# Secure Cloud Computing in Legal Metrology

vorgelegt von
**Dipl.-Inform.**
**Alexander Oppermann**

an der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

**Promotionsausschuss:**

| | | |
|---|---|---|
| Vorsitzender: | Prof. Dr. Sebastian Möller | |
| Gutachter: | Prof. Dr. Jean-Pierre Seifert | |
| Gutachter: | Prof. Dr. Marian Margraf | Freie Universität Berlin |
| Gutachter: | Prof. Dr. Frederik Armknecht | Universität Mannheim |

Tag der wissenschaftlichen Aussprache: 13.03.2019

Berlin 2020

Security in Telecommunications

Technische Universität Berlin

## Eidestattliche Erklärung / Statutory Declaration

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed.

Berlin, 13. März 2019 Alexander Oppermann

# Abstract

In Europe, all measuring instruments under legal control have to pass a conformity assessment to prove compliance with the European Measuring Instrument Directive (MID). In Germany, the MID is regulated via the German Measures and Verification Act (MessEG) that imposes additional requirements for nationally regulated measuring instruments. According to estimations, about four to six percent of the gross national income in European countries is generated by transactions in Legal Metrology, which equals annual turnover of 500 billion Euros.

An ongoing transition can be observed from a local and concentrated measuring instrument to a distributed and interconnected one. In recent years Cloud Computing has been developed constantly, overcoming different challenges to mature in the fields of security, stability and reliability. However, a lack of trust and verifiability of outsourced computations are still major hindrances for employing Cloud Computing solutions in sensitive and security-conscious industries. These properties are challenging to protect by classical approaches.

In this thesis, a Secure Cloud Reference Architecture for measuring instruments is presented, addressing both requirements and roles of the Legal Metrology framework. Splitting a well contained measuring instrument into a distributed measuring system, creates new challenges to guarantee security and integrity of the measurements. Addressing these challenges, Fully Homomorphic Encryption (FHE) is employed to enable calculations on encrypted measurements. FHE suffers from time intensive and complex computations. However, by introducing multithreading to the employed FHE schema, a significant speed-up in all arithmetic operations is achieved. A secure communication protocol for encrypted data is presented to take account of integrity of encrypted measurements for data in transit. The feasibility of FHE is proven by applying it to real-world tariff-applications in the smart-meter domain. Furthermore, verification methods for the reference architecture are presented to classify the system behaviour. A risk analysis is performed to detect potential vulnerabilities, identify contemporary threats as well as possible countermeasures demonstrating the suitability of the proposed architecture for actual use. With this architectural approach all legal requirements are met and the needs of all stakeholder are addressed.

# Zusammenfassung

In Europa müssen sich alle Messgeräte, die dem gesetzlichen Messwesen unterliegen, einer Konformitätsprüfung unterziehen, um zu zeigen dass sie alle Anforderungen der europäischen Messgeräterichtlinie (MID) erfüllen. In Deutschland wird die MID durch das Mess- und Eichgesetz (MessEG) abgebildet, das zusätzlich noch Anforderungen für national geregelte Messgeräte enthält. Anhand von Schätzungen ist das gesetzliche Messwesen für vier bis sechs Prozent des Bruttoinlandsprodukt in Europa verantwortlich.

Ein anhaltender Trend von lokalen und isolierten Messgeräten hinzu verteilten und vernetzten Messgeräten kann beobachtet werden. In den letzten Jahren hat sich die Cloud-Computing-Technologie ständig weiterentwickelt und ist an verschiedenen Herausforderungen gewachsen, im Besonderen in den Bereichen Sicherheit, Stabilität und Zuverlässigkeit. Allerdings ist ein Mangel an Vertrauen und Verifizierbarkeit ausgelagerter Berechnungen immer noch ein großer Hinderungsgrund, um Cloud-Computing-Lösungen, besonders in sensiblen und sicherheitsbewussten Bereichen, einzusetzen. Diese Sicherheitseigenschaften sind herausfordernd für klassische Sicherheitsansätze umzusetzen, besonders in einer verteilten Umgebung.

In dieser Dissertation wird eine sichere Cloud-Referenzarchitektur für Messgeräte präsentiert, die die Anforderungen und Rollen im gesetzlichen Rahmenwerk abdeckt. Durch das Aufspalten herkömmlicher Messgeräte und das Abbilden auf eine verteilte Architektur werden neue Herausforderungen an die Sicherheit und Integrität des Messgeräts gestellt. Um diesen Herausforderungen angemessen zu entsprechen, kommt die vollständig homomorphe Verschlüsselung (FHE) zum Einsatz. Diese Verschlüsselung ermöglicht sichere Berechnungen auf verschlüsselten Daten, ohne die Preisgabe des Klartextes. FHE-Berechnungen sind zeitintensiv und sehr komplex. Im Rahmen dieser Arbeit wird das zugrundeliegende FHE-Schema durch Multithreading erweitert, das zu signifikanten Beschleunigungen in allen arithmetischen Operationen führt. Außerdem werden Ansätze zur Integrität der Daten auf dem Transportweg durch ein spezielles Kommunikationsprotokoll präsentiert. Zusätzlich werden Verifikationsmethoden vorgestellt, die die Integrität sichern und das Verhalten der gesamten Architektur bewerten. Abschließend wird eine Risikoanalyse, die speziell für Messgeräte im gesetzlichen Messen entwickelt wurde, auf der Grundlage der sicheren Referenzarchitektur durchgeführt. Dabei sollen potentielle Sicherheitslücken aufgedeckt und zeitgemäße Angriffsvektoren identifiziert werden. Mit diesem architektonischen Ansatz werden alle gesetzlichen Anforderungen erfüllt und die Bedürfnisse aller beteiligten Parteien berücksichtigt.

# Publication List

The primary results of this work have been presented in the following peer-reviewed publications:

- **A. Oppermann**, M. Esche, F. Thiel, J.-P. Seifert, Secure Cloud Computing: Risk Analysis for Secure Cloud Reference Architecture in Legal Metrology, Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS), IEEE, 2018, pp. 593–602, DOI: 10.15439/2018F226 (Chapter 6)
- **A. Oppermann**, F. Grasso Toro, F. Thiel, J.-P. Seifert, Anomaly Detection Approaches for Secure Cloud Reference Architectures in Legal Metrology. In Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER 2018), pages 549-556, DOI: 10.5220/0006777105490556 (Chapter 2)
- **A. Oppermann**, F. Grasso Toro, F. Thiel, J.-P. Seifert, Secure Cloud Computing: Reference Architecture for Measuring Instrument under Legal Control. Journal Security and Privacy 2018;e18. DOI: 10.1002/spy2.18 (Chapter 1, 2, 3, 4, 5 and 7)
- **A. Oppermann**, F. Grasso Toro, A. Yurchenko, J.-P.Seifert, Secure Cloud Computing: Communication Protocol for Multithreaded Fully Homomorphic Encryption for Remote Data Processing in IEEE International Symposium on Parallel and Distributed Processing with Applications (IEEE ISPA 2017) (pp. 503-510), DOI: 10.1109/ISPA/IUCC.2017.00084 (Chapter 2 and 3)
- **A. Oppermann**, A. Yurchenko, M .Esche, J.-P. Seifert, Secure Cloud Computing: Multithreaded Fully Homomorphic Encryption for Legal Metrology, in International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments (ISDDC 2017) 2017 Oct 25 (pp. 35-54), DOI: 10.1007/978-3-319-69155-8, (Best Paper Award), (Chapter 3, 4 and 5)
- **A. Oppermann**, J.-P. Seifert, F. Thiel, Distributed Metrological Sensors managed by a secure Cloud-Infrastructure, 18. GMA/ITG Fachtagung, Sensoren und Messsysteme 2016, Nürnberg, 10.-11. Mai, (2016), ISBN: 978-3-9816876-0-6, DOI: 10.5162/sensoren2016/P7.5 (Chapter 1 and 2)
- **A. Oppermann**, J.-P. Seifert, F. Thiel, Secure Cloud Reference Architectures for Measuring Instruments under Legal Control, in Proceedings of the 6th International Conference on Cloud Computing and Services Science (CLOSER 2016) - Volume 1, pages 289-294, 23.-25. April, (2016), DOI: 10.5220/0005909902890294 (Chapter 1 and 2)

# Contents

# List of Figures

# List of Tables

# List of Abbreviation

| | |
|---|---|
| *ActiveMQ* | Active Message Queue |
| *AD* | Anomaly Detection |
| *AI* | Artificial Intelligence |
| *API* | Application Programming Interface |
| *ARP* | Adress Resolution Protocol |
| *AtPt* | Attack Probability Tree |
| *BSI* | Bundesamt für Sicherheit in der Informationstechnik |
| *CA* | Certificate Authority |
| *CERT* | Computer Emergency Response Team |
| *CLA* | Carry Look Ahead Adder |
| *CLI* | Command Line Interface |
| *CPU* | central Processing Unit |
| *CSA* | Carry Select Adder |
| *CSP* | Cloud Service Provider |
| *CVE* | Common Vulnerabilities and Exposures |
| *CVP* | Closest Vector Problem |
| *DDoS* | Distributed Denial of Service Attacks |
| *EFTA* | European Free Trade Association |
| *ENISA* | European Union Agency for Network and Information Security |
| *FHE* | Fully Homomorphic Encryption |
| *FLINT* | Fast Library for Number Theory |
| *GCHQ* | British Government Communications Headquarters |
| *GMP* | GNU Multiple Precision Arithmetic Library |
| *HE* | Homomorphic Encryption |
| *HSM* | Hardware Security Modules |
| *IaaS* | Infrastructure as a Service |

| | |
|---|---|
| *ID* | Identification/Identitiy/Identifier |
| *IoT* | Internet of Things |
| *I/O* | Input/Output |
| *LHE* | Levelled Homomorphic Encryption |
| *LM* | Legal Metrology |
| *LSIM* | LibScarab extended for Integer Arithmetic and Multithreading |
| *MessEG* | Mess- und Eichgesetz |
| *MessEV* | Mess- und Eichverordnung |
| *MID* | Measuring Instrument Directive |
| *MITM* | Man-In-The-Middle attack |
| *ML* | Machine Learning |
| *MPFR* | Multiple-Precision Binary Floating-Point Library |
| *NIC* | Network Interface Card |
| *NIST* | National Institute of Standards and Technology |
| *OIML* | International Organization of Legal Metrology |
| *OpenMP* | Open Multi-Processing |
| *OpenSSL* | Open Secure Socket Layer |
| *OSI* | Open Systems Interconnection model |
| *PaaS* | Platform as a Service |
| *PCP* | Performance Co-Pilot |
| *PKI* | Public Key Infrastructure |
| *PPT* | Probabilistic Polynomial Time |
| *PTB* | Physikalisch-Technische Bundesanstalt |
| *PThread* | POSIX Threads |
| *RCA* | Ripple Carry Adder |
| *SaaS* | Software as a Service |
| *SELinux* | Security Enhanced Linux |
| *SHE* | Somewhat Homomorphic Encryption |
| *SLA* | Service Level Agreements |
| *SME* | Small and Middle size Enterprises |
| *SMGW* | Smart Meter Gateway |
| *SOA* | Service Oriented Architectures |

| | |
|---|---|
| *SSH* | Secure Shell |
| *SVP* | Shortest Vector Problem |
| *TAF* | Tariff Application Scenarios |
| *TLS* | Transport Layer Security |
| *TOE* | Target of Evaluation |
| *TR* | Technische Richtlinie (Technical Requirement Document) |
| *USA* | United States of America |
| *VM* | Virtual Machine |
| *WAN* | Wide Area Network |
| *WELMEC* | (Western) European Legal Metrology Cooperation |
| *WG* | WELMEC Working Groups |

# 1 Introduction

*"They don't call it the Internet anymore, they call it Cloud Computing. I'm no longer resisting the name. Call it what you want."*

*—Larry Ellison, CEO of Oracle*

IN THE LAST YEARS, an exponential growth has been observable in interconnected devices, such as computers, smart phones and numerous embedded devices. The amount of these interconnected devices will increase immensely rather than decline in the foreseeable future. In the field of Legal Metrology, are already 850 million measuring instruments employed throughout the European Union (EU) [1]. These measuring instruments are responsible for an annual turnover of four to six percent or 500 billion Euros of the gross national income of the EU. Because of their significant financial contribution to the economy, a fundamental purpose is to establish trust in the market between all stakeholders. To be able to fulfill this obligation, all measuring instruments that are subject to legal control have to pass a conformity assessment, in order to prove that all requirements of the European Measuring Instrument Directive (MID) are met [2]. In Germany, the MID is regulated via the German Measures and Verification Act (MessEG) which further imposes requirements for national regulated measuring instruments. Additional technical support to fulfill the MID is provided by the WELMEC Software Guide [3].

In this increasing interconnected world, a significant role is played by Cloud Computing. It is a key to enable technologies such as Internet of Things (IoT), Big Data and machine learning (ML). By Centralizing the distributed resources, it paves the way for a service-oriented society. Furthermore, measuring instruments will be reduced in size and costs, combining virtualization and externalization. Consequently, it is assumed that future measuring instruments consist only of a basic sensor and a communication unit for an Internet connection in the field [4]. The remainder of the measuring instrument with metrological relevance, like data processing and storing units will be centralized in a secure Cloud Computing solution.

Requests are increasing from manufacturers of measuring instruments, Notified Bodies and market surveillance authorities for Cloud Computing solutions that are in conformity with the law. First promising approaches to integrate Cloud Computing into existing business models or products are brought forward to Notified Bodies, such as Physikalisch-Technische Bundesanstalt (PTB) in Germany. The focus lies on the advantages of virtualization, setting up infrastructure quickly, to scale it accordingly and to map these requirements to industry processes, so that modern and cost-efficient solutions can emerge.

## 1.1 Problem Statement

Despite the constant improvements in the field of Cloud Computing over the last years, a lack of trust and verifiability of outsourced computations are still major challenges for employing Cloud Computing solutions in sensitive and security-conscious industries. In these fields confidentiality, integrity and availability of services are very important security properties. Furthermore, in respect to Legal Metrology reliability, reproducibility and repeatability of measurements have to to be ensured. These properties and assets are challenging to protect by classical approaches, especially in a distributed environment. In the context of Cloud Computing, locality becomes a legal issue, because national bodies are operating within national boundaries while Cloud Computing solutions can operate technically on a global scale with virtually no borders. This also addresses off- and on-premise solutions for Cloud Computing, which restricts initially the solution base and thus impedes taking full advantage of Cloud Computing. Often, classical security measures, such as static encryption, lead to restricted Cloud Computing solutions in terms of flexibility, agility and usability. Cloud Computing will introduce a new role of a Cloud Service Provider. This role has to be further determined in the legal framework.

## 1.2 Contributions

In this thesis, the development of a secure cloud reference architecture for measuring instruments is presented, addressing both requirements and roles of the Legal Metrology framework. In a bottom up approach, each layer of the Cloud is addressed and carefully tested against the essential requirements in Legal Metrology. Nowadays, the majority of the measuring instruments in the field are local and concentrated. Splitting a well contained measuring instrument into a distributed measuring system, creates new challenges to guarantee security and integrity of the measurements. Addressing these challenges, Fully Homomorphic Encryption (FHE) is improved, extended and implemented to enable calculations on encrypted measurements. FHE suffers from time intensive and complex computations. However, by introducing multithreading to the employed FHE schema, a significant speed-up in all arithmetic operations is achieved. Furthermore, the operation space is extended to 32 bit and 64 bit numbers for all homomorphic operations. The key generation and recryption routines are accelerated by a high factor, through modernizing and optimizing the implemented schema. The feasibility of FHE is proven by applying it to real-world tariff-applications in the smart-meter domain. This is achieved by adding simple comparison, zero testing and multithreading for homomorphic operations.

Also, a secure communication protocol for encrypted data is presented to address the demand of integrity of encrypted measurements for data in transit. Furthermore, verification methods for the reference architecture are presented such as a continuous monitoring approach and a verification monitor to classify the system behaviour. A risk analysis is performed to detect potential vulnerabilities and identify contemporary threats. A tailored risk assessment method for measuring instruments under legal control is applied to objectify the derived probability score for identified threats. By objectifying the resulting risk score the comparability of risk assessments can be increased among Notified Bodies in Europe. This secure cloud reference architecture fulfills the highest security requirements of the European regulations and needs of all stakeholders.

## 1.3 Outline

The remainder of this thesis is organized as follows:

- Chapter 2 gives an overview of the legal requirements and the technical preliminaries;
- Chapter 3 presents the secure Cloud Computing architecture in a bottom up approach including a monitoring approach for the secure cloud architecture.
- Chapter 4 introduces homomorphic encryption and highlights the contribution to this field.
- Chapter 5 applies the prototype to real world application scenarios and

presents results.

- Chapter 6 presents the risk assessment of the secure reference architecture with an attack probability tree especially tailored for Legal Metrology.
- Chapter 7 provides conclusions and further work.

# 2

# Technical Background, Provisions and Assumptions

*"I don't need a hard disk in my computer if I can get to the server faster... carrying around these non-connected computers is byzantine by comparison."*

*—Steve Jobs, Co-founder, CEO and Chairman of Apple Inc.*

THIS CHAPTER PROVIDES an introduction to the field of Legal Metrology. The most relevant regulations are highlighted, such as the European measuring instrument directive (MID), WELMEC and OIML guide. All stakeholders in Legal Metrology will be summed up and a new role of the Cloud Service Provider will be introduced and discussed. Furthermore, an overview of the legal requirements for a Cloud Computing architecture will be given. The general provisions and technical background of Cloud Computing is thoroughly described and explained.

## 2.1 Legal Metrology

Legal Metrology covers a wide range of measuring instruments from commercial or administrative purposes to measurements in the public interest. In Germany over 100 million legally relevant meters are in use (see [5]). The vast majority of them are employed for business purposes, commodity meters in the field of water, gas, electricity or heat. Furthermore, common applications for meters are petrol pumps or scales in e.g. super markets. The traffic system, for example, requires meters for speed or alcohol meters in a large amount, in order to guarantee safety. All of these applications have in common, that neither the user nor the affected person can check the validity of the determined result. Instead, they rely on the accuracy of the measurement as well as the official calibration of these measuring instrument. The key role of Legal Metrology is to assure the correctness of measurements and further to protect public trust in them. Additionally, Legal Metrology fulfills the purpose to ensure the functioning of the economic system as well as to protect the consumer at the same time.

The *International Organization of Legal Metrology* (OIML) was founded with the aim to harmonize regulations across national boundaries worldwide and to avoid trade barriers due to legal requirements. The document OIML D 31 [6] focuses especially on software requirements for legal measuring instruments (see [7]).

*WELMEC* is the European committee responsible for harmonizing legal regulations in the area of Legal Metrology. The committee publishes guides and supports notified bodies (public or private organized departments to certify measurements devices) throughout Europe and manufacturers alike, which implement the Measuring Instruments Directive (MID) (see Section 2.2.3 and [7], [8]).

## 2.2 Regulations in Legal Metrology

In the following subsections a comprehensive overview of the paramount regulations in Legal Metrology are given. Furthermore, the stakeholders are introduced, and a new role of the Cloud Service Provider is discussed. Lastly the concept of the reference architecture is presented that serves as a basis for the developed secure Cloud Computing architecture.

### 2.2.1 Measuring Instrument Directive

The MID comprises ten types of measuring instruments that are of special interest and importance to the economy due to their wide-spread or cross-border use. These are: water meters, gas meters and volume conversion devices, active electrical energy meters, heat meters, measuring systems for the continuous and dynamic measurement of quantities of liquids other than water, automatic weighing instruments, taximeters, material measures, dimensional measuring instruments, and exhaust gas analyzers. In the annex of the directive there are definitions, fault tolerances and specific requirements outlined for each type of the prior mentioned measurement instruments.

Before putting a new measuring instrument on the market, the manufacturer has to declare the conformity with the MID based on the assessment by a Notified Body in Europe (see Figure 2.2). The *Physikalisch-Technische Bundesanstalt (PTB)* is a Notified Body in Germany. Aside from that role, the PTB is the German national metrology institute and acts as an interface between scientific research and economic interests. Furthermore, the PTB is responsible for technical expertise related to measuring instruments, conformity assessment, monitoring of product-related quality assurance systems and European regulations. Given the scope of these responsibilities, it is crucial that Notified Bodies are independent and neutral in order to be able to fulfill duties impartially.

### 2.2.2 Essential Software Requirements from the MID

Annex I of the MID defines a set of essential requirements, which all measuring instruments covered by the directive need to comply with. These requirements cover physical conditions, under which the instrument has to function correctly, accuracy requirements for the measuring result as well as requirements concerning the prevention of manipulation. The requirements from the latter category may be interpreted from an information security angle as a list of assets to be protected [9]. As an example, essential requirement 8.4 will be briefly examined here. It states, "Measurement data, software that is critical for measurement characteristics and metrologically important parameters stored or transmitted shall be adequately protected against accidental or intentional corruption." The assets defined in the requirement are transmitted and stored parameters, measurement data as well as the software of the instrument itself. As all of these are to be protected against intentional and accidental corruption, at least their integrity and authenticity need to be guaranteed. In the context of this thesis, the main focus will be on the integrity of both the software and the measurement data. Availability of the software is not required by the MID as no false measurement data can be generated if the instrument is out of order.

Figure 2.1 displays platform requirements and additionally the most important MID software requirements are listed below:

- Reproducibility of measurement results must be ensured, even if different users make use of it.
- Durability of the measuring instrument's software must be ensured over a period of time. The measuring instrument's design shall decrease the effect of a defect as far as possible that would lead to an inaccurate measurement result, unless the presence of such a defect is obvious.

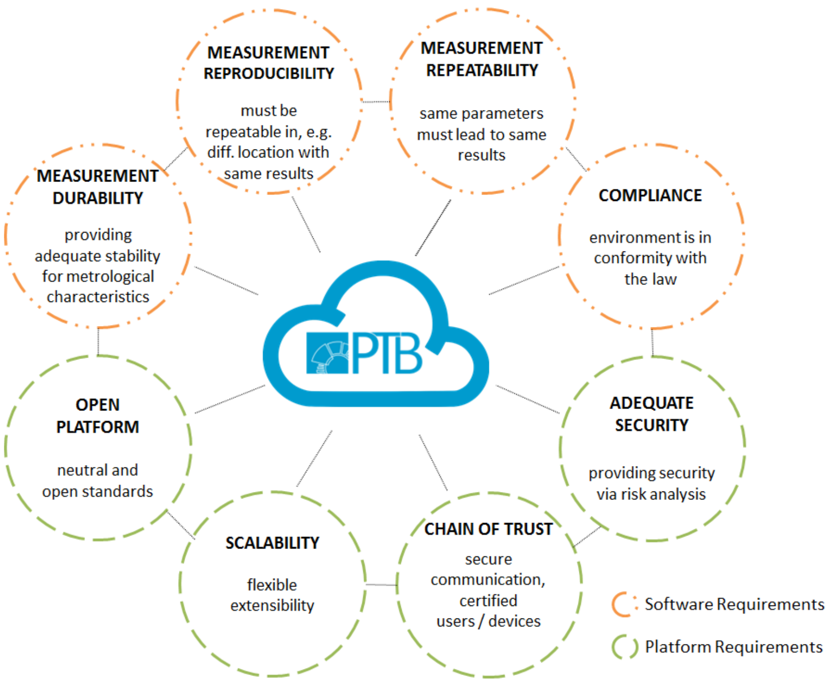

MEASUREMENT REPRODUCIBILITY
must be repeatable in, e.g. diff. location with same results
MEASUREMENT REPEATABILITY
same parameters must lead to same results
MEASUREMENT DURABILITY
providing adequate stability for metrological characteristics
COMPLIANCE
environment is in conformity with the law
PTB
OPEN PLATFORM
neutral and open standards
ADEQUATE SECURITY
providing security via risk analysis
SCALABILITY
flexible extensibility
CHAIN OF TRUST
secure communication, certified users / devices
Software Requirements
Platform Requirements

**Figure 2.1:** Overview of the different requirements in Legal Metrology for measuring systems. *Orange* are the demands for handling measurement data. *Green* are the platform requirements

- The characteristics of a measuring instrument shall minimize the fraudulent use, and possibilities for unintentional misuse. Theses claims request that the effect of manipulations and defects are reduced to a feasible minimum.
- After the instrument has been placed on the market and put into use, the measuring instrument's design shall allow the control of the measuring tasks. Also, software identification shall be easily provided by the measuring instrument.

