encrypted – ID_A and N_B are sent unencrypted but protected against unauthorized modification. The main aim of including ID_A is to identify the message's sender. This is necessary since the underlying data link layer address is not protected. Furthermore, it may change and so it cannot be used for this purpose. However, note that this ID cannot be used to guarantee data origin authentication since each valid network member can simply forge any ID. N_B avoids that an adversary re-routes a broadcast message to a different destination network.

7.2.2 Secured unicast

To protect unicast communication i.e., data exchange between two devices, the SSTS that is shared between two devices has to be used. An SSTS is always dedicated to a single session which is generated and distributed during the session establishment process. Again, symmetric and asymmetric cryptographic schemes can be used. Using symmetric schemes, an SSTS that is shared between the devices A and B consists of the following items:

$$SSTS_{AB,P} = SSTS_{AB,T'} = SSTS_{AB,T} = t_g||t_v||k_{AB,m}$$

$$SSTS_{AB,C'} = SSTS_{AB,C} = \begin{cases} t_{hg}||t_{hv}||k_{AB,h} & \text{if a hybrid scheme is used} \\ t_g||t_v||k_{AB,m}||t_e||t_d||k_{AB,c} & \text{otherwise} \end{cases}$$

where $k_{AB,m}$, $k_{AB,h}$, and $k_{AB,c}$ denote the used secret keys that are shared between the two devices. Note that a session relationship always consists of exactly two members. As a result, the message sender can be identified. Therefore, data origin authentication is also guaranteed in a native way.

[5]For global broadcasts, N_B is set to a reserved address (e.g., 0).

If asymmetric algorithms are used, device A must have the following SSTS:

$$SSTS_{AB,P} = SSTS_{AB,T'} = SSTS_{AB,T} = t_g||t_v||(k_{A,p}, k_{A,s})||k_{B,p}$$
$$SSTS_{AB,C'} = SSTS_{AB,C} = t_g||t_v||t_e||t_d||(k_{A,p}, k_{A,s})||k_{B,p}$$

where $(k_{A,p}, k_{A,s})$ denotes the public/private key pair of A and $k_{B,p}$ the public key of B. The corresponding SSTS of device B has the following structure:

$$SSTS_{BA,P} = SSTS_{BA,T'} = SSTS_{BA,T} = t_g||t_v||(k_{B,p}, k_{B,s})||k_{A,p}$$
$$SSTS_{BA,C'} = SSTS_{BA,C} = t_g||t_v||t_e||t_d||(k_{B,p}, k_{B,s})||k_{A,p}$$

Again, due to the asymmetric nature, data origin authentication is guaranteed in a native way.

A secured unicast message consists of:

$$(msg_{secured})_{x,TEXT} = (ID_A||ID_B||msg_{plain})_{x,TEXT}$$

where $x \in \{P, T, C, T', C'\}$. ID_A denotes the ID of the sender, ID_B the ID of the receiver, and msg_{plain} the data that has to be transmitted by the next higher sublayer. If security level `Confidential` is chosen, msg_{plain} is encrypted – ID_A and ID_B are sent in clear. The main aim of including ID_A and ID_B is to uniquely bind the message to the session and the involved devices. The enclosed IDs also help to distinguish between multiple, parallel sessions.

7.2.3 Secured multicast

To securely communicate within a group of devices, the GSTS that is dedicated to the group has to be used. Since a GSTS is only shared between members of the communication group, sending and receiving of messages that are secured with the group's GSTS are reserved to group members exclusively. The GSTS is retrieved during the group join that is initiated by the device's management entity.

Again, each communication group can choose between symmetric and asymmetric schemes. Using symmetric schemes, a GSTS that is used to secure communication within group G_x consists of the following items:

$$GSTS_{G_x,P} = GSTS_{G_x,T} = t_g||t_v||k_{G_x,m}$$
$$GSTS_{G_x,C} = \begin{cases} t_{hg}||t_{hv}||k_{G_x,h} & \text{if a hybrid scheme is used} \\ t_g||t_v||k_{G_x,m}||t_e||t_d||k_{G_x,c} & \text{otherwise} \end{cases}$$

where $k_{G_x,m}$, $k_{G_x,h}$, and $k_{G_x,c}$ denote the secret keys that are shared between all members of the communication group. As mentioned before, guaranteeing data origin authentication is only possible if special, additional measures are implemented.

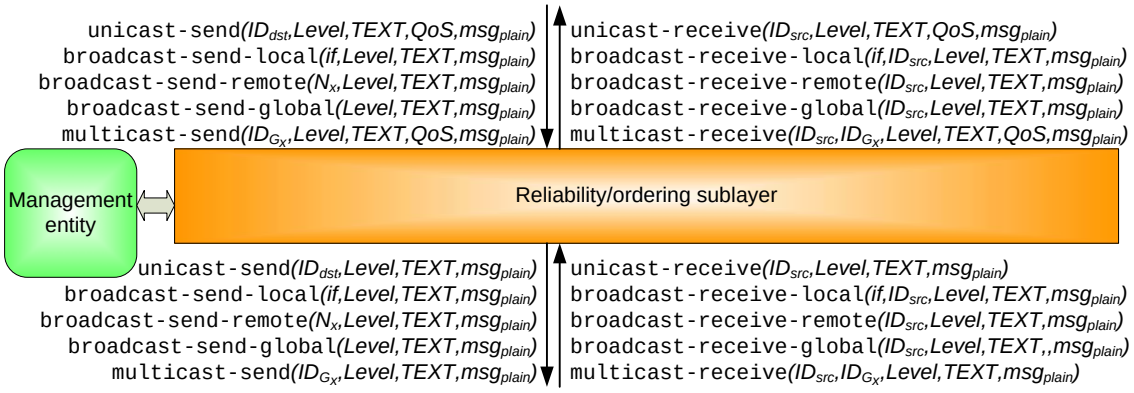


Figure 7.6: Reliability/ordering sublayer

Similar to a secured broadcast, asymmetric schemes are only reasonable for protected and trusted messages. The GSTS of device A that is used to handle secured multicasts using asymmetric schemes consists of the following items:

$$GSTS_{G_x,P} = GSTS_{G_x,T} = GSTS_{G_x,T'} = t_g||t_v||(k_{A,p}, k_{A,s})||k_{B,p}||k_{C,p}|| \ldots$$

where $(k_{A,p}, k_{A,s})$ denotes the public/private key pair of A and $k_{B,p}||k_{C,p}|| \ldots$ the public keys of the other group members. Since asymmetric algorithms provide data origin authentication, $GSTS_{G_x,T'}$ is equal to $GSTS_{G_x,T}$.

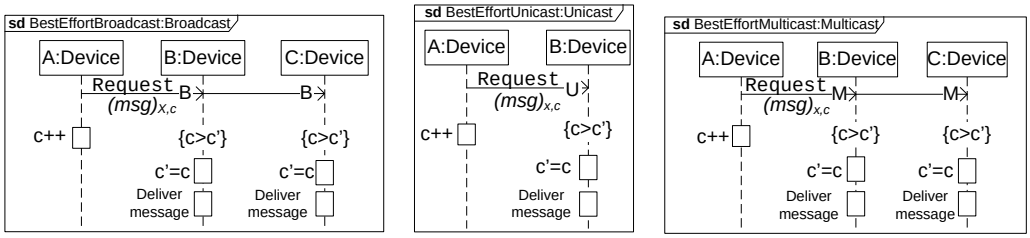
A secured multicast message consists of:

$$(msg_{secured})_{x,TEXT} = (ID_A||ID_{G_x}||msg_{plain})_{x,TEXT}$$

where $x \in \{P, T, C\}$ for symmetric schemes and $x \in \{P, T, T'\}$ for asymmetric schemes. ID_A denotes the ID of the sender, ID_{G_x} the ID of the group, and msg_{plain} the data that has to be transmitted by the higher sublayer. If security level `Confidential` is chosen, msg_{plain} is encrypted – ID_A and ID_{G_x} are sent in clear.

7.3 Reliability/ordering sublayer

The main aim of the reliability/ordering sublayer is to enhance the secure communication services with QoS properties. Figure 7.6 shows the resulting services. For unicast and multicast service primitives, an additional parameter (denoted as `QoS`) is available. Using this parameter together with the parameter `TEXT`, the required QoS properties can be specified. This concerns reliability, on the one hand, and an ordering of messages on the other hand. For broadcast services, these additional QoS properties are not available. As it will be shown in this section, the main reason is that the chosen QoS enhancements are based on mechanisms that require knowledge on member details. However, since it is not



(a) Broadcast (b) Unicast (c) Multicast

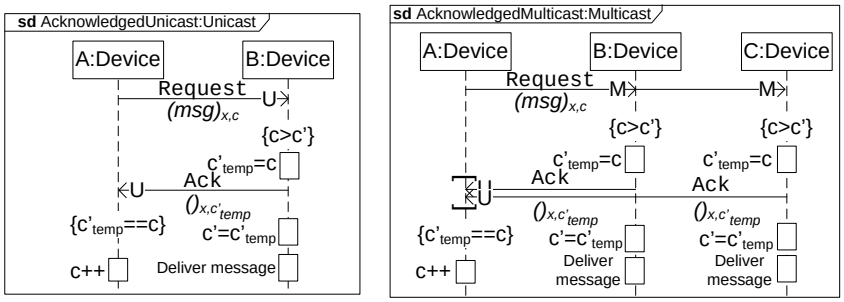
Figure 7.7: Best-effort communication services

required that the members of a broadcast relationship are aware of each other (loose group membership), QoS enhancements are not applicable to the provided broadcast services. Therefore, secured broadcast is only advisable for applications with relaxed requirements regarding QoS. A typical example would be the periodic broadcast of a temperature sensor where the loss of a value is not critical. For security-critical applications, secured unicast or secured multicast has to be used instead.

First of all, it is possible to select the required reliability level that the service has to provide. As mentioned in Section 6.2, the following three properties characterize the reliability of a service: *integrity*, *no duplicates*, and *liveness*. An advantage of the presented solution is that the security sublayer assists in supporting reliability since it already guarantees parts of these properties. If a security level of at least `Protected` is chosen, the security sublayer already provides an integrity check and discards messages that violate data integrity. Therefore, the secure communication services with security level `Protected` and above already satisfy the integrity condition of Definition 6.2.

If a security level of `Trusted` or above is chosen and a TVP is selected as parameter `TEXT`, message duplicates are prevented, too. Figure 7.7[6] illustrates how a monotonically increasing counter acts as TVP. Whenever a device sends a broadcast message (cf. Figure 7.7(a)), it passes the current counter value to the secured broadcast communication service. The security sublayer generates a trusted or confidential message $msg_{x,c}$ and broadcasts it. The other network members retrieve the message where the enclosed MAC or digital signature is verified. If it is valid (from a security point of view), the message is delivered together with the extracted TVP to the reliability/ordering sublayer where the time variant property of the received counter is checked. If the retrieved counter value (denoted as c) is greater than the locally stored one (denoted as c'), the message is accepted and the locally stored counter value is updated to the received one. If it is lower or equal, the message is discarded since it is identified as a duplicate. This duplication

[6]Arrows marked with a "B" denote broadcast messages, arrows marked with a "U" denote unicast messages, and arrows marked with an "M" are used to represent multicast messages.



(a) Unicast (b) Multicast

Figure 7.8: Acknowledged communication services

prevention scheme also guarantees data freshness since malicious replays are detected, too. At the sender site the counter is updated by simply incrementing it after sending the message. Figure 7.7(b) and Figure 7.7(c) show the unicast and multicast variant.

Guaranteeing liveness is more complex. Up to now, the communication services are based on a best-effort principle. Each message is simply sent to the receiver(s) – a dedicated feedback is not demanded. As a result, the sender cannot be sure whether the message was successfully retrieved by the receiver(s). Since liveness is not guaranteed by the communication services shown in Figure 7.7, they are also referred to as *best-effort communication services*.

To overcome this problem, the basic idea is to introduce some feedback mechanism that informs the sender whether the transmission was successful or not. Figure 7.8 shows a variant based on acknowledgments where a counter is used as TVP (*acknowledged communication services*). The sender transmits the desired message to all receivers of the communication relationship. After having received the message, each receiver individually checks the included counter. If the retrieved counter value is greater than the locally stored counter value, the message is correct and so, the receiver responds with a positive acknowledgment using a unicast message. For generating the acknowledgment message, the received counter value is used as TVP. Afterwards, the locally stored counter value is updated and the message is delivered to the next higher layer. If the retrieved counter is equal to the locally stored one, the receiver also sends an acknowledgment since the previous one may have been lost. However, the message is not delivered again. If the counter is lower, the message is identified as a duplicate and discarded[7].

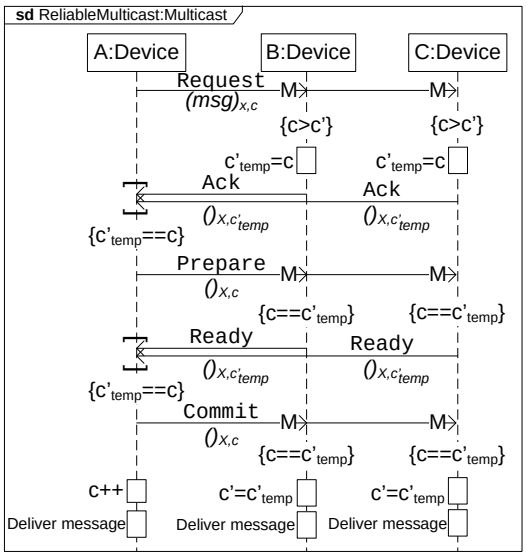
The sender collects all acknowledgments. If a negative acknowledgment is received or if an acknowledgment of a receiver is missing at all (e.g., the message was lost), the

[7]Since it may be possible that a previous request is lost, an alternative solution is to send a negative acknowledgment back. However, the main disadvantage is that adversaries may sent replayed messages to cause negative acknowledgments.

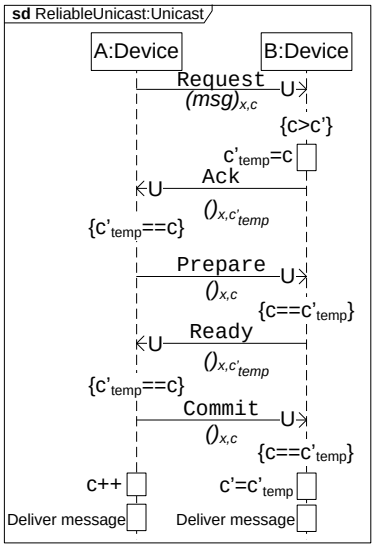
sender simply retransmits the original message. To prevent duplicates at the receiver site, the same counter value that was included in the initial message has to be used. The most critical factor is the amount of time that the sender waits before an absent acknowledgment is declared as missing. The value of this timeout depends on the maximum message delay within the network and on the processing time i.e., the time that the receiver needs to handle the message. In general, it is not possible to determine such an upper bound in asynchronous networks [23]. However, in practice, an upper bound can be estimated during the configuration phase of the HBA network. The timeout value consists of twice the maximum transmission delay plus the processing time of a message. The maximum transmission delay depends on the longest communication path which in turn depends on the used network technology (i.e., bandwidth), on the present topology, and on the time that a communication may be slowed down due to the medium access scheme used. The latter factor must not be underestimated especially in large communication groups where multiple members want to acknowledge an incoming message almost simultaneously. However, since the used network technology, the present topology, and the number of members within a communication group are mostly static in the HBA domain, an appropriate estimation of the maximum transmission delay is possible during the configuration phase. The processing time of a message depends on the time it needs to decode the message and to react on the content. This includes the time it takes to generate a response. Since this mainly depends on the used hardware and system software of a device, an estimation during configuration is also possible. Finally, after having finished the transmission, the sender updates its counter value by incrementing it.

Note that acknowledged communication is only available for unicast and multicast. Since the membership of network relationships is not known, the sender of a broadcast message is not able to identify which acknowledgments are missing. Therefore, an acknowledged broadcast service is not available.

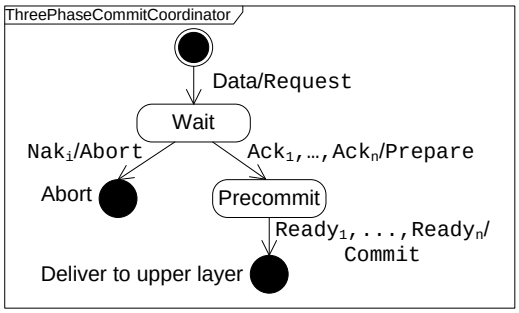
However, also when using acknowledged communication, a major problem remains unsolved. A device can never be sure when the data is regarded as valid and when it can deliver it to the upper layer. Consider, for example, device A sends a message to device B and C using acknowledged multicast. Assume that B successfully receives the message but C never gets it due to a network failure. In that case, B sends an acknowledgment back to A. However, B does not know that a retransmission is necessary and so it delivers the data to the next upper layer although C has not received the message yet. This leads to an inconsistent data view since some members already delivered the data while others are still in the retransmission phase. An inconsistent data view is also possible for unicast



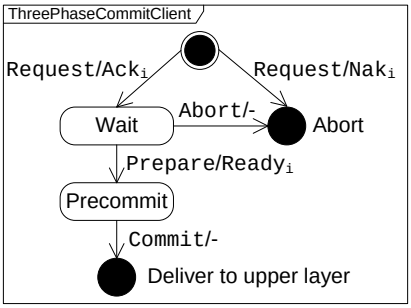
(a) Reliable multicast



(b) Reliable unicast



(c) Three phase commit: source site



(d) Three phase commit: sink site

Figure 7.9: Three-phase commit protocol

relationships. Consider, the case where the sender misses an acknowledgment. While the receiver already delivers the message to the upper layer, the sender is still waiting due to the missing acknowledgment. Inconsistent data views may also appear if one or several relationship members crash.

As a result, acknowledged communication cannot fully guarantee liveness. Therefore, its use is only advisable for applications that are able to tolerate inconsistent data views. To provide liveness, a commit protocol is necessary. The most important commit protocols mentioned in literature are the two-phase and the three-phase commit protocols. While the former is more simple and thus used more frequently in practice, the three-phase variant has advantages regarding fault tolerance. It can be shown that in the presence of two site failures there exists no non-blocking, two-phase commit protocol that uses an independent, local recovery strategy [112]. Three-phase protocols have the advantage that they can provide non-blocking behavior at the operational site. Recovering of

failing sites is commonly based on polling the current status from other, operational sites. Figure 7.9 shows the use of a three-phase commit protocol for providing reliable multicast [113]. Figure 7.9(a) and Figure 7.9(b) present the communication protocol in case of no failures whereas Figure 7.9(c) and Figure 7.9(d) show the state machines of the commit protocol that have to be implemented at the source and at the sink sites. The source site starts the commit protocol by sending a `Request` message containing the data that has to be distributed to the multicast group. Again, a monotonically increasing counter is used as TVP to avoid duplicates. Each sink site receives the message and verifies the enclosed counter value. If the counter is valid and if device is able to handle the retrieved data, the sink site responds with a positive acknowledgment `Ack`. Otherwise (e.g., the device's receive buffer is full), a negative acknowledgment `Nak` is sent back. In both cases, the received counter is used as TVP. The source collects the responses. If there is at least one negative acknowledgment or if there is at least one acknowledgment missing at all (within a defined timeout), the multicast transaction is canceled by sending an `Abort` message. If a positive acknowledgment is retrieved from all group members, the source sends a `Prepare` message to the multicast group. In both cases, the same counter value that has been included in the first message is used. To indicate that the `Prepare` message has been received and that the individual sinks are ready to deliver the data to the next higher layer, each sink responds with another `Ready` message. After having received all acknowledgments, the source sends a `Commit` message that initiates the final delivering of the message at all sites. Figure 7.9(b) shows the corresponding unicast variant. Note that a reliable broadcast variant is also not available.

In addition to the demanded reliability level of the communication service, a well-defined ordering of the exchanged data may also be a requirement. The main advantage of the proposed communication stack is that an adequate TVP selection can be used to control the ordering characteristic. However, since a defined ordering is only reasonable for acknowledged or reliable communication services, guaranteeing a well-defined ordering is not possible for best-effort communication services. As a result, demanding a well-defined ordering for broadcast communication is also not possible in the proposed solution.

The most simple kind of ordering is *SSF ordering*. Here, it is demanded that all messages that are sent by a single source are received by all sinks in the original order. SSF ordering can be implemented by using a separate counter for each device within each relationship and by modifying the counter verification mechanism at the receiver site. Instead of verifying whether the retrieved counter is greater than the locally stored one (i.e.,

$c > c'$), a message is only accepted if the retrieved counter value is equal to the locally stored value plus one (i.e., $c == c' + 1$). If the retrieved counter is greater than one plus the locally stored counter value, the message is "too recent" i.e., there must be a message that has not been received yet. Here, the receiver has to hold the message until the missing messages arrive. Additionally, the receiver can inform the sender by transmitting a negative acknowledgement including the last successfully received counter value. This gives the sender the opportunity to resend the missing messages.

A more general form of ordering is *causal ordering*. For unicast communication, it can be shown that causal ordering is equivalent to SSF ordering. However, while causal ordering always implies SSF ordering, the counter argument is not valid for relationships consisting of more than two members. Therefore, a single counter for each sender is not sufficient to provide causal ordering within multicast groups. One possibility to guarantee causal ordering is to use the concept of logical vector timestamps. Here, each device has to maintain a logical vector clock that is used to generate logical vector timestamps. For each message that has to be sent, the TVP has to be set to the local logical vector timestamp of the sending device. Based on this logical vector timestamp, the receivers are able to causally order the received messages. The main disadvantage of this scheme is that vector timestamps introduce a communication overhead in large communication groups since a logical vector timestamp contains one counter value for each group member. For more details on vector clocks and possible implementations see [23].

Vector timestamps are still insufficient to guarantee *total ordering*. As an alternative, synchronized clocks and timestamps can be used. However, since it can be shown that totally ordered, reliable multicast is impossible in asynchronous systems [23], assumptions have to be made. One important restriction is that the maximum transmission time within the network as well as the processing time have to estimated. Furthermore, using synchronized timestamps to guarantee total ordering demands support for clock synchronization services. This is clearly a major disadvantage since the need for synchronizing clocks has been avoided in the proposed solution. A detailed survey of clock synchronization can be found in [114].

Note that due to the uniqueness of vector and synchronized timestamps, both act as TVPs that prevent duplicates and guarantee data freshness. Figure 7.10 summarizes the different communication options and their provided objectives.

Communication service		Best effort		
TVP		N	C	
Encryption enabled		N	N	Y
Security	Data integrity	x	x	x
	Data freshness	-	x	x
	Data confidentiality	-	-	x
	Data origin	-	-	-
Reliability	Integrity	x	x	x
	No duplicates	-	x	x
	Liveness	-	-	-
Ordering	Single source FIFO	-	-	-
	Causal	-	-	-
	Total	-	-	-

TVP	Time Variant Parameter
N	No
Y	Yes
C	Counter
V	Vector timestamp
T	Timestamp (synchronized)
x	Satisfied
~	Partly satisfied
-	Not satisfied

(a) Broadcast

Communication service		Best effort							Acknowledged							Reliable						
TVP		N	C		V		T		N	C		V		T		N	C		V		T	
Encryption enabled		N	N	Y	N	Y	N	Y	N	N	Y	N	Y	N	Y	N	N	Y	N	Y	N	Y
Security	Data integrity	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	Data freshness	-	x	x	x	x	x	x	-	x	x	x	x	x	x	-	x	x	x	x	x	x
	Data confidentiality	-	-	x	-	x	-	x	-	-	x	-	x	-	x	-	-	x	-	x	-	x
	Data origin	-	x	x	x	x	x	x	-	x	x	x	x	x	x	-	x	x	x	x	x	x
Reliability	Integrity	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	No duplicates	-	x	x	x	x	x	x	-	x	x	x	x	x	x	-	x	x	x	x	x	x
	Liveness	-	-	-	-	-	-	-	-	~	~	~	~	~	~	-	x	x	x	x	x	x
Ordering	Single source FIFO	-	-	-	-	-	-	-	-	x	x	x	x	x	x	-	x	x	x	x	x	x
	Causal	-	-	-	-	-	-	-	-	x	x	x	x	x	x	-	x	x	x	x	x	x
	Total	-	-	-	-	-	-	-	-	-	-	-	-	x	x	-	-	-	-	-	x	x

(b) Unicast

Communication service		Best effort							Acknowledged							Reliable						
TVP		N	C		V		T		N	C		V		T		N	C		V		T	
Encryption enabled		N	N	Y	N	Y	N	Y	N	N	Y	N	Y	N	Y	N	N	Y	N	Y	N	Y
Security	Data integrity	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	Data freshness	-	x	x	x	x	x	x	-	x	x	x	x	x	x	-	x	x	x	x	x	x
	Data confidentiality	-	-	x	-	x	-	x	-	-	x	-	x	-	x	-	-	x	-	x	-	x
	Data origin[8]	-	~	~	~	~	~	~	-	~	~	~	~	~	~	-	~	~	~	~	~	~
Reliability	Integrity	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	No duplicates	-	x	x	x	x	x	x	-	x	x	x	x	x	x	-	x	x	x	x	x	x
	Liveness	-	-	-	-	-	-	-	-	~	~	~	~	~	~	-	x	x	x	x	x	x
Ordering	Single source FIFO	-	-	-	-	-	-	-	-	x	x	x	x	x	x	-	x	x	x	x	x	x
	Causal	-	-	-	-	-	-	-	-	-	-	x	x	x	x	-	-	-	x	x	x	x
	Total	-	-	-	-	-	-	-	-	-	-	-	-	x	x	-	-	-	-	-	x	x

(c) Multicast

Figure 7.10: High-level communication services

7.4 High-level communication interface

The main aim of the high-level communication interface of the SAL is to provide generic, configurable communication services that can be used by the ASL. Figure 7.11 shows the resulting interface. For joining group and session relationships, the following three

[8] Data origin authentication is only guaranteed with special measures.

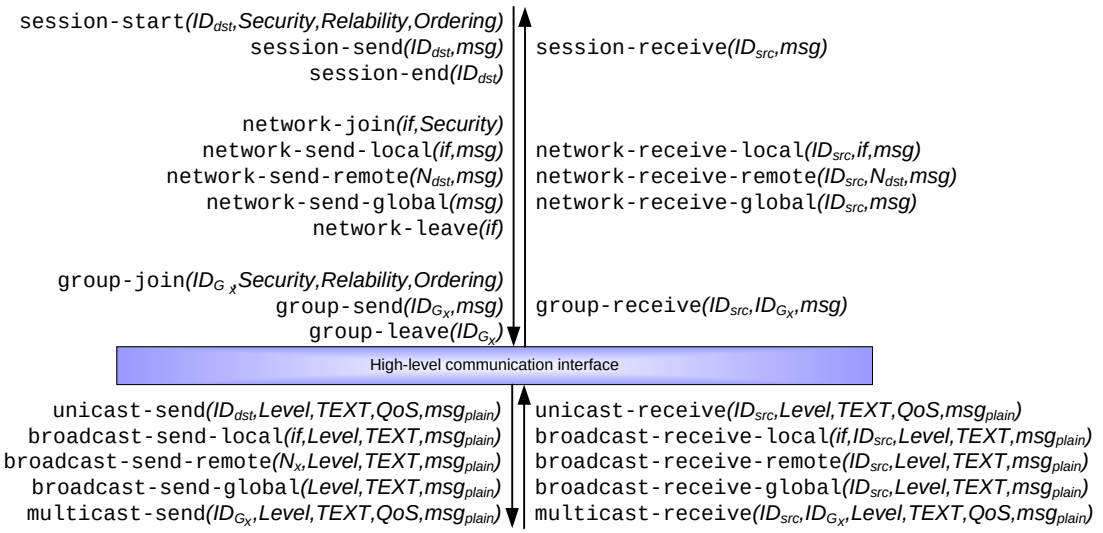


Figure 7.11: High-level communication interface

parameters can be specified:

- `Security`: `Raw`, `Protected`, `Trusted`, or `Confidential` + Option `Data origin`.
- `Reliability`: `Best-effort`, `Acknowledged`, or `Reliable`.
- `Ordering`: `SSF`, `Causal`, or `Total`.

Since network relationships are based on loose group communication, only the parameter `Security` is available. Performing relationship joins is the task of the management entity. After reception of a join request from the ASL, the management entity tries to join the requested communication relationship. If the join was successful i.e., the joining device has adequate access rights to participate in the relationship, the communication properties of the communication relationship are compared to the `Security`, `Reliability`, and `Ordering` properties that are demanded by the ASL (according to Figure 7.10). If they are sufficient, a positive confirmation is sent to the ASL. If joining is not possible (e.g., due to missing access rights or unsupported communication properties), the join process is aborted and the ASL is informed by sending a negative confirmation.