### 2.2.3 WELMEC

The European Free Trade Association (EFTA) and national authorities send members to WELMEC Committee. At the moment 37 countries are part of the WELMEC Committee. The committee has established eight WELMEC Working Groups (WG) and WG7 is in charge of software matters and the *WELMEC 7.2 Software Guide*. The current issue is WELMEC 7.2 from 2015 [3].

The WELMEC 7.2 Software Guide gives an overview on software security with special focus on measuring instruments. It helps manufacturers and Notified Bodies alike by providing examples and rules on how to achieve software security and guarantees, if followed, compliance with the software related part required by the MID. Furthermore, the WELMEC 7.2 Software Guide defines six risk classes in ascending order A-F and distinguishes from low to high the protection level, examination level and conformity level of software. This means the risk class A has the lowest security claims for software examination and class F the highest security demands vice versa. For this distributed reference architecture class F was chosen, in order to comply with the highest software security requirements available in the domain of legal metrology. Risk class F has to fulfill the following list:

- Software protection against intentional changes with sophisticated software tools.
- Thorough software examination including review of the submitted source code.
- The approved software of the type examination has to be reasonably identical with the later available measuring instrument on the market.

The Welmec 7.2 Software Guide categorizes all software modules as legally relevant that make a contribution or can influence the measurement result. This comprises software modules and auxiliary functions that are able to display measuring results, read and write legal relevant data, identifying software modules, protecting, logging, transferring, receiving, storing and executing data. These requirements are fundamental for the later design of the architectural approach of a distributed measuring instrument in Chapter 3.

Traditionally, Welmec 7.2 Software Guide distinguishes between purpose and universal measuring instruments type P and U. For this reference architecture a type U instrument and its requirements fits best, since it is not a micro-controller architecture, but rather a universal instrument with a operating system that is connected to an open network. Additionally, four extensions and its requirements are applied to the reference architectural design [3]:

- Extension L: Long-term Storage of Measurement Data
- Extension T: Transmissionof Measurement Data via Communication Networks
- Extension S: Software Separation
- Extension D: Download of Legally Relevant Software

The Welmec 7.2 Software Guide advises to design software for secure measuring instruments with isolation of the legally relevant software from the legally non-relevant one. Likewise, the non-relevant part may not influence the legally relevant one, this may be realized by means of protective interfaces. This approach is consequently enforced for each tier of the cloud reference architecture. On the IaaS tier subnetting and virtual machines are used to guarantee a low level separation. On PaaS tier, microservices are used to enforce the same separation, that can be consequently transferred to the SaaS tier (see Sections 2.3.2 and 3.1) Moreover, if the software is built modular it will ease the update process for manufacturers and Notified Bodies.

### 2.2.4 The roles of manufacturer, Notified Body, user and market surveillance

In the text of the MID, certain roles are defined for players in the field of Legal Metrology. Firstly, the manufacturer of an instrument is responsible for putting instruments on the market and into use that comply fully with the directive. To assure conformity with the MID, the manufacturer submits a prototype of the instrument to a so-called Notified Body for conformity assessment module B.

If the prototype is in conformance with the requirements, the Notified Body issues a certificate accordingly. The manufacturer will then sell an instrument with the same properties as the prototype to the user together with a declaration of conformity. A market surveillance authority is subsequently tasked with supervising the use of the instrument. In regular intervals, the authority will also reverify the instrument to ensure that it still performs within the parameters set by the MID and the certificate issued by the Notified Body. In this context, only the market surveillance authority and the Notified Body will be considered to be trustworthy as they are under constant supervision by the respective EU member states. All other parties (manufacturer, user of the instrument and the user's customer) can be deem as potential attackers (see Figure 2.2).

### 2.2.5 The role of Cloud Service Provider

In Legal Metrology four important roles are established: the Notified Body, the manufacturer of the measuring instrument, the user of the measuring instrument and the Market- and User Surveillance (see Section 2.2.4). By establishing Cloud Computing as a technology a new role has to be determined, that of the Cloud Service Provider. As pointed out this role does not exist yet in Legal Metrology and how to deal with it or where to place that role has not been conclusively decided (see Figure 2.2). Either the role of the Cloud Service will be filled by the manufacturer as part of his responsibilities and

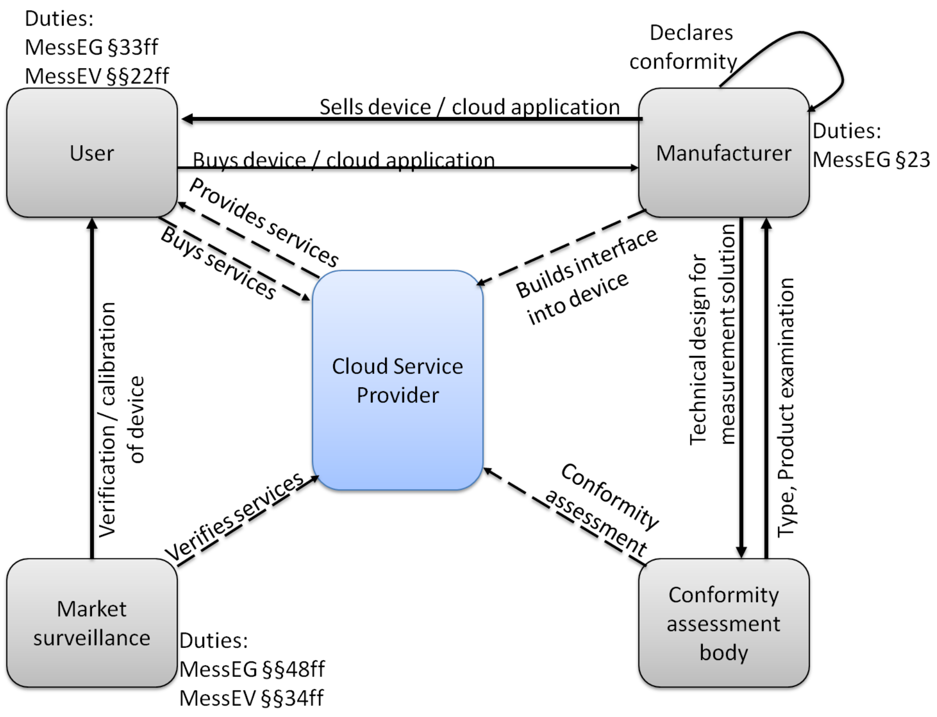


**Figure 2.2:** Overview of the different actors their roles, schedule of responsibilities and duties in Legal Metrology. The cloud service provider role has to be determined.

business case, so that he can provide an “on premise" solution for the customer, or the User of the measuring instrument fills the role of the Cloud Service Provider. A third option would be that either the manufacturer or the user will delegate the responsibilities to a third party provider. For the market surveillance and notified body, the legal liabilities will still stay with the manufacturer or the user respectively but under private law the Cloud Service Provider can held responsible via Service Level Agreements (SLA).

### 2.2.6 Reference Architecture

Just as all other fields, Legal Metrology is undergoing fundamental changes. The world and its technologies are changing at a pace that is incomparable to prior decades. In order to cope with new and fundamental technologies, knowledge is key. The basic idea behind creating general reference architectures is to gain the knowledge of implementing and utilizing key technologies that promise to enable the field of Legal Metrology to keep up with the state-of the-art.

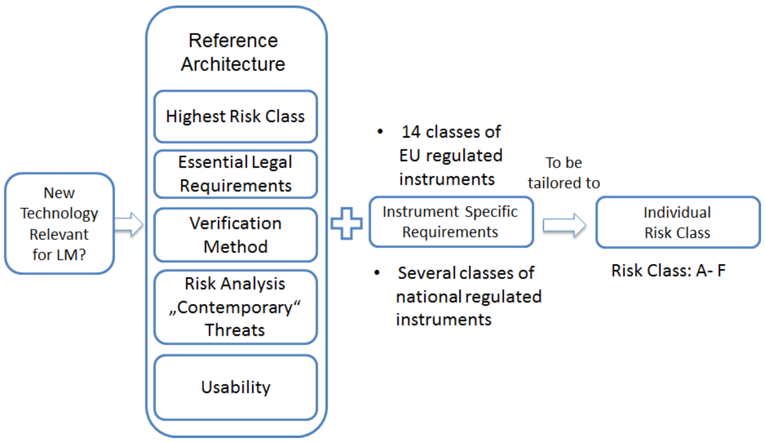

New Technology Relevant for LM?
Reference Architecture
Highest Risk Class
Essential Legal Requirements
Verification Method
Risk Analysis „Contemporary“ Threats
Usability
14 classes of EU regulated instruments
Instrument Specific Requirements
Several classes of national regulated instruments
To be tailored to
Individual Risk Class
Risk Class: A- F

**Figure 2.3:** Overview of the requirements for a reference architecture in Legal Metrology. Furthermore, measuring instrument specific requirements and the demand to be adaptable to different risk classes are displayed.

The PTB, as a Notified Body, builds up this important technological knowledge and shares it among all stakeholders in Legal Metrology to provide confidence and trust among all stakeholders. While creating a reference architecture using a new technology, in this particular case it is Cloud Computing, the technological knowledge is gained, and an architecture is built specifically tailored for the field of Legal Metrology. The processes of conformity assessment and market surveillance can be streamlined through the research of new technologies applied to prototyping architectures, risk assessment and remote verification of measuring instruments in the field see 3.3.1.

In order to prove that new technologies are important and viable to the field of Legal Metrology, the technology has to fulfill a number of requirements (see Figure 2.3). A successful reference architecture has to fulfill the essential requirements of the European directives [2], [10] and guidelines [3] (see Section 2.2.2 and Section 2.2.3).

Furthermore, it has to provide an applicable verification method for the market and user surveillance, since they do not have the time nor the financial resources to dig deep into the used technology stack. The verification method will be introduced in Section 3.3.1 and supports the work of the market and user surveillance in the field. The verification method and its result has to be understood by an average user of the market and user surveillance. Furthermore, the verification method is built deliberatly simple to encourage future manufacturer to build in such a feature in their products, aginst intentional and unintentional changes of their validated measuring software. It supports a simple integrity check in the field and raises the bar of the state-of-the-art for such instruments in the field. Also, a risk analysis of contemporary threats and attacks has to be carried out and evaluated as required by the MID [9]. In Chapter 6 such a risk analysis is carried out and is based on an attacker model with an insider and outsider attack. The reference architecture is designed for the highest risk class but is adaptable and can be scaled down according to the needs of the applied measuring instrument.

The reference architecture is deliberately built very general, in order to able to adapt it to the 14 classes of European and several national regulated measuring instruments, which range from gas, power, heat to water, oil and speed meters.

## 2.3 Cloud Computing

The concept of centralized computation has been introduced in the 1960s by John McCarthy with timesharing on mainframe computers. The main idea is that a group of people can increase the computing workload more efficiently and stabilize it than an individual alone, since idle times will be reduced enormously. In 1969 the Advanced Research Projects Agency Network (ARPANET) was introduced and enabled an early computer network, that is nowadays better known as the INTERNET. In the 1970s the first software Virtualization was introduced and early concepts of emulating complete computers, Virtual Machines, were researched. This concept introduced a massive paradigm shift in computer science. In the early 1990s a forecast for distributed computing was made.

Prof. Ramnath Chellappa coined the term Cloud Computing in 1997 and defined it as “a computing paradigm where the boundaries of computing will be determined by economic rationale rather than technical limits alone”[11]. At the end of the 1990s and in the early 2000s outsourcing IT-related tasks were very popular and led to the first enterprise related applications that were delivered via the Internet. This development fostered a research in web-based services and led to service-oriented architectures (SOA). The advent of Cloud Computing took place in 2009. Since then a rapid development and spread throughout the whole market can be observed. Nowadays, Cloud Computing is being viewed as an enabling technology for new developments like Big Data, the Internet of Things (IoT) and the evaluation of massive data by artificial intelligence (AI) and machine learning (ML).

Keeping Chellapa’s definition in mind, it held true that Cloud Computing is driven and sharpened by economic and marketing rationales. Thus, it is hard to find a precise and comprehensive definition for the technologies comprised by Cloud Computing. Mostly, it is just a description of its features.

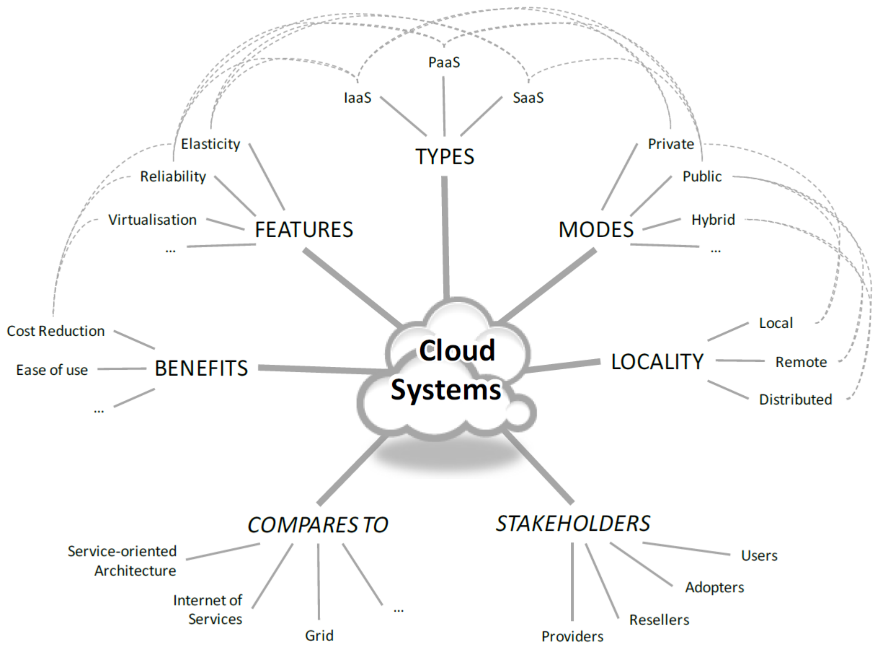


**Figure 2.4:** Overview of Cloud Computing and its main characteristics. [12]

Figure 2.4 gives a non-exhaustive overview of the areas touched by Cloud Computing. Furthermore, this Section describes the essential characteristics, service and deployment models of Cloud Computing that are defined by the National Institute of Standards and Technology (NIST).

## 2.3.1 Essential Requirements

In the following paragraphs the NIST Cloud Computing definition and their Cloud Computing concept with its five essential characteristics [13] will be briefly explained and mentioned, in order to grasp a better understanding of a cloud system:

*On-demand self-service.* A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

*Broad network access.* Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

*Resource pooling.* The provider's computing resources are pooled to serve multiple consumers using a multi tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, and network bandwidth.

*Rapid Elasticity.* Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly out ward and in ward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

*Measured service.* Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

### 2.3.2 Cloud Service Models

Cloud Computing is usually separated into three tiers: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). These tiers are distinguished by their functionality scope that will be covered by a cloud service provider (CSP) and the liabilities for the customer. In Figure 2.5 the three tiers are displayed side-by-side while the functionality is divided in each column. The responsibility scope is marked by the red border and tells the difference between the customer's and provider's responsibility. The more functionality is outsourced to the cloud service provider, the less freedom the customer has to manage and handle their application. However, the liabilities for security and updates are in the realm of the cloud service provider with professional expertise at its disposal.

The following paragraphs define the three Cloud Service Models (tiers) according to NIST [13] and give a precise overview:

*Infrastructure as a Service (IaaS).* The capability provided to the

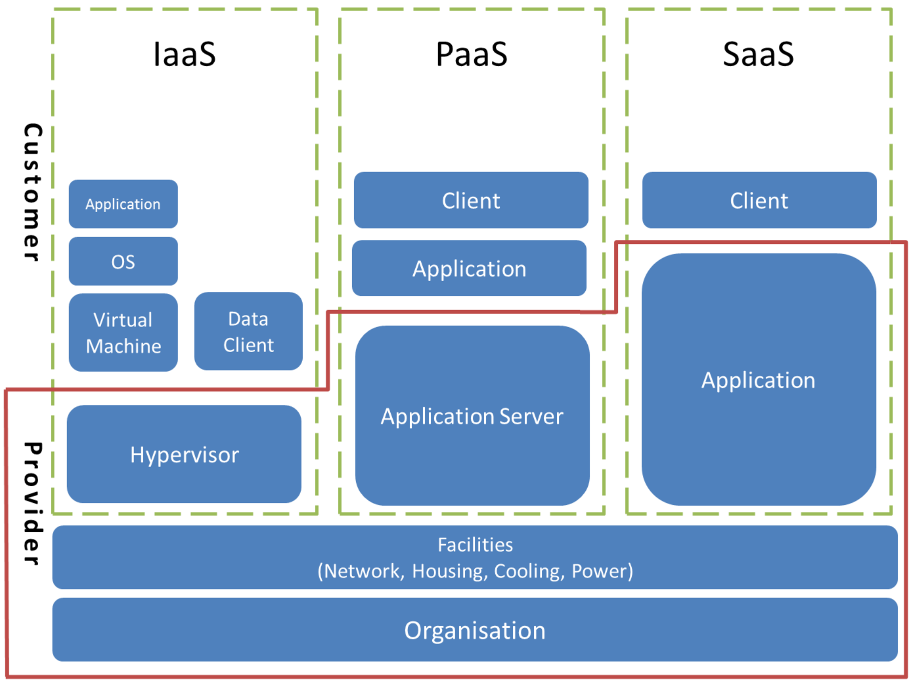


**Figure 2.5:** Overview of the different Cloud service models

consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

*Platform as a Service (PaaS).* The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application hosting environment.

*Software as a Service (SaaS).* The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

### 2.3.3 Cloud Deployments Models

The following paragraphs define the four Cloud Deployment Models according to NIST [13]. They differ with the grade of in house vs out sourced services and also the level of customization of services, that is buying standard services versus developing them especially for the use case. In Figure 2.6 this relationship is visualized.

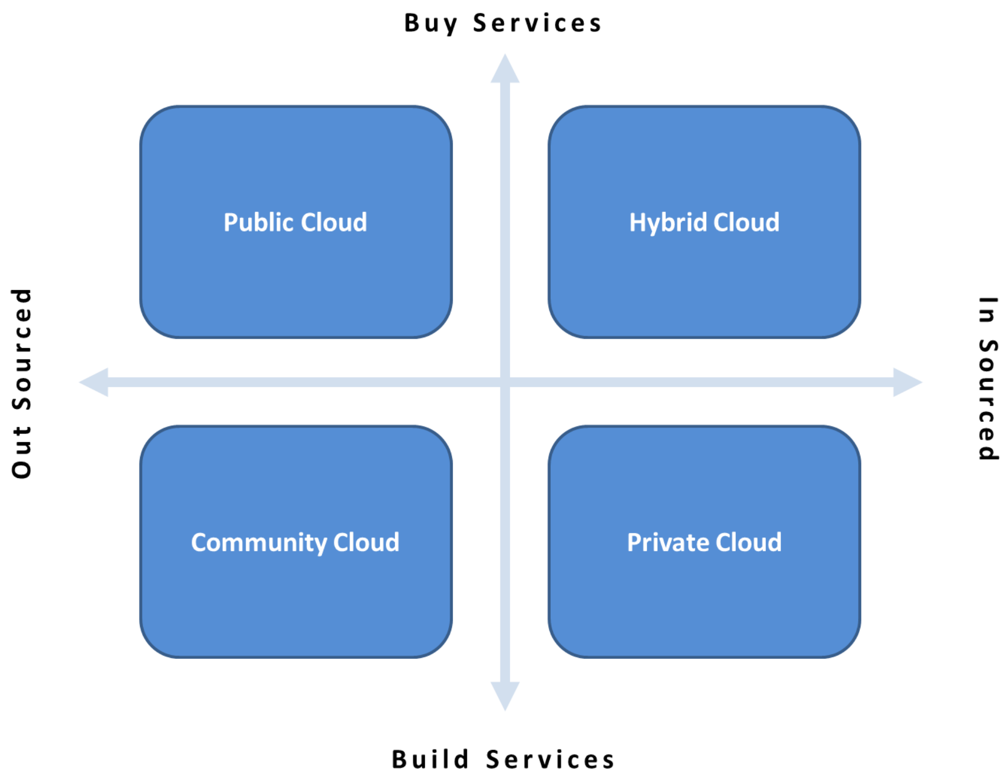


**Figure 2.6:** Overview of the different Cloud deployment models

*Private cloud.* The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

*Community cloud.* The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

*Public cloud.* The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

*Hybrid cloud.* The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability(e.g., cloud bursting for load balancing between clouds).

### 2.3.4 Advantages of Cloud Computing

The main advantages for Cloud Computing stem from externalizing costs, which frees financial resources especially for small and medium sized enterprises (SME) and keeps the up-front investment considerably low. Renting computing power through a cloud service provider reduces the investment costs in hardware. The operational costs are significantly reduced for maintaining hardware and keeping it state-of the-art, i.e. updating it on a regular basis. Furthermore, the externalization frees enterprises from labor costs and competition for rare IT-experts. The security costs are already included for maintenance and defense against common threats, like distributed denial of service attacks (DDoS). Depending on the service and deployment models, the security responsibilities reside with the CSP.

Increasing productivity through simplified deployments involves automatically a better time to market. This leads indirectly to a higher competitiveness for SME. Depending on the success and access rate, bursts of activities can be easily caught through auto-scaling the application thus increasing availability of data and services. In addition, the resilience factor is exorbitant higher by incorporating a scalable architecture and developing the own application towards it.

For a more recent holistic view of Cloud Computing Architectures and its advantages see Kratzke [14]. The most significant advantages are summarized in the following list:

- reduced operational costs
- reduced labor costs
- no bounded financial resources in hardware
- better time to market
- simplified deployment
- increased development speed
- simplified operation and auto-scaling based on event stimulus
- high availability of data
- backup and redundancy
- high resilience factor

### 2.3.5 Disadvantages of Cloud Computing

While cost saving is a huge factor for Cloud Computing, it can also lead into a trap. If the transition of classical in house hosted application to an external CSP is not carefully planned and the workflows are adapted to it, the costs will most likely explode (see [15]). A Cloud Application still needs a good and rigorous management by IT-experts. For example, instances that are not used anymore have to be closed.

Through centralization of services and data, the attack surface and the attractiveness for attackers are increased. Furthermore, each layer of abstraction introduces new attack surfaces and risks of configuration errors. This means, technology itself cannot replace proper planning, a reasonable technology stack and a careful role out of an application.

Moreover, moving applications into a centralized hub, can introduce latencies, for example in network, in functions and runtime. The industry offers solutions to reduce latencies by e.g. introducing content delivery networks (CDN) and thus placing relevant information closer to the consumer (Edge computing).

If a SME develops its application against an API of a CSP, it makes itself dependent on the API of this certain CSP, since APIs are not standardized across different CSPs. This phenomenon is called vendor lock in and makes a transition to a different CSP very hard. Furthermore, developing a distributed and scalable application increases complexity and exacerbates integration testing and maintenance.

Kratzke [14] gives a comprehensive overview of the difficulties using Cloud Computing. The most significant disadvantages are summarized in the following list:

- cost explosion through lack of suitable adaption of workflows and applications to the cloud paradigm
- maximum function runtime is limited through cost-benefit-ratio
- increased attack surfaces with each virtual machine
- latencies (network, functions, runtime) must be considered
- distributed system, parts of application logic is shifted out of the realm of the service provider to the client
- vendor lock in, no standardized APIs
- increased complexity in integration testing
- distributed systems are more complex to maintain and test

To avoid most of these drawbacks, a careful architectural planning phase is recommended before introducing Cloud Computing to a productive environment.

### 2.3.6 Security of Cloud Computing

In this subsection a very brief overview of the different layers of security is given. The broad spectrum of security reaches from information security within the infrastructure level to data security in the platform level and finally the security management and monitoring tools within the software level.

Table 2.1 summarizes the major security measures that Cloud Computing is built on. An overview is given of the different security controls, their families and organizational classes. Figure 2.7 visualizes the different layers of security techniques that are intertwined and build on each others security scopes and approaches. Mather et al.[16] explores the topic in more detail. In the following paragraphs, the responsibilities between a cloud customer and a CSP is also put briefly into perspective of the different deployment models across the service models.

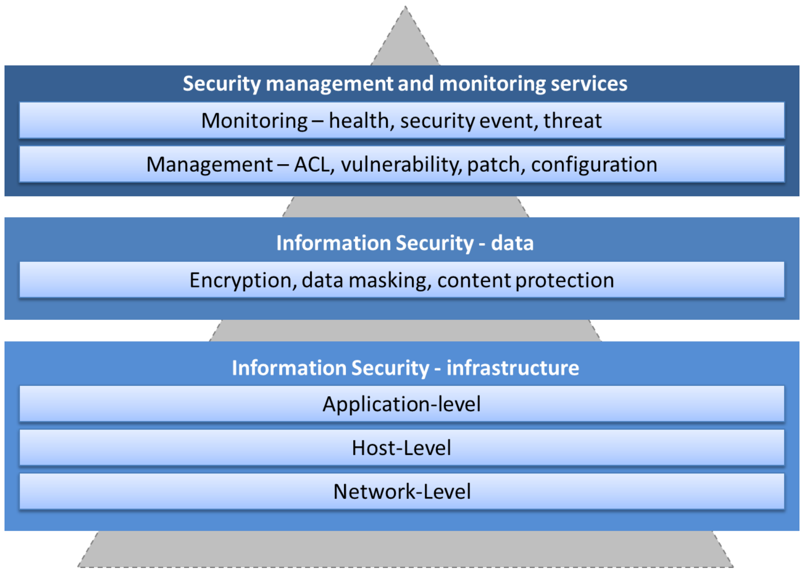

Security management and monitoring services
Monitoring – health, security event, threat
Management – ACL, vulnerability, patch, configuration
Information Security - data
Encryption, data masking, content protection
Information Security - infrastructure
Application-level
Host-Level
Network-Level

**Figure 2.7:** Overview of the different levels and scope of security in the cloud [16]

At the Infrastructure as a Service (IaaS) level, the CSP has to offer availability management for virtual machines, access control for users as well as resources, for example network. Furthermore, vulnerability management for the host level of operating systems and its applications on a higher level. This includes a patch management that incorporates update routines for operating systems and applications. The Customer has to actively manage these systems. It is not the provider's responsibility to maintain security within the customer's architecture.

**Table 2.1:** Security Controls and their Organizational Classes according to NIST [17]

| ID | Family | Class |
|---|---|---|
| AC | Access Control | Technical |
| AT | Awareness and Training | Operational |
| AU | Audit and Accountability | Technical |
| CA | Security Assessment and Authorization | Management |
| CM | Configuration Management | Operational |
| IA | Identification and Authentication | Technical |
| MA | Maintenance | Operational |
| PE | Physical and Environmental Protection | Operational |
| RA | Risk Assessment | Management |
| SC | System and Communication Protection | Technical |
| SI | System and Information Integrity | Operational |

At the Platform as a Service (PaaS) level, the CSP offers on top of the IaaS level a configuration management for libraries, databases, APIs and other tools for the application needed. Furthermore, an incident response (CERT) should be offered that notifies specific details about intrusions, compromised data and user accounts. The customer has to report to the affected users and fix the application, while the CSP only provides the information about the intrusion. Monitoring systems should be offered that log network, host, database and application activities, in order to provide a sufficient security level. The customers responsibilities reside with the application, that is deployed on the PaaS. The CSP is responsible for offering, maintaining and patching the monitoring services.

At the Software as a Service (SaaS) level, the CSP is held responsible for availability, patch and configuration, vulnerability and access control management. This includes all the activities that reside within each management block, detailed in the prior levels. The customer is only responsible for its own software and can focus on its security.

### 2.3.7 IT Compliance and Cloud Computing

Cloud Computing affects many fields in law (privacy, copyrights, intellectual property, etc.) and gives rise to a lot of jurisdictional questions. It is obvious that the law did not keep pace with the technological change around the globe. This leads to the situation that global companies with multiple national jurisdictions are caught between two regimes.

In recent history, the USA tried to get hold of data, which resided in Ireland and would have been out of its jurisdiction. However, the US government argued that the organization processing the data is a national company and is subject to US law as well as US national security interests. The European Union argues differently and holds the position that the country's law matters in which the company is active despite its origin [18].

A solution to this dilemma, can be a national company that hosts the data and relieves the global CSP from handing over data to the country of its origin. This is called a trust solution ("Treuhandlösung"). The CSP only offers services and cannot actively access the customer's data without the consent and active act of the customer.

Despite the global jurisdictional problems, clash of law and its interpretation, the regulations within the European Union are not conflict free especially with regard to Legal Metrology. Here, national regulations compete with each other and arise practical questions. For example, a Spanish manufacturer sells his devices in Germany but processes its services in Spain. The jurisdiction of the German market surveillance is limited and cannot inspect the servers in Spain. At this point, a transnational collaboration is needed and has to be regulated in the realm of the European legislation. Projects started in 2018 to tackle these problems, like the "European Metrology Cloud", that foster the digital transformation in Legal Metrology and initiate a mutual coordinated European digital infrastructure [19].

To sum up, ten years after the advent of Cloud Computing it is still challenging to comply with jurisdictional requirements without breaching directives of a foreign judiciary. With respect to current political developments [20], it might remain complicated for Cloud service providers to fully comply with all requirements and not get caught in the middle.

### 2.3.8 Service Level Agreements

With the introduction of shared resources and Cloud Computing, it is crucial to negotiate and arrange certain levels of reliability, availability and fault tolerance of outsourced processes. These management tasks are encompassed by service level agreements (SLA). SLAs can be considered as a negotiation layer between CSPs and consumers of services. It fulfills requirements that is handled by private law and provides securities for both parties.

For instance, a manufacturer outsources services to a third party Cloud service provider in Legal Metrology. The manufacturer is still liable to the law, while using SLAs he can demand financial compensation towards the CSP in case of not providing the promised services. However, the CSP is protected against not-contractual conditions and services. SLAs can serve as an indicator for quality characteristics between different CSPs. For a more in depth and comprehensive analysis on this matter see [21].

### 2.3.9 Certifications and Cloud Computing

There is plethora of security guidelines, norms and standards for Cloud Computing available. The most important are the ISO/IEC-2700x family, which describe fundamental information security techniques and best practices. It reaches from information security management systems via measurements to risk management. Dekker et al. [22] surveyed 12 different audit frameworks and certification schemes that are also listed in Figure 2.8. This in-depth analysis is beyond the scope of this work but is noteworthy because it is a good introduction to that topic.

The European Union Agency for Network and Information Security (ENISA) compiled an overview for Cloud Computing Certification and certification

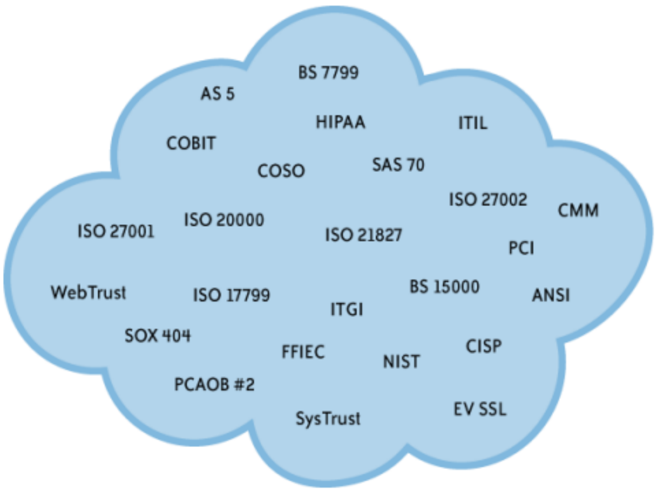


**Figure 2.8:** Overview of the different security standards [16]

schemes [23] as part of the European Cloud Strategy [24]. It offers a list of private certification frameworks and the underlying standards. This is particularly helpful for SMEs that need to make an educated decision if the Cloud Service Provider complies with the European legal framework.

## 2.4 Summary

In this Chapter a brief introduction into the field of Legal Metrology with its stakeholders and relevant regulations were given. Furthermore, with the introduction of Cloud Computing in that field, a new role for the Cloud Service Provider was postulated. However, due to the wide field of deployment models, the role of the Cloud Service Provider is not fixed. This role still needs to be determined in the future. Moreover, the essential requirements and the overall goals of reference architectures in Legal Metrology were explained. Given the background of the applied field, the preliminaries of the Cloud Computing techniques, service and deployments model and the advantages as wells disadvantages were given and discussed. A general security outline and IT-compliance were presented. Along with a brief summary of service level agreements (SLA) and existing certification guidelines for Cloud Computing. This Chapter provided the necessary assumptions for the concept of the Secure Cloud Computing Reference Architecture that will be presented in the next Chapter.

# 3 Secure Cloud Reference Architecture

*"Cloud Computing is a challenge to security, but one that can be overcome."*

*—Whitfield Diffie*

THIS chapter covers the secure Cloud Reference Architecture with exemplary use cases in Legal Metrology. The distributed measuring system with its characteristics will be described and surveyed on the basis of the previously introduced fundamentals for such an architecture. A glimpse into the applied security approaches and verification methods for integrity checking of the platform will be provided. Furthermore, the risk assessment method will be briefly described, but it is fully covered in Chapter 6. Ultimately, related work is presented.

## 3.1 Distributed Measuring System

A premise for the secure Cloud Computing architecture is the transition from well contained measuring instruments to a distributed measuring system (see Figure 3.1). The radical change that imposes the distribution of components, with only a physical sensor and a communication unit left in the field, while processing and storage will be centralized, was already pointed out in Chapter 2 with the description of the legal framework in Legal Metrology.

In this section the focus lies on the security approach for a secure cloud reference architecture, that fulfills the essential requirements of the MID (see

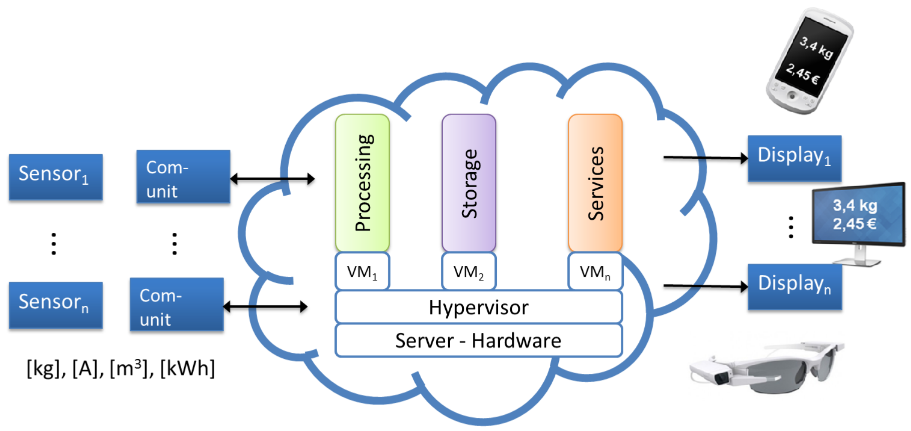


**Figure 3.1:** Overview of the reference architecture concept for a distributed measuring instrument with a sensor and a communication unit in the field. Processing, storage and service units are moved to the cloud. The client is left with a secure display on a variety of devices.

Section 2.2.2) while at the same time providing adequate security and preserving the flexibility and economical advantages of Cloud Computing, such as ease of update procedures for large numbers of measuring instruments by reducing hardware dependencies with software solutions. The different components of this architecture are described in a bottom up approach and are detailed in the following subsections. Furthermore, the communication and security measures of such an architecture are discussed.

### 3.1.1 Architectural Approach

The separation of legally relevant software processes and keeping them safe is the driving force to implement virtualization techniques. The most important processes are separated in different compartments. These processes are deduced from the MID, Welmec Guide and the national measurement and verification act (see Section 2.2). These processes reflect the legal relevant and regulated scope of software in common measuring instruments. Further it visualizes the separation from the commercial and unregulated part of the measuring instrument, in order to ease ,e.g., software updates in the unregulated part without the need of recertification of the whole measuring instrument (see Section 2.2.2). The legally relevant software modules are listed below. A summarized view of the architectural model is visualized in Figure 3.2. The measurement results are encrypted via fully homomorphic encryption (FHE) within the measuring instrument and then received and processed by the cloud. The encryption applies for the whole processing time in the cloud and will be only decrypted outside of the cloud at a secure end-user device. Thus, FHE impedes an internal attacker or the CSP to attack encrypted data in the cloud.

In Section 2.3 the requirements and service as well as the deployment models of cloud computing are introduced. This lays the important foundation for the architectural approach, since each tier of the cloud has to be carefully designed against the legal requirements of the aforementioned legal documents (see Section 2.1). In the next few sections the separation of legal and non-legal software parts is the driving primitive for the design of each tier of the cloud reference architecture. Furthermore, the legal requirements have to be tested for each service model and a design proposal for each tier has to be developed, so that a potential manufacturer could use this proposal as a blue print for a realization of a distributed architecture.

The distributed cloud reference architecture is designed from the bottom up. The Infrastructure as a Service (IaaS) tier will be described in Section 3.1.2. Here the concept of subnetworks and virtual machines guarantee the separation of legal relevant processes. While separating the networks into logically smaller entities via sub netting, it was decided to create separated virtual machines for the most important legally relevant tasks (see Section 2.2.2). The Platform as a Service (PaaS) tier is built on top of that. To guarantee a further separation of legal relevant processes, the microservice tier introduces another separation of legal relevant software. Noteworthy is here, that the separation of the IaaS is still intact and supports fully the separation of the microservice approach (see Section 3.1.3). The Paas tier defines already good part of the Software as a Service tier architecture, so that a clear distinction is hard to spot, but Figure 3.4 includes already the architecture of the SaaS tier. For clarification, Paas usually provides only tools for developers to deploy and develop their applications in a cloud (see Section 3.1.3). The SaaS tier includes additionally the communication concept for creating a simple signature of exchanges messages, which is formalized in a communication protocol and explained in Section 3.1.5.

The following list describe the most important legal relevant processes that are deduced from the MID and WELMEC and ar implemented in separated virtual machines:

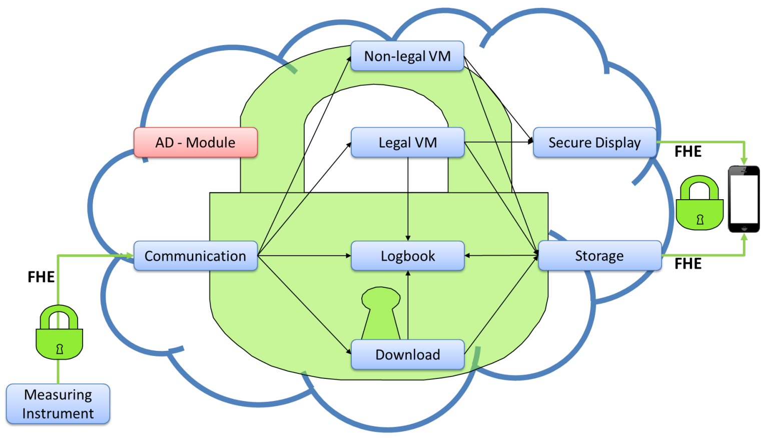


**Figure 3.2:** Architectural model overview of virtual machines and their communication paths

Logbook The Logbook VM hosts the software process responsible for logging all relevant activities around the measuring system, i.e. arrival, processing, saving of measurement data, user activities, software updates etc.

Legal VM The Legal VM is responsible for processing the legally relevant measurement data. This VM has the most CPU cores available, since it has to carry out all the computation for the measurement data.

Download Manager The Download Manager receives signed software updates from the manufacturer. After carefully checking the integrity and authenticity of the received update, it will spread the updates to the dedicated machine.

Storage Manager The Storage Manager is designed to store measurement data for a long time period. Thus, it will handle a database and will make the measurement data available through the secure display VM, which hosts services to demand data from the Storage Manager.

Monitoring Service The Monitoring Service, part of the Legal Network VM, continuously monitors all the VMs while learning their individual patterns for the running tariff applications, in order to detect anomalies within the system.

### 3.1.2 Infrastructure as a Service

The Cloud Computing infrastructure is built on the open source project OpenStack which delivers a modular construction kit for Cloud Computing architectures. For the prototype an All-in-One Single Machine approach was chosen to mimic a cloud environment on a server with an Intel Xeon CPU E5-2620 with 24 Cores. Nevertheless, the distributed characteristics are realized with the help of OpenStack to build the Infrastructure as a Service (IaaS). As a first step a virtual network was created to lay the foundation for a separated and secure network from the Internet to comply with the software requirements of the MID four subnetworks assure a low level separation (OSI level 3) of the virtual machines. The four subnetworks are divided into an Ingress/Entrance subnetwork, a Legal Metrology subnetwork, a Services subnetwork and an Egress/Exit subnetwork. If this concept should be distributed across different locations, e.g. data centers, countries etc., then it would be sensible to use Virtual LAN and to aggregate different virtual machines across physical boundaries. Also, it would enable a separation already on OSI layer 2. For reasons of simplicity subnetting was sufficient for this prototype, and dealt with the same problems of more complex networks.

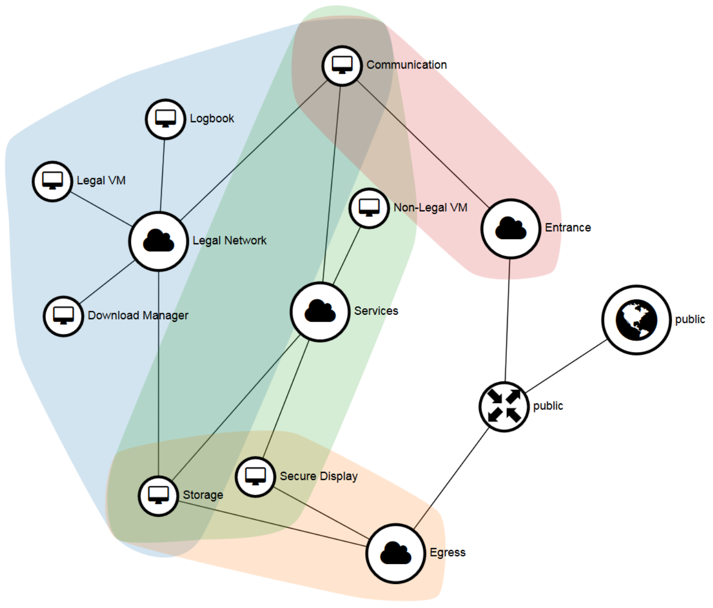


**Figure 3.3:** Network topology of the IaaS layer. *Red* is the ingress subnetwork. *Blue* comprises the Legal Network with all legal relevant VMs. *Orange* describes the egress subnetwork, which offers a Storage VM and a secure display that allows pulling measuring data for the end-devices. The *green* shape marks the third party subnetwork for non-legal relevant services.

**Subnetwork** - A subnetwork is a logical subdivision of an IP network [25]. Actively dividing a network in smaller networks is called subnetting. The main motivation for using this technology is to separate the legal relevant virtual machines from the non-legal relevant ones. The distinction for the ingress and egress network was made on purpose to support the API design for incoming and outgoing data and to shield the Legal Network from direct access to an open network.

Ingress/Entrance network The Ingress/Entrance network hosts the communication virtual machine (see Figure 3.3 is marked red), which agitates as a gateway to the Internet for the rest of the virtual machines. This design decision was done deliberately, in order not to expose the legal network directly to the Internet, and increase the security.

Legal Metrology network The Legal Metrology network hosts three legally relevant virtual machines (see Figure 3.3 is marked blue): the logbook that logs all relevant events, the legal processing (Legal VM) and a Download Manager for software updates.

Services network The Services network is created for third parties (see Figure 3.3 is marked green), that want to host legally non-relevant virtual machines and services, e.g. for advertisement or maintenance services.

Egress/Exit network The Egress/Exit network is designed to send data to the terminal devices in the field (see Figure 3.3 is marked yellow), thus it hosts the secure display virtual machine which will provide measurement data.

To sum up the IaaS-layer comprises the actual physical devices like servers, storage, and the network. Furthermore, it offers software functionality like virtualization to increase the workload of the hardware and decrease the idle times, speed up the provisioning time for new servers and decrease the costs per server. IaaS enables also a first low-level separation of legally relevant processes.

### 3.1.3 Platform as a Service

This section is dedicated to the Platform as a Service (PaaS) layer of the Cloud Computing architecture. PaaS can relate to the logic tier in a multi-tier architecture [26] as it places the developer tools at disposal to coordinate

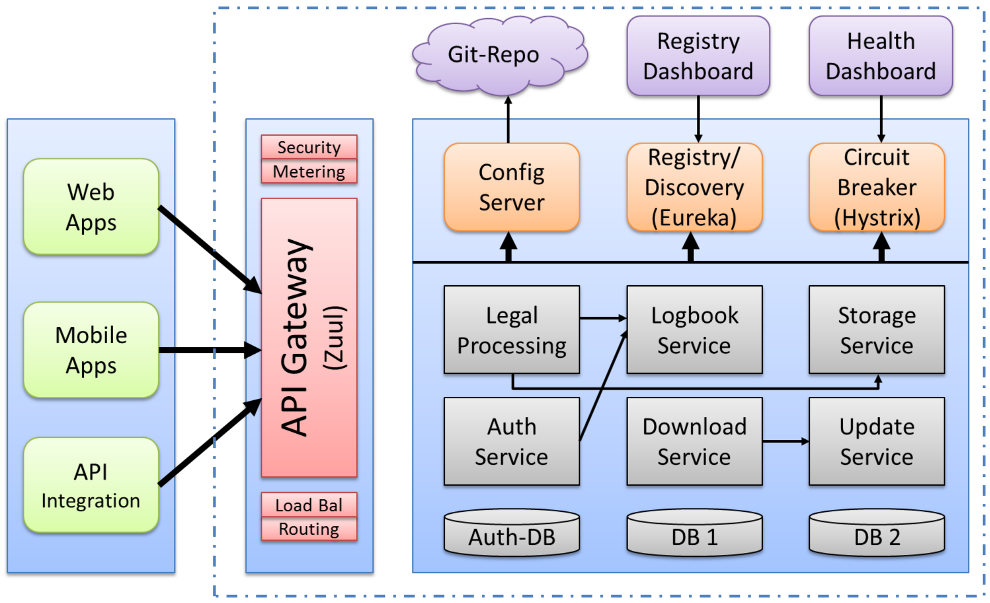


**Figure 3.4:** Overview of the platform concept using the open source Netflix-Stack. *Green* are the external devices that interact with the platform via a gateway (in *red*). *Orange* and *purple* represent the orchestration layer. *Grey* are chosen microservices with an exemplary data flow.

and handle processes, applications and further integrate them into the surrounding layers. A PaaS comprises usually execution runtime, Application Programming Interfaces (API), libraries, web servers and other development tools. While designing the concept for a secure Cloud Computing architecture the architectural pattern of Microservices had several advantages that would provide more flexibility, security and agility. Furthermore, it fits perfectly the architectural premises of Cloud Computing for designing a distributed service oriented architecture. The pattern was implemented by using the open source Netflix-Stack for Microservices.

**Microservice Architecture** - A microservice architecture is a fine-grained Service Oriented Architecture (SOA) pattern, that creates for each task and process a service. A service is focused only on one task, hence called microservice. Each microservice can exist independently and communicates via messages with other microservices. Thus, microservices are highly distributable and scalable.