After a relationship has been joined, the ASL is able to exchange data. The corresponding send and receive services are mapped to the primitives provided by the reliability/ordering sublayer. The parameters `Level`, `TEXT`, and `QoS` that are required by the underlying reliability/ordering sublayer have to be specified. These parameters depend on the current communication properties of the relationship and are provided by the management entity.

If a membership to a communication relationship is no longer necessary, the ASL is able to decide to leave the relationship. Performing relationship leaves is again the task

of the management entity.

For the rest of this dissertation, the following notation is used within the sequence diagrams. Arrows that represent a message exchange are marked with X_Y. X denotes the type of relationship where B is used for local broadcasts, GB for global broadcasts, RB for remote broadcasts, U for unicasts, and M for multicasts. For unicast and multicast communication, Y denotes the type of service where B is used for best-effort, A for acknowledged communication, and R for reliable communication. In case of broadcast, Y is omitted.

7.5 Management entity

A key component within the SAL is the management entity. It is responsible for the management of the membership to the different communication relationships. The main objective of the management entity is to manage the relationships' *Configuration Data Records (CDRs)*. Each CDR is associated with a single communication relationship. It consists of related meta-data required by the communication stack to fulfill its purpose – that is the reliable and secure exchange of data. A typical CDR contains the relationship's SAL and data link addresses, routing information to reach the members of the relationship, the relationship's STS, as well as parameters like the current value of the parameter `TEXT`, the security level, and the used reliability service type. CDRs are stored in the security data base of the SAL which is accessible by the communication stack via the management entity.

The operation of a management entity is divided into four stages: First, each device has to be configured once at installation time (*stage I: initial configuration* cf. Section 7.5.1). Using this initial knowledge, each device is able to prove its identity as well as its associated access rights and may dynamically join one or more secure communication relationships during runtime (*stage II: secure binding* cf. Section 7.5.2). Within such a secure relationship, devices are able to exchange data through a secured channel (*stage III: secure communication* cf. Section 7.5.3). To close communication, the device has to leave the relationship (*stage IV: secure unbinding* cf. Section 7.5.4).

7.5.1 Initial configuration

To be able to securely bind to communication relationships, the device has to be prepared by uploading a so called *Initial Configuration Data Record (ICDR)*. To minimize the

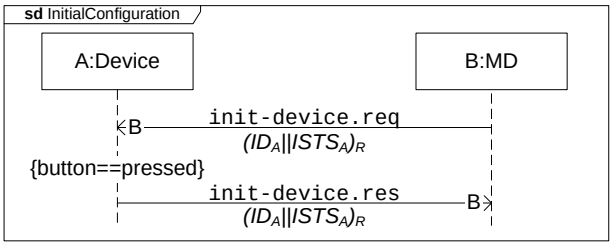


Figure 7.12: Stage I: Initial configuration using network mode

configuration effort, the amount of configuration data shall be reduced to a minimum. Therefore, for device A, this set of data consists of the following items.

$$ICDR_A = ID_A||ISTS_A$$

where ID_A denotes the device ID of A that uniquely identifies it within the whole HBA network. $ISTS_A$ denotes the *Initial Security Token Set (ISTS)* containing the security tokens that act as input parameters for the cryptographic schemes required in stage II. The exact structure of $ISTS_A$ depends on the supported binding protocol that is used during stage II (cf. Section 7.5.2).

Since there are no security tokens available prior to the initial configuration stage, the distribution of ICDRs cannot be cryptographically secured. To avoid that an adversary manipulates the ICDR, physical security must be guaranteed at the time of upload. However, this is the initial security problem – regardless of the used security protocol, a distribution of initial security tokens is always necessary [80]. A secure initial configuration can be achieved in multiple ways:

- *Network mode*: In this mode, an MD uploads the ICDR to the device via the network. Figure 7.12 shows the basic principle. Since the device that has to be configured is not in possession of any address information, a broadcast message has to be used (`init-device.req`). To avoid that multiple devices are programmed at the same time, only one configurable device must be connected to the network. An alternative solution would be to demand physical access to the particular device (e.g., pressing a button on the device to uniquely identify it). After having received the broadcast message, the device acknowledges with an `init-device.res` broadcast message containing the previously received ID_A and $ISTS_A$[9]. The MD receives the response and verifies the enclosed ID and ISTS. If they are equal to the previously sent ones, the configuration was successful. Otherwise, the configuration procedure is repeated. To protect the configuration process against malicious interference, it must be performed in a physically secure environment where it can

[9] If the MD has a local data link layer address, the device is able to use unicast instead of broadcast.

be guaranteed that only authorized devices have access to the network. One possible solution is to set up a minimal network containing only the device and the management station.

- *Local mode*: The upload is performed by an MD via a point-to-point connection (e.g., via an EIA-232 interface) or manually via a storage medium (e.g., smart card)[10]. The main benefit is that the ICDR is never transmitted over the network and thus, even adversaries with access to the network are not able to intercept or manipulate it. However, to avoid that an adversary uses the same local interface to manipulate the ICDR, the interface must be secured accordingly (e.g., with a secret PIN or using a secure enclosure).
- *Predefined mode*: In this mode, the initial configuration data cannot be altered neither physically nor via the network. It is provided by the device manufacturer. (e.g., fixed ICDR stored in ROM). Another variant would be to use probabilistic key distribution [115]. While this mode is the most secure, it is also the most inflexible way since changing the ICDR after manufacturing is not easy to achieve.

7.5.2 Secure binding

After a device has retrieved its ICDR, it is able to join one or more secure communication relationships. In contrast to the initial configuration stage which is performed only once, a device is able to bind to multiple communication relationships anytime during runtime.

After the device has gained physical access to the network medium, it has to become a logical member of the network segment where it is connected to (*network join*). To achieve this, the device has to discover the so called *coordinator* that is responsible for the network segment. After the coordinator has been identified, the coordinator proves the identity of the joining device and vice versa (*mutual entity authentication*). If the device possesses the required access rights (*authorization*), the coordinator distributes the so called *Network Configuration Data Record (NCDR)* to the device.

After becoming a member of the connected network segment, each device can become member of additional communication relationships (*relationship join*). Here, two cases are distinguished. First, a device is able to set up a session with other devices (*session establishment*). To achieve this, the initiating device as well as the second device that is involved in the session must retrieve a so called *Session Configuration Data Record*

[10]The upload protocol for local mode depends on the used mechanism. For local interfaces that use packet-oriented transmission protocols (e.g., an EIA-232 interface that uses the Universal Asynchronous Receiver/Transmitter (UART) protocol), a similar upload protocol as shown in Figure 7.12 can be used.

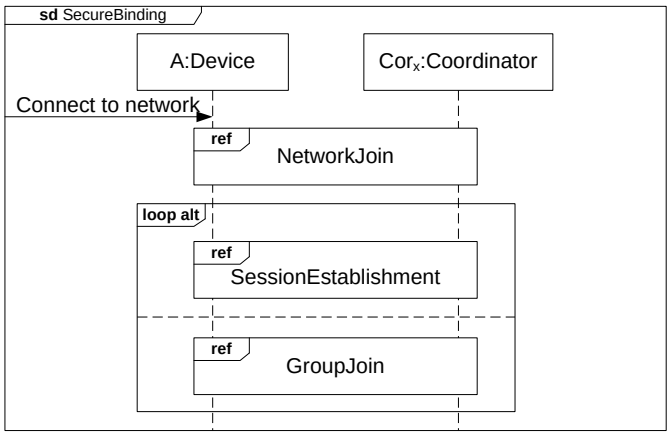


Figure 7.13: Stage II: Secure binding

(SCDR) from the coordinator. Additionally, a device can become member of one or more communication groups (*group join*). Joining a group is similar to joining a network. The device contacts the coordinator of the group. If the device has the necessary access rights, the coordinator sends the so called *Group Configuration Data Record (GCDR)* to the device.

Figure 7.13 presents the basic principle of the secure binding procedure. The generic sequence diagrams `NetworkJoin`, `SessionEstablishment`, and `GroupJoin` have to be replaced by *binding protocols*. The main objectives of such a binding protocol are:

- *Coordinator discovery*: Each joining device must be able to find the coordinator that is responsible for handling the joining request. Since the ICDR only consists of the ID and the ISTS of the device, some kind of discovery protocol is necessary.
- *Mutual entity authentication*: To prove the identities of the joining device as well as of the coordinator, an authentication protocol is necessary.
- *Authorization*: After the devices have been identified, the access rights of the joining device have to be verified.
- *Secure configuration data transfer*: To avoid security attacks, the distribution of the NCDRs, SCDRs, and GCDRs has to be protected using adequate security mechanisms. This is accomplished using cryptographic schemes that take the ISTS which has been received during stage I as input parameter. Note that the coordinators must also be configured accordingly in order to securely communicate with the joining devices.
- *Fault tolerance*: The use of a single coordinator clearly introduces a single point of failure. Therefore, a single coordinator is only advantageous for small HBA networks where the control applications have relaxed requirements regarding ro-

bustness and availability. If networks consist of many devices (e.g., a backbone or wide-ranging field network) and high service availability is mandatory (e.g., alarm systems), the binding protocol must be tolerant to faulty coordinators.

To fulfill these requirements, two different protocols that vary in the election of the coordinator exist. First, *predefined coordinators* can be used (*static binding protocol*). To be able to contact them, the address information of available coordinators is obtained during the network join. The protection of the communication between the joining devices and the coordinators is based on symmetric or asymmetric cryptographic schemes. The second protocol uses *dynamic coordinators* where the election is based on a democratic approach (*dynamic binding protocol*). This approach is only reasonable if asymmetric cryptographic schemes are used.

Static binding protocol

This binding protocol is based on static coordinators which are elected during the initial configuration stage. Since a central coordinator that is dedicated for the whole HBA network introduces a single point of failure, the presented approach uses a distributed scheme. Each network segment has one *coordinator* that processes all incoming management requests that concern network management tasks within its segment. The advantage of this distributed scheme is that only one network segment is affected in case of a coordinator failure. The coordinator functionality can be implemented in two ways. First, it is possible to add a dedicated coordinator to each network segment. Second, an existing device (e.g, a router that interconnects the network segment with its parent segment) can be extended with coordinator functionality.

Although the use of one coordinator per network segment clearly confines the effects of a failure within this segment, a single point of failure (at least for the particular segment) remains. Therefore, a so called *coordinator cluster* can be used instead of a single coordinator. A coordinator cluster consists of multiple coordinators that are conjointly responsible for network management within one network segment. The advantage of a coordinator cluster is that a joining device can freely select any coordinator out of the cluster for management requests. If the selected coordinator fails, the device can switch to another coordinator from the same cluster. Figure 7.14 shows the basic principle. Although coordinator clusters are more fault tolerant than a single coordinator, some form of synchronization protocol among the cluster devices is required. A solution will be discussed later in this chapter.

Based on the used cryptographic schemes, two different static binding protocol options

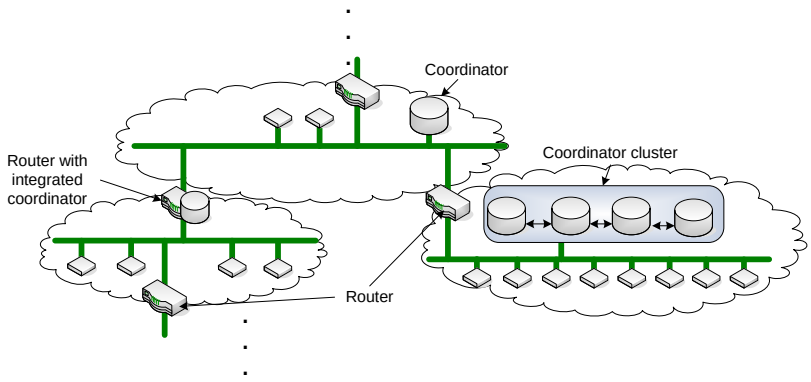


Figure 7.14: Coordinator clusters

are available. The first one is exclusively based on symmetric cryptographic schemes (*symmetric binding protocol with predefined coordinators*). Consider, for example, device A that wants to join network segment N_x. To achieve this, A has to be configured with the following ISTS during stage I:

$$ISTS_A = t_g||t_v||k_{A,m}||t_e||t_d||k_{A,c}$$

where t_g and t_v denote the public security tokens of the used MAC generation and verification transformation and $k_{A,m}$ denotes the corresponding shared secret key. Additionally, t_e and t_d represent the public security tokens of the used symmetric encryption/decryption transformation and $k_{A,c}$ denotes the corresponding shared secret key.

Furthermore, it is assumed that all coordinators that may come into consideration for network joins are also in possession of these shared secret keys. Therefore, the ISTS of each coordinator of network N_x consists of the following items:

$$ISTS_{Cor_{x,y}} = t_g||t_v||t_e||t_d||(ID_A, k_{A,m}, k_{A,c})||(ID_B, k_{B,m}, k_{B,c})||\ldots$$

where t_g and t_v denote the public security tokens of the used MAC generation and verification transformation, and t_e and t_d represent the public security tokens of the used symmetric encryption/decryption transformation. $(ID_A, k_{A,m}, k_{A,c})$ denotes the security tokens that are shared with device A. Obviously, depending on the number of devices, the ISTS of a coordinator may be large especially if it is assumed that a device may be able to join any network segment. However, the spreading of most of the devices within the HBA network is quite static. Only a few devices (e.g., MDs) are of dynamical nature where their position within the network may vary. Therefore, to minimize the ISTS for coordinators, the amount of network segments a particular device may join can be limited. Alternatively, the asymmetric variant of the binding protocol can be used.

While coordinators are also configured a priori, an update of the coordinators' ISTSs at a later point in time may be necessary. For example, if a new device is added to the HBA

network during runtime, the shared secret keys must also be uploaded to the coordinators that may come into consideration for network joins. Therefore, it must be possible to update the coordinators' ISTSs during runtime.

The second protocol option is based on asymmetric cryptographic schemes (*asymmetric binding protocol with predefined coordinators*). Here, the ISTS of the device that wants to join consists of the following items:

$$ISTS_A = t_g||t_v||t_e||t_d||(k_{A,p}, k_{A,s})||CERT_A||k_{CA,p}$$

where $(k_{A,p}, k_{A,s})$ denotes the public/private key pair of A, $CERT_A$ the so called *certificate* of A, and $k_{CA,p}$ the public key of a trusted third party (also known as Certification Authority (CA)). The main aim of a certificate is to bind a public key to the owner's identity. In practice, a certificate represents a special data record that consists of the public key of a device, some additional information, and a signature that proves the authenticity of the certificate. This signature is calculated by the CA during the certificate generation in stage I. In the proposed solution, the certificate of A consists of the following items:

$$CERT_A = ID_A||k_{A,p}||REV_{CERT_A}||s_{CERT_A}$$

where ID_A denotes the identity of A, $k_{A,p}$ A's public key, REV_{CERT_A} the revision number of the certificate, and s_{CERT_A} the signature of the certificate which has been calculated by the CA (i.e., $s = g(t_g, k_{CA,s}, ID_A||k_{A,p}||REV_{CERT_A})$. Using this certificate, the device is able to prove the validity of its public key by sending it to a requesting device which can verify the correctness by verifying the signature of the certificate. Additionally, using the public key of the CA ($k_{CA,p}$), A is able to prove the validity of certificates of other devices. In practice, the MD that distributes the ICDRs during stage I will assume the role of the CA.

Furthermore, it is assumed that all coordinators that may come into consideration for network joins are in possession of the following ISTS:

$$ISTS_{Cor_{x,y}} = t_g||t_v||t_e||t_d||(k_{Cor_{x,y},p}, k_{Cor_{x,y},s})||CERT_{Cor_{x,y}}||k_{CA,p}$$

where $(k_{Cor_{x,y},p}, k_{Cor_{x,y},s})$ denotes the public/private key pair of the coordinator, $CERT_{Cor_{x,y}}$ its certificate, and $k_{CA,p}$ the public key of the CA.

As can be seen, the ISTS of the asymmetric protocol option is much smaller than the one of the symmetric option. However, the disadvantage is the additional computational effort introduced by asymmetric cryptographic schemes (cf. Chapter 8). Therefore, it is suggested to use the symmetric protocol option for embedded devices with limited system

resources if their position within the network is static (e.g., embedded SAC devices). The use of the asymmetric protocol option avoids that coordinators must store a large number of security tokens. Therefore, the asymmetric protocol option is advisable for devices that may change their location within the network. However, due to the nature of asymmetric schemes, enough system resources must be available for these devices.

Regardless whether the symmetric or asymmetric protocol option is chosen, each device and each coordinator have a *monotonically increasing counter*. This counter is exclusively used for securing messages that are exchanged between coordinators and their devices[11]. The current counter value is used as input parameter $TEXT$ for the security sublayer. Since it is included in the MAC or digital signature generation, it acts as TVP. Due to the time variant property, it prevents replay attacks and thus guarantees data freshness. The counter value is set to zero during the configuration stage I. To avoid a loss of the current value after power-up, it is stored in non-volatile memory. However, in contrast to the STSs, the counter values do not have to be distributed a priori. It is sufficient to store a device's current counter value after the first request is retrieved from that device.

To start the network join, A sends a `network-search.req` message using local, best-effort broadcast (cf. Figure 7.15)[12]. If the asymmetric protocol option has been chosen, the certificate of A is included in the message. To avoid DoS attacks, the message is a trusted one where the current counter value c_A is used as parameter for the MAC or digital signature calculation. If the message would be a protected one, adversaries would be able to replay previously sent `network-search.req` requests that would generate unnecessary load at the coordinators. All available coordinators i.e., all coordinators of the coordinator cluster receive this message and verify it. In case of asymmetric binding, the received certificate is also verified and the public key of A is extracted. If the message is valid (i.e., the enclosed MAC or signature is correct and the included counter value is greater than the locally stored one), the identity of the device has been proved. Afterwards, each coordinator verifies whether the device has the necessary access rights to join the network (*authorization*). If it is allowed, each coordinator responds with a trusted `network-search.res` message. The message contains the ID of A, the device address of A, the device address of the coordinator, and a quality-of-service parameter $Q_{Cor_{x,n}}$ that indicates the current load of the coordinator. The message also includes

[11] As it will be shown later in this section, a coordinator's counter is also used for messages that are sent to other foreign coordinators.

[12] For the rest of this chapter, it is assumed that the management entity uses the services provided by the reliability/ordering sublayer. The used notation shows the security level and the QoS properties that are guaranteed.

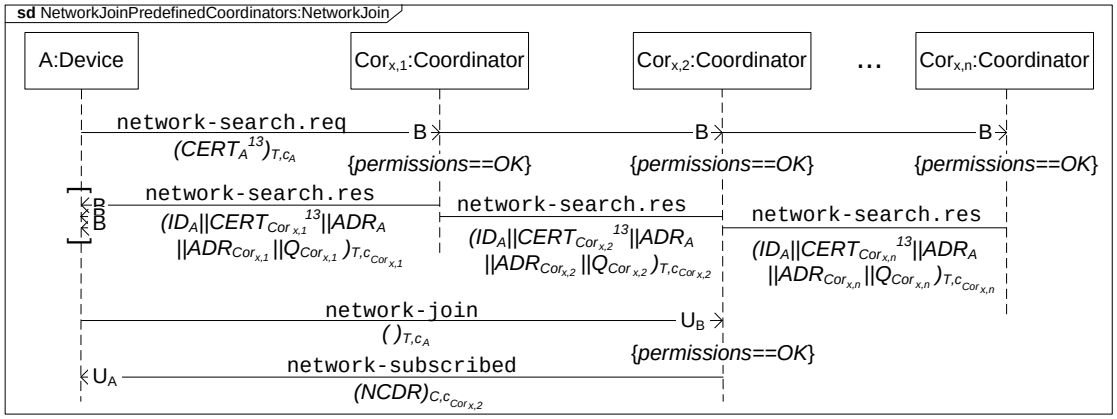


Figure 7.15: Network join using predefined coordinators

the certificate of the coordinator if asymmetric binding has been chosen. After having received these responses, the device proves the identities of the coordinators by verifying the received trusted message. If asymmetric binding is used, the certificate is also verified. If both are valid, the device is aware of the addresses of all coordinators within its network segment as well as their current load. To retrieve the NCDR, A sends a trusted, best-effort `network-join` unicast message to the coordinator with the lowest load. Alternatively, the coordinator can be chosen randomly. The coordinator responds with a `network-subscribed` message using acknowledged unicast.[14] The message contains the NCDR of the network segment. To avoid an unwanted disclosure of the transmitted NCDR, this message is a confidential one. As from now, the device is able to communicate within the network segment using the values of the received NCDR. Furthermore, the device is able to process incoming secured network management messages.

After a device has become member of the network, it is able to join one or more relationships. Consider, for example, a device A wants to establish a secure session to a device B. To start the session establishment process, A sends a trusted `session-start` message to one of the coordinators within the coordinator cluster using best-effort unicast. The request contains the ID of B. After having received this request, the coordinator generates an SSTS which is distributed to B using a confidential, acknowledged `session-init` message. The message includes the ID of A and the SCDR that among others contains the SSTS. If B is capable of accepting the session request of A (e.g., B has sufficient resources to host more sessions), B acknowledges the request from the coor-

[13]The `network-search.req` message only contains the device's certificate if the asymmetric protocol option is used.

[14]Using reliable unicast instead of acknowledged is not necessary. If the acknowledgement is missed by the coordinator, it will simply retransmit the message until it has been received correctly.

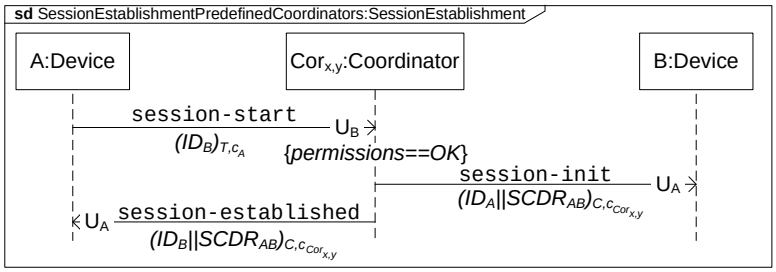


Figure 7.16: Session establishment using the static binding protocol

dinator[15]. The coordinator is now informed that B will accept the session request of A. Thus, it distributes the SCDR containing the SSTS to A using a confidential, acknowledged `session-established` message. After reception of the SSTS, A and B can securely communicate with each other.

To join a group, the device has to subscribe to it (cf. Figure 7.17). Consider, for example, device A wants to join a group G_z. To start the subscription process, A sends a trusted, best-effort `group-join` message to any coordinator within the coordinator cluster. This message contains the ID of the group. Then, the coordinator verifies whether A is allowed to join the group. If A is allowed to join the group, the coordinator sends a confidential, acknowledged `group-subscribed` message to A. This message includes the ID of the group and the GCDR which among others contains the GSTS. Using this GSTS, A is able to securely communicate within the group.

As shown in Figure 7.14, a two-tiered coordinator infrastructure is used in the proposed solution: First, each network segment has a dedicated coordinator cluster. Second, each of these coordinator clusters consists of one or more coordinators. Since a device can choose any coordinator out of the network segment's cluster, it must be avoided that the used CDRs become inconsistent. This especially concerns the used counter values for providing data freshness. To achieve this, two possible solutions exist. On the one hand, the CDR space can be split into equal parts with the CDRs being distributed across the different coordinators (*CDR distribution*). The main advantage is that a failure of a single coordinator only effects a smaller number of items. However, the main drawback is that the single point of failure remains. If a coordinator fails, all CDRs that are assigned to this coordinator become unavailable.

On the other hand, a redundancy scheme where each CDR is replicated to all coordinators can be employed (*CDR replication*). Using such a scheme, the failure of a single coordinator does not effect the availability of the CDRs since a device can simply contact

[15]The acknowledgment is not shown in the figure since it is part of the acknowledgment protocol (cf. Figure 7.8(a)).

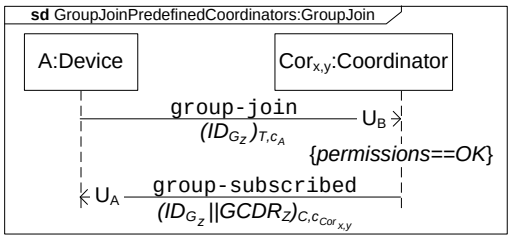


Figure 7.17: Group join using predefined coordinators

another coordinator. The price for this redundancy is the synchronization effort, as any changes must be propagated to all other coordinators to avoid inconsistencies.

The main aim of the proposed solution is to combine the advantages of both schemes. First, the CDR space is split and distributed across the network segments. This means that each coordinator cluster is responsible for a part of all CDRs. Second, these CDRs are replicated among all coordinators within a coordinator cluster.

In the following, this hybrid solution is discussed separately for each CDR type:

- *NCDR*: Each coordinator cluster is responsible for maintaining the NCDR of its network segment. Within the coordinator cluster, the NCDR is replicated.
- *GCDR*: In this dissertation, it is defined that exactly one coordinator cluster is responsible for a communication group. This maintaining coordinator cluster acts as a group coordinator and is responsible for managing the group membership and the GCDR of the corresponding communication group. The assignment scheme depends on the underlying technology. For example, the assignment of the communication group to its maintaining coordinator cluster can be based on the number of group members within the network segment in which the cluster is located. Alternatively, a special group address assignment where each group address is uniquely mapped to a network segment can be used. Definition of such an assignment scheme is left open to the initial configuration. Within the coordinator cluster, all GCDRs the coordinator cluster is responsible for are replicated.
- *SCDR*: Each coordinator cluster is responsible for handling session establishment requests within its network segment. Since an SCDR is only valid during a single session, a replication of SCDRs is not necessary.
- *ICDR*: Each coordinator cluster only stores the ICDRs of the devices that come into consideration for a network join. Within the coordinator cluster, these ICDRs are replicated among all coordinators. Using this scheme, each device can securely communicate with all coordinators of its network segment.

Following this approach, each device can freely choose any coordinator within the coordinator cluster without getting inconsistent CDRs. However, there are situations in

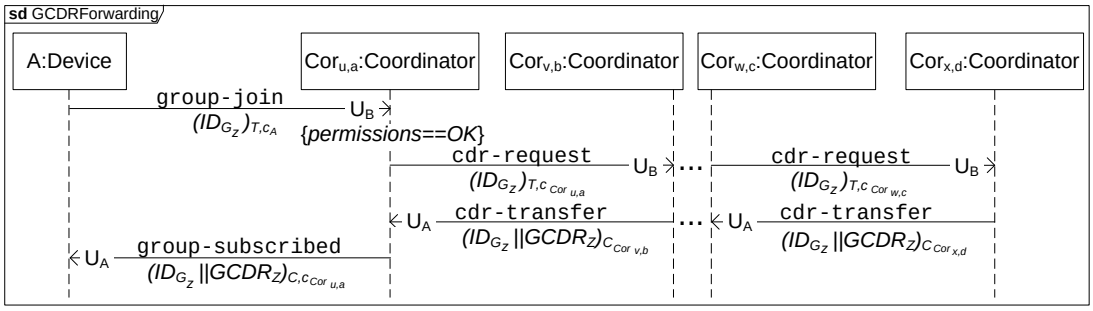


Figure 7.18: Resolving GCDRs misses

which a relationship join request cannot be immediately satisfied by the coordinator.

First, it is possible that members of a network segment want to securely communicate with a remote network segment (remote broadcast). Since the device is not in possession of the NCDR of the destination network segment, it is not able to directly communicate with the remote network segment (*NCDR conflict*). Second, it is possible that a device wants to join a group the coordinator cluster is not responsible for. In such a case the coordinator does not possess the GCDR and so it has to obtain it from the maintaining coordinator cluster (*GCDR miss*). Third, it is possible that two devices located in different network segments want to establish a session. If the symmetric protocol option is used, the coordinator contacted by the session initiator does not hold the ICDR of the second communication party. If the asymmetric variant is chosen, the coordinator is not in possession of the certificate of the second communication party. As a result, the coordinator cannot distribute the generated SSTS to the second communication party since the coordinator is not able to secure the `session-init` request. Furthermore, it is not able to verify the corresponding acknowledgment (cf. Figure 7.16). Therefore, the missing security tokens have to be requested from the coordinator of the network segment where the second device is located (*ICDR miss*).

To overcome an NCDR conflict, the routers of the network segment have to forward remote broadcast messages. This means that the router has to disassemble the message using the NSTS of the incoming network segment. Afterwards, it has to secure it with the NSTS of the next hop network segment. Then, it has to forward it. If it has reached the destination network segment, the members retrieve the message. Otherwise, the next router has to forward it again.