Advantages The advantages of a microservice pattern lies in its simplicity for each service, since each one focuses only on one task. This means less code to maintain for each service. The code base can be developed by different teams in individual pace and also deployed independently from each other without risking downtime of the whole system at any time.

Updating a service is very convenient. By keeping the older service running the new version can be deployed at the same time. After successfully deploying the new version, the system load can be slowly migrated to the newly deployed service with the help of a load balancer (see Figure 3.4). This prevents unwanted disruption of well established working processes and a small set of people can test the new service before making it available to everyone.

Microservices are polyglot. This means for each challenge the best suitable programming language can be chosen to solve the individual problem. This can reduce also the lines of code of a service. Furthermore, it increases the flexibility and eases the need for specialized programmers.

Microservices are highly scalable and resilient. Because of their self-sustaining nature, microservices can be run in parallel and as many instances as needed to cope with the working load. Beyond this each service has an API and communicates with the whole system via message that will be exchanged via an active message queue. In case a service has to be restarted, the messages will be queued and will be processed in time. No messages will be lost. This guarantees a pseudo-resilience, since the downtime can only last as long as the size of the queue can handle the incoming messages.

**Monolithic Architecture** - A monolithic software architecture combines all processes and tasks into one structure. The separation and distribution of subprocesses across an architecture is not provided nor encouraged.

Disadvantages Despite all the advantages for a microservice pattern, there are some challenges. The flexibility and scalability comes at a price of complexity. Each layer that is added increases the complexity and the maintenance effort. To debug microservices and tracing failures can be more challenging as a traditional monolithic application, since of their distributed nature. They externalize their communication, and this can become a serious problem when network latency adds up.

Furthermore, handling multiple databases and transaction management can become unclear and needs more attention than a traditional application. For developers the testing phase is more convoluted, since each service has to be setup and registered before being able to start testing a new service.

Most of these difficulties are already addressed by the Netflix framework.

There is Zipkin service (trace monitor) which helps to trace network problems and measures latency and responsiveness of each service. Hystrix provides latency and fault tolerance to each service. If a service should be not available a default fall-back can be implemented. This service increases the overall resilience of the distributed system.

### 3.1.4 Software as a Service

The software as a service (SaaS) layer is the last tier of the Cloud Computing platform. This tier hosts just the application and runs on top of the PaaS tier. This tier enables the client to run just a very thin application, often only a web browser, to access the software relevant functions offered by web services. The software is usually independent of the underlying hardware and profits from its highly abstract design, so that it can be easily distributed within the Cloud Computing nodes.

In the proposed reference architecture implementation approach, this tier offers the fully homomorphic encryption application for smart meter gateway tariff applications, which takes full advantage of the LSIM extension.

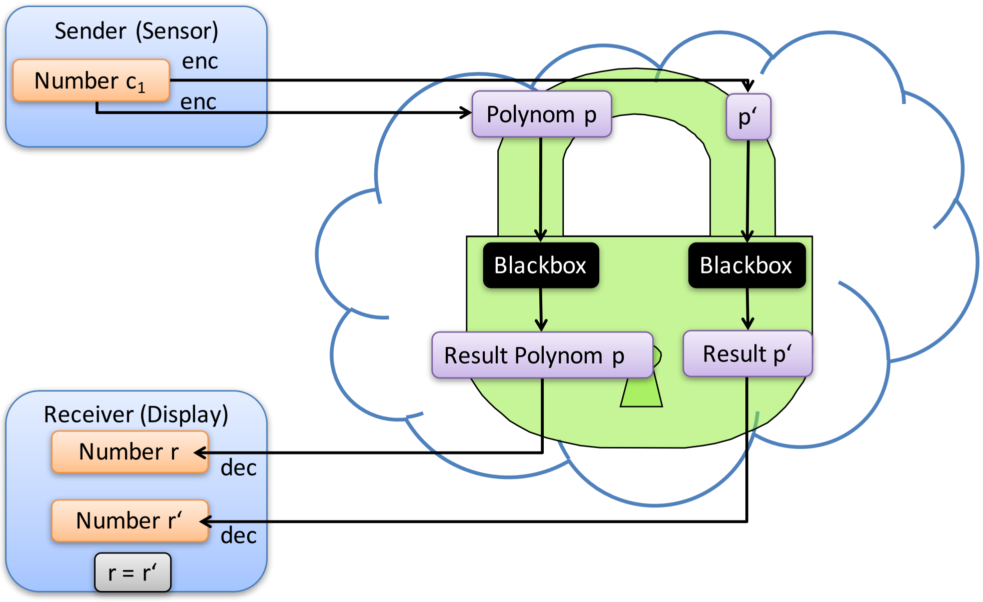


**Figure 3.5:** Overview of encrypted communication protocol.

### 3.1.5 Communication Protocol

An overview of the proposed communication protocol is described in detail below (see Figure 3.5). It has been developed to ensure a distributed, authentic and secure communication using FHE to secure the computation within a Cloud Computing architecture. The parts of sensor, processing and display correlate in the protocol as the following three sections (see Algorithm 1) : Client (Steps 1-7), Server (Steps 8-14) and Receiver (Steps 15-22).

The client generates the Public Key $P_k$ and Secret Key $S_k$. The $S_k$ will be send via a secure channel directly to the trustworthy receiver (Step 6-8). In Step 1 the client generates the measurement data $m_i$. In Step 2 the client encrypts the same measurement data $m_i$ twice with the Public Key $P_k$ and sends the encrypted measurement data $c_i$ and $c_i'$ to the Cloud (see Step 3). The same procedure will be done for the index $i$ that will be encrypted and result in the encrypted indices $v_i$ and $v_i'$ (Step 4). This procedure creates the simplest form of a signature to ensure authenticity and integrity of the measurement results, since both encrypted versions will automatically look different to an outsider (see Step 5). If the key exchange $S_k$ is successful between client and receiver (see Step 8) then the encrypted measurement data and encrypted index will be send to the cloud service (see Step 9.).

The Server will check the TLS certificate of the client and the origin of the receiving messages (see Step 10). If the check is successful (Step 11) then the applied tariff application (see Section 5.1) will process the measurement data $c_i$ and $c_i'$ (see Step 12). Noteworthy is here that the encrypted measurement data $c_i$ and $c_i'$ are treated equally under the same conditions. In Step 13, the applied tariff application will compute the encrypted measurement results $r_i$ and $r_i'$.

The receiver will authenticate against the server (see Step 18) and then pull the encrypted measurement results $r_i$ and $r_i'$ and the encrypted index $v_i$ and $v_i'$ from the server (see Step 19). Then the receiver will decrypt the encrypted data with the earlier received Secret Key $S_k$ (see Step 20). If the decryption of measurement results $r_i$ and $r_i'$ and index $v_i$ and $v_i'$ are successful and the results are the same (see Step 21) then no manipulation can be detected and the encrypted measurement result $mr_i$ must be correct. In case of unmatching measurement results or indices, the measurement results will be purged (see Step 24) and an error message will be written to the logbook (Step 25) as well as a measuring system failure will be flagged (see Step 26).

Even though homomorphic encryption usually aims at ensuring data privacy [27], it may be used in other areas and for other purposes as well. A potential attacker (Inside-Attacker) trying to manipulate the measurement results from within the Cloud Computing architecture will face no clear-text processing and thus can only tamper randomly a measurement or the attacker consequently manipulates all measurement data in the same way. If the first approach is chosen, the receiver will detect the manipulation (Step 19) and discard the measurement result. If all measurements are tampered in the same way only test data with a known outcome can detect this kind of manipulation. Implementing aperiodic test runs with precomputed data will decrease the possibility for an attacker to stay undetected for a long period.

If homomorphic encryption is carefully combined with testing a running algorithm via precomputed test data, even such random effects may be detected. Subsequently, the scheme detailed here [28] may be used to achieve a certain degree of robustness towards algorithm and data manipulation, too.

For reasons of simplicity and legibility of algorithm 1 the usage of a nonce and the coverage of replay attacks are not part of the described protocol. Nevertheless, it is assumed that the connection between all participants are secured by TLS (Transport Layer Security), which itself protects against possible replay attacks using message authentication codes (MAC) and the sequence number to prohibit a recorded message stream between client and server. Further, to secure messages on an application level, it is possible to implement nonce-based security mechanism, to sign each request to exchange measurement data and thus protect its originality. Since, this security approach is addressing the security concern on a different layer, it is not part of the communication protocol. For more information on this matter refer to [29], [30].

**Algorithm 1** LSIM client-server communication protocol (overview)

```
Input: Measurement data $m_i$, Encryption Parameter $eP$, Public Key $P_k$, Secret
    Key $S_k$, Usable bandwidth upload/download (U,D), Index of elements
    $i = 1 \ldots n$
Output: Measurement results $mr_i$

Client:
 1: creates measurement data $m_i$
 2: encrypts measurement data and index twice:
 3: $c_i \leftarrow enc(m_i)$ and $c'_i \leftarrow enc(m_i)$,
 4: index $v_i \leftarrow enc(i)$ and $v'_i \leftarrow enc(i)$
 5: note: $c_i \neq c'_i$ but $dec(c_i) = dec(c'_i)$, $v_i \neq v'_i$ but $dec(v_i) = dec(v'_i)$
 6: repeat
 7:     send $S_k$ to trustworthy receiver,
 8: until key exchange is successful
 9: sends two encrypted messages with $c_i$, $v_i$ and $c'_i$, $v'_i$ to the server, respec-
    tively

Server:
10: checks certificate and the origin of messages
11: if check is successful then
12:     process encrypted measurement data $c_i$ and $c'_i$, they will be treated
    equally
13:     receive encrypted measurment results $r_i$ and $r'_i$, respectively
14: else
15:     purge data,
16:     return error message to logbook
17: end if

Receiver:
18: authenticates against server
19: pulls encrypted measurement results and encrypted index from server
20: decrypts measurement results and index with $S_k$
21: if $dec(r_i) = dec(r'_i)$ and $dec(v_i) = dec(v'_i)$ then
22:     no manipulation and $mr_i$ is correct
23: else
24:     purge measurement results,
25:     return error message to logbook
26:     flag measuring system failure.
27: end if
```

## 3.2 Security Approaches and their limitations

Classical security approaches use encryption to ensure privacy and signatures for verifying the authenticity of messages. These attempts aim for securing the communication between two endpoints. In the Cloud Computing context, the attention has shifted to secure computation (see Figure 3.6). Classical encryption is static, thus messages have to be decrypted first before being able to be processed. Similar conditions apply for classical signature schemes. They are designed for static messages. In order to verify remote computation, a proof is needed for correct and trustworthy results. Verifiable Computing demands that the evaluation of an interactive proof [31] needs to be more compact than the actual computation. FHE offers an elegant way to ensure privacy by design since it prevents clear-text information leaks. While externalizing confidential processes to the cloud, industry places little trust in external service providers. FHE was chosen over a homomorphic signature scheme not only to help solving this conflict, but it also offers a more powerful, practical and flexible approach to secure remote computing. Furthermore, there are several approaches that combine homomorphic encryption with verifiable computation [32][33], so that it is still feasible at a later time to integrate verifiable computation with this line of research.

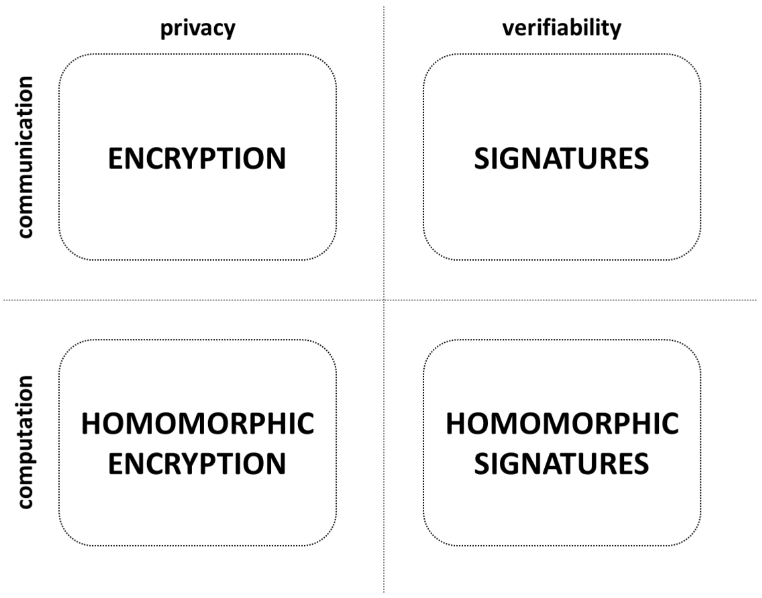


**Figure 3.6:** Overview of different security techniques and their area of application.

### 3.2.1 Measuring Instrument and Trustworthy Components

The reference architecture is a general approach for all measuring instruments, however it will be applied to real world smart gateway tariffs and intelligent measuring systems (see Section 5). In the context of such a system, all parties need to authenticate themselves before initiating a communication connection. To this end, the smart meter gateway (SMGW) as well as partners communicating with it over the wide area network (WAN) are in possession of Hardware Security Modules (HSM), which are in charge of key handling, signature verification and white listing of communication partners.

A HSM is a physical computing device that stores safely and manages cryptographic keys for critical functions. It provides extra security for sensitive data. A HSM is an essential security measure in order to enable secure key handling and exchange of keys between associated clients and devices within the public key infrastructure (PKI). It also acts as a trust anchor to guarantee the integrity and authenticity of a Root Certificate Authority (CA) and its Sub CAs. The HSM can be used to sign and secure TLS-certificates, OpenSSL-certificates for end-to-end cryptography and signature certificates to sign measurement data and thus prove their integrity as well as authenticity [34]. The HSM performs various checks with the certificates before being able to continue with the respective procedure, i.e. it checks the signature of the certificate, the lifetime span (maximum of 7 years) of the issued certificates, it checks revocations lists and the issuer of the certificate, reviews the mode of usage such as key usage validation and extended key usage validation.

For the proposed architecture a potential measuring instrument with a HSM is taken into account and can be easily integrated. For reasons of integrity and in respect to the BSI technical regulations of the smart meter gateway [34], a HSM in conjunction with a sensor are mentioned here. However, HSM will not be further discussed, since it is outside of the scope of this thesis.

In Legal Metrology, the concept of a trustworthy system administrator does not exist. Therefore, all parties must be considered untrustworthy, i.e. the HSM is the only trust anchor for security concepts, which is considered as completely unbiased and thus can be accepted by all parties involved.

## 3.3 Verification Methods

Through technologies, like Cloud Computing, manufacturers can provide customers a modernized and flexible way to access their meters, e.g. via mobile devices or by providing better service via intelligent data services. The radical change through the drift of well contained measuring instruments nowadays to distributed measuring systems confronts Legal Metrology with many challenges. Via fully homomorphic encryption (FHE) [35] the main threats will be addressed in Chapter 4, like an insider attack and data manipulation in the cloud, in order to create an acceptable solution that provides adequate security.

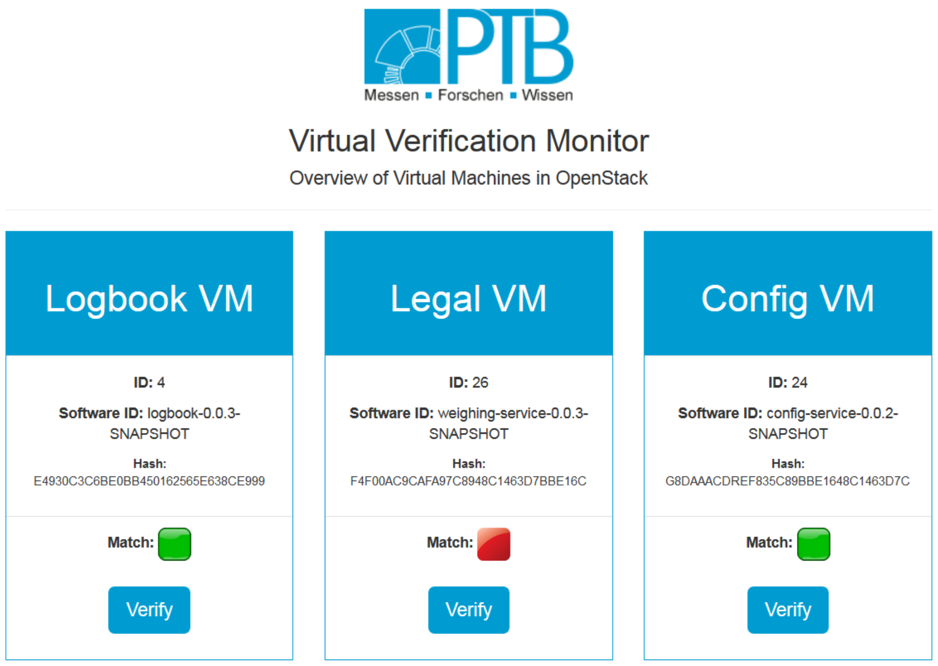
In this section two verification methods are presented. Section 3.3.1 describes an integrity checking method via a web service, that provides a simple interface for quick in-depth analysis of the used virtual machine and their actual state. It is paramount for market surveillance to be able to cope with a complex technology stack without investing too much time in a thorough integrity check of the used instruments.

Section 3.3.2 describes a continuous anomaly detection approach for the prior detailed secure cloud architecture (see Section 3.1). The behaviour and the tariff application pattern (see Chapter 5) of the platform will be constantly monitored by a software module residing within the Platform as a Service (PaaS) level as part of a monitoring service (see Figure 3.2). Depending on the severity of detected anomalies, the module automatically classifies them into three categories: green, the system is in a normal state; yellow, the system has an anomaly but its stability is not affected; red, the system behaves anomalously and its stability is affected.

### 3.3.1 Verification Monitor

As part of the envisioned reference architecture (see Figure 2.3) a verification method has to be developed, in order to support the user and market surveillance to verify measuring systems in the field. Since the technology stack becomes too complex to validate quickly in the field of Legal Metrology, the Notified Body endorses simple verification methods to help the authorities in the field to accomplish their tasks.

If an end-user expresses doubts to the user and market surveillance of, e.g. a power meter in his home and an associated billing, the addressed authorities must have the possibilities to verify the used meter, the processing unit and associated logbook. The metrological software has to provide a software identification. An acceptable solution according to the WELMEC Guide would be:


Virtual Verification Monitor

Overview of Virtual Machines in OpenStack

Logbook VM

**ID:** 4

**Software ID:** logbook-0.0.3-SNAPSHOT

**Hash:** E4930C3C6BE0BB450162565E638CE999

**Match:** 🟩

Legal VM

**ID:** 26

**Software ID:** weighing-service-0.0.3-SNAPSHOT

**Hash:** F4F00AC9CAFA97C8948C1463D7BBE16C

**Match:** 🟥

Config VM

**ID:** 24

**Software ID:** config-service-0.0.2-SNAPSHOT

**Hash:** G8DAAACDREF835C89BBE1648C1463D7C

**Match:** 🟩

**Figure 3.7:** Screen shot of the prototype verification monitor for the market and user surveillance.

1. a software name consisting of a string of numbers, letters or other characters,
2. a string added by a version number,
3. a checksum over code.

In Figure 3.7, a simple web application is displayed, which shows examples of three legally relevant virtual machines of the earlier described reference architecture. This web application can reside in the realm of the market and user surveillance and pulls the data exclusively of the virtual machines directly from the used cloud. Each virtual machine provides a unique ID, a Software ID consisting of a string of letters and version number and lastly a checksum over the used code. The checksum will be compared to a checksum residing with the surveillance authority. If the checksum match, a green batch otherwise a red batch will be displayed (as one can see by the example of the Legal VM).

### 3.3.2 Anomaly Detection Approach

To guarantee security over time it is advisable to monitor the system constantly, in order to be able to detect anomalies in system behaviour. At first, the monitoring system collects data to learn the normal system workload and behaviour. In a second stage it is planned that the anomaly detection system categorizes autonomously occurring anomalies into three categories: green, the system works flawlessly and no anomalies are detected; yellow, the system works unaffected and no critical anomaly occurred; red, the system is affected by critical anomalies and the measurement results cannot be trusted.

For Cloud Computing architectures two main approaches are worthy to further investigate: *Console log based anomaly detection* and *System metrics based anomaly detection* [36].

**Console Log based anomaly detection** is found by a set of rules to verify the system behaviour. The process of setting up these rules is complex, error prone and expensive. It is possible to optimize these inefficiencies by automatically generating system models for anomaly detection. However, creating a reliable feature vector access to the source of the monitored application is needed in order to learn the structure of the log files.

**System metrics based anomaly detection** is less invasive, needs less privileges and has less impact on the monitored system. This approach is more suitable especially for a distributed system when scalability, elasticity, flexibility and migration of virtual machines are from utter importance.

In the following subsection the selected strategy of system metrics based anomaly detection is explained and put into use within the Legal Metrology application lifecycle. Due to time constraints the research on this anomaly detection approach is not finished yet and will be part of future work. Early results have been published in [37].

#### 3.3.2.1 Distributed Data Collection

In a distributed environment it is of utmost interest to keep the employed modules and services flexible as well as easy deployable to meet the demands of virtual machines that are spread across various time-zones, networks and teams. Also, it is important to limit the monitoring overhead as much as possible to stay competitive and cost effective.

Bearing these demands in mind, to monitor the deployed virtual machines the highly distributable and modularized open source monitoring software Performance Co-Pilot (PCP) [38] is used. Over 300 metrics for example CPU utilization, network bandwidth, disk I/O and memory usage is collected by

PCP. Through a highly modular approach of PCP, it is possible to segregate collecting, logging and evaluating of metrics. PMlogger is a headless logging agent, that can be deployed on any VM. PMCD is a collection daemon and acts as a centralized instance for aggregating log files from all logging agents.

To analyze system anomalies, PCP provides collection archives to replay and analyses step-by-step the system behaviour. These archives can be visualized with the *pmchart*-module. Furthermore, PCP supplies real-time monitoring to visualize the current state of the system. Vector [39] is one of many tools to render plots for real time-monitoring.

For this current AD-Module approach the most significant and promising metrics are collected and evaluated. In future developments, the number of metrics and different levels of logging will increase, i.e. on service level of each microservice, metrics can be collected and help to state the activity and overall system health in more detail.

At this time, the yielded results are very encouraging to further pursue the chosen approach. By finding correlations between specific metrics related to the Legal Metrology Application Lifecycle (see [37]), normal and anomalous behaviour of the cloud environment can be represented. However, finding good indicators of compromise is hard and leads to a lot of false positives. Furthermore, detecting unknown attacks and patterns is very complex and not straightforward. This line of research is worth continuing, but is outside of the scope of this thesis.

## 3.4 Risk Assessment

A risk analysis is applied to the Secure Cloud Reference Architecture to fulfill the legal requirements (see fully applied in Chapter 6). This risk analysis is based on software risk assessment for measuring instruments under legal control proposed by WELMEC Working Group 7 [9]. By objectifying the derived probability score for identified threats while following at the same time the guidelines of ISO/IEC 27005, ISO/IEC 15408 and ISO/IEC 18045, this risk assessment method enables comparability and standardizes the otherwise highly subjective assessment process. Furthermore, potential countermeasures are identified and quantified using Attack Probability Trees (AtPT) [1] for derived assets to be suitably protected.

## 3.5 Related Work

Similar approaches like SensorCloud[40], SealedCloud[41] and TRESOR[42] are tackling related parts of our research problems, but are either not using the computation advantages of the cloud or are focusing only on parts of our problem spectrum. SensorCloud for example provides an end-to-end encryption for sensor data but reduces the cloud to a secure storage solution. The SealedCloud approach deals with an untrustworthy system administrator and deletes all data in case of an unauthorized access attempt. TRESOR handles patient's health data and offers security by distributing data through several cloud services and employing a special cloud broker to access the data and assembling it in a secure environment.