To resolve GCDRs misses, a mechanism has to be provided that allows the retrieval of the required GCDR from the remote coordinator cluster that is responsible for it (cf. Figure 7.18). The GCDR exchange is initiated by sending a trusted `cdr-request` message to any coordinator of the next coordinator cluster. To achieve this, each coordinator must

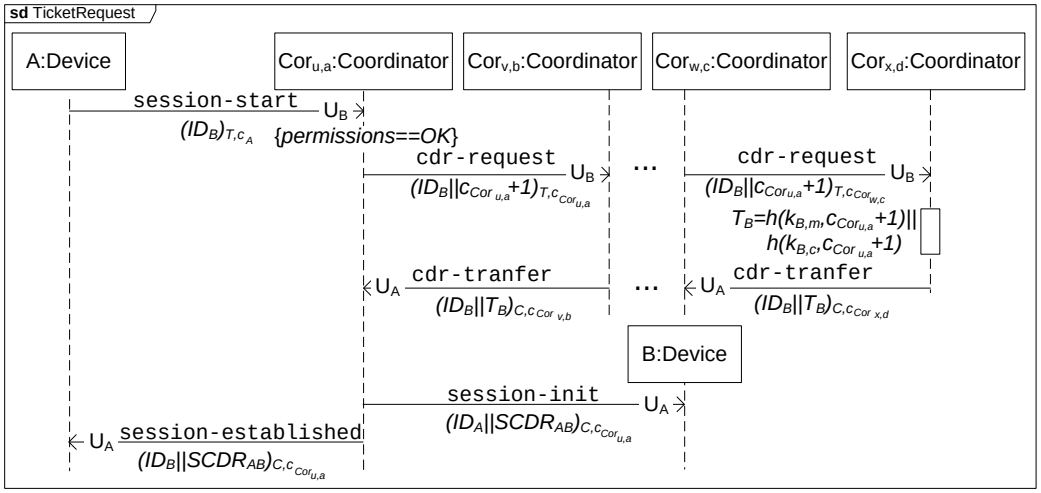


Figure 7.19: Resolving ICDRs misses for symmetric protocol option

be aware of the device addresses of the coordinators of its neighbor network segments. Additionally, it must be possible to securely communicate with them by either sharing secret keys for the symmetric protocol option or by having the certificates of the remote coordinators for the asymmetric protocol option. Furthermore, it is assumed that a coordinator is able to determine which of its neighbor coordinator clusters it has to contact next. This can for example be accomplished using a hierarchical addressing scheme or with the help of (static) routing tables. After having received a `cdr-request` message, the remote coordinator has two possibilities. If it has the desired GCDR, it transfers it using a confidential, acknowledged `cdr-transfer` message. Otherwise, it has to forward the request by sending another `cdr-request` message to the next cluster.

If the asymmetric protocol option of the binding protocol is used, this forwarding mechanism can also be used to resolve ICDR misses. Here, the `cdr-request` contains the ID of the requesting device instead of the ID of the group. If the remote coordinator has the desired ICDR, it responds with a `cdr-transfer` message that contains the certificate of the second device. If the symmetric protocol option is used, it would be necessary to transfer the device's shared secret keys. However, due to security reasons, it is more secure to only transfer a so called *ISTS ticket*. Figure 7.19 shows the basic concept. If a coordinator that wants to establish a session between device A and B, but does not have the shared secret key of B, it sends a trusted `cdr-request` message to the remote coordinator containing the ID of B as well as the current counter value of the coordinator incremented by one. If the remote coordinator is not the coordinator of device B, it forwards the request to the next cluster. If the remote coordinator is responsible for B, it generates a ISTS ticket. This ticket consists of a dynamic key pair that is generated us-

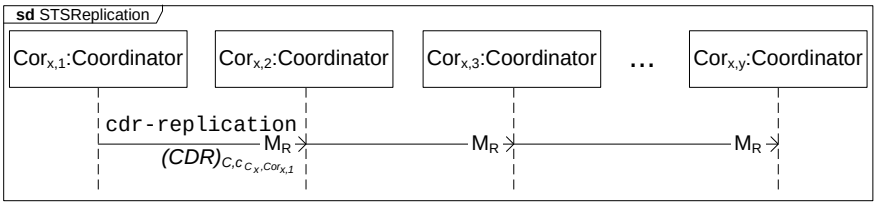


Figure 7.20: Replication of CDRs

ing the shared secret keys of B and the retrieved counter value. This ticket is transferred to the coordinator that initiated the request by using a confidential, acknowledged `cdr-transfer` message. Using this temporally valid key pair, the initiating coordinator is able to finish the session establishment by sending the `session-init` message and the corresponding `session-established` message.

As mentioned before, a mechanism that allows to synchronize the CDRs within the coordinator cluster is necessary. To achieve such a *CDR replication*, each member of the cluster is member of a special communication group called *cluster group*. Within this cluster group, the communication is secured with the cluster group's GSTS. The CDR replication is done using reliable multicast (cf. Figure 7.20). The initiating device sends a confidential, reliable `cdr-replication` multicast message to the cluster group. The message contains the new CDR that has to be replicated. Since reliable multicast guarantees that the message is received by all members or by none of them, inconsistent CDRs are avoided.

Dynamic binding protocol

In contrast to the binding protocol where static coordinators are used, this protocol is based on a democratic approach where coordinators can be dynamically elected during runtime. The main advantage is that the distribution of the coordinator functionality is not static – if a coordinator fails any other device is able to assume the role of the coordinator. Another important difference is that coordinators may not be responsible for all request within their network segments – each secure communication relationship is able to have a different coordinator.

After a device has been configured for the dynamic binding protocol, it is in possession of the following ISTS:

$$ISTS_A = t_g||t_v||t_e||t_d||(k_{A,p}, k_{A,s})||CERT_A||k_{CA,p}$$

where $(k_{A,p}, k_{A,s})$ denotes the public/private key pair of A, $CERT_A$ the certificate of A, and $k_{CA,p}$ the public key of the CA. While the structure of the ISTS is identical to the one

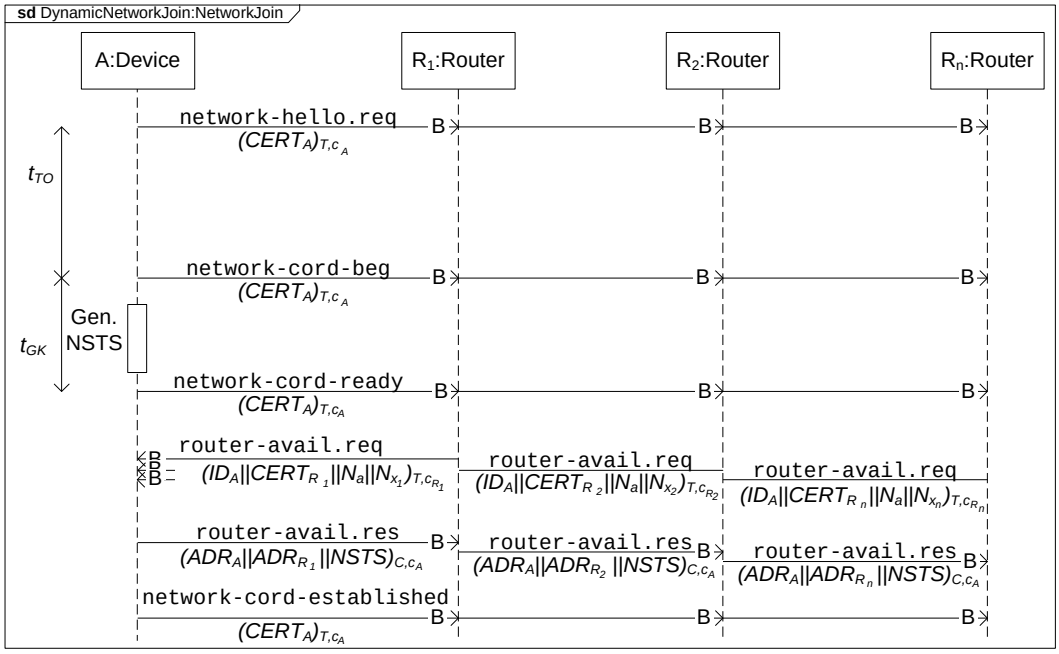


Figure 7.21: Network join using dynamic binding protocol: No coordinator present

of the asymmetric static binding protocol, the certificate of A is different:

$$CERT_A = ID_A||k_{A,p}||REV_{CERT_A}||ACL_A||s_{CERT_A}$$

The only difference to the certificate used by the asymmetric static binding protocol is the existence of ACL_A which denotes the *Access Control List (ACL)* of device A. It consists of a list of group and device IDs that the device is allowed to communicate with. This ACL determines the device's access rights – using the ACL, coordinators as well as other devices are able to prove whether the device has sufficient access rights to join a relationship. Since the ACL is part of the certificate, the containing access rights are uniquely bound to the device. The ACL is specified by the CA during the configuration stage.

The dynamic binding protocol is only reasonable if asymmetric algorithms are used. Due to the democratic concept, each device is a possible candidate for the coordinator role and thus it needs to securely communicate with all other devices. As a result, each device must hold the shared keys of all other devices, if symmetric algorithms are used. Therefore, the exclusive use of symmetric algorithms is not an option for the dynamic binding protocol.

To start a network join, device A sends a trusted `network-hello.req` message to all members of the network segment using broadcast. The message is protected using the ISTS retrieved during the configuration stage and the device's current counter value. Depending on the state of the network segment, three different cases are distinguished.

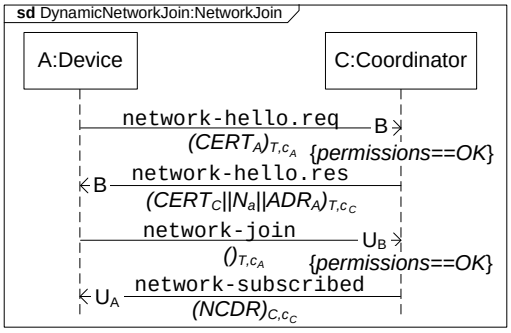


Figure 7.22: Network join using dynamic binding protocol: Coordinator available

If there is no response from other devices within the timeout t_{TO}, the device assumes that it is the first active device within the network segment[16] and so it takes over the network coordinator role (cf. Figure 7.21). To announce that it is the new coordinator and to prevent other devices from becoming the coordinator at the same time, it sends a trusted `network-coord-beg` message using broadcast. After having sent this message, it generates the NSTS for this network segment. The generation process takes the time t_{GK}. After having finished the NSTS generation process, it sends a `network-coord-ready` message. After having sent the `network-coord-ready` message, the device is coordinator but it does neither know the address of the network segment nor which routers are available at the network. Therefore, each router sends a trusted `router-avail.req` message which contains the router's certificate, the address of the network segment (N_a) as well as a list of all network segments (denoted as $N_{x_1}, N_{x_1}, \ldots, N_{x_n}$) that the router is connected to (directly or indirectly). The new coordinator receives these broadcast requests and responds with an individual, confidential `router-avail.res` broadcast message[17]. Each response contains the device address of the coordinator, the device address of the router, as well as the NSTS. Afterwards, the coordinator finishes the coordinator establishment process by sending a trusted `network-coord-established` message.

Figure 7.22 shows the situation where a coordinator is present. After having received the initial `network-hello.req` message, it extracts the enclosed certificate of the device and verifies it. If the certificate is valid, the coordinator responds with a trusted `network-hello.res` message. This message contains the certificate of the coordinator, the address of the network segment, and the device address of A that is dynamically

[16] If the HBA network is divided into more than one network segment, routers are always present. To avoid that routers always have to assume the role of the network coordinator, they are not considered as potential candidates. However, the design of the protocol does not prohibit that a router may also be a network coordinator.

[17] Since the routers do not have a device address yet, broadcast instead of unicast has to be used.

assigned by the coordinator. Using this address, the device is able to send unicast messages. To request the NCDR, the device sends a trusted `network-join` message to the coordinator. The coordinator verifies the request and responds with a confidential, acknowledged `network-subscribed` message that contains the NCDR of the network segment.

If there exists a coordinator that does not respond for any reason (e.g., the coordinator has crashed), the new device will assume the coordinator role by sending a `network-coord-ready` message. The other network members (including the routers) will recognize this and verify both the `network-hello.req` and the `network-coord-ready` message. If they are valid, all remaining network members invalidate their device addresses as well as the current NCDR. At the same time, the coordinator establishment process continues as indicated in Figure 7.21. After having sent the `network-coord-ready`, the remaining devices have to rejoin the network by using the joining procedure shown in Figure 7.22. In principle, the new coordinator would have the opportunity to retrieve the current NCDR from any other network member instead of generating a new one. However, due to security reasons it is advisable to generate a new one since a security attack may be the reason of the coordinator failure.

In addition to the situations mentioned above, it may be possible that there is no coordinator available and at least two devices compete for the coordinator role at the same time. If two devices are sending a `network-hello.req` message at the same time (i.e., within the time interval t_{TO}) and there is no coordinator available yet, the one with the smaller ID shall become coordinator – after having proved the identity of the winning device, the other device shall cancel the coordinator election process. To achieve this, t_{TO} has to be greater than t_{GK}. This avoids conflicts during the NSTS generation process (i.e., within t_{GK}), since the second device will receive a `network-coord-ready` message before it initiates a `network-coord-beg` message. If a device sends a `network-hello.req` message between the `network-coord-ready` and the `network-coord-established` message, the new coordinator immediately responds with a `network-hello.res` and the second device has to continue the normal join procedure by sending a `network-join` message.

After a device has become member of the network, it is able to join one or more relationships. To establish a session, the initiating device A sends a trusted `session-hello.req` message that contains its certificate as well as the ID of the other device (cf. Figure 7.23). Since A is not aware of the device address of B and of the network address where B is connected to, the message is sent as global broadcast. If there is no response

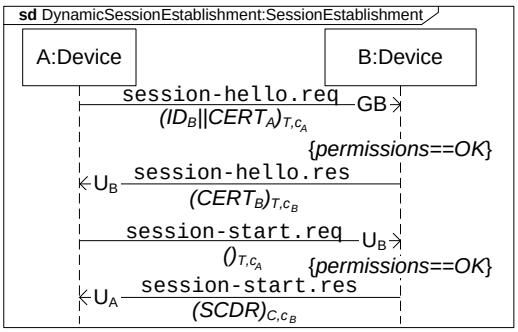
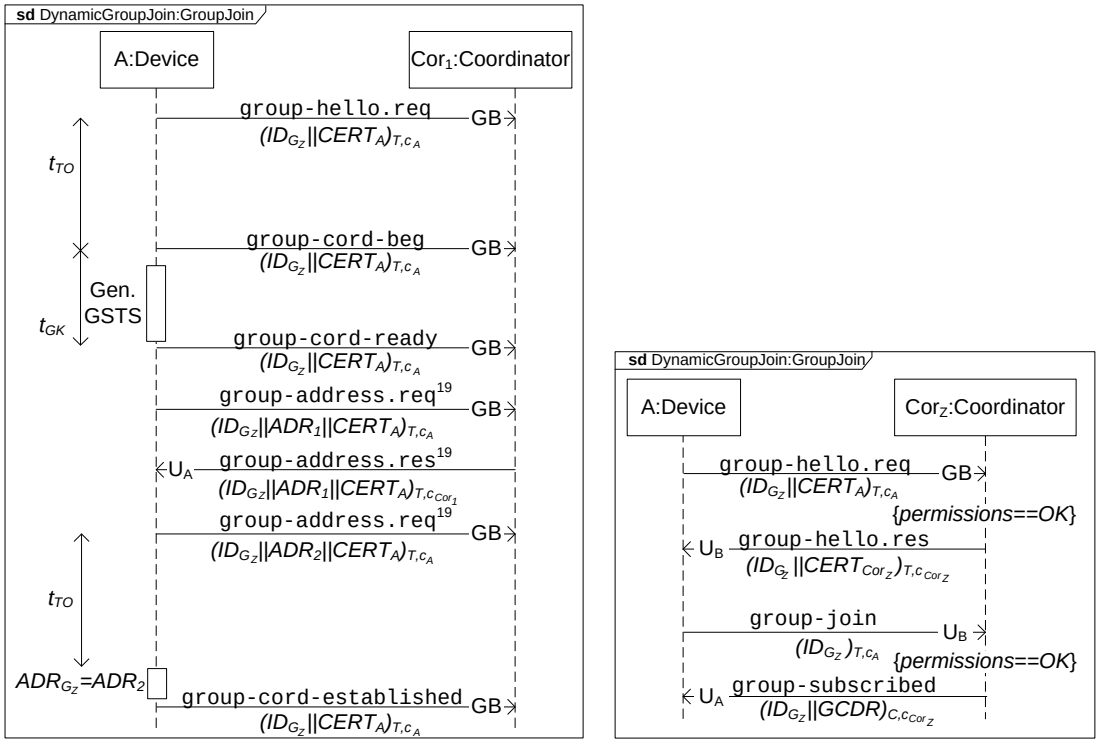


Figure 7.23: Session establishment using the dynamic binding protocol

within a specific timeout t_s[18] (e.g., B is not active), the session establishment process is retried or aborted. If B receives the message and the encapsulated ID matches its own, it responds with a trusted `session-hello.res` message containing its certificate. Since B already knows the address of A, a unicast message can be used. Afterwards, A sends a trusted `session-start.req` message using unicast to B. B receives this message, generates an SCDR, and transmits it to A using a confidential, acknowledged `session-start.res` message.

Joining a group is similar to joining a network segment. Consider, for example, A wants to join a communication group G_z. To begin the joining process, A sends a trusted `group-hello.req` message. Since A only knows the ID of the group, it is not in possession of the address dedicated to the group. Therefore, `group-hello.req` is sent using global broadcast. If there is no response within timeout t_{TO}, A is the first device that wants to join the communication group (cf. Figure 7.24(a)). This implies that there is no group coordinator available and so it assumes the role of the group coordinator. To announce that A will become group coordinator, A sends a trusted `group-cord-beg` message. Again, this message avoids that another device tries to become group coordinator at the same time. Afterwards, A generates the GSTS that will be used in stage III to securely communicate with the other group members. After having generated the GSTS, A sends a `group-cord-ready` message. If native multicast is not supported by the data link layer, the coordinator establishment process is finished by sending a `group-cord-established` message. However, if the underlying data link layer supports native multicast, additionally a data link address has to be assigned to the group. Since A is not aware of other communication groups, it has to find an address that is not used by another communication group. Therefore, A randomly chooses an address out of the address space for communication groups. Then, it sends a `group-address.req` mes-

[18] Again, this timeout depends on the used network technology and on the present topology. For more details see Section 7.3.



(a) No coordinator available (b) Coordinator available

Figure 7.24: Group join using the dynamic binding protocol

sage containing the randomly generated address to all network members using global broadcast. If this address is already in use by another communication group, the corresponding coordinator responds with a trusted `group-address.res` message and a new address has to be chosen. This procedure is repeated until there is no response from any coordinator within timeout t_{TO}. To finish the coordinator establishment process, A sends a `group-cord-established` message.

If there is already a coordinator available, the coordinator responds with a `group-hello.res` message (cf. Figure 7.24(b)). The remaining steps are similar to joining a network – A sends a trusted `group-join` message to the coordinator which responds with a confidential, acknowledged `group-subscribed` message that contains the GCDR.

7.5.3 Secure communication

After a device has joined a communication relationship, it is able to securely communicate with the other members of this relationship. The communication is performed using the services of the reliability/ordering sublayer (cf. Figure 7.6). However, due to performance

[19] These messages are only required if native multicast is supported.

reasons, symmetric schemes are exclusively used during the secure communication stage – the use of asymmetric schemes is limited to the secure binding and unbinding stage. The required parameters for these services (i.e., $Level$, $TEXT$, and optional QoS) are specified by the corresponding CDR of the relationship which is received during the relationship binding. The management entity has the aim to maintain the CDRs and to pass the current values to the reliability/ordering sublayer whenever a send or a receive service is invoked.

The content of the CDR depends on the relationship type. In case of a network relationship, the corresponding NCDR is identical to all network members. For a network segment N_x, the NCDR consists of the following data items:

$$NCDR_x = N_x||Level_{N_x}||R_{N_x}||TEXT_{N_x}||NSTS_x$$

where N_x denotes the network address, $Level_{N_x}$ the used security level of the network relationship, and $TEXT_{N_x}$ the parameter that is integrated into the MAC calculation. R_{N_x} contains the routing information of the network which consists of a list of available routers and their connected network segments. $NSTS_x$ denotes the NSTS of network N_x (cf. Section 7.2). The NSTS contains the security tokens that are assigned to network N_x. The contents of $TEXT_{N_x}$ depend on the chosen security level:

$$TEXT_{N_x} = \begin{cases} SEED & \text{if } Level_{N_x} == Protected \\ c_{N_x} & \text{if } Level_{N_x} == Trusted \vee Confidential \end{cases}$$

where $SEED$ specifies a fixed binary string and c_{N_x} a monotonically increasing counter that is shared between all members. Whenever a device sends a local broadcast message with security level $Trusted$ or $Confidential$, it uses the common network counter as parameter $TEXT$ and increments it afterwards. The main benefit of using a single counter instead of one for each sender is that the members of the network segment need not to be aware of each other. The result is a loose group membership which is also advantageous if a new device joins the network since a notification of the other members is not necessary. However, while duplicates are always prevented, it may be possible that two devices transmit a message using the same counter value. Consider, for example, device A broadcasts a message. If B does not receive it for any reason, B will not increment the counter. Since all remaining members (including A) increment it, a message that is sent by B will be detected as a duplicate and discarded. However, since a broadcast does not guarantee liveness at all, multicast has to be used if a possible loss of messages cannot be tolerated by the application.

In case of a session relationship, the SCDR that is shared between device A and B consists of the following items:

$$SCDR_{AB} = ADR_A||N_A||ADR_B||N_B||Level_{AB}||Order_{AB}||QoS_{AB}||TEXT_{AB}||SSTS_{AB}$$

where ADR_A, N_A as well as ADR_B, N_B denote the addresses of the involved devices. $Level_{AB}$ specifies the session's security level, $TEXT_{AB}$ the binary strings that are used for MAC calculation, $Order_{AB}$ the required ordering (SSF, $Causal$, or $Total$), and QoS_{AB} the communication service type that has to be used by the reliability/ordering sublayer ($Best-effort$, $Acknowledged$, or $Reliable$). $SSTS_{AB}$ denotes the SSTS that is dedicated to the session. The content of $TEXT_{AB}$ depends on the security level and on the chosen ordering requirement:

$$TEXT_{AB} = \begin{cases} SEED & \text{if } Level_{AB} == Protected \\ c_{AB} = 0||c_{BA} = 0 & \text{if } (Level_{AB} == (Trusted \vee Confidential)) \wedge \\ & \quad (Order_{AB} == (SSF \vee Causal)) \\ current\_timestamp & \text{if } (Level_{AB} == (Trusted \vee Confidential)) \wedge \\ & \quad (Order_{AB} == Total) \end{cases}$$

where $SEED$ specifies a fixed binary string. If SSF or causal ordering is demanded, two monotonically increasing counters are used – one for each sender. Since a session is newly created during the establishment process, both counters are initialized to zero. If total ordering is required, synchronized timestamps are used. Whenever a device sends a message, it obtains the current timestamp and passes it to the security layer. However, since synchronized timestamps are set by the synchronization protocol, there is no need for an initialization.

In case of multicast communication, each communication group has a dedicated GCDR. A GCDR consists of the following items:

$$GCDR_x = \begin{cases} ADR_{G_x}||M_{G_x}||Level_{G_x}||Order_{G_x}||QoS_{G_x}|| & \text{if data link multicast is supported} \\ TEXT_{G_x}||GSTS_x & \\ M_{G_x}||Level_{G_x}||Order_{G_x}||QoS_{G_x}||TEXT_{G_x}|| & \text{if global broadcast is used} \\ GSTS_x & \\ ADR_A||N_A||\ldots||ADR_N||N_N||M_{G_x}||Level_{G_x}|| & \text{if multiple unicasts are used} \\ Order_{G_x}||QoS_{G_x}||TEXT_{G_x}||GSTS_x & \end{cases}$$

If the data link layer supports native multicast, the GCDR contains the data link multicast address ADR_{G_x}[20] to which the communication group is mapped to. If native multicast

[20] If the device has two or more network interfaces, one data link multicast address for each network interface has to be specified.

is not supported, global broadcast or multiple unicasts have to be used. While, for global broadcast, there is no need for any additional address information, a list of all group members' addresses has to be provided if multiple unicasts are used. M_{G_x} specifies the IDs of all group members, $Level_{G_x}$ the group's security level, $Order_{G_x}$ the required ordering, $TEXT_{G_x}$ the used parameter for MAC calculation, and QoS_{G_x} the communication service type that has to be used for exchanging data within the group. $GSTS_x$ denotes the GSTS of relationship G_x. The contents of $TEXT_{G_x}$ depend on the security level and on the chosen ordering requirement:

$$TEXT_{G_x} = \begin{cases} SEED & \text{if } Level_{G_x} == Protected \\ c_{G_x,A}, c_{G_x,B}, \ldots & \text{if } (Level_{G_x} == (Trusted \vee Confidential)) \wedge \\ & (Order_{G_x} == SSF) \\ v_{G_x} & \text{if } (Level_{G_x} == (Trusted \vee Confidential)) \wedge \\ & (Order_{G_x} == Causal) \\ current\_timestamp & \text{if } (Level_{G_x} == (Trusted \vee Confidential)) \wedge \\ & (Order_{G_x} == Total) \end{cases}$$

where $SEED$ specifies a fixed binary string. If SSF ordering is demanded, each member (or at least each member that is able to send messages) has its own monotonically increasing counter. In contrast to sessions, a communication group may already exist. Therefore, these counters are initialized with the current values during the group join. If causal ordering has to be guaranteed, vector timestamps have to be used instead. Here, each device has its own local vector clock (denoted as v_{G_x}) which is used to generate vector timestamps whenever it sends a message. Obviously, the local vector clock of the joining device has to be initialized during the binding process, too. If total ordering is required, synchronized timestamps are used. Since synchronized timestamps are set by the synchronization protocol, there is no need for an initialization.

In contrast to broadcast communication, the members must know each other since each device must monitor the current counter values of the other members. Therefore, the group coordinator must notify the communication group whenever a new device joins the group. This done by sending a `group-inform` message to all group members after the join process was successful. The message includes the ID of the joining device.

To protect the communication within the relationship, the content of the STS that is included in the relationship's CDR is used as input parameter for the symmetric cryptographic schemes. Due to security reasons, the lifetime of the shared secret parts of the STSs (i.e., the shared secret keys) has to be limited. In general, the lifetime of secret keys shall correlate with the number of their uses (cf. Figure 7.25). Shared secret keys of com-

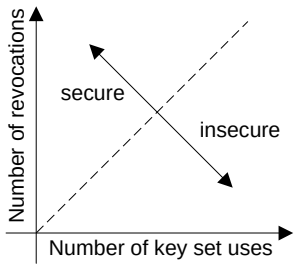


Figure 7.25: Revocation vs. number of uses

munication relationships that are used more often shall be changed more frequently than the keys that are used rather rarely. However, since changing shared secret keys introduces additional effort, it is not always possible to change shared secret keys as frequently as required for maximum security.

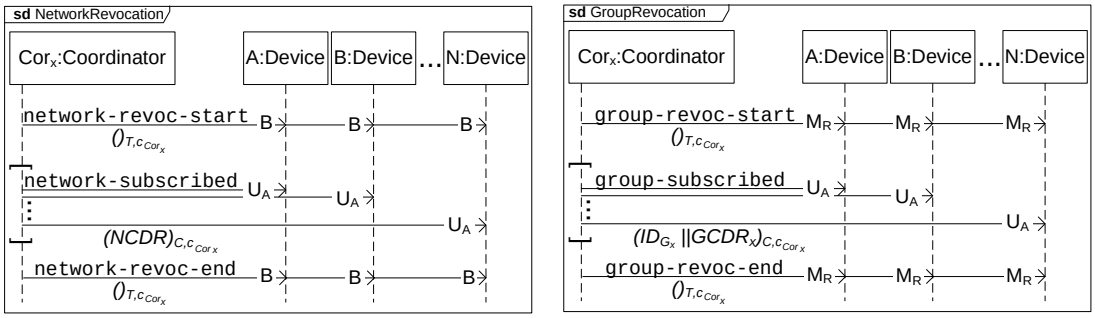
Concerning the relation between old and new shared secret keys, secret key changing methods can be classified according to the following three properties:

- *Backward secrecy* guarantees that an adversary that knows a contiguous subset of keys cannot discover preceding keys.
- *Forward secrecy* guarantees that an adversary that knows a contiguous subset of keys cannot discover subsequent keys.
- *Key independence* is supported if both backward and forward secrecy are provided.

Depending on the key type that is changed, two different mechanisms for changing shared secret keys are provided. The first one is called *key refreshing*. Here, only the dynamic key that is calculated out of the secret key and the parameter $TEXT$ is changed (cf. Figure 7.4 in Section 7.2). This key refreshing mechanism takes advantage of the properties of the hash function. If a cryptographic hash transformation that fulfills the properties defined in Section 4.1 is used, key independence of the used dynamic keys is provided. This means that if an adversary with access to the network is able to compromise a dynamic key (e.g., using brute force), it is not able to calculate preceding or subsequent dynamic keys as long as the used cryptographic hash transformation is secure.

For trusted and confidential messages, key refreshing is implicitly provided. Due to the time variant property of the used parameter $TEXT$, it is guaranteed that a new dynamic key is generated for each request. For protected messages, the fixed binary string $SEED$ has to be changed. This can be done by sending a trusted `sts-refresh` message that contains the new $SEED$ value.

However, if a relationship member gets compromised (e.g., due to a device attack), the shared secret keys contained in the STS gets disclosed, too. In such a case, a so called *key revocation* is necessary. Here, the shared secret keys within the STS have to be in-

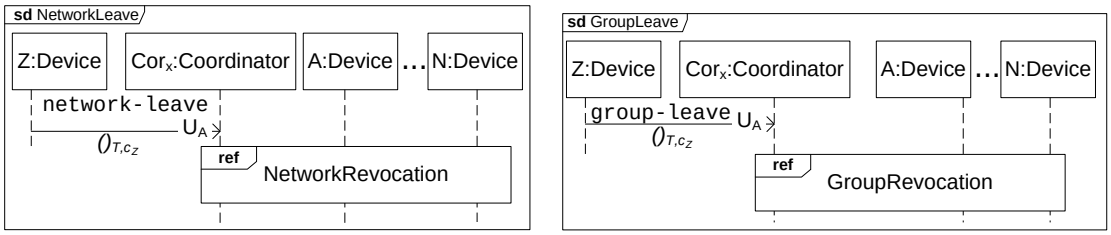


(a) Network revocation (b) Group revocation

Figure 7.26: STS revocation

validated and new ones must be distributed by the coordinator. To start the revocation process of a relationship's STS, the responsible coordinator (or one coordinator out of a coordinator cluster) sends a `revocation` message to all relationship members – that is `network-revoc-start` for network relationships and `group-revoc-start` for group relationships. This message is secured with the current (old) STS and indicates that a revocation process is under way. From now on, the relationship is in state "locked" i.e., sending data messages as well as joining to the relationship are forbidden. After having sent the message, the coordinator generates a new STS and distributes it to all relationship members. The distribution of the new STS is done by redistributing the relationship's CDR using the acknowledged `subscribed` service of the relationship binding process – for network relationships, `network-subscribed` is used and for group relationships, `group-subscribed` is taken. If the new CDR has been retrieved, the used TVPs are also reinitialized. After all relationship members are in possession of the new STS, the coordinator sends a `revocation-finished` message to all relationship members to inform them about the completion of the revocation process. This message is secured with the new STS and triggers the mandatory use of the new STS. Figure 7.26(a) and Figure 7.26(b) show how a revocation within a network or a group relationship is performed. The revocation of a session relationship is much easier. Here, the current session is simply terminated and a new one is established using the session establishment procedure.

In network and group relationships with a high node count, a revocation can clearly lead to considerable network traffic. If coordinator clusters are used, a possible improvement would be to spread the responsibility of the STS distribution to the different coordinators. However, for relationships where the devices are located at the same network segment, the problem of high traffic (at least at the local network segment) remains. To further reduce the distribution overhead, hierarchical group communication schemes can be used. Here, the relationship members are divided into subgroups where each subgroup has its own



(a) Network leave (b) Group leave

Figure 7.27: Relationship leave

STS. The main advantage of such a scheme is that the revocation can be done separately for each subgroup. However, the main disadvantage of a hierarchical scheme is that it introduces additional communication and management effort during normal operation. For example, coordinators for each subgroup are required. Furthermore, it is necessary to send data that is dedicated to the whole group multiple times – one message to each subgroup. A survey of hierarchical group communication schemes can be found in [116].

7.5.4 Secure unbinding

In addition to join a relationship, it must also be possible to explicitly exclude a device from a relationship (*secure unbinding*). There are two reasons for secure unbinding. First, members of a relationship may decide to actively banish a device from a relationship. A typical example would be a compromised device where an abnormal behavior is detected by an IDS. In such a case, it must be guaranteed that the device is no longer able to participate in the communication. From the point of unbinding on, the leaving device must be precluded from all communication within the relationship. To achieve this, the relationship's STS has to be replaced by a new one using the revocation mechanism. During the revocation, a new STS is generated which is only distributed to the remaining members. Since the excluded device does not have the new STS, it is no longer able to securely communicate with the relationship members.

Second, a device may decide on its own to leave a communication relationship because it is no longer interested in the exchanged data. An example is an MD that temporarily performs some management tasks. To indicate a relationship leave, the device sends an unbinding request to the coordinator of the relationship. The coordinator revokes the current STS and distributes a new one to the remaining members. Again, the same revocation mechanism can be used. Figure 7.27 shows how leaving a group or a network relationship is performed. Again, leaving a session is much easier. Since there is no need for revoking an SSTS, the session is simply terminated by sending a `session-close` message.

8 Evaluation

To be able to show the feasibility of the presented security concept, an evaluation is necessary. The method of choice is a hybrid one. First, the used security protocols and algorithms are formally evaluated relative to a well-defined adversary model. The evaluation relies on generic cryptographic transformations that are assumed to be secure. Second, a prototype implementation is presented. It shows the practicability of the proposed architecture and allows to study the real-world behavior. Within this prototype implementation, state of the art implementations of cryptographic transformations are used.

8.1 Formal evaluation

To be able to formally evaluate the presented security concept, an *adversary model* has to be specified. This model defines the capabilities that an adversary may have. It consists of the following assumptions:

- Considering the current computing hardware, it is assumed that an adversary has reasonable system resources both in terms of time and computational resources like processing power and memory capabilities.
- An adversary has full expertise of current techniques regarding algorithms and mathematical knowledge. This means that adversaries are able to use the best-known techniques to break security algorithms and protocols. Additionally, all cryptographic protocols and schemes are based on Kerckhoff's principle – an adversary has full access to the description of cryptographic protocols, schemes, and algorithms. Furthermore, it is assumed that an adversary knows all public security tokens that are used by cryptographic transformations. Secret security tokens like shared and private keys are not known. In a nutshell, there is no "Security by Obscurity".
- An adversary has full access to the network medium. This means that an adversary can intercept, fabricate, and modify any message at any time. Interruption threats are possible in a reasonable way – an adversary is able to redirect and drop

messages. Attacks where an adversary is able to completely interrupt the communication between two devices (e.g., by cutting the physical connection or by jamming) are considered as permanent interruption attacks. Here, it is sufficient that the attack is detected by at least one device. Additionally, it is assumed that this device is able to inform a trusted third-party (e.g., an IDS) which initiates further countermeasures (e.g., isolating the source of the attack).

- It is assumed that an adversary can access the network at any point within the network topology i.e., he can access field networks, the backbone, or WANs with access to the HBA network.
- Since device attacks are not considered within this dissertation, the following assumptions are made. As defined in Chapter 3, the smallest security context is a device. Regarding security, there are two different types of devices. Trusted devices are considered as unconditionally secure. This means that access to the trusted devices as well as to their software and stored data is not possible. Typical examples are coordinators and routers. Untrusted devices, however, are not fully protected against device attacks. It is assumed that compromising a device by an adversary is detected and reported within a time interval so that an adversary cannot cause further damage (e.g., comprise other devices, access secret keys).
- In general, legitimate devices (both trusted and untrusted devices but not adversaries) are considered as fail-silent – if they crash (either as a consequence of a security attack or due to other reasons), they do not send any messages. A recovery after a crash is done with the help of manual intervention (e.g., by a system operator).
- Furthermore, the following assumption is made. User data (i.e., data exchanged between control applications) that has to be protected against security attacks is only exchanged within the communication relationship of its interest. This means that there is no "hidden channel" which can be used to retrieve data other than from the relationship's communication channel. This also implies that the data exchanged between any two distinct communication relationships is independent.

Taking this adversary model into account, the formal evaluation is performed relative to the generic cryptographic transformations defined in Chapter 4. Thus, the evaluation relies on these cryptographic transformations that are assumed to be secure. Selecting appropriate implementations of these cryptographic transformations is task of the prototype implementation.

The ultimate goal of this formal evaluation is to analyze all communication services supported by the high-level communication interface. Therefore, the following approach is used. The communication stack of the SAL provides services that are based on a modular concept. Each sublayer enriches the native data link communication services with additional features. Therefore, the evaluation is performed "bottom-up" – beginning at the data link layer, all subsequent layers are separately analyzed up to the communication services of the high-level communication interface.

8.1.1 Data link services

As shown in Figure 7.2, the available data link/physical layer combination(s) provide native communication services that are used to send and receive messages over the native network media. In general, these data link/physical layer combinations are considered as "black channels" – regarding security and QoS no assumptions are made. From a security point of view, an adversary is able send any arbitrary data link message as well as receive all data link messages at the connected network segment. As a result, an adversary is able to perform the following security attacks at the data link layer (cf. data format defined in Section 6.1)[1]:

Interception attacks

- (B.i) Broadcast
  $(ADR_{src}||msg) \overset{\Uparrow}{\rightarrow} (ADR_{src}||msg)$
- (U.i) Unicast
  $(ADR_{src}||ADR_{dst}||msg) \overset{\Uparrow}{\rightarrow} (ADR_{src}||ADR_{dst}||msg)$
- (M.i) Multicast (if supported)
  $(ADR_{src}||ADR_{G_x}||msg) \overset{\Uparrow}{\rightarrow} (ADR_{src}||ADR_{G_x}||msg)$

[1]For the rest of this section, the following notation is used. Security attacks are classified in interception, modification, fabrication, and interruption attacks. For each group, the attacks on available services are listed. Each attack is identified by an abbreviation in the form of (X.y) where X stands for the service type and y corresponds to the attack group. Below, the attack is described. $(msg) \overset{\Uparrow}{\rightarrow} (msg)$ denotes that the message is intercepted, $(msg) \rightarrow (msg')$ indicates that message msg is modified to msg', $() \rightarrow (msg')$ represents a fabrication of a message, and $(msg) \rightarrow ()$ denotes that message msg is dropped.

Modification attacks

- (B.m) Broadcast
  $(ADR_{src}||msg) \rightarrow (ADR'_{src}||msg')$
- (U.m) Unicast
  $(ADR_{src}||ADR_{dst}||msg) \rightarrow (ADR'_{src}||ADR'_{dst}||msg')$
- (M.m) Multicast (if supported)
  $(ADR_{src}||ADR_{G_x}||msg) \rightarrow (ADR'_{src}||ADR'_{G_x}||msg')$

Fabrication attacks

- (B.f) Broadcast
  $() \rightarrow (ADR'_{src}||msg')$
- (U.f) Unicast
  $() \rightarrow (ADR'_{src}||ADR'_{dst}||msg')$
- (M.f) Multicast (if supported)
  $() \rightarrow (ADR'_{src}||ADR'_{G_x}||msg')$

Interruption attacks

- (B.x) Broadcast
  $(ADR_{src}||msg) \rightarrow ()$
- (U.x) Unicast
  $(ADR_{src}||ADR_{dst}||msg) \rightarrow ()$
- (M.x) Multicast (if supported)
  $(ADR_{src}||ADR_{G_x}||msg) \rightarrow ()$

Without loss of generality these security attacks represent the basic capabilities of an adversary. Obviously, it is possible that an adversary is only interested in parts of the transmitted data. Consider, for example, security attack (U.m). It may be possible that an adversary only modifies the user data msg while keeping the rest of the message (i.e., ADR_{src} andADR_{dst}) unmodified. As a result, variants of modification attacks where parts of the message are left untouched are also considered by the basic security attacks. Furthermore, an adversary is able to perform any combination of these basic security attacks. For example, replaying of a unicast message can be seen as a combination of security attack (U.i) and (U.f) i.e., a message is intercepted and resent at a later point in time. Converting messages of one type into messages of another type can also be

constructed by using these basic attack types. Consider, for example, a unicast message. From a formal point view, discarding the destination address field (denoted as ADR_{dst}) converts a unicast message into a broadcast message – replacing msg by $ADR_{dst}||msg$ converts a broadcast message into a unicast message.

8.1.2 Routing/naming sublayer

The routing/naming sublayer enriches the basic communication services of the data link layer with routing capabilities and provides a global naming scheme. While the data link layer is capable of services that are limited to a communication within a single network segment, the routing/naming sublayer supports services for communication across network borders. The resulting communication services are classified according to the communication type and destination. In case of broadcast communication, three different services are distinguished. Local broadcasts are addressed to all devices located at the same network segment as the receiver. Remote broadcast is used if a device wants to send a message to a foreign network segment. Global broadcasts have the aim to address all devices within the entire network. In case of unicast communication, the routing/naming sublayer distinguishes between messages that are exchanged between devices hosted at the same network segment (direct unicast) and devices located at different network segments (forwarded unicasts). The available multicast services depend on the underlying data link layer. If native multicast is not supported by the underlying data link layer, it has to be simulated by the routing/naming sublayer. This can be achieved by using multiple unicasts or global broadcast.

Based on this service classification, the security attacks that an adversary is able to perform at the data link layer directly affect the services provided by the routing/naming sublayer. These effects are as follows (cf. data format in Section 7.1):

Interception attacks

- (BL.i) Local broadcast
  $(ADR_{src}||msg) \overset{\Uparrow}{\rightarrow} (ADR_{src}||msg)$
- (BR.i) Remote broadcast
  $(ADR_{src}||N_A||ADR_A||N_B||msg) \overset{\Uparrow}{\rightarrow} (ADR_{src}||N_A||ADR_A||N_B||msg)$
- (BG.i) Global broadcast
  $(ADR_{src}||N_A||ADR_A||msg) \overset{\Uparrow}{\rightarrow} (ADR_{src}||N_A||ADR_A||msg)$
- (UD.i) Direct unicast

$(ADR_{src}||ADR_{dst}||msg) \overset{\Uparrow}{\rightarrow} (ADR_{src}||ADR_{dst}||msg)$

- (UF.i) Forwarded unicast
  $(ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||msg)$
  $\rightarrow (ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||msg)$
- (MN.i) Native multicast
  $(ADR_{src}||ADR_{G_x}||msg) \overset{\Uparrow}{\rightarrow} (ADR_{src}||ADR_{G_x}||msg)$
- (MU.i) Multicast using multiple unicasts
  $(ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||ID_{G_x}||msg)$
  $\overset{\Uparrow}{\rightarrow} (ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||ID_{G_x}||msg)$
- (MG.i) Multicast using global broadcast
  $(ADR_{src}||N_A||ADR_A||ID_{G_x}||msg) \overset{\Uparrow}{\rightarrow} (ADR_{src}||N_A||ADR_A||ID_{G_x}||msg)$

Modification attacks

- (BL.m) Local broadcast
  $(ADR_{src}||msg) \rightarrow (ADR'_{src}||msg')$
- (BR.m) Remote broadcast
  $(ADR_{src}||N_A||ADR_A||N_B||msg) \rightarrow (ADR'_{src}||N'_A||ADR'_A||N'_B||msg')$
- (BG.m) Global broadcast
  $(ADR_{src}||N_A||ADR_A||msg) \rightarrow (ADR'_{src}||N'_A||ADR'_A||msg')$
- (UD.m) Direct unicast
  $(ADR_{src}||ADR_{dst}||msg) \rightarrow (ADR'_{src}||ADR'_{dst}||msg')$
- (UF.m) Forwarded unicast
  $(ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||msg)$
  $\rightarrow (ADR'_{src}||ADR'_{dst}||N'_A||ADR'_A||N'_B||ADR'_B||msg')$
- (MN.m) Native multicast
  $(ADR_{src}||ADR_{G_x}||msg) \rightarrow (ADR'_{src}||ADR'_{G_x}||msg')$
- (MU.m) Multicast using multiple unicasts
  $(ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||ID_{G_x}||msg)$
  $\rightarrow (ADR'_{src}||ADR'_{dst}||N'_A||ADR'_A||N'_B||ADR'_B||ID'_{G_x}||msg')$
- (MG.m) Multicast using global broadcast
  $(ADR_{src}||N_A||ADR_A||ID_{G_x}||msg)$
  $\rightarrow (ADR'_{src}||N'_A||ADR'_A||ID'_{G_x}||msg')$

Fabrication attacks

- (BL.f) Local broadcast
  $() \rightarrow (ADR'_{src}||msg')$
- (BR.f) Remote broadcast
  $() \rightarrow (ADR'_{src}||N'_A||ADR'_A||N'_B||msg')$
- (BG.f) Global broadcast
  $() \rightarrow (ADR'_{src}||N'_A||ADR'_A||msg')$
- (UD.f) Direct unicast
  $() \rightarrow (ADR'_{src}||ADR'_{dst}||msg')$
- (UF.f) Forwarded unicast
  $() \rightarrow (ADR'_{src}||ADR'_{dst}||N'_A||ADR'_A||N'_B||ADR'_B||msg')$
- (MN.f) Native multicast
  $() \rightarrow (ADR'_{src}||ADR'_{G_x}||msg')$
- (MU.f) Multicast using multiple unicasts
  $() \rightarrow (ADR'_{src}||ADR'_{dst}||N'_A||ADR'_A||N'_B||ADR'_B||ID'_{G_x}||msg')$
- (MG.f) Multicast using global broadcast
  $() \rightarrow (ADR'_{src}||N'_A||ADR'_A||ID'_{G_x}||msg')$

Interruption attacks

- (BL.x) Local broadcast
  $(ADR_{src}||msg) \rightarrow ()$
- (BR.x) Remote broadcast
  $(ADR_{src}||N_A||ADR_A||N_B||msg) \rightarrow ()$
- (BG.x) Global broadcast
  $(ADR_{src}||N_A||ADR_A||msg) \rightarrow ()$
- (UD.x) Direct unicast
  $(ADR_{src}||ADR_{dst}||msg) \rightarrow ()$
- (UF.x) Forwarded unicast
  $(ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||msg) \rightarrow ()$
- (MN.x) Native multicast
  $(ADR_{src}||ADR_{G_x}||msg) \rightarrow ()$
- (MU.x) Multicast using multiple unicasts
  $(ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||ID_{G_x}||msg) \rightarrow ()$
- (MG.x) Multicast using global broadcast

$$(ADR_{src}||N_A||ADR_A||ID_{G_x}||msg) \rightarrow ()$$

8.1.3 Security sublayer

The task of the security sublayer is to avoid or detect security attacks. To achieve this, the security sublayer uses cryptographic techniques. Depending on the security objectives that are guaranteed within a relationship, four different security levels are distinguished: `Raw`, `Protected`, `Trusted`, and `Confidential`. In the following, each security level is evaluated separately. If a relationship has security level `Raw`, messages exchanged within the relationship are not secured at all. To avoid security attacks, other countermeasures (e.g., physical security) have to be used.

Security level `Protected` guarantees that unauthorized modification is detected by all receivers of the relationship.

Theorem 8.1. Let R_x denote a relationship with security level `Protected` where A_1, $A_2, \ldots, A_n$ are the only members of R_x. An external entity X ($X \neq A_x, \forall x \in \{1 \ldots n\}$) is not able to modify any message $(msg)_{P,TEXT}$ in a way that the resulting message $(msg')_{P,TEXT}$ is accepted by any member A_i where $i \in \{1 \ldots n\}$.

Proof. Assume to the contrary that an adversary X ($X \neq A_x, \forall x \in \{1 \ldots n\}$) is able to modify a message $(p)_{P,TEXT}$ in a way that the resulting message $(p')_{P,TEXT}$ is accepted by a member A_i where $i \in \{1 \ldots n\}$. If a symmetric scheme is used (cf. Figure 7.4(a)), $(p)_{P,TEXT}$ equals to $p||s$ where $s = g(t_g, k^*_{R_x,m}, p)$. To achieve that $(p')_{P,TEXT}$ is accepted by member A_i, X must either find a p' that fulfills the equation $s = g(t_g, k^*_{R_x,m}, p')$ or it must also modify s to $s' = g(t_g, k^*_{R_x,m}, p')$. However, since g is a cryptographic hash transformation (cf. Definition 4.6), given a specific p it is impossible to find a single p' ($p \neq p'$) such that $g(t_g, k^*_{R_x,m}, p) = g(t_g, k^*_{R_x,m}, p')$. Therefore, X must also modify s by calculating $s' = g(t_g, k^*_{R_x,m}, p')$. To achieve this, X has to determine $k^*_{R_x,m}$ i.e., it has to calculate $k^*_{R_x,m} = h(k_{R_x,m}, TEXT)$. Due to the properties of the cryptographic hash transformation h, X must know $k_{R_x,m}$ and thus $STS_{R_x,P}$. However, this is contradicting the definition of $STS_{R_x,P}$ since the secret part of the STS of relationship R_x (i.e., $k_{R_x,m}$) must only be known by a member of R_x.
If an asymmetric scheme is used (cf. Figure 7.5(a)), $(p)_{P,TEXT}$ equals to $p||s$ where $s = g(t_g, k_{A_j,s}, p)$ where A_j is the sender of the message ($j \in \{1 \ldots n\}, j \neq i$). To achieve that $(p')_{P,TEXT}$ is accepted by member A_i, X must either find a p' that fulfills the equation $s = g(t_g, k_{A_j,s}, p')$ or it must also modify s to $s' = g(t_g, k_{A_j,s}, p')$. However, since g is a cryptographic hash transformation (cf. Definition 4.6), given a specific p it is impossible

to find a single p' ($p \neq p'$) such that $g(t_g, k_{A_j,s}, p) = g(t_g, k_{A_j,s}, p')$. Therefore, X must also modify s by calculating $s' = g(t_g, k_{A_j,s}, p')$. To achieve this, X must know $k_{A_j,s}$ and thus $STS_{R_x,A_j,P}$. However, this is contradicting the definition of $STS_{R_x,A_j,P}$ since the secret part of the STS of A_j (i.e., $k_{A_j,s}$) must only be known by entity A_j. □

If a relationship is declared with security level `Trusted`, a modification by non-members is also detected.

Theorem 8.2. Let R_x denote a relationship with security level `Trusted` where $A_1, A_2, \ldots, A_n$ are the only members of R_x. An external entity X ($X \neq A_x, \forall x \in \{1 \ldots n\}$) is not able to modify any message $(msg)_{T,TEXT}$ in a way that the resulting message $(msg')_{T,TEXT}$ is accepted by any member A_i where $i \in \{1 \ldots n\}$.

Proof. The proof is identical to the one of Theorem 8.1. □

Additionally, an unauthorized fabrication of messages is also detected by the members of a relationship.

Theorem 8.3. Let R_x denote a relationship with security level `Trusted` where $A_1, A_2, \ldots, A_n$ are the only members of R_x. At a specific logical point in time, an external entity X ($X \neq A_x, \forall x \in \{1 \ldots n\}$) is not able to send any message $(msg)_{T,TEXT}$ that is accepted by any member A_i ($i \in \{1 \ldots n\}$) and that has not been sent by any member at the same logical point in time.

Proof. Assume to the contrary that at a specific logical point in time t, an adversary X ($X \neq A_x, \forall x \in \{1 \ldots n\}$) is able to send a message $(p)_{T,TEXT}$ that is accepted by a member A_i ($i \in \{1 \ldots n\}$) and that has not been sent by any member at the same logical point in time t. If a symmetric scheme is used (cf. Figure 7.4(b)), $(p)_{T,TEXT}$ equals to $p||s$ where $s = g(t_g, k^*_{R_x,m}, p||TEXT)||TEXT$. To achieve that $(p)_{T,TEXT}$ is accepted by member A_i, X must find the corresponding $s = g(t_g, k^*_{R_x,m}, p||TEXT)||TEXT$. Since X is not able to calculate s (cf. Theorem 8.2), it has to extract s' from a previously intercepted message $p'||s'$ where s' equals to s. However, since g is a cryptographic hash transformation (cf. Definition 4.6), given a specific p it is impossible to find a single p' ($p \neq p'$) such that $g(t_g, k^*_{R_x,m}, p||TEXT)||TEXT = g(t_g, k^*_{R_x,m}, p'||TEXT)||TEXT$. Therefore, X is only able to resend a message $(p)_{T,TEXT}$ that is equal to a previously intercepted message $p'||s'$ i.e., $p = p'$ and $s = s'$. According to the precondition that $(p)_{T,TEXT}$ has not been sent by any member at the same logical point in time t, X is only able to resend a message $(p')_{T,TEXT'}$ that has been intercepted at a preceding logical

point in time t'. Due to the definition of security level `Trusted`, A_i only accepts message $(p')_{T,TEXT'}$ if $TEXT'$ (which is the TVP at time t') is equal to the TVP value of the current point in time t. However, this is contradicting to the definition of TVP (cf. Definition 4.13), since at any logical point in time the present value of a TVP is always different to all values at preceding logical points in time.
If an asymmetric scheme is used (cf. Figure 7.5(b)), $(p)_{T,TEXT}$ equals to $p||s$ where $s = g(t_g, k_{A_j,s}, p||TEXT)||TEXT$ and A_j is the sender of the message ($j \in \{1 \dots n\}, j \neq i$). To achieve that $(p)_{T,TEXT}$ is accepted by member A_i, X must find the corresponding $s = g(t_g, k_{A_j,s}, p||TEXT)||TEXT$. Since X is not able to calculate s (cf. Theorem 8.2), it has to extract s' from a previously intercepted message $p'||s'$ where s' equals to s. However, since g is a cryptographic hash transformation (cf. Definition 4.9), given a specific p it is impossible to find a single p' ($p \neq p'$) such that $g(t_g, k_{A_j,s}, p||TEXT)||TEXT = g(t_g, k_{A_j,s}, p'||TEXT)||TEXT$. Therefore, X is only able to resend a message $(p)_{T,TEXT}$ that is equal to a previously intercepted message $p'||s'$ i.e., $p = p'$ and $s = s'$. According to the precondition that $(p)_{T,TEXT}$ has not been sent by any member at the same logical point in time t, X is only able to resend a message $(p')_{T,TEXT'}$ that has been intercepted at a preceding logical point in time t'. Due to the definition of security level `Trusted`, A_i only accepts message $(p')_{T,TEXT'}$ if $TEXT'$ (which is the TVP at time t') is equal to the TVP value of the current point in time t. However, this is the contradicting to the definition of TVP (cf. Definition 4.13), since at any logical point in time the present value of a TVP is always different to all values at preceding logical points in time. □

A relationship that is declared as `Confidential` provides the strongest form of security. Since it uses the same countermeasures against modification and fabrication, the following theorems hold.

Theorem 8.4. Let R_x denote a relationship with security level `Confidential` where $A_1, A_2, \dots, A_n$ are the only members of R_x. An external entity X ($X \neq A_x, \forall x \in \{1 \dots n\}$) is not able to modify any message $(msg)_{C,TEXT}$ in a way that the resulting message $(msg')_{C,TEXT}$ is accepted by any member A_i where $i \in \{1 \dots n\}$.

Proof. The proof is identical to the one of Theorem 8.1. □

Theorem 8.5. Let R_x denote a relationship with security level `Confidential` where $A_1, A_2, \dots, A_n$ are the only members of R_x. At a specific logical point in time, an external entity X ($X \neq A_x, \forall x \in \{1 \dots n\}$) is not able to send any message $(msg)_{C,TEXT}$ that is

accepted by any member A_i ($i \in \{1 \dots n\}$) and that has not been sent by any member at the same logical point in time.

Proof. The proof is identical to the one of Theorem 8.3. □

Additionally, a disclosure of the transmitted message by non-members is also avoided.

Theorem 8.6. Let R_x denote a relationship with security level `Confidential` where $A_1, A_2, \dots, A_n$ are the only members of R_x. If the encrypted part and the unencrypted part of a message $(msg)_{C,TEXT}$ are independent, an external entity X ($X \neq A_x, \forall x \in \{1 \dots n\}$) is not able to derive the plain text version of the encrypted part for any message $(msg)_{C,TEXT}$ exchanged within R_x.