## 3.6 Summary

In this chapter, the envisioned distributed measuring instrument was introduced and set the foundation for the proposed secure cloud reference architecture. As a centerpiece, virtualization on different levels, i.e. infrastructure, platform and services, enables software separation for legally relevant and non-relevant processes. The different techniques such as subnetting and microservices were discussed and evaluated to ensure flexibility, scalability and security for the secure cloud architecture. Furthermore, a communication protocol was proposed to guarantee security, integrity and authenticity of measurement results throughout their lifecycle.

An overview of security approaches and trustworthy components of a measuring instrument were given. Classical encryption and security approaches usually focus on securing communication, while traditionally the computation was carried out in a secure realm. Using distributed systems, this assumption no longer holds true. In this thesis, the use of fully homomorphic encryption technique focuses on covering this security gap. Furthermore, a remote verification tool was presented to fulfill the integrity demand of the reference architecture. A continuous monitoring approach based on anomaly detection was introduced that emphasizes the integrity demand in the near future.

4

# Homomorphic Encryption

*"It was elegant math with real world applications, but it was also hairy and disgusting – I was instantly attracted"*

*—Shai Halevi, Cryptographer, IBM Research*

The main motivation of using fully homomorphic encryption within the reference architecture is to increase data security and integrity for processing measurement data in an external cloud. The principal attack vectors (malicious insider and software security vulnerabilities) will be decreased immensely by implementing a homomorphic encryption scheme that performs operations on encrypted data. Thus, neither a malicious insider nor a corrupted VM can easily change or spy on data. While reducing the most important risks for utilizing cloud computing in a well-regulated area, another goal is achieved by avoiding special requirements for cloud service providers offering a trustworthy environment without specialized hardware.

This chapter gives a condensed introduction into the field of homomorphic encryption. Bearing in mind that most readers are not cryptologist, the most important preliminaries of lattice-based cryptography are repeated but only briefly explained. For a comprehensive overview and a detailed introduction into lattice-based cryptography refer to the following primary sources [43],[44], [45]. Especially for the employed fully homomorphic encryption refer to [27], [46],[47], and [48].

The first three sections lead to the necessary preliminaries of this field. In Section 4.4 the process of choosing the suitable homomorphic library for the envisioned use case is discussed and put into perspective. The functionality of the cryptographic library LibScarab is described in Section 4.5. In Section 4.6 the contribution and modification are highlighted and comprised in an extension to the LibScarab library.

## 4.1 Preliminaries of homomorphic encryption

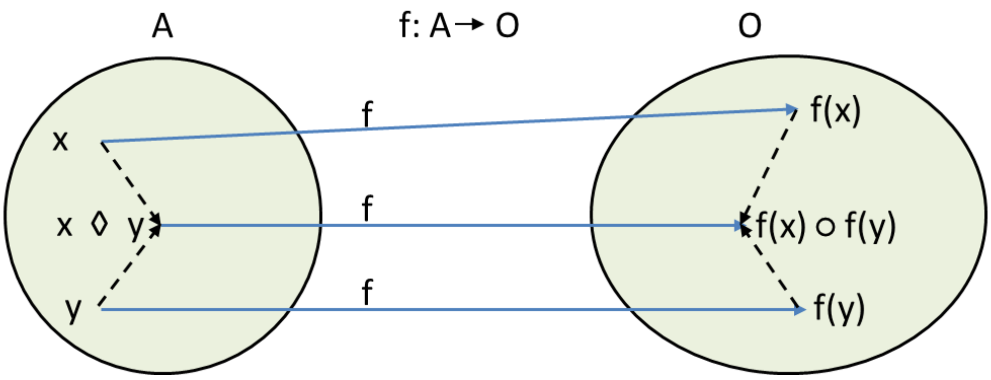


**Figure 4.1:** Graphical example of a homomorphism.

**Homomorphism** - Is a structure preserving transformation in linear algebra between two types of sets (structures) $A$ and $O$ with a function $f : A \to O$ which preserves the operations of the original set (structure) (see Figure 4.1 for a graphical representation).

Furthermore, it must guarantee the closure of addition $\oplus$ and multiplication $\otimes$ operations, i.e.,

$$\forall x, y \in A : f(x) \oplus f(y) = f(x \oplus y), f(x) \otimes f(y) = f(x \otimes y). \tag{4.1}$$

A homomorphism not only allows structure preserving operations, like addition and multiplication, but can also be used for Boolean circuits, where these basic arithmetic operations will be reduced to a XOR- and AND-Gate (for an applied approach see Section 5).

**Lattice** - A lattice is a discrete additive subgroup $\mathcal{L} \subset R^m$. The $rank(\mathcal{L})$ of $\mathcal{L}$ is the dimension of the vector subspace $span(\mathcal{L})$. Each lattice $\mathcal{L}$ with rank $n$ has a basis of $[b_1, ..., b_n]$ given by,

$$\mathcal{L} = \left\{ \sum_{i=1}^{n} x_i b_i | x_i \in \mathbb{Z} \right\}. \tag{4.2}$$

**Sublattice** - Let $\mathcal{L}, \mathcal{U} \subset R^m$ be lattices. Then $\mathcal{U}$ is a sublattice to $\mathcal{L}$, only if $\mathcal{U} \subseteq \mathcal{L}$.

We need to define further a basis matrix and the area of a basis matrix, in order to have the tools at hand to successfully understand a lattice-based cryptosystem.

**Basis matrix** - The matrix $B := [b_1, ...b_n] \in \mathbb{R}^{m \times n}$ is called basis matrix to the lattice $\mathcal{L}$ and say the lattice $\mathcal{L}(B) := \mathcal{L}(b_1, ..., b_n)$ is spanned by $B$. [47]

**Area of a basis matrix** - For a basis matrix $B := [b_1, ...b_n] \in \mathbb{R}^{m \times n}$ we call $\mathcal{P}(B)$ a parallelepiped given by

$$\mathcal{P}(B) := \left( \sum_{i=1}^{n} x_i b_i \mid 0 \leq x_i \leq 1 \right). \tag{4.3}$$

## 4.2 Definitions and Classifications

Armknecht et al. [49] give a comprehensive overview of the different homomorphic schemes and their properties that are commonly used. Furthermore, they provide a good introduction to the field of homomorphic encryption and their numerous definition throughout literature. In the following subsections three classifications of homomorphic schemes are given that are paramount for the later choice of a suitable homomorphic library. The following paragraphs only focus on the formal definition of a homomorphic encryption schema following Brakerski et al. [50] and their classifications following Armknecht's overview. It is not a catholic set of definitions to formally introduce and describe homomorphic encryption schemes entirely.

A generic homomorphic encryption scheme HE consists of the following four probabilistic polynomial time (PPT) algorithms (Gen, Enc, Dec, Eval) such that:

**Gen($1^\lambda$, $\alpha$)** The key generation algorithm has usually two inputs: security parameters $\lambda$ and often additional auxiliary parameters. It outputs a triple consisting of a public key $p_k$, a secret key $s_k$ and a public evaluation key $ev_k$.

**Enc($p_k$, m)** The encryption algorithm has two inputs: public key $p_k$ and one bit cleartext message $m \in \{0, 1\}$ . The ciphertext $c$ is received as an output.

**Dec($s_k$, c)** The decryption algorithm takes the secret key $s_k$ and a ciphertext $c$ and delivers a one bit cleartext $m \in \{0, 1\}$

**Eval($ev_k$, f, $c_1 \ldots c_n$)** The evaluation algorithm has three inputs: A public evaluation key, a function $f : \{0, 1\} \rightarrow \{0, 1\}$ and a set of $n$ ciphertexts $c_1 \ldots c_n$. It produces a ciphertext $cf$ that serves as an evaluation.

Furthermore, Armknecht et. al [49] define and formally introduce three additional attributes for homomorphic encryption schemes, that are important to have in mind for the following classifications. Instead, a compact informal summarization is given:

**Correct decryption:** A HE scheme is able to produce a flawless decryption of a ciphertext.

**Correct evaluation:** A HE scheme correctly evaluates all functions $f$ with a negligible deviation $\epsilon$. It follows that HE-evaluation scheme is *correct* if the evaluation and decryption are correct.

**Compactness:** The ciphertext of a HE scheme does not grow significantly in size after homomorphic operations. Furthermore, this requirement excludes the identity function as a trivial HE scheme, which in term would prevent the delegation of computation [51].

### 4.2.1 Somewhat Homomorphic Scheme

A HE scheme (Gen, Enc, Dec, Eval) that fulfills correct decryption and evaluation but lacks compactness. This leads to only limited number of homomorphic operations before the ciphertext becomes useless for decryption, since the ciphertext grows in size (noise ratio increases). Often times, these schemes only fulfill partial requirements, for example, only provide addition or multiplication but not both at the same time.

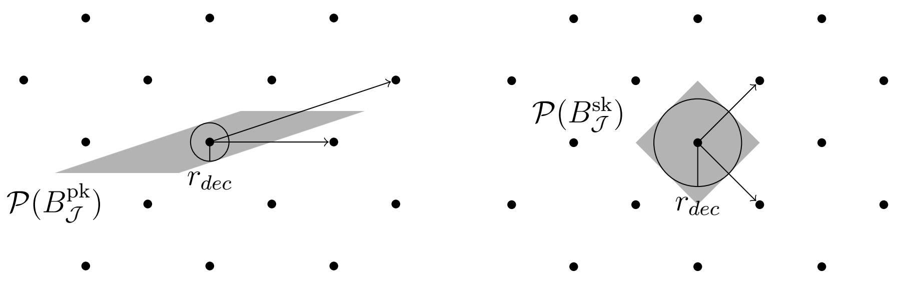
### 4.2.2 Leveled Homomorphic Scheme

A HE scheme (Gen, Enc, Dec, Eval) that fulfills correct decryption and evaluation and has an additional input parameter $d$ that indicates how many homomorphic operations are possible, before the ciphertext renders futile. Such a scheme does not provide unlimited homomorphic operations with further enhancement, such as modulus switching the number of homomorphic encryption can be increased but is still limited.

### 4.2.3 Fully Homomorphic Scheme

A HE scheme (Gen, Enc, Dec, Eval) that fulfills correct decryption and evaluation and is compact at the same time. Such a scheme permits an arbitrary number of homomorphic operations.

## 4.3 Gentry-based homomorphic cryptosystems


$\mathcal{P}(B^{\text{pk}}_{\mathcal{J}})$
$r_{dec}$
$\mathcal{P}(B^{\text{sk}}_{\mathcal{J}})$
$r_{dec}$

(a) Small radius $r_{dec}$ for public basis $B^{pk}_{\mathcal{I}}$ (b) Big radius $r_{dec}$ for secret basis $B^{sk}_{\mathcal{I}}$

**Figure 4.2:** Comparison of public and secret basis $B$ of a lattice $\mathcal{L}$ [47]

In 2009 Gentry et al. [27] created the first fully homomorphic cryptosystem based on ideal lattices. An ideal lattice is a special subclass of a lattice (see lattice definition 4.1) and is defined as:

**Ideal lattice** - Let $R = \mathbb{Z}[x]/\langle f \rangle$ be quotient ring with the property to be isomorphic to $\mathbb{Z}^n$, where $f$ is an irreducible monic polynomial of degree $n$ to the integer lattice $\mathbb{Z}^n$, so that any ideal $\mathcal{I} \subseteq \mathbb{Z}[x]/\langle f \rangle$ creates an integer sublattice $\mathcal{L}(\mathcal{I}) \subseteq \mathbb{Z}^n$. Subsequently an ideal lattice is an integer lattice $\mathcal{L}(B) \subseteq \mathbb{Z}^n$, such that $B = \{g \ mod \ f : g \in \mathcal{I}\}$ for an irreducible monic polynomial $f$ of degree $n$ and ideal $\mathcal{I} \subseteq \mathbb{Z}[x]/\langle f \rangle$.

Building a cryptosystem on ideal lattices with a quotient ring result into a homomorphic property, since ideals guarantee closure of addition and multiplication (see homomorphism definition 4.1). Furthermore, Gentry et al. are proposing, in order to generate an asymmetric key pair, to create two basis $B^{pk}_{\mathcal{I}}, B^{sk}_{\mathcal{I}}$, where the ideal $\mathcal{I}$ represents the cryptographic domain. The first basis creates the public key and therefore the basis is common knowledge (see Figure 4.2a). It will be used for encryption and can be easily reduced by Gaussian elimination procedure, i.e. simple computation and no need for expensive hardware.

The second basis creates the secret key (see Figure 4.2b). The basis vectors of this secret basis are orthogonal to each other and thus enable to find a representative of a vector in this lattice. It is noteworthy, that depending on the degree of the orthogonality of the basis vector, the precision of the approximation of the modulus operation is interlinked. This is the reason why the public Basis $\mathcal{B}^{pk}_{\mathcal{I}}$ gives no hint for the underlying CV problem (see Closest-Vector-Problem **??**).

The biggest disadvantages of the original Gentry-based-approach were the enormous key sizes, especially for the public key which averaged in Gigabytes, and time consuming recrypt operations. These problems are addressed among others by Smart and Vercauteren [52], which lay the foundation for the cryptographic library LibScarab by Brenner et. al [53] that is used for our contribution in the following sections.

## 4.4 Choosing the homomorphic library

Before describing the features of the LSIM implementation in Section 4.5, an overview is given over the most prominent representative algorithms from the wide field of fully homomorphic encryption schemes. Each of them is briefly characterized concerning their implementation, their advantages and disadvantages as well as their suitability for the real world application scenarios in Section 5. The libraries listed in Table 4.1 are shortly summarized and described.

**Table 4.1:** Overview of fully homomorphic libraries

| Library | Language | Encryption Scheme | Key handling |
|---|---|---|---|
| HElib | C++ | BGV | Asymmetric |
| SEAL | C++ | Fan & Vercauteren | Asymmetric |
| LibScarab | C | Smart-Vercauteren | Asymmetric |
| FHEW | C | LWE | Symmetric |
| HE | R | Fan & Vercauteren | Asymmetric |

### 4.4.1 HELib

HELib [54] is a C++ implementation of the homomorphic encryption scheme of Brakerski-Gentry-Vaikuntanathan (BGV) [55]. It provides Smart-Vercauteren [56] cipher text packing techniques and Gentry-Halevi-Smart optimizations. In 2015, the library was extended to include bootstrapping, which allows cipher text refreshes in the encrypted domain. Very flexible and configurable but at the same time quite complex to handle for non-experts, it provides low-level routines like *set, add, multiply, shift,*. Furthermore, it is highly optimized for single instruction multiple data (SIMD) operations, which are unrewarding for our application scenarios, since the library cannot compare numbers in the encrypted domain. While testing it with small numbers for the security parameters, the memory usage for 6 multiplications on 7-bit integers amounted to over 6 GBytes. The library is still under active development.

### 4.4.2 SEAL

The Simple Encrypted Arithmetic Library (SEAL)[57] is being developed by the Microsoft Research Group and originally implemented the YASHE scheme. In the most recent version, the scheme was switched to Fan and Vercauteren [58]. In the tested version 2.1 of the library it supported only leveled homomorphism, i.e. by using modulus switching, one can only perform a predefined amount of multiplications. Since a flexible approach is needed for the targeted application scenarios, it would be counter-productive to limit the amount of operations by the library beforehand. The library itself is well documented and relatively easy to use, but it is restricted to a Microsoft Windows environment. In addition, it does not support bootstrapping and comparisons in the encrypted domain.

### 4.4.3 LibScarab

LibScarab is a C implementation of the Smart-Vercauteren scheme [52] created by Perl, Brenner and Smith [53]. It supports bootstrapping, XOR as well as AND operations on single bits. The library is easy to use, along with a clear and compact structure. The architecture is clearly structured and easy to extend. The library had to be ported to the latest FLINT 2.5.2, GMP 6.1.1 and MPFR 3.1.5 libraries, since LibScarab was last updated in 2013 and does not support multi-threading out of the box. A recrypt operation is executed automatically after every arithmetic operation.

### 4.4.4 FHEW

The “Fastest Homomorphic Encryption in the West" (FHEW) [59] is a C implementation of the Learning With Errors (LWE) scheme. It uses a symmetric key, i.e. the same key is used for encryption and decryption. It also offers bitwise encryption and NAND-Gates. Because of the symmetric key procedures this library is unsuitable for the envisioned application scenarios (see Section 5.1). This library was last updated in 2015.

### 4.4.5 Homomorphic Encryption (HE)

The Homomorphic Encryption (HE) [60] is a package for the statistic language R and implements a scheme from Fan and Vercauteren [58]. The R package itself is a wrapper for the arithmetic functions written in C/C++. While easy and intuitive to use, it lacks bootstrapping and therewith the possibility for a cipher-text-refresh, preventing running several multiplications consecutively, making it not practical for the application scenarios.

To sum up, it was decided to use the Libscarab Library over the FHEW library, because of its easy extensibility, its clear structure and asymmetric key handling. If the asymmetric key handling would not be an essential design requirement, the FHEW library would be a good choice otherwise. The HE-Lib has a lot of settings and improvements that all the other libraries lack, but the implemented application scenarios cannot benefit from any of those features. The SEAL library fails because of its limited number of multiplication as well as the HE library.

## 4.5 Functionality of the cryptographic library LibScarab

This section describes the functionality of the chosen homomorphic cryptographic library LibScarab with a proper description of encryption and decryption routines, homomorphic operations (circuits) and a brief overview of the bootstrapping routine. Smart and Vercauteren already provided the mathematical definitions and proofed its security and correctness in [52].

This work approaches the encryption schema from a practical point of view and how to best apply the findings in real world applications. Furthermore, this section facilitates for the reader the preliminaries for the extension of the cryptographic library that will be described in section 4.6.

**Table 4.2:** Parameters of key generator

| **Parameter** | **Description** |
|---|---|
| $Log_{NU}$ | sets the bit length of the coefficient of the polynomial G(x) |
| $n$ | sets the degree of the polynomial G(x) and F(x) |
| $S_1$ | sets the total amount of elements within the hint |
| $S_2$ | sets the amount of elements of the secret key within the hint |

### 4.5.1 Key generation

The key generator uses four input parameters $Log_{NU}$, $n$, $S_1$ and $S_2$ (see Table 4.2). Depending on the size of these parameters, the key generator can be one of the longest and computing-intensive routines of the library. However, since this routine is regarded as *static*, the duration is not important for the efficiency of the envisioned distributed measuring system (described in Chapter 3). The key generation lies outside of the time critical scope. Furthermore, keys are preliminaries for the communication protocol and have to be available before the protocol is initiated.

Brenner and Perl [53] described their key generation routine based on Smart and Vercauteren's schema [52] very briefly in pseudo code. Algorithm 2 gives a detailed overview of the key generation routine.

**Notation:** Polynomials are noted in upper case letters and their coefficients in lower case letters. For example, a given Polynomial $G(x)$ of degree $n$ with coefficients $(g_0 \ldots g_n)$, so that $G(x) = \sum_{i=0}^{n} g_i x^i$.

The basic idea is that $G(x)$ generates an Ideal $\mathfrak{p} = (G(x))$ in $\mathbb{Z}[x]/F[x]$, while $\mathfrak{p} = \langle p, x - \alpha \rangle$ is the two element representation of a prime ideal, where $p$ is the norm of the ideal and $\alpha$ is a root of $F(X)$ modulo $p$. For further information refer to Smart and Vercauteren"s elucidations [52]. The homomorphism reads as follows:

$$\begin{aligned} \varphi_{\mathfrak{p}_i} : \mathbb{Z}[x] &\to (\mathbb{Z}[x]/F(x))/\mathfrak{p} \\ C(x) &\mapsto C(\alpha) \; mod \; p \end{aligned} \tag{4.4}$$

This homomorphism resembles the basis for the encryption routine, later. In Algorithm 2 $F(X)$ is of the form of $x^n + 1$, since it will have no root in $\mathbb{Z}[x]$ and $n$ is of the power of two. Via the Miller-Rabin primality test the property "$p \neq 0$ is prime" will be checked. The implications of this property test are twofold: First, $F(x)$ and $(Gx)$ are coprime and $G(x)$ is irreducible like F(x). Second, $G(x)$ generates a principal ideal $\mathfrak{p}$ in $\mathbb{Z}[x]/F[x]$ (as required see 4.4).

This requirement and its check are the most costly and time intensive computations of the key generation routine. Gentry already simplified that claim from Smart and Vercauteren's schema and proved that such a strong security requirement is not necessary, instead they require only the Hermite normal form [61]. However, the implementation of Brenner et al. did not offer any optimization and works with the original approach.

From here the source code is sequential with a constant runtime. By means of Euclid's greatest common divisor (gcd) algorithm, $D_p(x)$ is calculated (see Line 6). The only root of $D_p(x)$ determines $\alpha$, which will be used for a part of the public key. The following equation 4.5 is solved by the extended Euclidean algorithm (xgcd) (see line 8):

**Algorithm 2** Key generation without bootstrapping hint

```
Input: $p = 0$                         ▷ modulus of encrypted text
       $G(X) = 0$                      ▷ Polynomial of degree n
       $F(x) = x^n + 1$                ▷ monic, irreducible polynomial in $\mathbb{R}$
       $g_i$                           ▷ Coefficient at $i \in [0; n-1]$ of the polynomial $G(x)$
Output: Public key $pk$, Secret Key $sk$

1: while p is not prime do
2:     make constant coefficient $g_i$ randomly odd,
         $g_i \in [-2^{LOG_{NU}}, 2^{LOG_{NU}} - 2]$
3:     $g_0 = g_0 + 1$
4:     $p = resultant(G(X), F(x))$
5: end while

6: $D_p(x) = gcd(G_p(x), F_p(x))$
7: $D_p(\alpha) = 0$                          ▷ determine $\alpha$
8: $r = xgcd(Z(x), G(x), t(x), F(x))$         ▷ determine $Z(x)$
9: $B = z_0\ mod(2p)$                         ▷ determine $B$,part of secret key
```

$$r = Z(x) \cdot G(x) + t(x) \cdot F(x) \tag{4.5}$$

The variables $r$, $t$ and the polynomial $Z(x)$ have to be determined, whereas $r$ is equal to the resultant p that has been calculated beforehand (see line 4). From the polynomial $Z(x)$ only the constant $z_0$ is used for calculating $B$ (see line 9), that is needed for the secret key $S_k$. All relevant parts for the key generation are done, so that the secret key $S_k = (B, p)$ and $P_k = (\alpha, p)$ can be calculated for the somewhat homomorphic encryption schema. With this version a limited number of homomorphic operations is already possible (see Section 4.2.1).

In order to be able to carry out unlimited homomorphic operations, a "'hint" for the secret key has to be integrated into the public key, so that a bootstrapping operation (see Section 4.5.4) will be possible. Via bootstrapping the noise level introduced by each homomorphic operation will be reduced through generating a new ciphertext with the same cleartext. Here it is paramount that the reencryption does not reveal the clear text at any time. A decryption and encryption routine is carried out in the encrypted domain, such that the confidentiality, integrity and availability (see CIA principle) of the cleartext is maintained. Also, the secret key will never be revealed in cleartext space at any given moment. Instead, it is kept in the ciphertext realm as well.