Proof. Assume to the contrary that an adversary X ($X \neq A_x, \forall x \in \{1 \dots n\}$) is able to intercept the plain text version of a message $(p)_{C,TEXT}$ exchanged within R_x. Without loss of generality, it is assumed that the unsecured version of p consists of $p = p_p||p_s$ where p_p denotes the part that can be disclosed and p_s the part of the message that has to be kept confidential. If separated schemes for encryption/decryption and MAC generation/verification are used (cf. Figure 7.4(c)), $(p)_{C,TEXT}$ equals to $p_p||c||s$ where $c = e(t_e, k^*_{R_x,c}, p_s)$ and $s = g(t_g, k^*_{R_x,m}, p||TEXT)||TEXT$. To intercept p_s, X has to derive p_s either from p_p, c, or s, since p_p, c, and s are the only items that are transmitted over the network. Since p_s and p_p are independent from each other, p_s cannot be derived from p_p. Furthermore, since g is a cryptographic hash transformation (cf. Definition 4.6), it is impossible to calculate p out of s and thus it is also impossible to derive p_s from s. Therefore, p_s has to be derived out of c. To achieve this, X has to perform the calculation $d(t_d, k^*_{R_x,c}, c)$. To do so, X has to determine $k^*_{R_x,c}$ i.e., it has to calculate $k^*_{R_x,c} = h(k_{R_x,c}, TEXT)$. Due to the properties of the cryptographic hash transformation h, X must know $k_{R_x,c}$ and thus $STS_{R_x,C}$. However, this is contradicting the definition of $STS_{R_x,C}$ since the secret part of the STS of relationship R_x (i.e., $k_{R_x,c}$) must only be known by a member of R_x.
If a hybrid scheme is used (cf. Figure 7.4(d)), $(p)_{C,TEXT}$ equals to $p_p||c||s$ where $c||s = g(t_{hg}, k^*_{R_x,h}, p||TEXT)||TEXT$. To intercept p_s, X has to derive p_s either from p_p, c, or s, since p_p, c, and s are the only items that are transmitted over the network. Since p_s and p_p are independent from each other, p_s cannot be derived from p_p. Therefore, p has to be derived out of $c||s$. To achieve this, X has to perform the calculation $v(t_{hv}, s\{n \dots |TEXT|\}, k^*_{R_x,h}, c||TEXT)$. To do so, X has to determine $k^*_{R_x,h}$ i.e., it has to calculate $k^*_{R_x,h} = h(k_{R_x,h}, TEXT)$. Due to the properties of the cryptographic hash

transformation h, X must know $k_{R_x,h}$ and thus $STS_{R_x,H}$. However, this is contradicting the definition of $STS_{R_x,H}$ since the secret part of the STS of relationship R_x (i.e., $k_{R_x,h}$) must only be known by a member of R_x.
If an asymmetric scheme is used (cf. Figure 7.5(c)), $(p)_{C,TEXT}$ equals to $p_p||c||s$ where $c = e(t_e, k_{A_j,p}, p_s)$ and $s = g(t_g, k_{A_i,s}, p||TEXT)||TEXT$ if the message is sent from A_i to A_j for any $i, j \in \{1 \dots n\}$ and $i \neq j$. To intercept p_s, X has to derive p_s either from p_p, c, or s, since p_p, c, and s are the only items that are transmitted over the network. Since p_s and p_p are independent from each other, p_s cannot be derived from p_p. Furthermore, since g is a cryptographic hash transformation (cf. Definition 4.9), it is impossible to calculate p out of s and thus it is also impossible to derive p_s from s. Therefore, p has to be derived out of c. To achieve this, X has to perform the decryption calculation $d(t_d, k_{A_j,s}, c)$. To do so, X must know $k_{A_j,s}$ and thus $STS_{R_x,A_j,C}$. However, this is contradicting the definition of $STS_{R_x,A_j,C}$ since the secret part of $STS_{R_x,A_j,C}$ (i.e., $k_{A_j,s}$) must only be known by entity A_j. □

Based on proved theorems, the identified security attacks at the routing/naming sublayer are analyzed.

Interception attacks

- (BL.i) Local broadcast
  $(ADR_{src}||(ID_A||N_B||msg_{plain})_{x,TEXT}) \overset{\Uparrow}{\rightarrow} (ADR_{src}||(ID_A||N_B||msg_{plain})_{x,TEXT})$
- (BR.i) Remote broadcast
  $(ADR_{src}||N_A||ADR_A||N_B||(ID_A||N_B||msg_{plain})_{x,TEXT})$
  $\overset{\Uparrow}{\rightarrow} (ADR_{src}||N_A||ADR_A||N_B||(ID_A||N_B||msg_{plain})_{x,TEXT})$
- (BG.i) Global broadcast
  $(ADR_{src}||N_A||ADR_A||(ID_A||N_B||msg_{plain})_{x,TEXT})$
  $\overset{\Uparrow}{\rightarrow} (ADR_{src}||N_A||ADR_A||(ID_A||N_B||msg_{plain})_{x,TEXT})$
- (UD.i) Direct unicast
  $(ADR_{src}||ADR_{dst}||(ID_A||ID_B||msg_{plain})_{x,TEXT})$
  $\overset{\Uparrow}{\rightarrow} (ADR_{src}||ADR_{dst}||(ID_A||ID_B||msg_{plain})_{x,TEXT})$
- (UF.i) Forwarded unicast
  $(ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||(ID_A||ID_B||msg_{plain})_{x,TEXT})$
  $\overset{\Uparrow}{\rightarrow} (ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||(ID_A||ID_B||msg_{plain})_{x,TEXT})$
- (MN.i) Native multicast
  $(ADR_{src}||ADR_{G_x}||(ID_A||ID_{G_x}||msg_{plain})_{x,TEXT})$

$\overset{\Uparrow}{\rightarrow} (ADR_{src}||ADR_{G_x}||(ID_A||ID_{G_x}||msg_{plain})_{x,TEXT})$

- (MU.i) Multicast using multiple unicasts
  $(ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||ID_{G_x}||(ID_A||ID_{G_x}||msg_{plain})_{x,TEXT})$
  $\overset{\Uparrow}{\rightarrow} (ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||ID_{G_x}||(ID_A||ID_{G_x}||msg_{plain})_{x,TEXT})$
- (MG.i) Multicast using global broadcast
  $(ADR_{src}||N_A||ADR_A||ID_{G_x}||(ID_A||ID_{G_x}||msg_{plain})_{x,TEXT})$
  $\overset{\Uparrow}{\rightarrow} (ADR_{src}||N_A||ADR_A||ID_{G_x}||(ID_A||ID_{G_x}||msg_{plain})_{x,TEXT})$

An interception of the address information included by the data link layer (i.e., ADR_{src}, ADR_{dst}, ADR_{G_x}) as well as the address information added by the routing/naming sublayer (i.e., N_A, ADR_A, N_B, ADR_B,ID_{G_x}) cannot be avoided since they are not incorporated into the secured part of the message. This information can be used by an adversary to determine information about the communication behavior of devices (e.g., "which devices communicate with each other?", "when do they communicate?", "where is a device located in the network?"). If this information can be related to the identities of devices, the anonymity may be disclosed. However, as mentioned in Chapter 3, anonymity is only of lesser concern in the HBA domain. An alternative solution would be to encrypt the entire message including the whole address information. However, this solution has several drawbacks. First, the security sublayer has to be integrated into the data link layer and so a transparent use of existing network technologies would not be possible anymore. Second, encrypting the address information is disadvantageous for routing since intermediate routers are not able to read the address information required for routing. Third, providing end-to-end security would not be possible anymore since intermediate routers must be able to decrypt forwarded messages.

According to Theorem 8.6, an adversary is not able to derive the plain text version of the encrypted part of a message if a security level of `Confidential` is chosen. Since the enclosed IDs (i.e., ID_A, ID_B, ID_{G_x}) are not encrypted, an adversary is able to intercept them. However, similar to the disclosing of the address information, identifying the IDs of devices is not critical in the HBA domain as long as anonymity is not of concern. The most important part of a message that has to be kept confidential is the user data portion (i.e., msg_{plain}). Due to the "no hidden-channel" assumption and due to the proof that the plain text version of the encrypted part of a message can only be derived by relationship members (cf. Theorem 8.6), non-disclosure of the user data is guaranteed. Therefore, all interception attacks are either identified as not critical or avoided at all.

Modification attacks

Since a modification cannot be avoided in the assumed adversary model, it has to be detected in order to prevent a misinterpretation of the modified data. As a result, the following modifications are possible:

- (BL.m) Local broadcast
  $(ADR_{src}||(ID_A||N_B||msg_{plain})_{x,TEXT})) \rightarrow (ADR'_{src}||(ID'_A||N'_B||msg'_{plain})_{x,TEXT})$
  - ADR'_{src}: The routing/naming sublayer of the receiving device maps ADR'_{src} to the corresponding ID. If a mapping is not possible or if ADR'_{src} is converted to a wrong ID, the security sublayer will detect the wrong mapping since the ID will be different to the enclosed ID_A.
  - $(ID'_A||N'_B||msg'_{plain})_{x,TEXT}$: A modification will be detected if a security level of `Protected` or above is chosen (cf. Theorem 8.1).
- (BR.m) Remote broadcast
  $(ADR_{src}||N_A||ADR_A||N_B||(ID_A||N'_B||msg_{plain})_{x,TEXT})$
  $\rightarrow (ADR'_{src}||N'_A||ADR'_A||N'_B||(ID'_A||N'_B||msg'_{plain})_{x,TEXT})$
  - ADR'_{src}: A modified ADR'_{src} does not matter at all, since it is not used by the routing/naming sublayer.
  - N'_A,ADR'_A: The final receivers map N'_A and ADR'_A to ID'_A respectively. If a mapping is not possible or the address information is converted to a wrong ID, the security sublayer will detect the malicious mapping since the ID will be different to the enclosed ID_A.
  - N'_B: Router will use N'_B to identify the next hop router. Final receivers will verify N'_B to determine whether they are located in the destination network. In both cases, the security sublayer is able to detect an unauthorized modification by comparing it with the enclosed destination network ID N_B.
  - $(ID'_A||N'_B||msg'_{plain})_{x,TEXT}$: cf. local broadcast.
- (BG.m) Global broadcast
  $(ADR_{src}||N_A||ADR_A||(ID_A||N_B||msg_{plain})_{x,TEXT})$
  $\rightarrow (ADR'_{src}||N'_A||ADR'_A||(ID'_A||N'_B||msg'_{plain})_{x,TEXT})$
  - ADR'_{src}: cf. remote broadcast.
  - N'_A,ADR'_A: cf. remote broadcast.
  - $(ID'_A||N'_B||msg'_{plain})_{x,TEXT}$: cf. local broadcast.
- (UD.m) Direct unicast
  $(ADR_{src}||ADR_{dst}||(ID_A||ID_B||msg_{plain})_{x,TEXT})$
  $\rightarrow (ADR'_{src}||ADR'_{dst}||(ID'_A||ID'_B||msg'_{plain})_{x,TEXT})$

- ADR'_{src}: cf. local broadcast.
- ADR'_{dst}: If ADR'_{dst} is mapped to a none existing data link address, the message will be lost since all members of the relationship will discard it. This is equivalent to an interruption attack (UD.x). If ADR'_{dst} is modified to a different but existing address, the device with that address will wrongly receive the message. However, since the enclosed ID_B is not equal to the own ID, the modification will be detected by the security sublayer.
- $(ID'_A||ID'_B||msg'_{plain})_{x,TEXT}$: cf. local broadcast.

- (UF.m) Forwarded unicast
  $(ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||(ID_A||ID_B||msg_{plain})_{x,TEXT})$
  $\rightarrow (ADR'_{src}||ADR'_{dst}||N'_A||ADR'_A||N'_B||ADR'_B||(ID'_A||ID'_B||msg'_{plain})_{x,TEXT})$
  - ADR'_{src}: cf. remote broadcast.
  - ADR'_{dst}: Replacing ADR_{dst} to a none existing address results in an interruption attack, since the message gets lost (UF.x). If ADR_{dst} is converted to an existing data link address, the message gets re-routed which is considered as an interruption attack if the destination is not reachable anymore (UF.x).
  - N'_A,ADR'_A,N'_B,ADR'_B: Routers will use N'_B,ADR'_B to identify the next hop router. An unauthorized modification will re-route the message which may result in an interruption attack (UF.x). The final receiver maps N'_A,ADR'_A,N'_B, and ADR'_B to ID'_A and ID'_B respectively. If a mapping is not possible or the address information is converted to wrong IDs, the security sublayer will detect the malicious mapping since the IDs will be different to the enclosed ID_A and ID_B.
  - $(ID'_A||ID'_B||msg'_{plain})_{x,TEXT}$: cf. local broadcast.
- (MN.m) Native multicast
  $(ADR_{src}||ADR_{G_x}||(ID_A||ID_{G_x}||msg_{plain})_{x,TEXT}))$
  $\rightarrow (ADR'_{src}||ADR'_{G_x}||(ID'_A||ID'_{G_x}||msg'_{plain})_{x,TEXT})$
  - ADR'_{src}: cf. local broadcast.
  - ADR'_{G_x}: cf. ADR'_{dst} direct unicast.
  - $(ID'_A||ID'_{G_x}||msg'_{plain})_{x,TEXT}$: cf. local broadcast.
- (MU.m) Multicast using multiple unicasts
  $(ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||ID_{G_x}||(ID_A||ID_{G_x}||msg_{plain})_{x,TEXT})$
  $\rightarrow (ADR'_{src}||ADR'_{dst}||N'_A||ADR'_A||N'_B||ADR'_B||ID'_{G_x}||(ID'_A||ID'_{G_x}||msg'_{plain})_{x,TEXT})$
  - ADR'_{src}: cf. forwarded unicast.
  - ADR'_{dst}: cf. forwarded unicast.

  - N'_A,ADR'_A,N'_B,ADR'_B: cf. forwarded unicast.
  - ID'_{G_x}: The final receivers will detect an unauthorized modification of ID'_{G_x} since the ID will be different to the enclosed ID_{G_x}.
  - $(ID'_A||ID'_{G_x}||msg'_{plain})_{x,TEXT}$: cf. local broadcast.
- (MG.m) Multicast using global broadcast
  $(ADR_{src}||N_A||ADR_A||ID_{G_x}||(ID_A||ID_{G_x}||msg_{plain})_{x,TEXT})$
  $\rightarrow (ADR'_{src}||N'_A||ADR'_A||ID'_{G_x}||(ID'_A||ID'_{G_x}||msg'_{plain})_{x,TEXT})$
  - ADR'_{src}: cf. global broadcast.
  - N'_A,ADR'_A: cf. global broadcast.
  - ID'_{G_x}: The final receivers will detect an unauthorized modification of ID'_{G_x} since the ID will be different to the enclosed ID_{G_x}.
  - $(ID'_A||ID'_{G_x}||msg'_{plain})_{x,TEXT}$: cf. local broadcast.

Converting a message of one type into a message of another type is also regarded as a modification attack. Consider, for example, a remote broadcast message. By suppressing the address of the destination network, the remote broadcast is converted to a global broadcast message. However, due to the included source and destination IDs, modification attacks that convert message types can be detected (if a security level of `Protected` or above is chosen) or result in an interruption attack[2]:

- Unicast $\rightarrow$ Multicast: detectable since ID_B cannot be converted to ID_{G_x}.
- Multicast $\rightarrow$ Unicast: detectable since ID_{G_x} cannot be converted to ID_B.
- Unicast $\rightarrow$ Broadcast: detectable since ID_B cannot be converted to N_B.
- Broadcast $\rightarrow$ Unicast: detectable since N_B cannot be converted to ID_B.
- Multicast $\rightarrow$ Broadcast: detectable since ID_{G_x} cannot be converted to N_B.
- Broadcast $\rightarrow$ Multicast: detectable since N_B cannot be converted to ID_{G_x}.
- Direct unicast $\rightarrow$ forwarded unicast: The address information field N_A,ADR_A,N_B, and ADR_B have to be added. If a wrong address information is added, the security sublayer will detect the attack since the enclosed ID_A and ID_B will not correspond. If N_A,ADR_A,N_B, and ADR_B map to the ADR_{src} and ADR_{dst}, this attack does not have any consequences since there is no semantic difference between direct and forwarded unicasts.
- Forwarded unicast $\rightarrow$ direct unicast: If the modification attack is performed on the network segment where the destination device is located, it does not have any consequences. Otherwise, the message gets lost (UF.x).
- {Local|Remote|Global} broadcast $\rightarrow$ {Local|Remote|Global} broadcast: detectable

[2]It is assumed that the address spaces of device IDs, group IDs, and network addresses are different.

since N_B is included in the secured part of the message and therefore it cannot be modified.

- Multicast → Multicast: Converting a multicast message into a multicast that uses a different data link layer mechanisms does not have any consequences since there is no semantic difference at higher layers.

Fabrication attacks

An adversary is able to fabricate any message:

- (BL.f) Local broadcast
  $() \rightarrow (ADR'_{src}||(ID'_A||N'_B||msg'_{plain})_{x,TEXT})$
- (BR.f) Remote broadcast
  $() \rightarrow (ADR'_{src}||N'_A||ADR'_A||N'_B||(ID'_A||N'_B||msg'_{plain})_{x,TEXT})$
- (BG.f) Global broadcast
  $() \rightarrow (ADR'_{src}||N'_A||ADR'_A||(ID'_A||N'_B||msg'_{plain})_{x,TEXT})$
- (UD.f) Direct unicast
  $() \rightarrow (ADR'_{src}||ADR'_{dst}||(ID'_A||ID'_B||msg'_{plain})_{x,TEXT})$
- (UF.f) Forwarded unicast
  $() \rightarrow (ADR'_{src}||ADR'_{dst}||N'_A||ADR'_A||N'_B||ADR'_B||(ID'_A||ID'_B||msg'_{plain})_{x,TEXT})$
- (MN.f) Native multicast
  $() \rightarrow (ADR'_{src}||ADR'_{G_x}||(ID'_A||ID'_{G_x}||msg'_{plain})_{x,TEXT})$
- (MU.f) Multicast using multiple unicasts
  $() \rightarrow$
  $(ADR'_{src}||ADR'_{dst}||N'_A||ADR'_A||N'_B||ADR'_B||ID'_{G_x}||(ID'_A||ID'_{G_x}||msg'_{plain})_{x,TEXT})$
- (MG.f) Multicast using global broadcast
  $() \rightarrow (ADR'_{src}||N'_A||ADR'_A||ID'_{G_x}||(ID'_A||ID'_{G_x}||msg'_{plain})_{x,TEXT})$

If a security level of `Trusted` or above is chosen, an adversary is not able generate and send a valid, secured part of a message that has not been sent at the same logical point in time (cf. Theorem 8.3). This includes the generation of new messages as well as replaying of messages that have been intercepted at a preceding logical point in time. It is important to note that address information added by the data link layer and by the reliability/routing sublayer is of no relevance since an invalid secured part is sufficient to detected a fabricated message (cf. Theorem 8.3).

The most critical point is the conformance of the TVP and the definition of logical time since replaying of messages at the *same* logical point in time cannot be detected. The so called *freshness window* i.e., the absolute time frame where replayed messages

cannot be detected depends on the used TVP type. However, guaranteeing the time variant property of parameter $TEXT$ as well as maintaining the logical clock is the task of the reliability/ordering sublayer.

Interruption attacks

According to the assumed adversary model, communication can be interrupted by discarding messages:

- (BL.x) Local broadcast
  $(ADR_{src}||(ID_A||N_B||msg_{plain})_{x,TEXT}) \rightarrow ()$
- (BR.x) Remote broadcast
  $(ADR_{src}||N_A||ADR_A||N_B||(ID_A||N_B||msg_{plain})_{x,TEXT}) \rightarrow ()$
- (BG.x) Global broadcast
  $(ADR_{src}||N_A||ADR_A||(ID_A||N_B||msg_{plain})_{x,TEXT}) \rightarrow ()$
- (UD.x) Direct unicast
  $(ADR_{src}||ADR_{dst}||(ID_A||ID_B||msg_{plain})_{x,TEXT}) \rightarrow ()$
- (UF.x) Forwarded unicast
  $(ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||(ID_A||ID_B||msg_{plain})_{x,TEXT}) \rightarrow ()$
- (MN.x) Native multicast
  $(ADR_{src}||ADR_{G_x}||(ID_A||ID_{G_x}||msg_{plain})_{x,TEXT}) \rightarrow ()$
- (MU.x) Multicast using multiple unicasts
  $(ADR_{src}||ADR_{dst}||N_A||ADR_A||N_B||ADR_B||ID_{G_x}||(ID_A||ID_{G_x}||msg_{plain})_{x,TEXT})$
  $\rightarrow ()$
- (MG.x) Multicast using global broadcast
  $(ADR_{src}||N_A||ADR_A||ID_{G_x}||(ID_A||ID_{G_x}||msg_{plain})_{x,TEXT}) \rightarrow ()$

Since the security sublayer does not support mechanisms that prevent or detect these attacks, the task is deferred to the next higher layers.

8.1.4 Reliability/ordering sublayer

According to the assumed adversary model, the security sublayer is able to prevent and/or detect most of the basic security attacks originated at the data link layer. From a security point of view, the reliability/ordering sublayer is responsible for two different security tasks. First, it is responsible to prevent or at least detect interruption attacks. Second, in order to provide an effective protection against fabrication attacks, the reliability/ordering sublayer has to provide and maintain the required TVPs.

Interruption attacks

Counteracting interruption attacks is by no means a trivial task. To prevent them, it must be guaranteed that an adversary is not able to discard messages during transmission. However, since the used adversary model assumes that an adversary is able to drop any network message, a prevention is not possible. Therefore, mechanisms that provide at least a detection of lost messages are necessary.

Detecting message losses is closely related to reliability. If it can be guaranteed that either all relationship members receive a message or none of them, interruption attacks are detected in a native way.

Theorem 8.7. Let R_x denote a relationship where $A_1, A_2, \ldots, A_n$ are the only members of R_x and where the communication is said to be reliable (according to the Definition 6.2). An external entity X ($X \neq A_x, \forall x \in \{1 \ldots n\}$) is not able to drop any message $(msg)_{P,TEXT}$ that is sent by a non-faulty relationship member without being detected by all non-faulty relationship members $A_1, A_2, \ldots, A_n$.

Proof. Assume to the contrary that an external entity X ($X \neq A_x, \forall x \in \{1 \ldots n\}$) is able to drop a message $(msg)_{P,TEXT}$ that is sent by a non-faulty relationship member without being detected by at least one non-faulty relationship member. This is contradicting to the liveness property of reliable communication since it demands that every message sent by a non-faulty relationship member is received at least once by all non-faulty relationship members. □

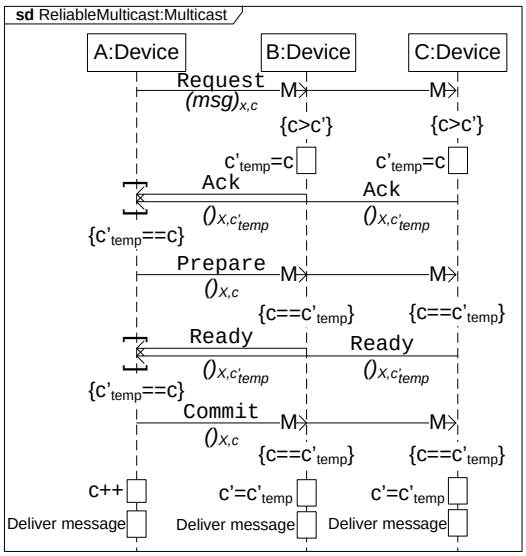
This theorem demands that only non-faulty relationship members must be able to detect interruption attacks. This requirement corresponds to the used adversary model since devices are considered as fail-silent where recovery is done manually. As a result, proving that a communication service guarantees liveness implies that interruption attacks are detected, too.

To be able to analyze the ability to detect interruption attacks, a more precise definition of the case "a message loss is detected" is required. In this context, a message is regarded as a request that contains user data – losses of acknowledgments or control messages that are part of the reliability protocol are not considered. Furthermore, a lost message at the receiver site is said to have been detected if the loss is identified before data from subsequent messages is delivered to the next higher layer. At the sender site, a lost message is said to have been detected once the next higher layer is notified. Therefore, for both sites, a lost message detection is only useful if it is performed before the service to the next higher layer is invoked.

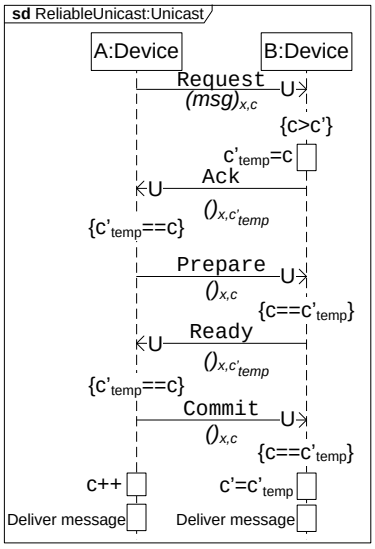
The reliability of communication services can be controlled by choosing one of three different communication service types. The first service type is called best-effort (cf. Figure 7.7). As it can easily be seen, best-effort communication does not provide liveness since a message loss is neither detected by the sender nor by the receiver. Therefore, interruption attacks are not detected, too. The second communication service type is based on acknowledgments (cf. Figure 7.8). Here, each receiver responds to a `Request` message with an individual `Ack` message. From the sender's point of view, it is possible to detect a lost message. If a `Request` message is lost on the way to a receiver, the sender will detect the loss since the acknowledgment will be missing. On the other hand, whenever an acknowledgment gets lost, the sender will also detect the loss after a specific timeout. Therefore, if all acknowledgments have been collected, it can be sure that no interruption attack occurred. However, detecting a message loss at the receiver site is not always possible. Consider, for example, a device A sends a multicast message to device B and C. Further, assume that an adversary is able to drop the `Request` message before C gets it. Since B successfully receives it, B responds with an acknowledgment and delivers the data to the next higher layer. While A is able to recognize the message loss due to the missing acknowledgment, C is not able to detect it. Therefore, acknowledged communication services are only advisable if a detection of interruption attacks at the sender site is sufficient.

The third communication service variant is based on a three-phase commit protocol. For better understanding, Figure 7.9 is shown here again (cf. Figure 8.1). In [113], it is shown that the presented protocol is non-blocking at operational sites. In this context, non-blocking means that the operational site does not have to wait until a failed site recovers. However, as shown in [112], there is no protocol that uses independent recovery being resilient to two-site failures. Independent recovery refers to a protocol where a failed site is able to recover without contacting other relationship members. Since message losses are considered as two-site failures, a recovery protocol is necessary. Based on a variant of the recovery protocol presented in [117], the three-phase commit protocol can be used to detect interruption attacks:

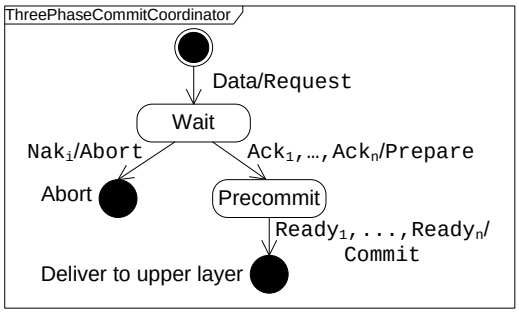
- `Request`: At the sender site, a loss of a `Request` message is detected since the following `Ack` from the corresponding receiver is missing. Therefore, the sender will abort the transaction by sending an `Abort` message to all receivers. The receivers will also recognize the loss since they will receive the `Abort` message from the sender. As a result, the transaction is terminated, too.
- `Ack`: If an `Ack` is dropped by an adversary, the sender will detect the loss and an



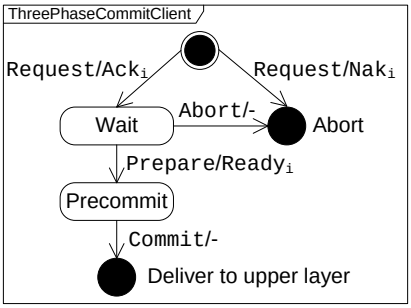
(a) Reliable multicast



(b) Reliable unicast



(c) Three phase commit: source site



(d) Three phase commit: sink site

Figure 8.1: Three-phase commit protocol

`Abort` message is sent. Due to the `Abort` message, the receivers will detect the loss and abort the transaction, too.

- `Nak`: If a receiver is not able to process an incoming request, it responds with a `Nak` message. If this message is dropped by an adversary, the sender of the `Request` message will detect the loss due to the missing response of that receiver. As a result, an `Abort` message is sent. Due to the retrieval of the `Abort` message, the receivers cancel the transaction, too.
- `Abort`: At the sender site, a detection of a lost `Abort` message is not necessary, since the sender already knows that it has to abort the transaction. The receivers that successfully get the `Abort` message do not care about the loss, too, since they will abort the transaction anyway. At the receiver site where the `Abort` message is lost, three different cases are distinguished.
  - The receiver has missed the `Request` message, too. Since it has not started

the transaction yet, a detection is not necessary.