**Algorithm 3** Construction of bootstrapping hint

**Input:** Array $B_h$ with $S_1$ elements
Array $C_h$ with $S_1$ elements

**Output:** Two Arrays $B_h$, $C_h$ with an encrypted and interlinked hint for the secret key.

```
1:  for i = 0; i < $S_2$; i++ do
2:      $B_h[i] = floor(\frac{B}{S_2})$
3:      $C_h[i] = encrypt(1, pk)$
4:  end for
5:  $B_h[0] = B_h[0] + (B - floor(\frac{B}{S_2}) \cdot S_2)$

6:  for i = 0; i < $S_2$; i++ do
7:      Add/subtract values randomly $k \in [-p-1, p-1]$
         to/from $B_h[i]$ to keep the sum total
8:      $C_h[i] = encrypt(1, pk)$
9:  end for

    //fill array $B_h[i]$ with random values
10: for i = $S_2$; i < $S_1$; i++ do
11:     $B_h[i] = random(-p, p-1)$
12:     $C_h[i] = encrypt(0, pk)$
13: end for

    //swap values randomly between arrays $B_h$ and $C_h$
14: for i = $S_2$; i < $S_1$; i++ do
15:     $j = random(0, S_1 - 1)$
16:     $swap(B_h[i], B_h[j])$
17:     $swap(C_h[i], C_h[j])$
18: end for
```

In Algorithm 3, two arrays $B_h$ and $C_h$ are created and in the end added to the public key $P_k$. The array $B_h$ consists of the prior calculated number $B$ (see Algorithm 2) that will be divided equally in $S_2$" summands. The rest will be added to the summand $B_h[0]$. In an obfuscation step, values will be added randomly and subtracted from each summand. Here it is paramount, that the sum still equals $B$. In parallel, the array $C_h$ equal in size will contain an encrypted "1" at the same index where in the array $B_h$ a valid summand of $B$ is placed. Vice versa, an encrypted "0" indicates a random value in $B_h$. Finally, the values will be shuffled synchronously within in the arrays $B_h$ and $C_h$ and resulting in the following structure of the public and secret keys:

$$P_k = (\alpha, p, B_h, C_h) \tag{4.6}$$
$$S_k = (B, p). \tag{4.7}$$

### 4.5.2 En- and decryption

**Algorithm 4** Encryption algorithm

```
Input: $P_k = (\alpha, p, B_h, C_h)$          ▷ Public key
       $C(x)$                                   ▷ Polynomial of degree n
       $c_i$                                    ▷ Coefficient of the polynomial $C(x)$
       $M \in \{0,1\}$,                         ▷ cleartext
Output: $c$                                     ▷ ciphertext

   //set all coefficients $c_i \in [-2^{MU}; 2^{MU} - 2]$ randomly and even
1: $c_0 = c_0 + M$
2: $c = C(\alpha) mod\ p$
```

The encryption routine follows the pseudocode of Smart and Vercauteren [52] and the implementation of Perl and Brenner [53]. Nevertheless, in Algorithm 4 an overview is given. A polynomial $C(x)$ of the degree n is generated and used as an input parameter. The message will be added to the constant $c_0$ of the Polynomial $C(x)$. With the help of the public key $P_k$ it is encrypted and resembles in $C(\alpha)$ (see also 4.4) and can be evaluated by $C(\alpha)\ mod\ p$.

The decryption is very fast and only takes the secret key $S_k$ and the ciphertext $c$ as an input (see Algorithm 5). It basically evaluates the following equation, in order get the cleartext (message):

$$m = (c - (\left\lfloor \frac{c \cdot B}{p} \right\rceil)) mod\ 2. \tag{4.8}$$

**Algorithm 5** Decryption algorithm

```
Input: $S_k = (B, p)$                          ▷ Secret key
       $c$                                     ▷ ciphertext
Output: $M \in \{0, 1\}$,                      ▷ cleartext

1: $[Q, R] = \lfloor c \cdot B/p \rceil$        ▷ Quotient, Remainder
2: if $(R/p) > 0,5$ then                       ▷ Rounding parameter
3:     Q = Q+1
4: end if
5: $M = (c - Q) mod\, 2$
```

### 4.5.3 Homomorphic operations

The homomorphic operations can be represented by Boolean circuits. The closure of addition and multiplication find their real world counter part in a modulus-2-addition respectively in a logical XOR operation. The multiplication is represented as a modulus-2-multiplication or a logical AND operation. These operations can be summed up in the following equations:

$$(m_1 + m_2) \bmod 2 = (\text{encrypt}(m_1) + \text{encrypt}(m_2)) \bmod p \qquad (4.9)$$

$$(m_1 \cdot m_2) \bmod 2 = (\text{encrypt}(m_1) \cdot \text{encrypt}(m_2)) \bmod p \qquad (4.10)$$

After each homomorphic operation a bootstrapping or recrypt routine has to be launched, to reduce the noise level. This will be further explained the next section.

### 4.5.4 Bootstrapping

Gentry [61] introduced the Bootstrapping (“recrypt”) technique to extend somewhat homomorphic encryption schemes to fully homomorphic schemes by enabling unlimited homomorphic operations. After each homomorphic operation the noise level increases until a certain threshold is met, and a flawless decryption becomes impossible. Gentry came up with the idea to decrypt the ciphertext without leaving the cryptographic realm and giving up the confidentiality of the ciphertext. Instead, the resulting ciphertext is a new encrypted version of the original message with a reduced noise level and its initial value. Informally speaking, bootstrapping creates a “clean” noise free ciphertext with the same cleartext value.

In order to be able to decrypt the ciphertext with only a public key $P_k$, a hint for the secret key $S_k$ is hidden within $P_k$ as already described in Section 4.5.1. The hint consists of two arrays $B_h$ and $C_h$. In array $B_h$ parts of the secret key are hidden and distributed, while the array $C_h$ indicates the location of the hints in $B_h$ with an encrypted zero. The rest of the irrelevant indexes hold an encrypted one.

Smart and Vercauteren [52] analyzed their recrypt routine and described it in seven stages. Brenner's implementation [53] followed this approach very closely except for the rounding function. A brief overview of the routine is given in the next paragraphs, but for more details refer to [52].

As a first step, the message has to be decrypted. Instead of using the secret key directly, it has to be constructed from the hints distributed across the two arrays $B_h$ and $C_h$. :

$$m = c - \left( \left\lfloor \sum_{i=1}^{S_1} C_{h_i} \frac{(c \cdot B_{h_i} \bmod 2p)}{p} \right\rceil \right) \bmod 2 \tag{4.11}$$

Next, the values of $S_1$ have to be determined in $d_i$ as follows:

$$d_i = \frac{(B_{h_i} \cdot \; c \bmod 2p)}{p} \tag{4.12}$$

The first t bits values are taken from $d_i$ forming an $S_1 \; \times t$ matrix $A_1$, while the accuracy of $t = \lceil log_2(S_2) \rceil + 2$. The values of matrix $A_1$ are encrypted and will be homomorphically multiplied by each element of the vector $C_h$. At this point the matrix holds a binary representation of $C_{h_i}(c \cdot B_{h_i} \bmod 2p)/\; p$. In order to save homomorphic additions, the sum will be computed indirectly via Hamming weights of each column of $A_1$ using symmetric polynomials. The result matches a $t \times S$ matrix $A_2$, such that $A_2$ equals the i-th row of $A_1$ with $S$-bit precision, while $S + \lfloor (log_2(S_2)) \rfloor + 1$. Adding up the rows of matrix $A_2$ while its paramount that the current row of $A_2$ equals the current column of $A_1$ and shifting the Hamming weight $(i-1)$ of the column of $A_1$. This results in a vector $m$ representing a binary encrypted sum of:

$$\sum_{i=1}^{S_1} C_{h_i} \frac{(c \cdot B_{h_i} \bmod 2p)}{p}. \tag{4.13}$$

An element of vector $m_i$ matches an encrypted bit with the adicity of $2^{i-1}$. The resulting ciphertext $c_{new}$ after rounding can be determined as follows:

$$c_{new} = (c \bmod 2 + m_1 + m_2) \bmod p \tag{4.14}$$

The ciphertext $c_{new}$ has the same cleartext as $c$ with a reduced noise level.

## 4.6 LSIM - Extension

The following section describes the LSIM-extension (LibScarab extended for Integer arithmetics and Multithreading) of the homomorphic library LibScarab and its improvement for a practical usage in the field of Legal Metrology. It gives an overview of the added functionality and the enhanced capability of 32bit and 64bit wide homomorphic operations. *This practical implementation and extension arose as a student master thesis under the author's supervision. This work is published in [35].*

Fully homomorphic encryption schemes support two operations: addition and multiplication. Depending on the implemented scheme, also a sign change is possible, which enables subtraction. A lot of simple algorithms are based on these three operations and thus can be implemented. But nevertheless, no solution exists for comparing two integer numbers in the encrypted domain, which makes it impossible to make a decision in the encrypted domain and consequently to implement a division algorithm for example.

One possible way to implement a division algorithm is to represent numbers as fractions by saving nominator and denominator separately in order to bypass division. But this approach does not solve the problem of comparisons in the encrypted domain. It also seems impossible to render an unencrypted result from an encrypted operation, especially while using lookup tables, without giving up security and privacy. This means that all possibilities have to be decided without guessing and avoiding a shortcut like lookup tables for the division algorithm. This would lead to an improvident amount of computing power, in order to calculate all possible results in parallel.

A final decision should always be made in the unencrypted domain for the targeted algorithms. Since they cannot completely be calculated in sufficient time and without increasing the computing power enormously nor is the decryption of data in an insecure environment an option, a different approach was pursued replacing integers with binary numbers.

The LibScarab library is a good choice for the prototype, since it provides all necessary tools without adding too much complexity. In addition, it is easily configurable yet simple to modify and to extend. On top of that it is very fast. A "recrypt" procedure is executed after each operation. Thus, there is no further need for noise control to fulfill the requirement of unlimited multiplications. The library did not support multi-threading out of the box, since it was built on old versions of the libraries FLINT, GMP and MPFR. Therefore, it was ported to newer versions in order to comply with the requirements.

The following Subsections explain the extensions made to enhance the original LibScarab implementation, that had to be done, in order to build the smart meter gateway tarrifs in Section 5. First of all, to meet the time constraints the library was enhanced to take advantage of multithreading (see Section 4.6.1). In Section 4.6.3, advantage is taken of the binary representation of numbers from the original library, in order to be able to implement a zero test of encrypted integers. This test enables simple decisions in the encrypted domain (see Section 4.6.5) in conjunction with a simple comparison of encrypted integers (see Section 4.6.4). These extensions are paramount and essential, in order to implement the smart meter gateway tariffs. Furthermore, the enhancements of the addition (see Section 4.6.6), multiplication (see Section 4.6.7) and division (see Section 4.6.8) operations contribute to a performant and improved implementation of the smart meter gateway tariffs.

### 4.6.1 Multithreading

The complex and time intensive computation, that homomorphic encryption requires makes it a perfect candidate for parallelization via multithreading. At a first glance the Bootstrapping routine would be perfect for parallelization since it is a very complex and the most often repeated routine because it will be executed after each homomorphic operation. However, this routine offers only small parts for distributing the computation among parallel working threads and is followed by stark sequential, non-parallelizable stages. The overhead for managing the thread generation and -synchronization does not balance the time savings via parallelization and is just not cost-efficient enough. Instead, it was decided to work at the hierarchy level of encrypted bits and parallelize each homomorphic integer operation, as well as at the application level for the tariff models. In order to get optimal results to reduce execution time for tariff models, it is necessary to employ multithreading at the lowest execution level possible. The homomorphic integer operations offer many regular parts and easy parallelizable structures at the level of each Boolean operators. The reduction of execution time and acceleration of computation is won via multithreading at the level of each Boolean circuit. The computation time was reduced in a first iteration of the implemented library by a factor of 7 and in a second iteration it was further reduced by factor 2. For more details refer to Section 5.

Several tools are available to realize multithreading within applications. The easiest and rudimentary way is using Posix Thread (pthread). This library is independent of a programming language, operating system and execution model. Furthermore, it allows a fine grained control of the thread generation, synchronization and termination. However, pthread introduces a huge overhead in maintenance and controlling dynamic threads is complicated to handle. A more suitable solution is the OpenMP library that automates the generation and synchronization of threads. Via #pragma preprocessing directives even single thread environments are able to execute the code. The OpenMP library is natively available via the Microsoft Visual C++ compiler and reduces the difficulty of platform independent programming. For the LSIM extension OpenMP is used to implement the multithreaded environment.

Thread safety was of the main aspects of the parallelization for each routine of the LSIM extension. While parallelizing calculations on several threads, the order of interim results can be paramount for the end result. Thus it is important to avoid race conditions and nondeterministic programming errors. The existing implementation of LibScarab and its libraries are outdated. Therefore it had to be adapted to be compatible with the latest libraries. While the old version of FLINT 1.6 could not guarantee thread safety for all its methods, a switch to the newer version 2.5.2 had to be made to assure thread safety for all methods. In this context, the application interface had to be updated and additionally the extended Euclidean algorithm of the LibScarab library was replaced by the FLINT 2.5.2 library. Furthermore, through dependencies to the MPFR and GMP libraries an additional update (MPFR 3.1.1 $\mapsto$ 3.1.5, GMP 5.1.1 $\mapsto$ 6.1.2) was necessary.

### 4.6.2 Boolean operators

The LibScarab-Library offers two basic homomorphic arithmetic operations; the modulo-2-addition which corresponds to an XOR-gate and a modulo-2-multiplication that corresponds to an AND-gate. With these two operations it is possible to derive all the Boolean functions and, as a consequence, to provide arithmetic and logical operations on encrypted integers, which are represented as arrays of encrypted bits. An OR operation can be realized with De Morgan's Law (see 4.15) and the binary complement of the encrypted number which corresponds to toggling bits in the encrypted domain can be realized via XOR gate (see 4.16). The addition, subtraction and multiplication operations for integers had to be reimplemented according to the chosen binary word sizes (i.e. 32 bit and 64 bit length see Section 4.6.6).

$$A \vee B = \neg(\neg A \wedge \neg B) \tag{4.15}$$

$$((1 \oplus A) = \neg A). \tag{4.16}$$

### 4.6.3 Zero-Test of encrypted integer

This binary approach yields the opportunity to implement a zero-test as a simple bitwise $\vee$ on all bits of the represented number. The result of this operation should be complemented to return an encrypted 1 in case the examined number was 0. This corresponds to a logical NOR operation with 32 bit inputs. The encrypted result can be directly used as an input for further encrypted arithmetic and logical operations. To verify the computed result, it has to be decrypted in the end.

### 4.6.4 Comparison of encrypted integer

A simple implementation of a comparator consists of the subtraction of both input parameters $(A - B)$ and of the sign-evaluation of the result. This operation delivers two possible results $A < B$ or $A \geq B$. The latter needs to be checked with the help of the zero-test, so that it can be distinguished between $A > B$ and $A = B$. If this clarification is not needed, this approach can be reduced to the calculation of borrow-bits only, since the result of the subtraction itself is not important.

### 4.6.5 Simple decision of encrypted integer

It is significant to highlight that the result of a decision should remain in the encrypted domain. Thus, it is not possible to externally influence the program flow. Nevertheless, simple algorithmic constructs are feasible based on the result of the comparison or zero-test operator. These are *source* and *destination selections*. The source selection, on the one hand, is represented by the following C-construct, which is similar to an if-then-else-construct with the constraint that only data flow can be controlled:

$$Y = (condition)?A : B \tag{4.17}$$

To implement this decision a 2:1-multiplexer is required, which can be described as:

$$Y = \neg C \cdot A + C \cdot B \tag{4.18}$$

where $C$ is the output of the zero-test, the comparison operator or any other possible Boolean equation.

The destination selection, on the other hand, consists of a demultiplexer and binary adders. The selection is triggered by a Boolean equation (e.g. comparison or zero-test). After the output selection operation, the selected output or destination contains the input number and all the other outputs are set to an encrypted 0. All outputs will be added to the corresponding registers. All registers are modified, but only one will be increased by the input value of the demultiplexer. All the others will be increased by an encrypted 0. It is important to mention that the addition of an encrypted zero always produces a different encrypted version of the same number. Hence it is not possible to say which of the registers are actually changed by a FHE-operation.

### 4.6.6 Addition and subtraction of encrypted integer

The implementation of arithmetic circuits is done with special regard to multithreading. A particular challenge was to find a solution with a small number of gates and a small circuit depth at the same time. An analogue problem is known in the field of digital design as “Area-Speed-Tradeoff”, since a lower circuit depth usually results in a higher number of gates used. Each additional gate needs computing time, thus fast digital circuits have often poor performance executing homomorphic operations. Especially for multiplication and division some advanced algorithms exist, but due to the massive use of multiplexers or lookup-tables these solutions are not reasonable with respect to the execution time.

Three different versions of an adder were implemented and compared. A Carry Look Ahead Adder (CLA) [62] with block size of 4 and 8 bit, also known as “fast adder" proved to be the slowest adder within the constraints of this thesis, especially for the implemented tariff applications (see Section 5). The Carry Select Adder (CSA) [62] rendered mid-range performance. The modified Ripple Carry Adder (RCA) [62] revealed the shortest execution time, since the operations of the first half adder were executed completely in parallel. The same adder could be used to implement subtraction, but this would contain an array of XOR-gates to form the complement. Considering performance, the subtraction was outsourced to a separate routine, in order to save time during addition operations.

### 4.6.7 Multiplication of encrypted integer

Fast multiplications using higher radix solutions [62][63] require a massive use of multiplexers or/and operand recoding which are costly in size as well as time consuming.

In case of conventional binary multiplications, two problems can be identified: 1.) the generation of the partial products, 2.) their summation. While the generation of partial products can be carried out completely (bitwise) in parallel, the summation can only be partially parallelized. A tree structure of conventional word adders provides insufficient performance. For this reason, a multi-level bit-wise tree adder is used here (see Figure 4.3).

The main principle of a tree adder is that a full adder is used to add three bits. At the output, one sum bit and one carry bit are generated. The resulting number, consisting of carry bits, is shifted one place to the left. Due to the fact that there is no carry propagation in the case of the 32 partial products of the multiplication of two 32 bit numbers, 30 ($x_0$ *to* $x_{29}$) of them can be processed by bit-wise full adders in a bit-parallel manner in the first stage. The resulting 20 numbers are supplemented by $x_{30}$ to deliver 21 numbers and are processed in the next stage. After several iterations, there are only two numbers left which are added together by a conventional adder. The difficulty here is the bitwise addition of the shifted partial products with respect to the significant bit position, which requires manual optimization and proper planning of the multi-threaded operations. This approach is the fastest solution found to our problem in the encrypted domain (see Section 5 for performance results).

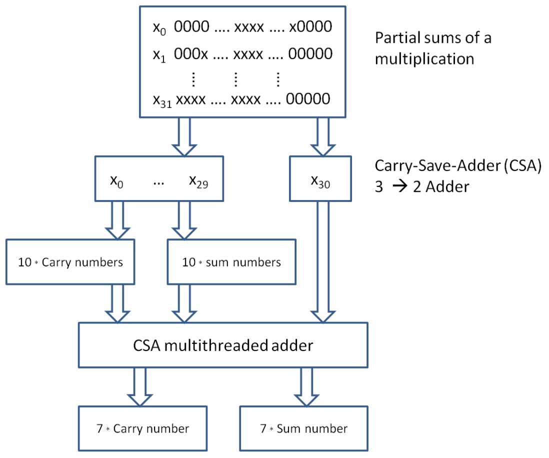

x0 0000 .... xxxx .... x0000
x1 000x .... xxxx .... 00000
x31 xxxx .... xxxx .... 00000
Partial sums of a multiplication
x0 ... x29
x30
Carry-Save-Adder (CSA) 3 → 2 Adder
10 ◦ Carry numbers
10 ◦ sum numbers
CSA multithreaded adder
7 ◦ Carry number
7 ◦ Sum number

**Figure 4.3:** Tree structure of the optimized adder part

### 4.6.8 Division of encrypted integer

The division operation is the most difficult one to parallelize because of inter-step dependencies. One possible improvement leading to a reduced number of iterations is the use of high-radix number systems to perform the division. Unfortunately, the complexity of each iteration is also increased in this case. Advanced higher radix algorithms like Sweeney-Robertson-Tocher (SRT) division uses lookup-tables and/or a lot of multiplexers [62][63]. These are very expensive in case of a FHE software implementation, so that a simpler solution was investigated.

The simplest way to implement the division is the well-known paper and pencil method adapted to the binary number system also known as restoring-division [62]. The non-restoring algorithm has been verified in [64]. The divisor is subtracted from the shifted dividend, in case of a negative remainder the divisor is added back to the remainder and the result will be shifted. The restoring operation implicates one extra addition or use of a multiplexer in each iteration. In contrast to the restoring division the non-restoring-algorithm requires only one addition or subtraction in each iteration. The underlying concept can be easily derived from the restoring algorithm. In case of a negative remainder after one subtraction the divisor (D) should be added back to the remainder (R) and the result (S) shifted to the left, i.e. $S = 2 \cdot (R + D)$.

In the next iteration D will be subtracted again: $2 \cdot R + 2 \cdot D - D = 2 \cdot R + D$, which is equal to the simple shift of the negative remainder of the first iteration and replacing of the subtraction by an addition in the next iteration. Addition and subtraction are performed by the same Boolean circuit (see Figure 4.4). The sign-bit of the result is used to select the operation (addition/subtraction) for the next iteration.

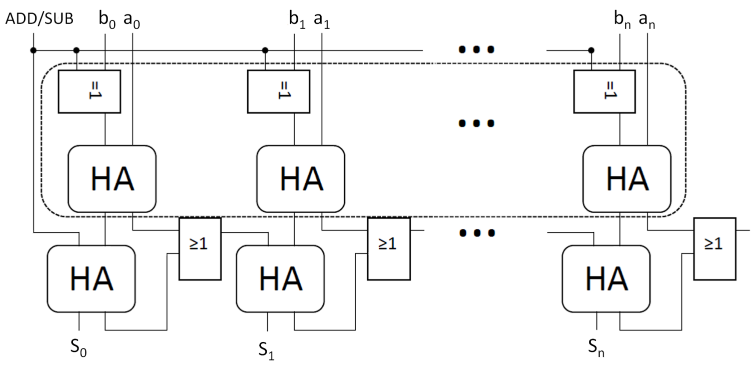


**Figure 4.4:** Multi-threaded combined adder and subtractor

## 4.7 Summary

In this section the field of fully homomorphic encryption (FHE) was introduced and their mathematical groundwork briefly explained. The base problems such as shortest vector problem (SVP) and closest vector problem (CVP) for lattice-based cryptography were introduced and their significance emphasized. Advantages of lattice-based cryptography in comparison to traditional cryptography were summed up. The contribution was highlighted by extending the FHE library LibScarab for zero-test, comparison, simple decisions for encryption integers. Moreover, basic arithmetical operations, such as addition and multiplication were extended for 32 and 64 bit usage, as well as division and subtraction were added to the library content. These contributions were necessary to allow implementing encrypted tariff models for smart meter gateways (SMGW).

# 5 Evaluation & Utilization

*"In the real world, nothing happens at the right place at the right time."*

—*Mark Twain*

While Chapter 4 explained the foundation of FHE and highlighted the authors contribution, this chapter focuses on the applied application scenarios for smart meter gateway tariffs according to the German Federal Office for Information Security (BSI). Four tariffs will be briefly explained and the algorithmic requirements that are needed to implement these scenarios. The results of the 32 bit and 64 bit-homomorphic operations along of the yielded results of the application scenarios are presented.

## 5.1 Tariff applications

Even though homomorphic encryption usually aims at ensuring data privacy [27], it may be used in other areas and for other purposes as well. If an attacker is unaware of the actual values of the data currently being processed, intentional manipulation is no longer possible. Instead, random changes to data and likewise random manipulation of the executed algorithm are the only aims an attacker may achieve. If homomorphic encryption is carefully combined with testing a running algorithm via precomputed test data, even such random effects may be detected. Subsequently, the scheme detailed here may be used to achieve a certain degree of robustness towards algorithm and data manipulation, too. This is especially useful in the area of Legal Metrology, where all parties involved in a transaction are considered untrustworthy and the only trust-anchor is the measuring instrument itself.

As indicated in the introduction, Legal Metrology covers all areas of measurements where lawmakers consider the outcome of a measurement to be crucial for consumer protection. In the following paragraphs, the different tariffing scenarios will be examined in more detail. Tariffing generally refers to the process of price calculation based on one or more determined measurement values consisting of a physical quantity together with the appropriate SI unit [65]. Here the term will be used in the context of smart meters for electrical energy, but the concept may easily be applied to other areas of Legal Metrology as well.