- The receiver got the `Request` message and sends an `Ack` back. Since, it is waiting for a `Prepare` message without success, it does not know whether it missed the `Prepare` message or the `Abort` message. If the `Prepare` message was lost, the other receivers will decide to commit – if the `Abort` message was lost, the remaining devices will abort the transaction. To solve this situation, the affected receiver has to resend the `Ack` message. If it still gets no answer (after several retransmissions), it has to assume that the communication to the sender has been interrupted. To come to a final decision, it contacts the other relationship members. If one receiver has committed the transaction or the majority of them is waiting for the final `Commit` message, the affected receiver commits, too. Otherwise, the transaction is aborted.
- The receiver retrieved the `Request` message but it sends a `Nak` message. Since the receiver has already decided that is not able to commit the transaction, it can abort it anyway. A detection of the missing `Abort` message is not necessary.

- `Prepare`: After all acknowledgments have been received, the sender transmits a `Prepare` message. At the sender site, a lost `Prepare` is detected since the following `Ready` message will be missing, too. The receivers that successfully retrieve the `Prepare` message need not care about the lost `Prepare` message, since they will wait for the final `Commit` message. The receiver that misses the `Prepare` will reside in a state where it waits for an incoming `Prepare` or `Abort` message. This case is identical to the one where the receiver has missed an `Abort` message after it sent an acknowledgment. Again, the affected receiver has to resend the `Ack` message. If the sender does not respond, it has to contact the remaining relationship members to come to a final decision.
- `Ready`: After having sent a `Prepare` message, all receivers respond with a `Ready` message. If a `Ready` message is missing, the sender will detect the loss. However, since all receivers have already acknowledged the successful retrieval of the initial `Request` message, the transaction can be finished by sending a `Commit` message. The other receivers do not care about a lost `Ready` since they are waiting for the final `Commit` message.
- `Commit`: At the sender site, a detection is not necessary since it will commit the transaction anyway. At the receiver sites, the situation is identical to the one where a receiver is waiting for an `Abort` or `Prepare` message. It retransmits the `Ready`

message to the sender. If there is no answer, it has to contact the remaining receivers to come to a decision.

In contrast to acknowledged communication, the reliable communication service types are able to detect interrupt attacks at both sites. The only attacks that cannot be detected by a receiver are permanent interruption attacks i.e., attacks where the communication between the sender and the affected receiver is completely terminated (e.g., by cutting the communication line). Since the receiver does not get any message at all (neither a `Request` message nor an `Abort` message), it is not able to distinguish between "no communication" and "interrupted communication". One possible approach is to require that the sender has to transmit `Request` messages at regular intervals – even if there is no new user data at all (heartbeat message).

Time Variant Parameter

Choosing an appropriate TVP is of utmost importance for detecting fabrication attacks. The used type of a TVP and the used definition of logical time specify the size of the freshness window i.e., the absolute timing frame between two logical points in time where replayed messages cannot be detected.

Figure 7.7 shows how a counter in combination with best-effort communication services is used. The counter is incremented by the sender after each successful message transmission. Therefore, sending a message can be seen as the event that triggers the logical clock. The counter represents the logical clock where the present value is used as the current logical timestamp. As can be seen, the used counter value is strict monotonically increasing and so, the time variant property is satisfied since at any logical point in time, the present value of a TVP is always different to all values before. The freshness window is defined as the interval between the point in time where the sender sends the message and the time where the receiver updates its local counter value. The size of this window depends on the message's transmission time and the time it takes to process the message. During this timing frame, an adversary is able to replay messages without a detection at the receiver site. However, receiving replays during this small interval can be tolerated since the content of the replayed message is exactly the same. As a result, the receiver can simply take the first message and the other ones are discarded.

Figure 7.8 shows the acknowledged variant. Here, the size of the freshness window is the time between sending the initial `Request` message and the final delivery at the receiver. Since a retransmission is possible, the interval may be larger. This is also true for reliable communication (cf. Figure 7.9) where the window size is the time between

the start and the successful end of the transaction. In both cases, receiving multiple copies of the same message can also be tolerated since the content is exactly the same. This also applies to acknowledgments and control messages of the commit protocol since they are idempotent. Note that the counter is only incremented after the successful transmission to all receivers. This means that the initial request message as well as the following acknowledgments and control messages use the same TVP. However, since they have a different content anyway, each message is unique even if the same TVP is used. Retransmissions also take the TVP that was used for the initial `Request` message. This provides the opportunity to distinguish between retransmissions and new messages. Since a receiver only updates the local counter value after a transmission was successful, retransmissions pass the initial counter verification. If the transmission was successful, the local counter is updated and subsequent retransmissions or malicious replays are detected and discarded.

The used TVP type also specifies the guaranteed ordering. If SSF ordering is required, monotonically increasing counters (one for each sender) are sufficient. If causal ordering is required, logical vector timestamps have to be used – for total ordering, timestamps based on synchronized clocks are necessary. Possible implementations and their corresponding proofs of correctness can be found in [23].

From a security point of view, logical vector timestamps and timestamps based on synchronized clocks also fulfill the time variant property. A logical vector clock consists of multiple counters – one for each device. As a result, a vector clock of a device also contains a counter for the device itself. Since a device always increments its counter after sending or receiving a message, the time variant property is also satisfied. The freshness window is the same as it is for monotonically increasing counters.

If synchronized clocks are used, the current timestamp is used as TVP. At the receiver sites, the enclosed timestamp is compared with the present value of the local clock. If it is within the specified freshness window, the message is accepted. The size of the freshness windows depends on the maximum transmission time of the message, the processing time at the receiver, as well as on the granularity (i.e., time between two ticks) and precision (i.e., the deviation to the clock of other devices) of the used clocks. The precision in turn depends on the used synchronization algorithm [114, 118]. As long as it can be guaranteed that only one message is sent between two clock ticks, timestamps of synchronized clocks fulfill the time variant property. This limitation applies to the entire relationship. This means that only one member is allowed to send a message between two clock ticks. To weaken this requirement, the current timestamp can be extended with the sender's ID. The advantage of the included ID is that the resulting TVPs are different for each sender

			Local				Remote				Global			
			BL.i	BL.m	BL.f	BL.x	BR.i	BR.m	BR.f	BR.x	BG.i	BG.m	BG.f	BG.x
Raw			-	-	-	-	-	-	-	-	-	-	-	-
Protected			-	x	-	-	-	x	-	-	-	x	-	-
Trusted	Best-effort	SSF	-	x	x	-	-	x	x	-	-	x	x	-
		C	-	x	x	-	-	x	x	-	-	x	x	-
		T	-	x	x	-	-	x	x	-	-	x	x	-
Confidential	Best-effort	SSF	x[3]	x	x	-	x[3]	x	x	-	x[3]	x	x	-
		C	x[3]	x	x	-	x[3]	x	x	-	x[3]	x	x	-
		T	x[3]	x	x	-	x[3]	x	x	-	x[3]	x	x	-

(a) Broadcast

			Direct				Forwarded			
			UD.i	UD.m	UD.f	UD.x	UF.i	UF.m	UF.f	UF.x
Raw			-	-	-	-	-	-	-	-
Protected			-	x	-	-	-	x	-	-
Trusted	Best-effort	SSF	-	x	x	-	-	x	x	-
		C	-	x	x	-	-	x	x	-
		T	-	x	x	-	-	x	x	-
	Acknowledged	SSF	-	x	x	~[4]	-	x	x	~[4]
		C	-	x	x	~[4]	-	x	x	~[4]
		T	-	x	x	~[4]	-	x	x	~[4]
	Reliable	SSF	-	x	x	x[5]	-	x	x	x[5]
		C	-	x	x	x[5]	-	x	x	x[5]
		T	-	x	x	x[5]	-	x	x	x[5]
Confidential	Best-effort	SSF	x[3]	x	x	-	x[3]	x	x	-
		C	x[3]	x	x	-	x[3]	x	x	-
		T	x[3]	x	x	-	x[3]	x	x	-
	Acknowledged	SSF	x[3]	x	x	~[4]	x[3]	x	x	~[4]
		C	x[3]	x	x	~[4]	x[3]	x	x	~[4]
		T	x[3]	x	x	~[4]	x[3]	x	x	~[4]
	Reliable	SSF	x[3]	x	x	x[5]	x[3]	x	x	x[5]
		C	x[3]	x	x	x[5]	x[3]	x	x	x[5]
		T	x[3]	x	x	x[5]	x[3]	x	x	x[5]

(b) Unicast

Figure 8.2: Evaluation of high-level communication services

even if they happen at the same point in time. In this case, it is sufficient to guarantee that a single device transmits at most one message between two clock ticks.

8.1.5 High-level communication interface

Figure 7.10 summarizes the resulting high-level communication interface as well as the provided services. Figure 8.2 lists the identified security attacks that are derived from the basic security attacks at the data link layer. According to the chosen security level, communication service type, and TVP, it is shown which of these security attacks are

[3] Only the user data part of the message cannot be intercepted.

[4] A detection is only possible at the sender site.

[5] A detection of a permanent interruption at the receiver site requires special measures (e.g., periodically sending of heartbeat messages).

			Native				Multiple unicasts				Global broadcast			
			MN.i	MN.m	MN.f	MN.x	MU.i	MU.m	MU.f	MU.x	MG.i	MG.m	MG.f	MG.x
Raw			-	-	-	-	-	-	-	-	-	-	-	-
Protected			-	x	-	-	-	x	-	-	-	x	-	-
Trusted	Best-effort	SSF	-	x	x	-	-	x	x	-	-	x	x	-
		C	-	x	x	-	-	x	x	-	-	x	x	-
		T	-	x	x	-	-	x	x	-	-	x	x	-
	Acknowledged	SSF	-	x	x	~[4]	-	x	x	~[4]	-	x	x	~[4]
		C	-	x	x	~[4]	-	x	x	~[4]	-	x	x	~[4]
		T	-	x	x	~[4]	-	x	x	~[4]	-	x	x	~[4]
	Reliable	SSF	-	x	x	x[5]	-	x	x	x[5]	-	x	x	x[5]
		C	-	x	x	x[5]	-	x	x	x[5]	-	x	x	x[5]
		T	-	x	x	x[5]	-	x	x	x[5]	-	x	x	x[5]
Confidential	Best-effort	SSF	x[3]	x	x	-	x[3]	x	x	-	x[3]	x	x	-
		C	x[3]	x	x	-	x[3]	x	x	-	x[3]	x	x	-
		T	x[3]	x	x	-	x[3]	x	x	-	x[3]	x	x	-
	Acknowledged	SSF	x[3]	x	x	~[4]	x[3]	x	x	~[4]	x[3]	x	x	~[4]
		C	x[3]	x	x	~[4]	x[3]	x	x	~[4]	x[3]	x	x	~[4]
		T	x[3]	x	x	~[4]	x[3]	x	x	~[4]	x[3]	x	x	~[4]
	Reliable	SSF	x[3]	x	x	x[5]	x[3]	x	x	x[5]	x[3]	x	x	x[5]
		C	x[3]	x	x	x[5]	x[3]	x	x	x[5]	x[3]	x	x	x[5]
		T	x[3]	x	x	x[5]	x[3]	x	x	x[5]	x[3]	x	x	x[5]

(c) Multicast

Figure 8.2: Evaluation of high-level communication services

prevented and/or detected by the countermeasures of the security and reliability/ordering sublayer. The used abbreviations of the security attacks are identical to the ones used in Section 8.1.2 and 8.1.3.

Since the sending and receiving services of the high-level communication interface are directly mapped to the sending and receiving services of the reliability/ordering sublayer, the results of the evaluation also apply to the sending and receiving services of the different communication relationships. The remaining services are used to join and leave relationships. Each of these services implements a protocol where multiple messages are exchanged. According to the chosen security level, communication service type, and TVP, each message is individually protected against security attacks. To verify that a prevention or at least a detection of all relevant security attacks is possible, it has to be verified whether the parameters are chosen correctly.

For the rest of this section, the remaining services of the high-level communication interface (i.e., joining and leaving relationships) as well as internal services of the management entity (i.e., services to revoke STSs) are analyzed. To achieve this, each message of the corresponding protocols is listed and the resulting vulnerabilities are identified (cf. Figure 8.2). Then, it is shown how these vulnerabilities can be detected or why they are tolerable.

network-join

If the static binding protocol is used, the following messages are exchanged during a network join (cf. Figure 7.15):

- `network-search.req`: Since best-effort local broadcast with security level `Trusted` is used, the message is vulnerable to interception and interruption. Due to the fact that the message only contains public values, confidentiality is not necessary. A lost message will be discovered by the sender since the following `network-search.res` will be missing, too. A detection at the receiver site is not necessary.
- `network-search.res`: This message is also vulnerable to interception and interruption. Since the entire message consists of public values, guaranteeing a non-disclosure is not required. The receiver is able to detect message loss since it is awaiting a response to the previously sent `network-search.req` message. A detection at the coordinator site is not necessary.
- `network-join`: This message is sent using best-effort unicast with security level `Trusted`. Avoiding an interception is not required since it does not contain any user data. Message loss is detected by the sender since the following `network-subscribed` message will be missing, too. A detection at the receiver site is not necessary.
- `network-subscribed`: Since the message is transmitted using confidential, acknowledged unicast, the message is only vulnerable to interruption attacks at the receiver site. However, since the receiver is awaiting a response to the previous `network-join` message, a missing `network-subscribed` message is also discovered by the receiver.

During a relationship join, it may also be necessary that a coordinator has to request a CDR from another coordinator (cf. Figure 7.18 and 7.19). To achieve this, the following messages are exchanged between the corresponding coordinators:

- `cdr-request`: To send this message, best-effort unicast with security level `Trusted` is used. Avoiding an interception is not necessary since it does not contain secret data. At the sender site, a message loss is detected since the following `cdr-transfer` message will be missing, too. A detection at the receiver site is not required.
- `cdr-transfer`: Since the message includes secret data, acknowledged unicast with security level `Confidential` is used. The receiver is able to detect the message since it is waiting for a response to the previous `cdr-request` message.

If a network segment contains a coordinator cluster, it may be necessary to synchronize CDRs. To synchronize a CDR, a `cdr-replication` message is sent to the cluster group. To avoid inconsistent data views, the message is transmitted using confidential, reliable multicast. The only possible attack scenario that remains is a permanent interruption attack. In such a case, a trusted third-party has to be informed about the attack.

For an analysis of the dynamic binding protocol, it has to be distinguished between the situation whether a coordinator is available or no coordinator is present. A network join where a coordinator is available consists of the following messages (cf. Figure 7.22):

- `network-hello.req`: Since a best-effort local broadcast with security level `Trusted` is used, the message is vulnerable to interception and interruption. An interception is not considered as critical since it does not contain secret values. If the message is dropped, the sender will detect the message loss since the following `network-hello.res` message is missing, too. However, the sender is not able to distinguish between a message loss and a situation where no coordinator is present. Therefore, the sender proceeds by sending a `network-cord-beg` message to assume the role of the coordinator. After having sent the `network-cord-beg` message, the current coordinator recognizes the situation and so it sends a `network-hello.res` message. This message informs the joining device that a coordinator is already present. If the second `network-hello.res` message is also lost, the joining device proceeds by sending a `network-cord-ready` message. To inform the joining device about the message loss, the original coordinator sends another `network-hello.res` message. If this message is also lost, a permanent interruption attack has to be assumed and a trusted third-party has to be informed about the attack. However, to resume operation, the situation is considered as a coordinator crash and so the role of the coordinator is taken over by the joining device.
- `network-hello.res`: Again, a best-effort local broadcast with security level `Trusted` is used. Since the message only contains public values, an interception needs not be avoided. If the `network-hello.res` message is lost, the coordinator detects the loss, since the joining device will send a `network-cord-beg`. To inform the joining device about the loss, a second `network-hello.res` is sent. If this message is also lost, the joining device sends a `network-cord-ready` message. To inform the joining device, it sends another `network-hello.res` message. If this message is also lost, three subsequent `network-hello.res` messages have been discarded and so a permanent interruption attack is assumed.

This situation is identical to one mentioned above – after having informed a trusted third-party, the role of the coordinator is handed over to the joining device.

- `network-join` and `network-subscribed`: These messages are identical to the ones that are used during the static binding protocol.

If there is no coordinator available, the following messages are exchanged (cf. Figure 7.21):

- `network-hello.req`: Since there are no secret values, an interception is not considered as critical. An interruption, however, has to be detected. If there is no coordinator available, two situations exist. If the device (denoted as A) is the only one that may become coordinator, a discarded message does not matter at all. If there is another device (denoted as B) available that wants to assume the role of the coordinator too, the two devices compete for the coordinator role at the same time. If the `network-hello.req` message is lost, B does not recognize A. Depending on the state of B, three different cases are distinguished:
  - B sends a `network-cord-beg` message. If A has a higher ID, A cancels the coordinator establishment process since B has won the competition. If A has a lower ID, it informs B by sending a `network-hello.req` message. If the second `network-hello.req` is also lost, B will send a `network-cord-ready` message. In this case, A cancels the join process since B is already coordinator.
  - B sends a `network-cord-ready` message. A cancels the coordinator establishment process since B is already coordinator.
  - B sends a `network-cord-established` message. Again, A cancels the coordinator establishment process since B is already coordinator.
- `network-cord-beg`: Confidentiality is not demanded since the message does not contain secret values. If the device (denoted as A) is the only one at the network, a message loss does not have any consequences. Any device (denoted as B) that competes for the role of the coordinator at the same time detects a message loss since a `network-cord-ready` follows. If B has already sent a `network-hello.req` or a `network-cord-beg` message, B cancels the establishment process since A is already coordinator. If B has already sent a `network-cord-ready`, both devices assume the role of the coordinator. Such a conflict state is only possible, if all subsequent `network-hello.req` and `network-cord-beg` messages have been dropped. This situation is equivalent to a permanent interruption attack and thus a trusted third-party is informed. To continue operation,

the device with the lower ID remains coordinator.

- `network-cord-ready`: Again, confidentiality is not of concern. A lost `network-cord-ready` message, however, is detected by a router since a `network-cord-established` follows before it receives the `router-avail.res` message. A competing device discovers the message loss, since it receives a `network-cord-established` before the competing device is able to send a `network-cord-beg` message.
- `router-avail.req`: Since the message does not contain secret data, encryption is not required. At the router sites, a detection of a message loss is possible since the following `router-avail.res` will be missing, too. A detection at the device site is not necessary since the router retransmits the request if the response is missing.
- `router-avail.res`: To transmit the message, best-effort local broadcast with security level `Confidential` is used. At the router site, a loss is detected since the router is awaiting an answer to the preceding `router-avail.req`. A detection at the sender site is not required.
- `network-cord-established`: Confidentiality is not demanded since the message only contains public values. If there is a competing device that wants to be coordinator at the same time, a lost `network-cord-established` will be detect since the new coordinator will send a `network-hello.res` message if the competing device sends any further messages. A detection at the sender site is not necessary.

`network-revocation`

If a network coordinator starts a revocation, the following messages are exchanged (cf. Figure 7.26(a)):

- `network-revoc-start`: This message is sent using best-effort local broadcast with security level `Trusted`. Since it contains no secret values, an interception is not critical. Loosing this message may result in a situation where network members send messages that are secured by the old NSTS that is currently under revocation. As a result, members that have received the `network-revoc-start` message will drop such a message since sending messages with the old NSTS is forbidden. However, members that have also missed the `network-revoc-start` message will accept messages secured by the old NSTS. Since a reliable broadcast communication is not possible in the proposed solution, an inconsistent data view cannot

be avoided at all.

- `network-subscribed`: Since the network coordinator has a list of all connected devices, acknowledged unicasts with security level `Confidential` can be used to distribute the new NCDR. At the receiver site, a detection of a dropped `network-subscribed` message is not necessary since the coordinator immediately retransmits the message.
- `network-revoc-end`: Again, best-effort local broadcast with security level `Trusted` is used. An interception of this message is not critical since it does not contain any user data. At the receiver sites, detecting a lost `network-revoc-end` message is not necessary. If it is lost and another device already sends a message using the new NSTS, the device that lost the `network-revoc-end` message may try to verify the message with the new NSTS. If the verification is successful, it can assume that the previous `network-revoc-end` message was missed. A detection of a message loss at the coordinator site is not necessary.

`network-leave`

To leave a network, the leaving device sends a `network-leave` message to the coordinator using acknowledged unicast (cf. Figure 7.27(a)). Confidentiality is not necessary since it does not contain any user data. At the coordinator site, a detection of a lost `network-leave` message is not necessary since the leaving device retransmits the message if there is no acknowledgment from the coordinator. The remaining steps are identical to the one of a network revocation.

`session-start`

If static binding is used, the following messages are exchanged during a session establishment (cf. Figure 7.16):

- `session-start`: This message is sent to the coordinator using best-effort unicast with security level `Trusted`. Since it only contains the ID of the second device, an interception is not considered as critical. At the sender site, a message loss is detected since the `session-established` message will be missing, too. At the coordinator site, a detection is not necessary since the initiating device will resend the message.
- `session-init`: To transmit a `session-init` message, an acknowledged unicast with security level `Confidential` is used. A detection at the receiver site is

not necessary since the coordinator will initiate a retransmission.

- `session-established`: A detection at the receiver site is not required since the coordinator will retransmit it if an acknowledgment is missing.

In case of dynamic binding, the following messages have to be transmitted (cf. Figure 7.23):

- `session-hello.req`: To find the second device, this message is sent using best-effort global broadcast with security level `Trusted`. Confidentiality is not demanded since only public values are included. At the sender site, message loss is detected since an answer will be missing, too. A detection at the receiver site is not required.
- `session-hello.res`: The second device responds with a `session-hello.res` message using best-effort unicast with security level `Trusted`. Guaranteeing a non-disclosure of the user data is not required since the enclosed certificate is public anyway. At the receiver site, a detection of a lost message is possible, since the session initiator is awaiting the response to the preceding `session-hello.req` message. At the sender site, detecting a message loss is not necessary.
- `session-start.req`: Again, confidentiality is not required. Discovering a message loss is done at the sender site since the following `session-start.res` will be missing, too. At the receiver site, a detection is not necessary.
- `session-start.res`: To distribute the SCDR, a confidential, acknowledged unicast message is used. A detection at the receiver site is not necessary since the sender will retransmit a lost message.

session-revocation

To revoke a session, it is simply terminated. Afterwards, a new one has to be established.

session-end

To terminate a session, the initiating device simply sends a `session-close` message using acknowledged unicast with security level `Trusted`. At the receiver site, a detection of a lost message is not necessary since the sender will retransmit the message if the acknowledgement is missing.

group-join

In case of static protocol binding, the following messages are transmitted during a group join (cf. Figure 7.17):

- `group-join`: Since a best-effort unicast message with security level `Trusted` is used, this message is vulnerable to interception and interruption. Due to the fact that it does not contain secret data, interception can be tolerated. At the sender site, discarding this message is detected since the following `group-subscribed` message will be missing, too. A detection at the coordinator site is not required.
- `group-subscribed`: To transmit a `group-subscribed` message, acknowledged unicast with security level `Confidential` is used. Due to the initial `group-join` message, the receiver is awaiting this message. Therefore, a message loss is detected.

The dynamic binding protocol variant is analog to the one used for network joins (cf. Figure 7.24). Therefore, the analysis is similar. Compared to a network join, it is not required to search for available routers. As a result, there is no need to exchange `router-avail.req` and `router-avail.res` messages. However, if the underlying data link layer supports native multicast, the new group coordinator has to look for an unused data link group address. To achieve this, the following messages are exchanged:

- `group-address.req`: To send a `group-address.req` message, best-effort global broadcast with security level `Trusted` is used. Since the contained values are all public ones, confidentiality is not necessary. At the coordinator site, a lost message cannot be detected. This is also true for the sender site, since the sender is not able to distinguish between a lost message and the case where the chosen address is not in use. As a result, the new group coordinator may use a data link group address that is already used by another communication group. From a security point of view, using the same data link group address is not critical since securing the communication by the GSTS logically separates it from other communication groups. Nevertheless, if a group coordinator detects that a data link group address is also used by another group, it may change it by revoking the GCDR.
- `group-address.res`: This message is sent using acknowledged unicast with security level `Trusted`. As it does not contain secret information, guaranteeing confidentiality is not demanded. A detection at the receiver site is not necessary since the sender will retransmit the message if the acknowledgment is missing.

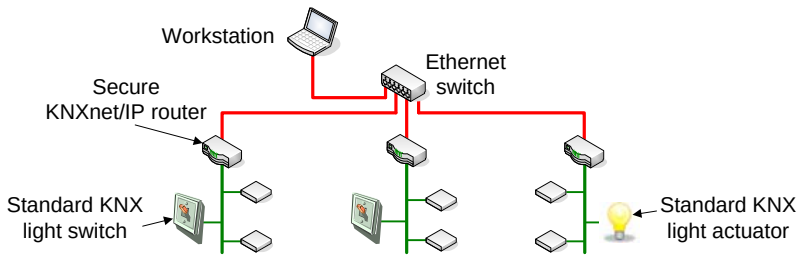


Figure 8.3: Prototype network

`group-revocation`

The revocation of a GCDR is similar to the revocation of an NCDR. The only difference is that reliable multicast is used for sending `group-revoc-start` and `group-revoc-end` messages. Using reliable multicast, message losses are prevented. As a result, it is avoided that group members send messages secured with the old GSTS during the revocation and so inconsistent date views are avoided.

`group-leave`

Leaving a communication group is identical to leaving of a network (cf. Figure 7.26(b)).

8.2 Prototype implementation

Formal evaluation provides the opportunity to verify a given solution in a formal and precise way. To achieve such a formal evaluation, basic assumptions about the system model have to be made. During the evaluation process, the proofs of correctness are done relative to these assumptions.

In the proposed evaluation, two major assumptions have been made. First, an adversary model that specifies the adversary's basic capabilities has been defined. Second, it has been assumed that the used cryptographic transformations satisfy properties formally described in Chapter 4. To verify whether the assumed adversary model is reasonable and that implementations of cryptographic transformations (that fulfill all requirements) exist, the real world behavior has to be investigated. This is accomplished best using prototyping.

8.2.1 Basic setup

To evaluate the presented security concept, a prototype implementation has been set up. As a proof of concept, the security concept has been applied to the IP extension

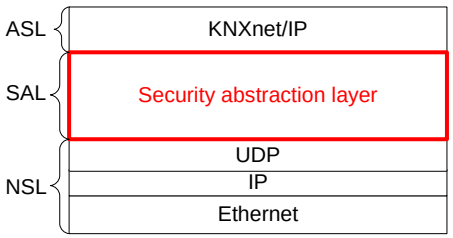


Figure 8.4: Communication stack for secure KNXnet/IP

of KNX also referred to as KNXnet/IP [59]. There are two reasons for this choice. First, KNXnet/IP does not provide any reasonable protection against security attacks. Second, due to the fact that securing multicast is more challenging than protecting unicast communication, KNXnet/IP is the ideal choice since the communication within KNXnet/IP is entirely based on IP multicast.

Figure 8.3 shows the basic structure of the test setup. The network consists of three KNX TP 1 networks which are interconnected by an IP backbone. Three AT91SAM7X-EK evaluation boards act as secure KNXnet/IP routers. While the onboard Ethernet controller provides the interface to the IP backbone, a standard ASIC (called TP-UART [119]) is used for access to the KNX TP 1 network segment. For testing purposes, two standard KNX light switches as well as a standard KNX light actuator are connected to the TP 1 networks. A workstation is connected to the Ethernet switch, too. This workstation is used to simulate an adversary – with the help of Wireshark [120] and a KNXnet/IP plugin [121] any network message can be intercepted. Additionally, it is possible to inject new messages. Furthermore, as a combination of interception and fabrication, previously intercepted messages can also be replayed.

To secure the communication at the IP backbone, the proposed security architecture has been integrated into the KNXnet/IP protocol[6]. As a result, the communication between the KNXnet/IP routers is protected against security attacks – once security is enabled, the KNXnet/IP frames are protected with security level `Confidential` and so the workstation is not able to maliciously interfere with the installation anymore.

8.2.2 Implementation details

In the proposed implementation, the SAL is located between UDP and KNXnet/IP (cf. Figure 8.4). This means that UDP in combination with IP and Ethernet acts as virtual data

[6]In this prototype implementation, the KNX TP 1 networks are not secured. However, due to the used protocol architecture (cf. Figure 8.4), the presented implementation is not limited to KNXnet/IP and can easily be ported to KNX TP 1.