While the measuring of commodities such as electrical energy, gas, water and heat are all regulated in the MID, national law may prescribe additional constraints if the measurements, for instance, touch upon informational privacy or other aspects subject to national legislation. In Germany, due to the implementation of the aforementioned Directive 2009/72/EC, the federal office for information security *Bundesamt für Sicherheit in der Informationstechnik (BSI)* has published a technical requirement document (TR) [34] with an associated protection profile [66]. While the first mainly covers compatibility requirements and secure communication protocols for the SMGW, the latter is only focused on the SMGW's integrated Hardware Security Module (HSM). Apart from defining communication protocols and the general environment of an intelligent measuring system, the TR also lists all approved tariff application scenarios (TAF) that may be used in an intelligent measuring system. These are of utmost importance for the SMGW, as it is usually in charge of connecting meter and time data (time stamping) and is also responsible for price calculation.

Out of the 12 TAFs defined in the TR, four will be examined and implemented in a secure cloud solution in this thesis. The four scenarios have been selected as they constitute a representative application of tariffing that may also be found in other measuring systems.

In this section, a number of common tariff models, which constitute a representative set of algorithms used in measuring instruments, are examined and requirements for the homomorphic encryption scheme detailed in Chapter 4 will be derived.

**Tariffs with low data usage - TAF I** Tariffs with low data usage, where energy is always billed with the same price and collected data are only sent at the end of the billing period requiring no external trigger.

Time-dependent tariff - TAF II Time-dependent tariffs, where energy is billed with different tariffs according to the time at which a certain amount of energy is consumed. The switching between tariffs is time-dependent but the switching points are static.

Power-dependent tariff - TAF III Power-dependent tariffs, where the price does not depend on the total energy consumed but on the current power consumption (energy per time interval) The switching into different price categories is therefore done according to the value of the current measurement result.

Consumption-dependent tariff - TAF IV Consumption-dependent tariffs, where a new price is used when a certain energy budget has been consumed. The budgets are statically predefined and the condition for assigning new measurement values to the next price category has to be checked regularly.

## 5.2 Required logical and arithmetic operations

The implemented library LSIM and its extension described in Section 4.5 aims at realizing the SMGW's tariffing functionality in a way that can be run on any system with suitable computation capacity without having to realize additional protective means. While SMGWs are very unlikely to be realized in the cloud any time soon due to the hardware requirements of the TR, the approach may easily be applied to many other measuring systems that all perform similar price calculations as listed below:

- addition, subtraction, multiplication, division,
- comparison,
- input-dependent source selection,
- input-dependent destination selection.

The addition operation is, of course, needed to add new energy values to an existing tariff register. Subtraction, division and negative numbers are likewise needed to calculate the current energy flow (the power) based on consecutive readings of a cumulative meter. The multiplication operation is required when a tariff and an energy amount are combined to form a price to be paid. Comparisons of input values are needed to realize input- and time-dependent switching statements, in order to be able to decide in which register the accumulated measurement result has to be saved based eiher on energy consumption (see TAF 3) or to distinguish day and night tariffs (see TAF 4) as described in Section 5.1.

### 5.2.1 Required time for processing.

Two separate tasks performed by the SMGW need to be distinguished: One is the accumulation of data coming continuously from the electricity meters. The other one is the monthly reading of the resulting registers. Data coming from smart meters will be accumulated within the SMGW to which the meters are connected. According to the current design of the SMGW, a maximum of 32 meters can be connected to the gateway simultaneously. Each individual meter sends a new measurement value every 15 minutes. In order for the gateway to efficiently cope with the arriving amount of data, every single request for data accumulation subsequently needs to be processed in under $\frac{15\times 60sec}{32} = 28sec$. Parallelization of the accumulation process is not feasible or wanted since there are inter-dependencies between measurements which may have an effect on the chosen register for subsequent values. The monthly retrieval of accumulated register values from the SMGW is not time-critical as the reading only works on static data that does not change during reading. The retrieval can thus happen in a separate process.

## 5.3 Evaluation of homomorphic operations

This section describes the experimental comparisons that were conducted on a Linux server with an Intel Xeon CPU E5-2620 v3 @2.40 GHz, 24 cores and 64 GB of RAM with a conventional hard-drive. In the first two subsections, key generation, recrypt operation and single arithmetic operations of FHE are compared, after improving and extending LibScarab as described in Section 4.6. Subsection 5.3.4 comprises results of the application scenarios TAF 1-4 that were already outlined in Subsection 5.1. The speedup and efficiency are calculated for both single operations as well as for the tariff applications.

The relative performance gain achieved by parallelization (speedup) can be measured and is defined in [67] as a metric. This can be written as:

$$S(n) = \frac{T_s(n)}{T_p(n)} \tag{5.1}$$

where $T_s$ is the execution time of the best sequential algorithm for solving the problem and $T_p$ marks the execution time of the parallelized algorithm using $n$ processing units.

The efficiency is also defined by [67] as a metric that measures the fraction of time in which a processing unit is usefully employed. This can be asserted as:

$$E(n) = \frac{S(n)}{n} \tag{5.2}$$

**Table 5.1:** Overview of key geometry and performance gain in comparison to Brenner et al. [53] key generation and recrypt operations in parenthesis.

| key geometry $[\|\alpha\|, S_1, S_2]$ | key size $P_k/S_k$ [kB] | keygen [s] | speedup | recrypt [ms] | speedup |
|---|---|---|---|---|---|
| 384/16/05 | 1.8/30 | 1.1 (17) | 15.5 | 34 (263) | 7.7 |
| 384/32/16 | 1.8/60 | 1.17 (14) | 11.9 | 90 (307) | 3.4 |
| 384/64/16 | 1.8/120 | 1.24 (15) | 12 | 181 (684) | 3.7 |
| 2048/64/16 | 9.6/626 | 516 (3180) | 6.1 | 670 (1350) | 2 |
| 4096/64/16 | 18/1250 | 5204 (14278) | 2.7 | 1660 (3280) | 1.9 |

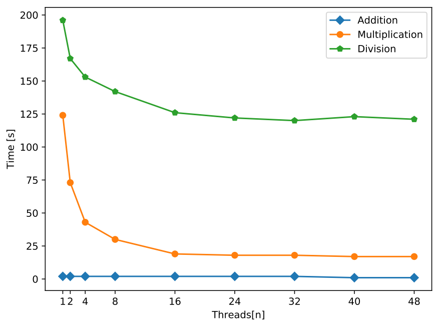
where $S$ is the speedup for the algorithm (see Equation (5.1)) and $n$ marks the number of processing units.

### 5.3.1 Key generation and recrypt operation

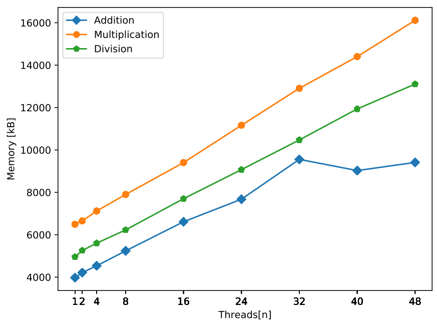
While parallelizing the basic arithmetic operations, e.g. addition and multiplication, it was determined that the speedup factor is not significant for creating a monic and irreducible polynomial $F(x)$ with a resultant $p$ being prime (see KeyGen [52]) in a multi-threaded environment compared to a single threaded one, i.e. no relevant time gain for key generation is realized. The author assume that performance gain is lost due to the randomized generation of the polynomials and their evaluation, because of the great spread of the results and the necessary synchronization of the threads. Thus, the times stated in Table 5.1 for key generation and recrypt operation are single threaded. Nevertheless, in comparison to Brenner's reference implementation [53] and the stated times for key generation for 384-bit key length a speed up of 15.5 could be yielded, i.e. instead of 17s it only took 1.1 second to generate a key. The main factors are optimizations in the implementation, modernized libraries as already explained in Section 4.6 and faster hardware than in 2012. The recrypt operation achieved a speed up factor of 7.7 for 384-bit key length, i.e. a recrypt costs only 34ms instead of 263ms. The amount of disk space for public and private keys do not distinguish from Brenner's data.

### 5.3.2 Arithmetic operations

By parallelizing the arithmetic operations, the greatest benefit was earned within multiplication by a factor of 7.29. In Figure 5.1 one can see the asymptotic characteristic of the optimization for the time usage. While executing a single thread utilization took 124s for a single multiplication, a 48 threads utilization only took 17s for a single multiplication. Considering memory



(a) Single 32 Bit-FHE-operations over time

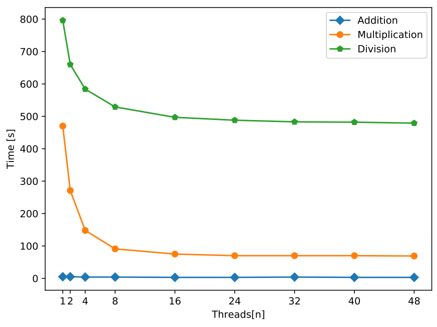


(b) Single 32 Bit-FHE-operations and the memory usage

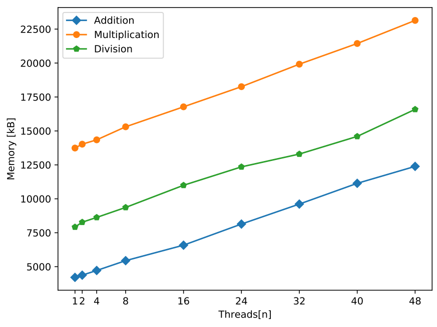
**Figure 5.1:** Plot of 32 Bit-FHE-operations (Add, Mult, Div) on a Server with an Intel Xeon CPU E5-2620 v3 @ 2.40 GHz and 64GB RAM.

usage, the optimum is reached by utilizing 16 threads with 19s for a single multiplication with a calculated efficiency gain of 41 % (see Equation 5.2) and a speedup of 6.53. The least significant benefit from parallelization was realized for the addition operation. It is already the fastest operation in the encrypted domain needing only two seconds for one addition for a single thread while it could be pushed down to one second for a single addition utilizing 48 threads. The efficiency optimum is reached here by using only two threads with a 50% efficiency and a speedup of 1 needing two seconds.

The division is traditionally a very complex arithmetic operation which often has its own compartment on modern CPUs in order to optimize its performance. Bearing this in mind for the software implementation in the encrypted domain, the benefit through parallelizing the division operation was achieved by a factor of 1.6. This means that a single thread for one division



(a) Single 64 Bit-FHE-operations over time



(b) Single 64 Bit-FHE-operations and the memory usage

**Figure 5.2:** Plot of 64 Bit-FHE-operations (Add, Mult, Div) on a Server with an Intel Xeon CPU E5-2620 v3 @ 2.40 GHz and 64GB RAM.

**Table 5.2:** Overview of 32 bit operation results

| | n | t (sec) | S(n) | E(n) in % |
|---|---|---|---|---|
| **Add** | 1 | 2 | - | - |
| | 2 | 2 | 1 | 50 |
| | 48 | 1 | 2 | 4 |
| **Mult** | 1 | 124 | - | - |
| | 16 | 19 | 6.5 | 41 |
| | 48 | 17 | 7.3 | 15 |
| **Div** | 1 | 196 | - | - |
| | 16 | 126 | 1.6 | 10 |
| | 48 | 121 | 1.6 | 3 |

took 196s while this was reduced to 121s utilizing 48 threads. The optimum for this operation is reached using 16 threads with a speedup of 1.6 with a 10 % efficiency. Similar results could be yielded for 64 bit arithmetic operations as seen in Figure 5.2. Obviously, more memory was needed for the single arithmetic operations due to the nature of the bigger operands. Again, the multiplication benefited the most by a factor of 6.8, needing almost 8 minutes (470s) for a single threaded multiplication compared to 69s using 48 threads. The optimum is reached using 16 threads with a speedup of 6.3 and an efficiency gain of 39 % needing 75s. Additions in the 64 bit space are optimized by a factor of 1.6, i.e. needing 5s for one addition using one thread compared to 48 threads lasting 3s. The optimum is reached for 4 threads with a speedup of 1.3 and a 31 % efficiency. Again addition is the fastest operation.

**Table 5.3:** Overview of 64 bit operation results

| | n | t (sec) | S(n) | E(n) in % |
|---|---|---|---|---|
| **Add** | 1 | 5 | - | - |
| | 4 | 4 | 1.3 | 31 |
| | 48 | 3 | 1.7 | 3 |
| **Mult** | 1 | 470 | - | - |
| | 16 | 75 | 6.3 | 39 |
| | 48 | 69 | 6.8 | 14 |
| **Div** | 1 | 796 | - | - |
| | 16 | 497 | 1.6 | 10 |
| | 48 | 467 | 1.7 | 3 |

For the division the same factor of 1.6 could be reached as for the 32 bit operands. Speaking in absolute numbers it is still a huge difference from roughly 13 minutes (796s) for a single threaded division compared to about 8 minutes (479s) using 48 threads. The optimum is reached using 16 threads with a speedup of 1.6 and 10 % efficiency needing 8.5 minutes (497s). An overview for the 32 and 64 bit arithmetic operation results can be seen in Table 5.2.

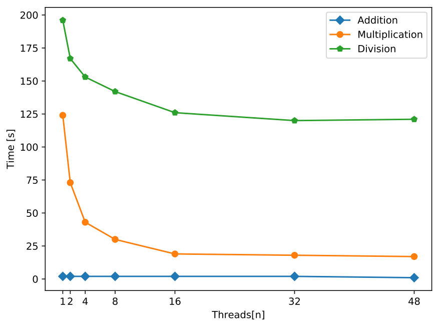
### 5.3.3 Comparison of the stack and heap implementation of LSIM

The implementation of the LSIM extension is still under active development. While writing this thesis, the work on the heap implementation for the memory handling was finalized. Switching from a stack implementation with a stark tailored use case approach to a more generalized and dynamic prototype implementation, improved the runtime of arithmetic operations drastically. Through this reimplementation, buffer overflows, memory leaks and index out of bound errors were fixed and do not affect the homomorphic operations and their runtime in an unauthorized way anymore.

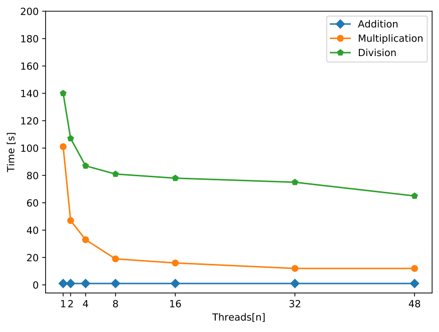
In direct comparison the speed of all 32 bit arithmetic operations was improved up to 50% on the same machine (see Figure 5.3). The homomorphic addition routine was already the fastest operation but was improved for all threads ranging from 1 to 32 needing only 1s gaining 50% in comparison to the already presented results. More interesting are multiplication and division. Here, the computations gain speed and the savings are more impressive. The new implementation is 36% faster for multiplication employing only two threads, needing 47s compared to 73s. Eight threads need only 19s compared to 30s and are 37% quicker. The maximum of 48 threads gain 12% against the old implementation and need only 12s compared to 17s.

The division routine is, because of its complexity, the longest running operation. Nevertheless, with only 4 threads employed a gain of 44% could be achieved, needing only 87s compared to 153s. The division routine almost cracks the psychologically important barrier in performing faster than within 60s. However, employing 48 threads resulted only in 65s compared to 121s gaining 46%.

These last minute results are encouraging in further optimizing the prototype implementation. They show that a lot of computational optimization can still be made. Keeping in mind that the first breakthrough of fully homomorphic encryption was in 2009 and the first fully homomorphic calculations lasted longer than 48 hours, it is quite impressive what the cryptographic community achieved so far. However, this is still an early stage of homomorphic encryption.



(a) Old implementation over time



(b) New implementation over time

**Figure 5.3:** Comparison of 32 Bit-FHE-operations (Add, Mult, Div) on a Server with an Intel Xeon CPU E5-2620 v3 @ 2.40 GHz and 64GB RAM.

### 5.3.4 Results of application scenarios

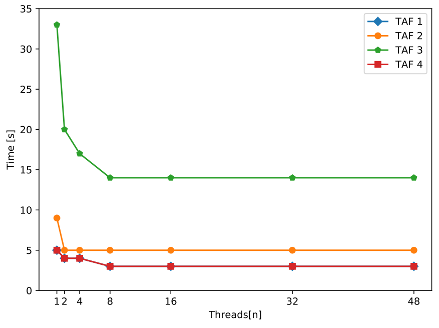
In contrast to the tests performed in Subsection 5.3 where only arithmetic operations are measured, these results cover the combination of arithmetic operations and comparisons applied to the application scenarios described in detail in Subsection 5.1. A recrypt is included after each arithmetic operation to reduce the noise.

The calculations for the application scenarios are split into accumulating the measurement data (see Figure 5.4) and summing them up on demand (see Figure 5.5), e.g. at the end of the month. The latter includes the more complex arithmetic operations and comparisons in the encrypted domain.

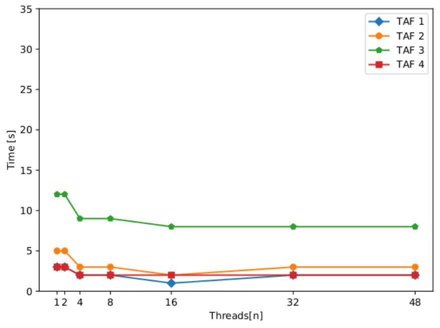
While TAF 1 and 4 performed very similar in accumulating measurement data (see Figure 5.4a) in respect to time and utilizing threads, the gain is 1.3 needing about 5s for a single thread to only 3.8s utilizing 48 threads. The optimum is reached using only 4 threads with a speedup of 1.3 and about 30% efficiency needing 4.2 and 4.3s.

TAF 2 gained a factor of 1.7 needing single threaded 9.3s and for 48 threads 5.6s. The optimum is reached using two threads with a speedup of 1.8 and a 91 % efficiency needing 5.4s to accomplish this task.

TAF 3 is the only scenario where a lot of comparisons and decisions were performed additionally, thus the time difference. A gain of factor 2.4 was yielded for a single thread needing 33.4s compared to 48 threads consuming only 14s. The optimum is reached using 8 threads with a speedup of 2.2 and 28 % efficiency needing 14.9s to finish. For all TAFs, parallelization helped to push execution time below the required boundary of 28s. The new implementation is 65 % faster needing only 12s for the single threaded accumulation process and 50 % faster with 4 threads needing only 8s for TAF 3



(a) Old implementation over time



(b) New implementation over time

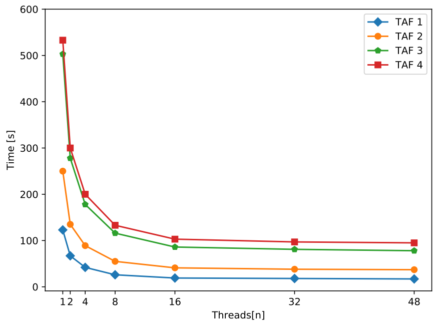
**Figure 5.4:** Comparison of accumulating measurement data on a Server with an Intel Xeon CPU E5-2620 v3 @ 2.40 GHz and 64GB RAM.

(see Figure 5.4b).

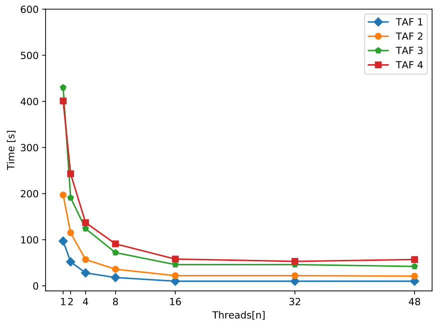
For the monthly "read out" of the summed-up measurement data the application scenarios (see Figure 5.5a) differ more than for the accumulation process. TAF 1, low data usage, is the fasted and simplest one gaining factor 7 for a single thread using 123.3s compared to 48 threads consuming only 17.6s. The optimum is reached using 16 threads with a speedup of 6.3 and a 39 % efficiency needing 19.6s to finish.

TAF 2, time-dependent tariff, records a gain of 6.7 for a single thread using 250.6s compared to 37.3s utilizing 48 threads. The optimum is reached using 16 threads with a speedup of 6 and a 38 % efficiency needing 41s.

The TAF 3, power-dependent tariff and TAF 4, consumption-dependent tariff, are more complex thus take around 9 minutes (503.3s / 533s) for a single thread and for 48 threads using around 1.5 minutes (78,6s / 95s).



(a) Old implementation over time



(b) New implementation over time

**Figure 5.5:** Comparison of "read out" measurement data on a Server with an Intel Xeon CPU E5-2620 v3 @ 2.40 GHz and 64GB RAM.

The performance gains are factor 6.4 and 5.6, respectively. The optimum is reached for both using 16 threads with a speedup 5.8 and 5.2 as well as a 36 % and a 32 % efficiency needing 86s and 103s to finish. The new implementation gains 150s in single threaded read out process and is 28 % faster needing only 400s for TAF4. With 4 threads employed the read out process for TAF4 is 38 % faster and needs only 120s (see Figure 5.5b).

## 5.4 Summary

This chapter covered the smart meter gateway tariffs (TAF) according to the German federal office for information and security (BSI) and their technical requirement document (TR)[34] with an associated protection profile [66]. The required logical and arithmetic operations were briefly summarized to be able to implement encrypted tariff models. Improving the used FHE library libScarab through multithreading and partly reimplementing basic FHE operations, the computation times for all the FHE operations was improved. Key generation improved by a factor of 15.5 needing only 1.1 instead of 17s and recryption was pushed down from 263ms to only 34ms.

Through the effort of reimplementing the memory handling and switching from a stack to a heap implementation a tremendous acceleration of homomorphic operations could be made possible. All arithmetic operations could benefit, especially interesting are multiplication which is 36 % faster than beforehand. Likewise all implemented tariff applications are benefiting from this speed up as well.

An overview of the arithmetic operations is given and summarized in a table. Through parallelizing the FHE library the time requirements can be pushed down for all TAFs below the required boundary of 28s [35]. This proves that fully homomorphic encryption can be used nowadays for real-world examples. Implementing FHE for the secure Cloud Computing reference architecture shall secure the measurement data through their lifecycle in the distributed system. Nevertheless, to prove the integrity of the distributed measurement system a condition monitoring approach is needed.

# 6 Risk Assessment

*"Named must your fear be before banish it you can"*

*—Yoda, Star Wars: The Empire Strikes Back.*

IN THIS CHAPTER, a risk analysis is applied to the presented Secure Cloud Reference Architecture to fulfil the legal requirements (see Section 2.2). This risk analysis is based on software risk assessment for measuring instruments under legal control proposed by WELMEC Working Group 7 [9]. By objectifying the derived probability score for identified threats while following at the same time the guidelines of ISO/IEC 27005, ISO/IEC 15408 and ISO/IEC 18045, this risk assessment method enables comparability and standardizes the otherwise highly subjective assessment process. Furthermore, potential countermeasures are identified and quantified using Attack Probability Trees (AtPT) [1] for derived assets to be suitable protected.

## 6.1 Derivation of assets to be protected

Esche et al. [9] developed a risk assessment method based on ISO/IEC 27005 [68] and WELMEC Risk Assessment Guide [69]. The approach consists of three stages and is shortly summarized in the following paragraphs. This method is applied to the secure cloud reference architecture (see Section 6.2).

Every measuring instrument that undergoes conformity assessment has to fulfill the essential software requirements listed in Annex I of the MID before being put on the market (see Section 2.2.2).