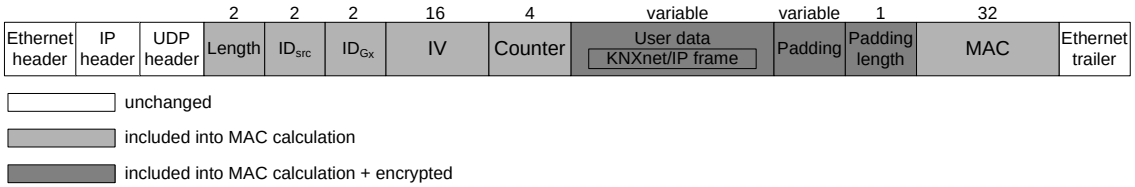


Figure 8.5: Format of secured KNXnet/IP frames

link layer whereas KNXnet/IP is used as ASL. The main advantage of this architecture is that the KNXnet/IP messages remain untouched. Thus, it is not necessary to change or extend the KNXnet/IP protocol.

In KNXnet/IP, devices that belong to different logical KNXnet/IP network segments may share the same physical IP network. To provide a logical separation, KNXnet/IP devices within the same logical KNXnet/IP network are members of the identical IP multicast group. To protect the communication within these IP multicast groups, the secure group communication facility of the SAL is used. Therefore, each device has to join the same group communication relationship. The communication setup is divided into four phases. First, each KNXnet/IP device must retrieve the initial configuration data once during deployment. The necessary steps during the initial configuration are done using network mode as defined in Section 7.5.1. Using the received ICDR, the device is ready to access the KNXnet/IP network. If it is finally connected, it tries to join the communication group. For this prototype implementation, the dynamic binding protocol as defined in Section 7.5.2 has been chosen. If the first device wants to join the communication group, it generates a GSTS, selects a free multicast address (within a predefined range), and becomes coordinator. If there is a coordinator already available, it retrieves the GCDR from the coordinator. Afterwards, the joined device is able to securely communicate with others of the same group.

Figure 8.5 shows the used frame format. The Ethernet, IP, and UDP header as well as the subsequent Ethernet trailer stay untouched. The remaining fields are added by the SAL. Light shaded blocks denote fields that are included into the MAC generation and dark shaded ones indicate fields that are additionally encrypted. The length field specifies the number of bytes that are added by the SAL. The following 4 bytes represent the ID of the sender as well as the ID of the destination group. The next 16 byte block specifies the IV that is used for the encryption algorithms. Afterwards, the used counter that acts as TVP is integrated. In the proof of concept implementation, a counter with a length of 16 bytes is used. For KNXnet/IP, this size is sufficient since it can be assumed that a KNXnet/IP device sends at most 50 frames per second[7]. In the worst case,

[7]This assumption is based on the upcoming KNX/IP specification [122] where 50 frames per second will

a wrap around of the counter occurs in 2.7 years. After the current counter value, the encrypted KNXnet/IP frame is included. Since a block-cipher is used as encryption/decryption transformation, padding is necessary. The secured part of the frame is completed by the MAC.

In addition to securing the data frame, the SAL is also responsible for providing the mapping of the IDs to the data link layer addresses. In the proposed solution, the source and destination IP addresses together with the corresponding UDP ports are used as virtual data link layer address. Since IP already provides support for multicast, the native multicast services are used by the SAL.

8.2.3 Selecting cryptographic transformations

As shown in Section 8.1, the presented formal evaluation relies on the assumption that the underlying cryptographic transformations are secure. Therefore, for real world implementations, it is necessary to select appropriate cryptographic schemes that satisfy the specified security objectives.

In general, the aim of a security algorithm is that it is not breakable. In this context, breakable means that a third party is able perform the included cryptographic transformation (e.g., decryption, MAC generation) within a reasonable time frame without knowing the secret keys [21]. To be unbreakable, the size of the time frame must be large enough so that the security of the data is guaranteed during the whole lifespan of it. When analyzing real world implementations of cryptographic transformations, the following definition of a secure cryptographic transformation is common. A cryptographic transformation is said to be secure if no algorithm is known that breaks a cryptographic transformation faster than exhaustive search of the key space. This means that if a cryptographic transformation is secure and the key space (i.e., the length of the secret keys) is chosen large enough, a brute force attack on finding the secret key becomes computationally infeasible and so the cryptographic transformation becomes practically applicable.

While it is not possible to prove the absence of algorithms that make a cryptographic transformation insecure, a number of instances of cryptographic schemes exist where no such algorithms have been found yet. From this large set of available cryptographic schemes that are assumed to be secure, the following ones come into consideration. For the dynamic binding protocol, asymmetric schemes for encryption/decryption and for digital signature calculation/verification are necessary. Asymmetric encryption/decryp-

be specified as a hard limit.

tion schemes that are commonly used today are RSA encryption scheme [45], ElGamal [47], or ECIES [48, 49]. Typical digital signature calculation/verification schemes are the RSA signature scheme [45], DSA [51], and ECDSA [50]. The main disadvantage of asymmetric schemes is that they are very resource-consuming and thus of limited applicability on embedded devices [123]. However, with the introduction of ECC, it is possible to use asymmetric algorithms even on low-end embedded devices. The main advantage of ECC is that compared to other asymmetric schemes, ECC schemes need a smaller key size. For example, RSA schemes with a key size of 3072 bits provide a comparable security level to ECC schemes with a key length of 256 bits [50]. As a result, ECC are more efficient concerning processing power, memory usage, and power consumption. In [77], it is shown how cryptographic schemes based on ECC can be used to efficiently implement TLS on embedded devices. Therefore, ECC has also been used in the proposed prototype implementation – ECIES is used for encryption/decryption and ECDSA for digital signature generation/verification.

The secure communication phase is based on symmetric algorithms exclusively. Therefore, appropriate symmetric schemes have to be selected. Typical examples of symmetric encryption/decryption schemes are AES [30], 3DES [31], Twofish [32], Blowfish [33], Camellia [34], and SAFER [35]. Since AES is the most common one and so many microcontrollers are already equipped with a hardware implementation of AES, it was elected here, too. The key size has been set to 128 bits. Since most KNXnet/IP frames exceed the block size of 128 bits, AES is used in combination with CBC mode [37]. The IV required for the generation of the first cipher block is contained within the frame (c.f. Figure 8.5). For MAC generation and verification, HMAC which is also used in many state of the art protocols (e.g., TLS, BACnet Addendum g) has been chosen [40]. Since HMAC relies on a cryptographic hash transformation, an appropriate implementation of a secure cryptographic hash transformation has to be chosen, too. While MD5 [26] can still be found in many designs, its use is not advisable anymore since MD5 is not collision resident. Therefore, in the proposed solution, SHA-256 has been selected. SHA-256 is also used in combination with ECDSA for generating digital signatures.

8.2.4 Hardware/software architecture

In Figure 8.6, the basic hardware layout as well as the software architecture of the secure KNXnet/IP router is shown. The software is implemented on AT91SAM7X-EK evaluation boards. These boards are equipped with an AT91SAM7X256 chip which is a 32-bit

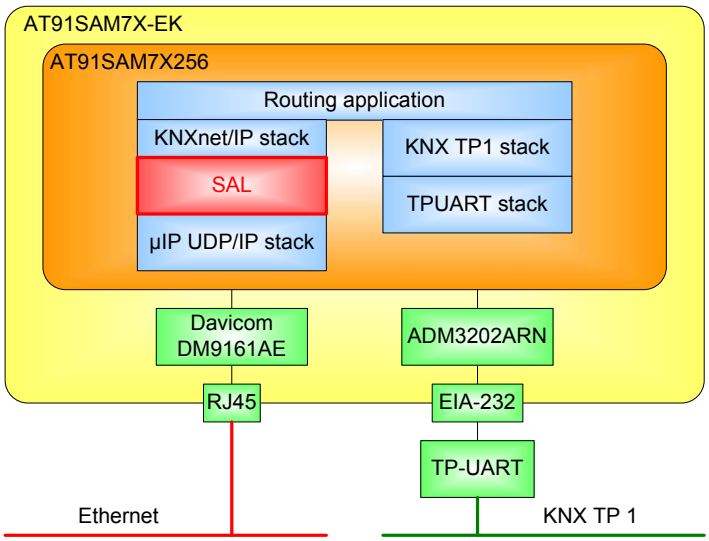


Figure 8.6: Secure KNXnet/IP router

ARM7 processor including 256 KB flash memory and 64 KB SRAM. It operates at up to 55 MHz. The CPU is connected to an Ethernet chip and to an EIA-232 interface which is used for the communication with the TP-UART.

The open source library *µIP* serves as UDP/IP stack [124]. *µIP* features a very small code size and low memory usage, hence it is suitable for the use in embedded microcontrollers. For cryptographic calculations, many arithmetic operations on large numbers have to be performed. Since native C cannot handle such large numbers, a special library called *Multiprecision Integer and Rational Arithmetic C Library (MIRACL)* has been used [125]. It provides implementations of cryptographic schemes like AES, RSA, DH, ECC over binary and prime fields, and various hashing transformations. Optimizations for different processor architectures are written in inline assembly for time-critical routines to enhance speed. The KNXnet/IP stack and the routing application support KNXnet/IP routing functionality. To communicate with KNX TP 1 devices, a KNX TP 1 stack as well as a TP-UART stack are used [126].

In Figure 8.7, a performance evaluation of the implemented ECC schemes is shown. The first row represents the measurements that are taken on the microcontroller ARM7 TDMI (32 bit RISC, 55 Mhz, 64KB SRAM) which is used on the evaluation boards. To compare the results, the ECC implementation has been ported to an embedded Linux platform called Grasshopper. This board is equipped with the AVR32 based microcontroller AT32AP7000 (32 bit RISC, 140 Mhz, 64MB SDRAM). Although the used ECC implementation does not include any optimizations, the performance values demonstrate the feasibility of the dynamic binding protocol on embedded devices. Especially if taking

MCU	Key pair generation	ECIES		ECDSA	
		Encrypt	Decrypt	Generate	Verify
ARM7	245.2 ms	379.4 ms	131.2 ms	128.5 ms	244.8 ms
AVR32	37.9 ms	57.5 ms	20.5 ms	20 ms	37.1 ms

Figure 8.7: Performance of ECC algorithms

into account that asymmetric operations are only executed during the group join and key revocation process – during the secure communication phase, only symmetric schemes are executed.

9 Guaranteeing data availability

Chapter 6 and Chapter 7 present a comprehensive security concept that protects the data exchange within HBA networks. The used mechanisms are based on secured channels where cryptographic techniques are used to guarantee the required security objectives. In Chapter 8, the proposed security approach is evaluated. As has been shown, permanent interruption attacks where adversaries are able to completely interrupt the communication between devices (e.g., by cutting the physical connection or by jamming) cannot be avoided using secured channels. Handling these DoS attacks is not a trivial task especially if open media like PL or RF are used [127]. However, in the presented concept, a detection by at least one device is possible. Therefore, advanced security concepts that initiate further countermeasures have to be integrated into a secure HBA network, too. The remainder of this chapter presents a brief survey of organizational countermeasures that counteract these DoS attacks.

9.1 Denial-of-Service detection

One possibility to detect DoS attacks is the use of a so called *Intrusion Detection System*. The objective of an IDS is to detect abnormal system behavior (e.g., abnormal network or device behavior). After having detected such an abnormal situation, it must be determined whether it may lead to a DoS attack. If it is considered as an attack, countermeasures have to be initiated to minimize its consequences i.e., to avoid further damage of the system components.

According to [128], an IDS commonly consists of four components (cf. Figure 9.1). The *data gathering component* is responsible for collecting the data by observing the network traffic as well as the behavior of the different network devices. IDSs are often classified according to the type of data collection (i.e., the location of the data gathering components). A host-based IDS observes the activities on a single device and compares the behavior pattern with a reference (profiling). Host-based intrusion detection is especially applicable for security-critical devices (e.g., coordinators). A network-based IDS

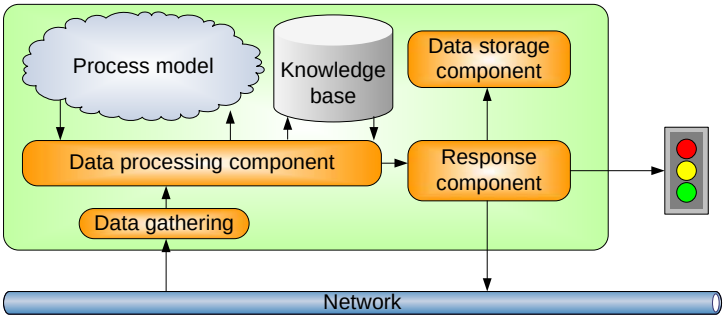


Figure 9.1: Intrusion Detection System

observes the network traffic. Therefore, these systems are able to discover anomalies which affect more than a single host.

The core unit of an IDS is the *data processing component*. This component processes the collected data and determines whether abnormal behavior is present. Again, different approaches exist. The two most important ones are called misuse based and anomaly based. Misuse based systems use a priori knowledge of activities that form an attack. This knowledge is stored in a database which contains typical patterns of known attacks (signatures). To determine whether an observation can be classified as an attack, different techniques such as expert systems or signature detection mechanisms can be used. Anomaly based intrusion detection tries to detect abnormal behavior by comparing the observed behavior with the normal and expected behavior (also called reference pattern). To achieve such a comparison, a system model must be specified. This model must define the default reference pattern (i.e., network traffic, device behavior) which represents the expected and normal behavior of the system. Obviously, this default behavior is not static since it can change during the lifetime of the system. Therefore, self-learning techniques (e.g., neural networks) are usually applied.

Collecting the results as well as the observed data (communication traces) is the task of the *data storage unit*. The *response unit*, finally, is responsible for initiating actions to minimize the consequences of a detected security attack. This can be done by performing a direct feedback to the network. For example, it could decouple the affected network segment(s).

In the IT domain, many different IDS solutions exist. However, their data gathering units are mainly designed for IP networks and the assumed system model is significantly different from the process model of an HBA system. Therefore, IDS from the IT world can only be used as a starting point – parts have to be redesigned. The functionality of the IDS is incorporated into the software implementation of routers, gateways, and/or coordinators.

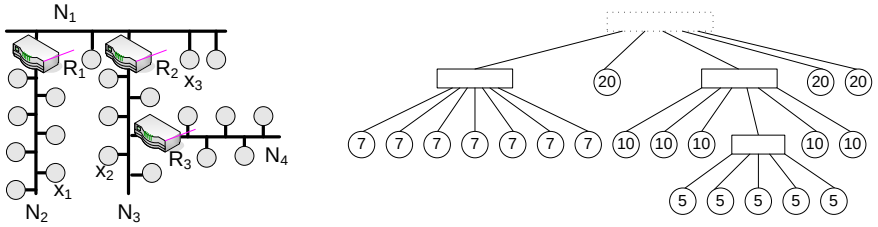


Figure 9.2: Device isolation

9.2 Denial-of-Service countermeasures

After a DoS attack has been detected by a communication party or by an IDS, it is essential to minimize the resulting consequences. The most effective solution is to stop the adversary from attacking the target. To achieve this, the source(s) of the DoS attack have to be identified and isolated from the rest of the network. This keeps the system operable and prevents a propagation of the DoS attack.

The most appropriate way to isolate the source(s) of the DoS attack is closely related to the physical topology of the affected network segment. In the HBA network, star (e.g., switched Ethernet networks), line, and even free topology are common.

In wired star topologies, an isolation can be accomplished by cutting the communication line to the source(s) of the attack. For example, in a switched Ethernet network, the port where a DoS attack has been detected can simply be deactivated.

In wireless networks, the isolation highly depends not only on the logical topology (e.g., star, mesh) but also on the used communication model (e.g., peer-to-peer or coordinator-to-peer). The basic idea is to isolate an affected zone and to find routes that bypass this zone [127].

Finally, in wired line or free topology, an isolation can only be achieved by decoupling the whole network segment. This action has to be executed by the ICD located at a network segment border. Consider, for example, device x_1 in Figure 9.2 starts a flooding attack. If this misbehavior is detected by the IDS module running in router R_1, the router can isolate network segment N_2 by stopping forwarding messages to N_1. Therefore, the rest of the network (i.e., N_1, N_3, and N_4) can continue normal operation. The main drawback of this solution is that all network members of N_2 become isolated. To estimate the damage a device may cause when performing a DoS attack, the so called *DoS Risk Factor (DoS-RF)* is defined. To calculate the DoS-RF, the network topology is represented as tree based graph. On the right hand side of Figure 9.2, the corresponding graph of the example network is shown. While ICDs are represented as rectangular nodes, field devices are shown as circular nodes. An edge between two nodes stands for a physical

network connection between two devices. Furthermore, the sequence of the children of an interconnection node defines the physical location of the device within the network segment – the left-most child is always the device located next to the interconnection device[1]. The DoS-RF of a device x is now calculated as the sum of all node weights of all children and sub-children of the parent of the device x. The weight represents the damage that occurs if a device fails due to a security attack. In a first approach, the amount of data points a device holds as well as their types (e.g., safety/security critical, system, normal, low) are used to determine the node weight. Consider, for example, device x_2. To isolate this device, the routers R_2 and R_3 have to decouple N_3. This means that all members of N_3 are no longer able to communicate with N_1, N_2, and N_4. However, since there is no route from N_4 to N_1 and N_2, also the members of N_4 are only able to communicate with each other. Any communication with the other remaining, operable network segments is interrupted. Due to the severity of an attack on node x_2, its DoS-RF equals to 10, which is the weighted count of all children and sub-children of its parent node R_2.[2]

While a low DoS-RF indicates that decoupling the device from the rest of the system influences only a smaller part of the network, a high DoS-RF represents the opposite. In case of an attack, a large number of devices would become isolated. To counter large DoS-RFs, a physical network segment has to be divided into smaller so called *virtual network segments*. Virtual network segments are not logically separated from each other i.e., they do not have a dedicated network address. Therefore, virtual network segments are invisible to network members. Dividing a physical network segment into two virtual network segments is done by placing a so called *virtual bridge* at the virtual segments' intersection point. A virtual bridge has two (or more) network interfaces and simply forwards incoming messages to the other network interface. However, in contrast to a layer 2 bridge, it is possible to request the virtual bridge to decouple virtual network segments by cutting the communication line between them. Consider, for example, device x_3. In the original network (cf. Figure 9.2), x_3 has a DoS-RF of 20. However, by placing a virtual bridge between R_2 and x_3, the DoS-RF of x_3 can be reduced to 2. This is due to the fact that now also the virtual bridge is able to decouple x_3 and the device next to x_3 (cf. Figure 9.3).

This solution has one drawback: if a (virtual) network segment is located between two other (virtual) network segments, decoupling it also isolates all succeeding network seg-

[1] It is assumed that the network topology is following a line topology. However, if there is free topology, the topology has to be converted to a line topology by inserting virtual interconnection nodes.

[2] It is assumed that all devices hold a single data point of type "low" and so each has a weight of 1.

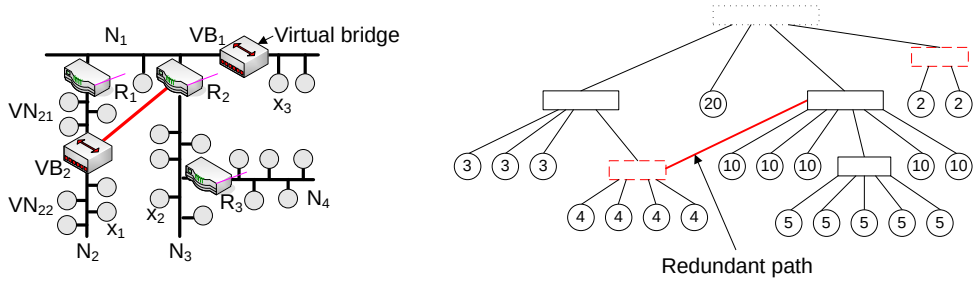


Figure 9.3: Virtual bridges

ments from the remaining network. Consider, for example, the virtual bridge VB_2 that is added in N_2. This virtual bridge reduces the DoS-RF of all succeeding devices to 4. However, the devices located between R_1 and VB_2 still have a DoS-RF of 7 since a DoS attack within VN_{21} also decouples VN_{22} from the remaining network. To avoid this, alternative communication paths have to be provided by using redundant interconnections between virtual bridges and ICDs. Suppose, for example, a redundant connection is added between VB_2 and R_2. Using this connection, the members of VN_{22} are still able to communicate with the remaining network. In this case, the DoS-RF of the members of VN_{21} is reduced from 7 to 3. However, adding redundant connections may introduce cyclic paths. Therefore, a mechanism is necessary that only enables these redundant connections when they are required. A possible realization includes a dedicated protocol that caters for communication among all interconnection devices and virtual bridges. Thus, information on attacks can be exchanged and the most appropriate countermeasures can be triggered. An alternative solution would be to permanently enable redundant connections and to discard all duplicate messages. Furthermore, adding redundant paths may introduce additional risk. However, this risk can be eliminated by limiting the physical access to the connection (e.g., immuring the redundant connection), since redundant paths are normally point-to-point connections.

10 Conclusion

"Is security important in the home and building automation domain?"

In the past years, this question was mostly negated. The impacts of violating security of HBA systems were underestimated. Typical examples of common responses that are still given today are the following:

"Why should I bother about someone who turns my light on and off?"
"If someone wants to know my room temperature, I have no objections."

At first sight, these statements seem to be true. However, as shown in the introduction of this dissertation, security threats and attacks must not be neglected. The resulting consequences of security attacks range from serious economic damage to a complete malfunction of the system. Consider, for example, a simple lighting system. A vandalism act may lead to massive economic impact (e.g., company-wide disturbance or system shutdown), but may also harm people (e.g., mass panic within public buildings or a failure of the HBA system within a hospital). While security attacks on such "apparently harmless" HBA systems may appear non-critical, it may be the case that they do have serious consequences. Consider, for example, an HVAC system of a home. A malicious interception of the present room temperature seems to be useless for adversaries. However, a constant, low room temperature may indicate that the HVAC system is in "vacation mode" which is of great interest for burglars. The situation is getting worse if HBA systems are extended to serve security-critical applications (e.g., security alarm systems, access control). Here, a protection against security attacks is obviously of utmost importance. This is especially true, if the functionality is coupled to safety-critical applications since a malicious interference may harm people, too.

Nevertheless, security in HBA systems was a side issue at best as a first analysis of available open standards revealed [129]. A common misunderstanding within this domain was that a physical isolation of the control system as well as the usage of proprietary, closed communication protocols ("Security by Obscurity") provide a guarantee for

a secure system. The following conversation between security experts and a power plant provider shall emphasize this circumstance [130]:

> *Sitting in conference room negotiating pen-test.*
> *"Why should we buy your services, we are secure so you won't be able to break in."*
> *"We have no WiFi."*
> *Turn notebook around and show that there is an open (no WEP) access point reachable from the conference room.*
> *"Oh, but you can't get an IP address or anything from it."*
> *We connect, it gives us a DHCP address.*
> *"It's just in the lab."*
> *We run scans and show that it's connected to the rest of the office network.*
> *"The office network (where people work) is not connected to the control network (where the power plant is)."*
> *We get into Solaris system using 10 year old exploit.*
> *"Please stop!"*
> *We had broken into a system that was on both networks and, indeed, was in direct control of something extremely sensitive and we were in danger of breaking it.*

It is a fact that neither relying on physical isolation nor trusting in "Security by Obscurity" provides an effective protection against security threats and attacks. Therefore, it is mandatory to secure HBA systems in a way as it has been done in the IT domain – otherwise it is only a matter of time until insecure HBA systems will become the next target of security attacks on a large scale.

On the road to reach this ambitious goal, existing HBA standards were analyzed [80, 129]. Since none of the available solutions provide an effective protection against security threats, the research project "Security in Building Automation" funded by FWF (Österreichischer Fonds zur Förderung der Wissenschaftlichen Forschung; Austrian Science Foundation) was launched. The aim of this project is to design a generic framework for secure building automation systems of all sizes and types. The project is divided into three major working packages – the first one deals with guaranteeing a secure data communication, the second one with organizational measures against DoS attacks, and the third one with counteracting device attacks. A comprehensive survey of the project can be found in [131].

The proposed dissertation represents the contribution and results of the research activities that were performed within the first working package as well as parts of the second one. At the beginning, a generic HBA system model was set up [13, 80]. This generic model was used as basis for an extensive security threat analysis. As shown within this analysis, a comprehensive security concept has to deal with device attacks on the one hand and with network attacks on the other hand. While the former one is part of the project's third working packages [17, 18], this dissertation is focused on counteracting network attacks. Afterwards, requirements and challenges for a secure communication within HBA networks were identified. Based on the requirements analysis, available HBA technologies were investigated. Nevertheless, this state of the art survey is not limited to HBA standards [80] – security schemes from the IT domain that may come into consideration for securing HBA systems have been analyzed, too [132]. While new technologies (e.g., ZigBee, OPC UA) as well as extensions to well-established standards (e.g., BACnet Addendum g) provide a good basis for secure HBA systems, many security issues (e.g., DoS attacks) and implementation details (e.g., management of secret keys) are still left open. As a result, existing solutions cannot be used as a comprehensive security concept for generic HBA systems since they do not fulfill all identified requirements. Applying solutions from the IT domain is also not possible since the domain specific challenges are not satisfied.

Therefore, a generic security architecture dedicated to HBA systems of all sizes and types has been developed. This architecture is based on a multi-protocol stack that can be used independently of the underlying network medium [133]. The key component within this architecture is the so called Security Abstraction Layer (SAL). It provides a generalization of the underlying, native communication primitives, enriches them with security and QoS properties, and supports secure communication services of different types to the application layer. The included security concept is based on secure communication relationships – each device is able to join one or more secure communication relationships where the communication between the relationships' members is protected. To securely join and leave communication relationships, two different protocols are available. The first one makes use of predefined coordinators that are responsible for performing relationship joins and leaves (static binding protocol). To avoid a single point of failure, coordinators can be replicated [134]. The second one is based on a democratic approach where the election of the coordinators is not fixed (dynamic binding protocol) [132]. To evaluate the presented concept, a hybrid approach was used. The feasibility of the used security protocols is shown using formal evaluation. This formal evaluation relies on a

well-defined adversary model as well as on the existence of secure cryptographic transformations. To verify the usability within real world environments, a prototype implementation was set up. As a proof of concept, the dynamic binding protocol was implemented for KNXnet/IP [126, 135].

The presented security architecture is based on cryptographic techniques. However, as shown during the formal evaluation, permanent interruption attacks (e.g., cutting the network cable) cannot be handled. To detect these kinds of DoS attacks and to imitate adequate countermeasures, an advanced security scheme based on organizational measures is necessary. The basic concept of such an approach that can be used to detect DoS attacks and to minimize the resulting consequences is described within this dissertation, too [136].

The feasibility of the proposed security architecture has been shown by using formal evaluation and prototyping. While the former verifies the correctness of the used communication protocols (with respect to the defined assumptions), the latter estimates the applicability under real world conditions. However, using prototyping, a large-scale analysis with more than a few participating devices is neither easily manageable, nor can it be implemented in a cost-efficient way. To verify the practicability of the presented concept within large-scale installations, simulation has to be used. Simulation is an approach mainly targeted to analyzing dynamic systems. Abstract models of a particular system are developed and evaluated using a simulator. It is thus possible to simulate large-scale installations consisting of thousands of devices. Compared to prototyping, a simulation is not bound to real-time. To simulate the behavior of a system over years, a simulator needs only a fraction of time that would be needed using prototyping. Therefore, further research activities will include a simulation of the presented security architecture. While many different simulators exist, OMNeT++ [137, 138] may be an adequate choice. OMNeT++ provides a component-based, modular, and open architecture for discrete event simulation. In [139], a framework for simulating KNX systems is presented. Extending this framework can act as an excellent basis for further evaluating the scalability of the presented security concept.

The security architecture presented within this dissertation is focused on providing secure communication services. What is still left open is the definition of a secure application layer model that makes use of communication services supported by the SAL. A possible approach has been presented in [140, 141]. This concept is based on a generic application object model which is independent of the used technology. Extending this model with security capabilities may act as a good starting point for fully fledged all-in-

one HBA systems.

Further steps of research include the refinement of the advanced security scheme to detect and counteract DoS attacks. The presented DoS detection mechanism is based on the use of an IDS. A critical point within this concept is the definition of a process model that acts as reference pattern for detecting abnormal system behavior. Modelling the default system behavior can be done using ontologies [142]. To be able to efficiently counteract identified DoS attacks, the accurate placement of virtual bridges and redundant paths is of importance. To assist the project engineer during installation time, engineering tools shall be developed that help to find critical points in the network topology and suggest where to add virtual bridges and redundant connections best.

Furthermore, it has to be mentioned that the presented security architecture is not only limited to HBA systems. In other domains like factory automation, security concepts are missing, too [143, 144, 145]. Therefore, it seems to be reasonable to set up similar security schemes there as it has to be done for the HBA domain. However, while the requirements and challenges of industrial automation systems are comparable to the ones identified within the HBA domain, there are also some significant differences. For instance, industrial automation systems are often related to safety-critical applications where hard real-time requirements have to be met. As a result, it is necessary to perform an extensive domain analysis before the proposed security architecture can be applied or even mapped to industrial automation systems.

Bibliography

[1] W. Kastner, G. Neugschwandtner, S. Soucek, and H. M. Newman, "Communication Systems for Building Automation and Control," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1178–1203, 2005.

[2] W. Kastner and G. Neugschwandtner, "Data Communications for Distributed Building Automation," in *Embedded Systems Handbook, Second Edition, Volume 2: Networked Embedded Systems*, ser. The Industrial Information Technology Series, R. Zurawski, Ed. Boca Raton: CRC Press, 2009, ch. 29, pp. 29–1 — 29–34.

[3] M. Thuillard, P. Ryser, and G. Pfister, "Life Safety and Security Systems," in *Sensors in Intelligent Buildings*. Wiley-VCH, 2001, vol. 2, pp. 307–397.

[4] K. Daniels, *Advanced Building Systems. A Technical Guide for Architects and Engineers*. Birkhäuser, 2002.

[5] W. Kastner and T. Novak, "Functional Safety in Building Automation," in *Proc. of the 14th IEEE Conference on Emerging Technologies and Factory Automation (ETFA '09)*, Sep. 2009, pp. 1 – 8.

[6] D. Dietrich, W. Kastner, T. Maly, C. Roesener, G. Russ, and H. Schweinzer, "Situation Modeling," in *Proc. of the 5th IEEE International Workshop on Factory Communication System (WFCS '04)*, Sep. 2004, pp. 93–102.

[7] R. Shirey, "Internet Security Glossary, Version 2," RFC 4949, Aug. 2007.

[8] International Organization for Standardization, "Information Technology – Vocabulary – Part 8: Security," ISO/IEC 2382-8, 1998.

[9] ——, "Information Technology – Security techniques – Evaluation Criteria for IT security," ISO/IEC 15408, 2005.

[10] M. Bishop, *Computer Security: Art and Science*. Addison-Wesley, 2002.

[11] International Organization for Standardization, "Building Automation and Control Systems (BACS) – Part 2: Hardware," ISO 16484-2, 2004.

[12] C. Reinisch, W. Kastner, G. Neugschwandtner, and W. Granzer, "Wireless Technologies in Home and Building Automation," in *Proc. of the 5th IEEE International Conference on Industrial Informatics (INDIN '07)*, vol. 1, Jul. 2007, pp. 93–98.

[13] W. Granzer, W. Kastner, G. Neugschwandtner, and F. Praus, "A Modular Architecture for Building Automation Systems," in *Proc. of the 6th IEEE International Workshop on Factory Communication Systems (WFCS '06)*, Jun. 2006, pp. 99–102.

[14] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing*, 4th ed. Prentice Hall, 2006.

[15] D. Hwang, P. Schaumont, K. Tiri, and I. Verbauwhede, "Securing Embedded Systems," *IEEE Security & Privacy Magazine*, vol. 4, no. 2, pp. 40–49, Mar. 2006.

[16] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in Embedded Systems: Design Challenges," *ACM Trans. on Embedded Computing Systems*, vol. 3, no. 3, pp. 461–491, Aug. 2004.

[17] F. Praus, T. Flanitzer, and W. Kastner, "Secure and Customizable Software Applications in Embedded Networks," in *Proc. of the 13th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA '08)*, Sep. 2008, pp. 1473–1480.

[18] F. Praus, W. Granzer, and W. Kastner, "Enhanced Control Application Development in Building Automation," in *Proc. of the 7th IEEE International Conference on Industrial Informatics (INDIN '09)*, Jun. 2009, pp. 390–395.

[19] J.-P. Thomesse, "Fieldbus Technology in Industrial Automation," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1073–1101, Jun. 2005.

[20] D. Dzung, M. Naedele, T. V. Hof, and M. Crevatin, "Security for Industrial Communication Systems," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1152–1177, Jun. 2005.

[21] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, 5th ed. CRC Press, 2001.

[22] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: Security Protocols for Sensor Networks," in *Proc. of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom '01)*, Jul. 2001, pp. 189–199.

[23] H. Attiya and J. Welch, *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*, 2nd ed. John Wiley and Sons Inc, 2004.

[24] "UML Unified Modeling Language 2.2," OMG Object Management Group, 2009.

[25] A. Kerckhoff, "La Cryptographie Militaire," *Journal des Sciences Militaires*, vol. 9, pp. 5–38, Jan. 1883.

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

[27] "Secure Hash Standard," NIST FIPS PUBS 180-2, Aug. 2002.

[28] X. Wang and H. Yu, "How to Break MD5 and Other Hash Functions," *Lecture Notes in Computer Science*, vol. 3494, pp. 19–35, 2005.

[29] "Recommendation for Random Number Generation Using Deterministic Random Bit Generators," NIST Special Publication 800-90, Mar. 2007.

[30] "Advanced Encryption Standard (AES)," NIST FIPS PUBS 197, Dec. 2001.

[31] "Data Encryption Standard (DES)," NIST FIPS PUB 46-3, Oct. 1999.

[32] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, *The Twofish Encryption Algorithm: A 128-Bit Block Cipher*. Wiley, Mar. 1999.

[33] B. Schneier, "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)," *Lecture Notes in Computer Science*, vol. 809, pp. 191–204, 1994.

[34] M. Matsui, J. Nakajima, and S. Moriai, "A Description of the Camellia Encryption Algorithm," RFC 3713, Apr. 2004.

[35] J. Massey, "SAFER K-64: A Byte-Oriented Block-Ciphering Algorithm ," *Lecture Notes in Computer Science*, vol. 809, pp. 1–17, 1994.

[36] B. Moeller. (Nov.) Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures. Last access: 12.02.2010. [Online]. Available: http://www.openssl.org/~bodo/tls-cbc.txt

[37] "Recommendation for Block Cipher Modes of Operation," NIST Special Publication 800-38A 2001 ED, Dec. 2001.

[38] B. Schneier, *Applied Cryptography Second Edition: Protocols Algorithms and Source in Code in C*. John Wiley and Sons, Oct. 1996.

[39] eSTREAM: ECRYPT Stream Cipher Project. Last access: 12.02.2010. [Online]. Available: http://www.ecrypt.eu.org/stream

[40] "The Keyed-Hash Message Authentication Code (HMAC)," NIST FIPS PUBS 198, Mar. 2002.

[41] J. Song, R. Poovendran, J. Lee, and T. Iwata, "The AES-CMAC Algorithm," RFC 4493, Jun. 2006.

[42] D. Whiting, R. Housley, and N. Ferguson, "Counter with CBC-MAC (CCM)," RFC 3610, Sep. 2003.

[43] P. Rogaway, M. Bellare, and J. Black, "OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption," *ACM Trans. on Information and System Security*, vol. 6, pp. 365–403, 2003.

[44] M. Bellare, P. Rogaway, and D. Wagner, "EAX: A Conventional Authenticated-Encryption Mode," Cryptology ePrint Archive, Report 2003/069, 2003, http://eprint.iacr.org/.

[45] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, 1978.

[46] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Trans. on Information Theory*, vol. 22, pp. 644–654, 1976.

[47] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Trans. on Information Theory*, vol. 31, pp. 469–472, 1985.

[48] N. Koblitz, "Elliptic Curve Cryptosystems," *Mathematics of Computation*, vol. 48, pp. 203–209, 1987.

[49] A. Cilardo, L. Coppolino, N. Mazzocca, and L. Romano, "Elliptic Curve Cryptography Engineering," *Proceedings of the IEEE*, vol. 943, no. 2, pp. 395–406, Feb. 2006.

[50] D. R. Hankerson, S. A. Vanstone, and A. J. Menezes, *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[51] "Digital Signature Standard (DSS)," NIST FIPS PUBS 186, May 1994.

[52] "BACnet – A Data Communication Protocol for Building Automation and Control Networks," ANSI/ASHRAE 135-2008, 2008.

[53] "Building Automation and Control Systems (BACS) – Part 5: Data Communication Protocol," ISO 16484-5, 2007.

[54] "Building Automation and Control Systems (BACS) – Part 5: Data Communication Protocol," ISO 16484-5/Amd 1, 2009.

[55] "Control Network Protocol Specification," ANSI/EIA/CEA 709 Rev. B, 2002.

[56] "Open Data Communication in Building Automation, Controls and Building Management – Control Network Protocol," EN 14908, 2005.

[57] "Open Data Communication in Building Automation, Controls and Building Management – Control Network Protocol," ISO/IEC 14908, 2008.

[58] "Information Technology - Home Electronic Systems (HES) Architecture," ISO/IEC 14543-3, 2006.

[59] "KNX Specification Version 2.0," Konnex Association, Diegem, 2009.

[60] "ZigBee Specification," ZigBee Alliance, San Ramon, 2007.

[61] "Digital Addressable Lighting Interface," IEC 62386, 2009.

[62] "Communication Systems for Meters and Remote Reading of Meters," EN 13757, 2004.

[63] "Modbus Application Protocol V1.1b," Modbus Organization, 2006.

[64] EnOcean. Last access: 12.02.2010. [Online]. Available: http://www.enocean.com

[65] "Z-Wave System Design Specification: Z-Wave Protocol Overview," Zensys A/S, Fremont, 2005.

[66] "OPC Data Access Specification," OPC Foundation, 2003.

[67] "OPC Unified Architecture Specification," OPC Foundation, 2009.

[68] "oBIX 1.0 Committe Specification," OASIS, 2006.

[69] "WS-I Basic Profile 1.0," Web Services Interoperability Organization, 2004.

[70] "Method of Test for Conformance to BACnet," ANSI/ASHRAE 135.1-2007, 2007.

[71] "Building Automation and Control Systems (BACS) – Part 6: Data Communication Conformance Testing," ISO 16484-6, 2009.

[72] C. Schwaiger and A. Treytl, "Smart Card Based Security for Fieldbus Systems," in *Proc. of the 9th IEEE Conference on Emerging Technologies and Factory Automation (ETFA '03)*, vol. 1, Sep. 2003, pp. 398–406.

[73] D. G. Holmberg, "BACnet Wide Area Network Security Threat Assessment," NISTIR 7009, National Institute of Standards and Technology, Tech. Rep., 2003.

[74] J. Zachary, R. Brooks, and D. Thompson, "Secure Integration of Building Networks into the Global Internet," NIST GCR 02-837, National Institute of Standards and Technology, Tech. Rep., 2002.

[75] Electronic Frontier Foundation, *Cracking DES: Secrets of Encryption Research, Wiretap Politics and Chip Design*. O'Reilly & Associates, 1998.

[76] "BACnet – A Data Communication Protocol for Building Automation and Control Networks," ANSI/ASHRAE 135-2008: Addendum g, 2009.

[77] V. Gupta, M. Wurm, Y. Zhu, M. Millard, S. Fung, N. Gura, H. Eberle, and S. C. Shantz, "Sizzle: A Standards-based End-to-End Security Architecture for the Embedded Internet," *Pervasive and Mobile Computing*, vol. 1, no. 4, pp. 425–445, 2005.

[78] W.-S. Juang, S.-T. Chen, and H.-T. Liaw, "Robust and Efficient Password-Authenticated Key Agreement Using Smart Cards," *IEEE Trans. on Industrial Electionics*, vol. 55, no. 6, pp. 2551–2556, Jun. 2008.

[79] "Tunneling Component Network Protocols Over Internet Protocol Channels," ANSI/EIA 852, 2002.

[80] W. Granzer, W. Kastner, G. Neugschwandtner, and F. Praus, "Security in Networked Building Automation Systems," in *Proc. of the 6th IEEE International Workshop on Factory Communication Systems (WFCS '06)*, Jun. 2006, pp. 283–292.

[81] "Home and Building Electronic Systems (HBES)," EN 50090, 2003.

[82] G. Westermeir and F. Schneider, "Methods for Cryptographically Secured Data Transmission in EIB/KNX Networks," in *Konnex Scientifc Conference*, 2004.

[83] "IEEE 802.15.4 Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)," IEEE Computer Society, New York, Sep. 2006.

[84] "IEEE 802.15.4 Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs): Amendment 1: Add Alternate PHYs," IEEE Computer Society, New York, Aug. 2007.

[85] "HART Field Communications Protocol Revision 7," HART Communication Foundation, 2007.

[86] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944, Sep. 2007.

[87] "ZigBee: Home Automation Public Application Profile," ZigBee Alliance, San Ramon, 2008.

[88] "ZigBee: Smart Energy Public Application Profile," ZigBee Alliance, San Ramon, 2008.

[89] "Bluetooth Specification Version 3.0," Bluetooth SIG, Apr. 2009.

[90] "IEEE 802.15.1, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)," IEEE Computer Society, New York, Jun. 2005.

[91] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC Unified Architecture*. Springer, 2009.

[92] "OPC Unified Architecture Specification," IEC 62541, 2008, Status: Committee Draft for Voting.

[93] "WS-I Basic Profile 1.1," Web Services Interoperability Organization, 2004.

[94] "WS-SecureConversation 1.3," OASIS, Mar. 2007.

[95] "WS-SecurityPolicy 1.2," OASIS, Jul. 2007.

[96] "WS-Trust 1.3," OASIS, Mar. 2007.

[97] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301, Dec. 2005.

[98] S. Kent, "IP Authentication Header," RFC 4302, Dec. 2005.

[99] D. Eastlake, "Cryptographic Algorithm Implementation Requirements For Encapsulating Security Payload (ESP) and Authentication Header (AH)," RFC 4305, Dec. 2005.

[100] C. Kaufman, "The Internet Key Exchange (IKEv2) Protocol," RFC 4306, Dec. 2005.

[101] B. Weis, G. Gross, and D. Ignjatic, "Multicast Extensions to the Security Architecture for the Internet Protocol," RFC 5374, Nov. 2008.

[102] T. Hardjono and B. Weis, "The Multicast Group Security Architecture," RFC 3740, Mar. 2004.

[103] A. Perrig, D. Song, R. Canetti, J. D. Tygar, and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction," RFC 4082, Jun. 2005.

[104] M. Baugher, B. Weis, T. Hardjono, and H. Harney, "The Group Domain of Interpretation," RFC 3547, Jul. 2003.

[105] H. Harney, U. Meth, A. Colegrove, and G. Gross, "GSAKMP: Group Secure Association Key Management Protocol," RFC 4535, Jun. 2006.

[106] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, Aug. 2008.

[107] M. Feilner, *OpenVPN: Building and Integrating Virtual Private Networks*. Packt Publishing, Oct. 2006.

[108] P. Palensky and T. Sauter, "Security Considerations for FAN-Internet Connections," in *Proc. of the 3rd International Workshop on Factory Communication Systems (WFCS '00)*, Sep. 2000, pp. 27–35.

[109] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "MiniSec: A Secure Sensor Network Communication Architecture," in *Proc. of the 6th International Conference on Information Processing in Sensor Networks (IPSN '07)*, 2007, pp. 479–488.

[110] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Springer, 1997.

[111] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, Jul. 1978.

[112] D. Skeen and M. Stonebraker, "A Formal Model of Crash Recovery in a Distributed System," *IEEE Trans. on Software Engineering*, vol. SE-9, no. 3, pp. 219–228, May 1983.

[113] D. Skeen, "Nonblocking commit protocols," in *Proc. of the ACM International Conference on Management of Data (SIGMOD '81)*, Jun. 1981, pp. 133–142.

[114] G. Gaderer, "Fault Tolerance Enhancements to Master/Slave Based Clock Synchronisation," Ph.D. dissertation, Vienna University of Technology, Oct. 2008.

[115] L. Eschenauer and V. D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," in *Proc. of the 9th ACM Conference on Computer and Communications Security (CCS '02)*, Nov. 2002, pp. 41–47.

[116] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication," *ACM Computing Surveys*, vol. 35, no. 3, pp. 309–329, Sep. 2003.

[117] A. S. Tanenbaum and M. van Steen, *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.

[118] P. Loschmidt, "On Enhanced Clock Synchronization Performance through Dedicated Ethernet Hardware Support," Ph.D. dissertation, Vienna University of Technology, 2010, to be published.

[119] Siemens, "Technical Data EIB-TP-UART-IC," 2001, version D.

[120] Wireshark Website. Last access: 12.02.2010. [Online]. Available: http://www.wireshark.org

[121] H. Weillechner. Wireshark KNXnet/IP plugin. Last access: 12.02.2010. [Online]. Available: https://www.auto.tuwien.ac.at/a-lab/wireshark-plugin-for-knxnet/ip.html

[122] W. Kastner and H. Weillechner, "KNX IP Simulation: First Insights," in *Konnex Scientifc Conference*, Nov. 2008.

[123] N. Okabe, S. Sakane, K. Miyazawa, K. Kamada, A. Inoue, and M. Ishiyama, "Security Architecture for Control Networks using IPsec and KINK," in *Proc. of the Symposium on Applications and the Internet (SAINT '05)*, 2005, pp. 414–420.

[124] uIP Website. Last access: 12.02.2010. [Online]. Available: http://www.sics.se/~adam/uip

[125] Shamus Software Ltd. Last access: 12.02.2010. [Online]. Available: http://www.shamus.ie

[126] W. Granzer, G. Neugschwandtner, and W. Kastner, "EIBsec: A Security Extension to KNX/EIB," in *Konnex Scientific Conference*, Nov. 2006.

[127] A. D. Wood and J. A. Stankovic, "Denial of Service in Sensor Networks," *IEEE Computer*, vol. 35, no. 10, pp. 54–62, Oct. 2002.

[128] C. Krügel, "Network Alertness: Towards an Adaptive, Collaborating Intrusion Detection System," Ph.D. dissertation, Vienna University of Technology, 2002.

[129] W. Granzer, "Security in Networked Building Automation Systems," Master's thesis, Vienna University of Technology, Institute of Computer Aided Automation, Automation Systems Group, Nov. 2005.

[130] D. Maynor and R. Graham, "SCADA Security and Terrorism: We're Not Crying Wolf," 2006, http://www.blackhat.com/presentations/bh-federal-06/BH-Fed-06-Maynor-Graham-up.pdf.

[131] W. Granzer, F. Praus, and W. Kastner, "Security in Building Automation Systems," *IEEE Trans. on Industrial Electronics*, vol. 56, 2009.

[132] W. Granzer, D. Lechner, F. Praus, and W. Kastner, "Securing IP Backbones in Building Automation Networks," in *Proc. of the 7th IEEE International Conference on Industrial Informatics (INDIN '09)*, Jul. 2009, pp. 410–415.

[133] C. Reinisch, W. Granzer, and W. Kastner, "Secure Vertical Integration for Building Automation Networks," in *Proc. of the 7th IEEE International Workshop on Factory Communication Systems (WFCS '08)*, May 2008, pp. 239–242.

[134] W. Granzer, C. Reinisch, and W. Kastner, "Key Set Management in Networked Building Automation Systems using Multiple Key Servers," in *Proc. of the 7th IEEE International Workshop on Factory Communication Systems (WFCS '08)*, May 2008, pp. 205–214.

[135] D. Lechner, W. Granzer, and W. Kastner, "Security for KNXnet/IP," in *Konnex Scientific Conference*, Nov. 2008.

[136] W. Granzer, C. Reinisch, and W. Kastner, "Denial-of-Service in Automation Systems," in *Proc. of the 13th IEEE Conference on Emerging Technologies and Factory Automation (ETFA '08)*, Sep. 2008, pp. 468–471.

[137] A. Varga, "The OMNeT++ Discrete Event Simulation System," in *Proc. of the 17th European Simulation Multiconference (ESM '03)*, Nov. 2003, pp. 171–180.

[138] ——, "Using the OMNeT++ discrete event simulation system in education," *IEEE Trans. on Education*, vol. 42, p. 11, Nov. 1999.

[139] W. Köhler, "Simulation of a KNX Network with EIBsec Protocol Extensions," Master's thesis, Vienna University of Technology, Institute of Computer Aided Automation, Automation Systems Group, Aug. 2008.

[140] W. Granzer and W. Kastner, "BACnet over KNX," in *Konnex Scientific Conference*, Nov. 2007.

[141] W. Granzer, W. Kastner, and C. Reinisch, "Gateway-free Integration of BACnet and KNX using Multi-Protocol Devices," in *Proc. of the 6th IEEE International Conference on Industrial Informatics (INDIN '08)*, Jul. 2008, pp. 973–978.

[142] C. Reinisch, W. Granzer, F. Praus, and W. Kastner, "Integration of Heterogeneous Building Automation Systems using Ontologies," in *Proc. of the 34th Annual Conference of the IEEE Industrial Electronics Society (IECON '08)*, Nov. 2008, pp. 2736–2741.

[143] W. Granzer and A. Treytl, "Security in Industrial Communication Systems," in *The Industrial Electronics Handbook, Second Edition, Part 4: Industrial Communication Systems*, J. D. Irwin and B. M. Wilamowski, Eds., 2010, to be published.

[144] A. Treytl, T. Sauter, and C. Schwaiger, "Security Measures for Industrial Fieldbus Systems – State of the Art and Solutions for IP-based Approaches," in *Proc. of the 5th IEEE International Workshop on Factory Communication Systems (WFCS '04)*, Sep. 2004, pp. 201–209.

[145] ——, "Security measures in automation systems – a practice-oriented approach," in *Proc. of the 10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA '05)*, Sep. 2005, pp. 847–855.

Curriculum Vitae

Personal Information:

Name:

Wolfgang Granzer

Address:

Neubaugasse 6/2

A-3363 Neufurth

Austria

Date-of-Birth:

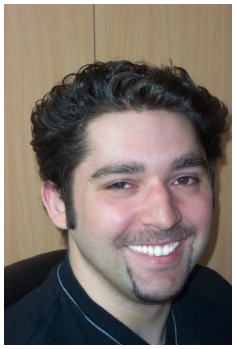
07.07.1980

E Mail:

wolfgang@granzer.org

Web site:

http://wolfgang.granzer.org



Work Experience:

Since 06/2007

Junior scientist (project assistant FWF) at TU Vienna,
Institute of Computer Aided Automation, Automation Systems Group.

09/2006 – 05/2007

Junior scientist at Research Unit of Integrated Sensor System,
Austrian Academy of Science.

Since 08/2006

System administrator and junior scientist at TU Vienna,
Institute of Computer Aided Automation, Automation Systems Group.

08/2003 – 09/2003; 08/2004 – 01/2005

Software engineer, Siemens AG Austria Department PSE CES4.

Teaching:

- Lecturer for: Wireless in Automation Systems (Lecture with Exercises),
  TU Vienna, Institute of Computer Aided Automation, Automation Systems Group.
- Lecturer for: Distributed Automation Systems (Lecture with Exercises),
  TU Vienna, Institute of Computer Aided Automation, Automation Systems Group.
- Lecturer for: Sensor/Actuator Systems (Laboratory Tutorial),
  TU Vienna, Institute of Computer Aided Automation, Automation Systems Group.
- Lecturer for: Grundlagen der Rechnerarchitektur u. Betriebssysteme (Exercise),
  FH Campus Vienna.

Education:

Since 2006

Working on Ph.D. in computer science at Vienna University of Technology.

2008

Mag. (MSocEcSc) in computer science management (with highest distinction) from Vienna University of Technology.

2000 – 2005

Dipl.-Ing. (M.Sc.) in computer science (with highest distinction) from Vienna University of Technology.

2000

Military Service.

1994 – 1999

Higher Technical School and Austrian Matura (with highest distinction) at HTL Waidhofen/Ybbs.

1992 – 1994

High school at Bundesrealgymnasium Waidhofen/Ybbs.

1990 – 1992

High school at Bundesrealgymnasium Vienna 16.

1986 – 1990

Elementary school Vienna 14.

Awards

- Best Paper Award at IEEE International Conference on Industrial Informatics (INDIN), July 2007.
- Konnex Scientific Award, November 2006.
- Best Paper Award at IEEE International Workshop on Factory Communication Systems (WFCS), June 2006.

Research Interests:

Security, Automation (especially Home and Building Automation), Embedded Systems, Distributed Systems, Simulation, HW/SW Codesign.

Additional Skills:

Foreign Languages:

English.

Hobbies:

Music, Mountain Climbing, Skiing, Billiard, Didgeridoo, Motorcycle.

Publications

- Wolfgang Granzer and Albert Treytl, *Security in Industrial Communication Systems*. In The Industrial Electronics Handbook, Second Edition, Part 4: Industrial Communication Systems, J. D. Irwin and B. M. Wilamowski, Eds., 2010, to be published.
- Wolfgang Granzer, Fritz Praus, and Wolfgang Kastner. *Security in Building Automation Systems*. IEEE Transactions on Industrial Electronics, 56, 2009.
- Wolfgang Granzer, Daniel Lechner, Fritz Praus, and Wolfgang Kastner. *Securing IP Backbones in Building Automation Networks*. In Proc. of the 7th IEEE International Conference on Industrial Informatics (INDIN '09), pages 410-415, July 2009.
- Fritz Praus, Wolfgang Granzer, and Wolfgang Kastner. *Enhanced Control Application Development in Building Automation*. In Proc. of the 7th IEEE International Conference on Industrial Informatics (INDIN '09), pages 390-395, July 2009.
- Daniel Lechner, Wolfgang Granzer, and Wolfgang Kastner. *Security for KNXnet/IP*. In Konnex Scientific Conference, November 2008.
- Christian Reinisch, Wolfgang Granzer, Fritz Praus, and Wolfgang Kastner. *Integration of Heterogeneous Building Automation Systems using Ontologies*. In Proc. of the 34th Annual Conference of the IEEE Industrial Electronics Society (IECON '08), pages 2736-2741, November 2008.
- Wolfgang Granzer, Christian Reinisch, and Wolfgang Kastner. *Denial-of-Service in Automation Systems*. In Proc. of the 13th IEEE Conference on Emerging Technologies and Factory Automation (ETFA '08), pages 468-471, September 2008.
- Wolfgang Granzer, Wolfgang Kastner, and Christian Reinisch. *Gateway-free Integration of BACnet and KNX using Multi-Protocol Devices*. In Proc. of the 6th IEEE International Conference on Industrial Informatics (INDIN '08), pages 973-978, July 2008. Best presentation award at INDIN '08.
- Wolfgang Granzer, Christian Reinisch, and Wolfgang Kastner. *Key Set Management in Networked Building Automation Systems using Multiple Key Servers*. In Proc. of the 7th IEEE International Workshop on Factory Communication Systems (WFCS '08), pages 205-214, May 2008.
- Christian Reinisch, Wolfgang Granzer, and Wolfgang Kastner. *Secure Vertical Integration for Building Automation Networks*. In Proc. of the 7th IEEE International Workshop on Factory Communication Systems (WFCS '08), pages 239-242, May 2008.
- Wolfgang Granzer and Wolfgang Kastner. *BACnet over KNX*. In Konnex Scientific Conference, November 2007.
- Fritz Praus, Wolfgang Granzer, Georg Gaderer, and Thilo Sauter. *A Simulation*

Framework for Fault-Tolerant Clock Synchronization in Industrial Automation Networks. In Proc. of the 12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2007), pages 1465-1472, September 2007.

- Christian Reinisch, Wolfgang Kastner, Georg Neugschwandtner, and Wolfgang Granzer. *Wireless Technologies in Home and Building Automation*. In Proc. of the 5th IEEE International Conference on Industrial Informatics, volume 1, pages 93-98, July 2007. Best paper award at INDIN '07.
- Wolfgang Granzer, Georg Neugschwandtner, and Wolfgang Kastner. *EIBsec: A Security Extension to KNX/EIB*. In Konnex Scientific Conference, November 2006.
- Christian Reinisch, Wolfgang Granzer, Fritz Praus, and Wolfgang Kastner. *Wireless Communication in KNX/EIB*. In Konnex Scientifc Conference, November 2006.
- Wolfgang Granzer, Wolfgang Kastner, Georg Neugschwandtner, and Fritz Praus. *Security in Networked Building Automation Systems*. In Proc. of the 6th IEEE International Workshop on Factory Communication Systems (WFCS '06), pages 283-292, June 2006. Best paper award of WFCS '06.
- Wolfgang Granzer, Wolfgang Kastner, Georg Neugschwandtner, and Fritz Praus. *A Modular Architecture for Building Automation Systems*. In Proc. of the 6th IEEE International Workshop on Factory Communication Systems (WFCS '06), pages 99-102, June 2006.