There are eight essential software requirements of the MID summed up in Table I [9] that have to be fulfilled. Requirement 7.6 and 8.3 which regulate

**Table 6.1:** Formal Definition of Threats

| ID | Threat Intention | Description |
|---|---|---|
| B1 | Integrity of transmitted measurement data | An attacker alters measurement data during transmission. |
| B2 | Authenticity of transmitted measurement data | An attacker creates tampered measurement data, that will be assigned wrongly to a verified measuring instrument. |
| B3 | Evidence of an intervention | An attacker prevents legally relevant events from being registered in the logbook. |
| B4 | Integrity of Parameters | An attacker alters persistence saved parameters, e.g. connection parameters. |
| B5 | Availability of the Logbook Service | An attacker prevents a legally relevant service from answering requests |

the identification and presentation of the legally relevant software are already addressed by the Verification Monitor introduced and explained in Section 3.3.1. These requirements are not part of the risk assessment, since these are not challenging assets to protect. Further the requirements 10.1 and 10.2 that regulate the indication of the measurement result are not part of the presented architecture and thus relevant for the risk assessment. Requirement 11.1 that regulates the record of measurement result is already addressed by the introduced communication protocol for creating a signature of homomorphic encrypted and secured measurement data in Section 3.1.5. Thus only three relevant assets remain here that are noteworthy to be protected for the introduced reference architecture, i.e.

1. measurement data
2. software that is critical for measurement characteristics
3. metrologically relevant parameters,

and are deduced from requirements 8.3 and 8.4 that regulate the evidence of an intervention as well as the protection of of measurement data, software and its metrologically relevant software parameters. B3 and B5 are linked to secondary assets of the second asset listed above. For each of the three requirements, the MID requires integrity and authenticity protection. Consequently, these assets must be secured against intentional or unintentional changes. By fulfilling this demand, integrity and authenticity of these assets are guaranteed. In addition, the MID requires evidence of an intervention, i.e. events registered in a logbook, to be available during verification.

### 6.1.1 Threat definition

A threat is any invalidation of a security property of a given asset. To define a threat, aside from the asset definition, several attacker models should be taken into account, for example, inside attacker and external attacker. Usually the market participant with the highest skill level can be used as a reference model. Additionally, different access levels and their associated roles within a measuring instrument take an important part in the risk assessment. In Table 6.1 five threats are given and a short description of what an attacker wants to achieve. The assets itself will be further described in separate tables, where the attack vectors (technical steps needed to implement a threat) are broken down into atomic attacks with a time, expertise, knowledge, window of opportunity and equipment column that are individually scored (see Section 6.2), according to [70]. This procedure has the advantage of objectifying the risk assessment procedure based on scores for well-defined features of any attack. This enables manufacturers and Notified Bodies alike, to be able to compare the same threats for different measuring instruments.

### 6.1.2 Identification of attack vectors

The second risk assessment phase is the least formalized stage. It starts with the examination of the manufacturer's documentation of the measuring instrument. Followed by creating a collection of possible attack vectors, needed to realize the prior identified threats from stage one. The collection comprises attack vectors reaching from simple to very complex structured attacks.

### 6.1.3 Calculating probability score and risk score

In phase three, the interim results from stage one and two are combined, i.e. an adverse action with at least one associated attack vector. Thereafter, the likelihood of implementing such an attack has to be calculated. The evaluation is based on the following five features [71] that lay the foundation to score and identify the resources that all attacks have in common:

- Elapsed Time (0-19 points)
- Expertise (0-8 points)
- Knowledge of the TOE (0-11 points)
- Window of Opportunity (0-10 points)
- Equipment (0-9 points)

The amount of *elapsed time* represents the time needed to implement a specific attack by any chosen attacker. The score ranges from 0 (equals 1 day) to 19 (more than half a year). *Expertise* represents the skill set of an attacker, where 0 is a layman and 8 is given when an attacker has to have competence in more than one field. *Knowledge of the Target of Evaluation (TOE)* scores the needed information on an attacked measuring instrument. It starts with publicly available knowledge (0) and ends with critical insider knowledge (11), that usually resides with the manufacturer. The *window of opportunity* evaluates the possibility available to an attacker, where 0 represents unlimited access, which would be common for measuring instruments connected to the Internet. If the access is difficult, a value of 10 should be given. In case it is impossible to obtain access, no rating is done, and the attack vector would be removed from the list. The last category scores the *equipment* needed to carry out the attack. Standard available hardware or software is described by 0, where 9 represents multiple bespoke devices or software.

After successfully calculating the sum yielded by the five categories for the chosen attack, a probability score is matched to the different ranges of the total sum. In Table 6.2 the Common Criteria evaluation [71] is also included in the final probability score calculation, so that a basic resistance results in a total sum of 10-13 points while 24 or more points represent a high resilience against the rated attack. Finally, the resistance evaluation is associated with the probability score, where 1 represents an unlikely occurrence while 5 stands for high probability to occur.

The final risk will be calculated by multiplying the impact score for the threat with the probability score of the most likely realized attack vector:

$$\text{risk score} = \frac{\text{impact score}}{5} \cdot \text{probability score} \tag{6.1}$$

## 6.2 Evaluation of threats

In this section, the risk assessment method will be applied to the secure cloud reference architecture, that was briefly introduced in the previous section. The threats listed in Table 6.1 will be treated sequentially and will pass the three stages of risk assessment. Afterwards, in Subsection 6.2.3 the Attack Probability Tree (AtPT) is introduced to describe more complex attack sce-

**Table 6.2:** Calculation of a TOE and association of a probability score according to [9]

| Sum of Points | TOE Resistance | Probability Score |
|---|---|---|
| 0-9 | No rating | 5 |
| 10-13 | Basic | 4 |
| 14-19 | Enhanced Basic | 3 |
| 20-24 | Moderate | 2 |
| >24 | High | 1 |

narios, by introducing a prescribed way to construct attack vectors in a standardized and compact way. At the end, suitable countermeasures for attack vectors will be discussed briefly.

### 6.2.1 Integrity of transmitted measurement data

The threat intention of the attacker is to undermine the integrity of transmitted measurement data by manipulating measurement data during transmission. The sensor unit will be considered, that collects the data and encrypts them with a protected public key via FHE before sending them to the cloud reference architecture. The transmission is secured by Transport Layer Security (TLS) and additionally by a x.509 certificate at the cloud service endpoint, so that the sensor unit usually knows the receiver. An insider attack is assumed with the attacker having the access rights of an administrator. For this threat, two attack vectors are taken into consideration, namely A3 and A4 (see Table 6.3). A3 needs two prerequisites A3.1 and A3.2 (see Table 6.4), in order to be feasible.

To manipulate the data in transit, the attacker has to carry out an active Man-In-The-Middle attack (MITM) (see Table 6.4 A3.1), that means the connection has to be rerouted via the attacker's interception device and the TLS-connection has to be captured during key exchange. Furthermore, the certificate has to be forged by, for example, getting the private key of the server and the client to establish active sessions at both ends with the impersonated certificates needed for authentication. The client's improper validation of the certificate would be a big advantage for the attacker.

**Table 6.3:** Attack vectors for Threat B1

| **Attack-ID** | **Attack Vector** | **Time** | **Expertise** | **Knowledge** | **Window of Opportunity** | **Equipment** | **Sum** | **Damage** |
|---|---|---|---|---|---|---|---|---|
| A3 | Manipulate data in transit | 19 | 8 | 11 | 10 | 0 | 48 | 1 |
| A4 | Exchange processing unit | 7 | 6 | 11 | 4 | 0 | 29 | 1 |

**Table 6.4:** Prerequisites for attack vector A3

| Attack-ID | Attack Vector | Time | Expertise | Knowledge | Window of Opportunity | Equipment | Sum | Damage |
|---|---|---|---|---|---|---|---|---|
| A3.1 | MITM-attack | 1 | 6 | 11 | 10* | 0 | 28 | 1 |
| A3.2 | decrypt-encrypt data | 19 | 8 | 11 | 0 | 0 | 38 | 1 |

The time needed to execute such an attack would be less than a day (1), if the attacker is an expert (6) and has critical knowledge of the system (11). While the window of opportunity is difficult (10), since the manipulation has to be carried out during transmission within the boundaries of transmission delay. There is no special equipment needed (1), that exceeds standard hardware. So the total sum of points for this attack (48) leads to high TOE resistance (see Table 6.2).

Even if A3.1 (MITM) is successfully established, the data itself is still encrypted by FHE. Lattice-based cryptography is provably secure and provides worst-case security that is still not broken by quantum computer algorithms. Therefore, the maximum time of more than half a year (19) assumed for A3.2. The attacker has to have expertise on several fields (8) to decrypt and/or break cryptography as well as having critical system knowledge (11) at their disposal. Once, the cryptography is broken, the window of opportunity is unnecessary (0). From the authors' point of view standard hardware (0) is sufficient. This yields a total sum of 38 points and again implies high resilience against the attack vector.

The two attack vectors A3.1 and A3.2 both need to be executed to form A3. The result is shown in Table 6.3 and implies a high resilience (48) for this attack vector. According to Table 6.2, the sum score translates to a probability score of 1. Since this threat has potential influence on all future measurement values, the impact score is 5 and the subsequent risk ($\frac{\text{impact score}}{5} \cdot \text{probability}$) also takes on a value of 1. PTB does not accept technical solutions with a risk greater than 3. This solution qualifies for PTB certification.

**Table 6.5:** Attack vectors for Threat B2

| Attack-ID | Attack Vector | Time | Expertise | Knowledge | Window of Opportunity | Equipment | Sum | Damage |
|---|---|---|---|---|---|---|---|---|
| A1 | Manipulate sensor unit | 4 | 8 | 11 | 0 | 7 | 30 | 1 |
| A2 | Replace sensor unit | 4 | 8 | 11 | 0 | 7 | 30 | 1 |
| A3 | Spoof identity | 19 | 6 | 11 | 0 | 0 | 36 | 1 |

Another attack vector is to exchange the FHE-processing unit (A4) in the cloud, in order to manipulate the data during processing. First, the attacker needs to have access to the software repository, to manipulate the FHE processing unit and then deploy the manipulated software into the cloud service. Furthermore, the hash of the manipulated software has to match the comparative hash, that the market surveillance monitor evaluates. Given the bonus of an insider attacker with the access level of an administrator, it should be feasible, yet the time frame for execution is less than two months (7). The attacker needs to be at least an expert (6) in IT and the window of opportunity is moderate (4), since a lot of security mechanisms have to be worked around. No special hardware (0) is needed. This yields a total sum of 29 and means a high TOE resistance and a probability score of 1. The threat influences all future measurements, the impact score is 5 and the resulting risk has a value of 1.

### 6.2.2 Authenticity of transmitted measurement data

The threat intention of B2 is to attack the authenticity of transmitted measurement data. In Table 6.5 three attack vectors A1-A3 are summed up, while the third is composed of three sub attack vectors displayed in Table 6.6.

The easiest way of attacking the authenticity is to manipulate the origin of the measurement data: the sensor unit itself (A1). The idea behind this attack vector is just to compromise the authenticity, thus it is enough to break the seal and replace the physical sensor with a tampered one, that calculates, for example, a smaller measurement value. Breaking the seal implicates forging a new seal, so that the instrument does not seem to be manipulated to market

**Table 6.6:** Prerequisites for attack vector A3

| Attack-ID | Attack Vector | Time | Expertise | Knowledge | Window of Opportunity | Equipment | Sum | Damage |
|---|---|---|---|---|---|---|---|---|
| A3.1 | Steal key from vault | 1 | 6 | 11 | 0 | 0 | 18 | 1 |
| A3.2 | Obtain certificate | 19 | 6 | 0 | 0 | 0 | 25 | 1 |
| A3.3 | Generate false data | 19 | 6 | 11 | 0 | 0 | 36 | 1 |

surveillance.

The time needed for this invalidation of authenticity (A1) is less than a month (4) and the attacker needs to be expert on several fields (8), since forging an official calibration seal needs knowledge and special equipment (7). Furthermore, replacing the physical sensor requires critical knowledge (8). The window of opportunity is unlimited (0) for this attack vector, because the instrument in the field is not subject to constant surveillance. In total, the attack vector reaches 30 points and represents a TOE with high resistance with an associated probability score of 1, which translates to a risk level of 1 because of its influence of all future measurement values (impact score of 5). However, it is noteworthy that in Legal Metrology there is no higher protection level achievable than a sealed hardware solution.

The second attack vector A2 deals with obtaining security features from the original sensor unit (physical sensor + communication unit) and replacing this unit with a tampered one that is identically constructed. Hereby, the attacker extracts, for example, the protected key (public key) needed for encryption from the original sealed instrument and then stores this security feature in an identical but tampered unit. A2 differs from A1 since it does not involve tampering original hardware, but buying malfunctioning hardware on purpose and putting it into use. The scores are the same as for the previous attack vector. It is again considered very hard to forge an official verification seal, which is reflected in the total sum of 30 points and offers high resilience.

With the last attack vector A3 the identity of the sensor unit will be spoofed by masquerading the IP address of the attacker's sensor unit, for example, by faking the source address field in the TCP header. In order to be successful at the cloud service endpoint, the attacker has to first obtain the protected key from the software vault in the cloud service, in order to be able to encrypt its fake measurement data (A3.1). Given the fact that an insider attacker with the privileges of an administrator is considered, the access to the cloud architecture is self-evident. The attacker will retrieve the information in less than a week (1). The postulated skill set of an expert (6) is needed in an IT related area and critical knowledge (11) of the system is demanded. A3.1 yields in total 18 points, which is considered as an enhanced basic resistance level.

As a next step (A3.2), the attacker has to get his hands on the private key of the x.509 certificate. It is assumed that this is very time consuming (>6 months) (19) but feasible for an expert (6), in order to forge the x.509 certificate and overcome the authentication barrier. The attack vector A3.2 has a total sum of 25 points and achieves high resilience against this threat.

As a last action, the attacker has to generate false measurement data with the stolen key from A3.1 and authenticates himself against the cloud service endpoint with a forged certificate, in order to achieve the objective to compromise the authenticity of the measurement data. Because of the logical AND operation of A3.1 and A3.2 the highest value will run into A3. That leads to the time frame of more than 6 months (19), an expert level (6) and the requirement of critical system knowledge (11), which totals into 36 points and reaches a high resistance level. The probability score evaluates to 1 with an associated risk level of 1 because of the influence of all future measurements (impact score 5).

For threat intention B3 the same risk assessment procedure is carried out and noted in tables. But this methodology is limited, and it quickly becomes extremely difficult to map all requirements and dependencies for all possible attack vectors. As a solution, Esche et al. introduced the attack probability tree that visualizes in a very compact manner the attack vectors and makes it easy to deduce a probable attack path. Furthermore, it enables them to derive the attacker motivation. In the next section a short theoretical introduction of the AtPT will be given and subsequently applied to B3 until B5.

### 6.2.3 Attack probability tree

Esche et al. introduced attack probability trees (AtPT) as an extension of attack trees by Mauw and Oostdijk [72] to tackle two main objectives: developing a method to standardize the deduction of attack vectors and to efficiently visualize the interdependencies of attack vectors in order to easily derive attacker motivation and as a result the most likely attacker path [1]. Additionally, each node embodies features with its own score, such as time, expertise,

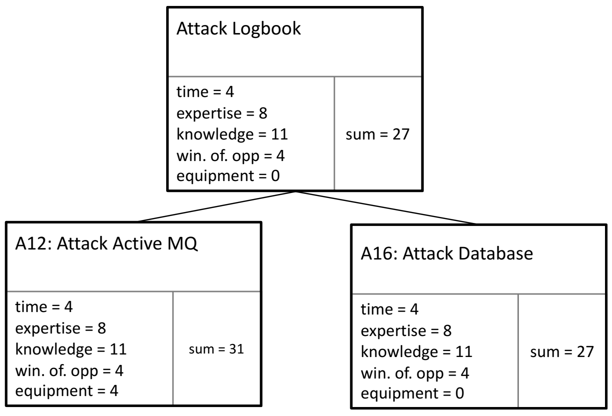


**Figure 6.1:** AtPT for threat intention B3. View: root node and two attack vectors.

knowledge, window of opportunity and equipment, that have been previously collected in tables. Furthermore, the logical relationship between parent and child attacks are visualized and attack nodes are linked either by an AND- or OR-statement.

Information enters the tree via the leaves, so that parent nodes' and finally the root's attributes can be calculated from the bottom to the top. The rules for both statements and each attribute/point score are extensively described in [1]. Briefly summarized: for AND-statements, the *maximum* for each attribute chosen; for OR-statements, the *smaller* sum score indicates the threat to select. A great side-effect of AtPTs is the reduction of required time for revaluation of individual attacks, because of the possibility of reusing attack nodes, that are common among different attacks without recalculating attributes.

The following subsections use the AtPT approach for risk assessment of the cloud reference architecture. Nevertheless, the corresponding tables were generated, as introduced in the previous sections, and can be found in the appendix.

### 6.2.4 Evidence of an Intervention

In this scenario an attacker prevents legally relevant events from being registered in the logbook. The threat intention is to attack the availability of the evidence of a intervention. In case of an successful manipulation, the user cannot present all relevant logbook entries that market surveillance demands.

In this thesis, only the AtPT for a logbook attack is presented. Another attack scenario with the same attack attributes is evaluated for the storage

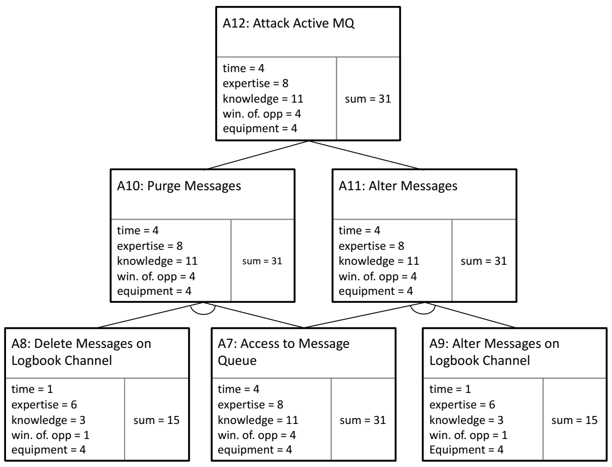


**Figure 6.2:** AtPT for threat intention B3. View: Subtree of attack vector Active MQ.

service of the instrument, with a similar-looking AtPT. Because of the complexity of the attack, the AtPT is divided into four subtrees (see Figure 6.1-6.4), that will be described in the next paragraphs.

An AtPT is read from the root to the leaves. For attacking the logbook, two possibilities are available. Either the attacker aims for the active message queue (Active MQ) or for the database of the logbook service (see Figure 6.1). Since these two attack vectors are alternatives, they are linked by an OR-connection. If the two vectors would be needed to be executed together, they would be linked by an AND-connection graphically expressed by an arc.

When attacking the Active MQ (A12), an attacker could either purge messages (A10) or alter messages (A11) on the logbook channel. For both actions, access to the message queue is required (A7) with the combination of deleting a message (A8) or changing a message (A9) on the logbook channel represented by an arc below the linked nodes (see Figure 6.2).

The actual scores in Figure 6.2 are calculated from the bottom to the top, for example, attack vector A10 consists of nodes A8 and A7. Since the latter two nodes are linked by an AND-statement the greater value is put across to A10. The time to purge a message takes less than a month (4) and stems from A7 accessing the message queue. Furthermore, it is required to be an expert in several areas (8), to have critical knowledge of the system (8) and the window opportunity is moderate (4). These attributes stem also from A7. However, the equipment to purge messages on the Active MQ is specialized (4), since the software is an expert tool written in python without a graphical user interface. The tool is publicly available.

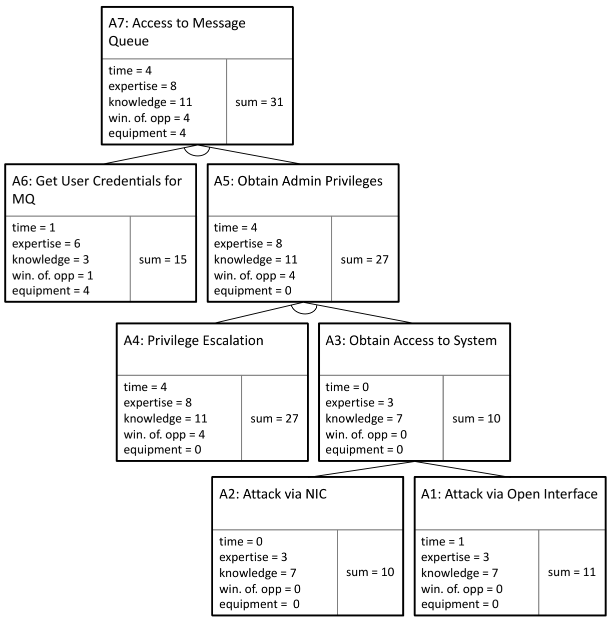


**Figure 6.3:** AtPT for threat intention B3. View: Subtree of attack vector Access to Message Queue.

Now, one could argue that using a specialized software and obtaining access to the message queue needs less time than proposed here. But the whole AtPT does not end with obtaining access to the message queue (A7), but rather continuous and becomes more detailed in how the access could be obtained in a malicious way.

In Figure 6.3 an exemplary attacking path is detailed. Node A7 consists of obtaining administrator privileges in the virtual machine (A5), that runs the Active MQ or is at least in the same subnet. With these new privileges the specialized software can be executed, which triggers node A6 to get the credentials for the message queue.

To get hold of the user credentials, less than a week (1) is estimated. An expert level (6) and restricted knowledge of the measuring system is required. The window of opportunity for an inside attacker is valued as easy (1) even so specialized software (4) is needed. Node A6 holds a total sum of 15 points which would be considered as an enhanced basic resistance level. However, A6 is to be evaluated in conjunction with A5 through the AND-connection.

The attack vector A5 depends again on a privilege escalation through exploiting Common Vulnerabilities and Exposures (CVE) of the underlying system (A4) and obtaining access to the virtual machine (A3). To accomplish a privilege escalation, the attack is assessed with less than a month (4), ex-

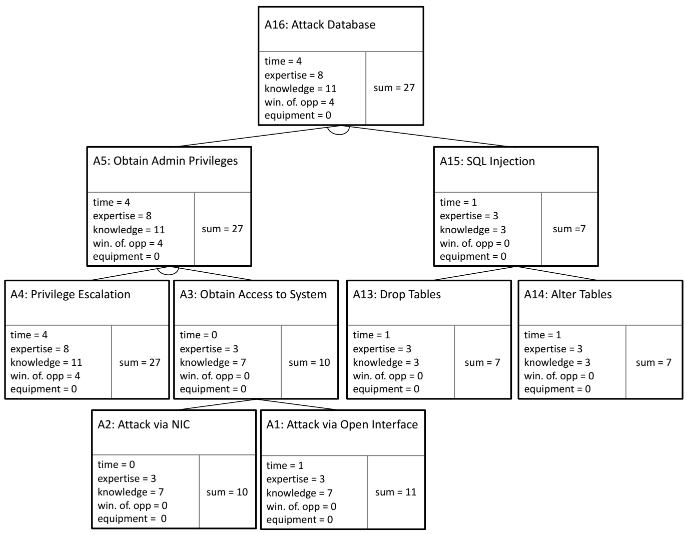


**Figure 6.4:** AtPT for threat intention B3. View: Subtree of attack vector Attack Database.

pertise on more than one field (8), critical system knowledge and moderate window of opportunity (4). Further, no special equipment (0) is expected. A privilege escalation is considered as a difficult endeavor with 27 points in total that translate to a high resilience. This corresponds again to a probability score of 1 with an impact score of 5 and results into a risk of 1.

Obtaining access to a virtual machine and therewith to the distributed measuring system (A3) is possible in two ways that are alternatives (OR-connection). Either the system is penetrated through a network interface card (NIC) (A2) or via an open physical interface (A1), such as a USB port. Considering the fact that an inside attacker with administrator privileges is assumed, that logs remotely into the measuring system for maintenance reasons, this attack is achievable in less than a day (0). To clarify, it is assumed that the inside attacker does not automatically have administrator privileges on the remote machine, but as an employee of the manufacturer. Furthermore, to login remotely requires only a proficient expertise and sensitive system knowledge (7). The window of opportunity is negligible (0), since this can belong to the attacker's daily routine. No special equipment is needed (0). The TOE resistance is basic (10 points in total).

The attack via an open interface (A1) differs from A2 only in the time attribute. It is assumed that the attacker has to physically approach the hardware to carry out the attack. That takes additional time (less than a week (1)) and is more inconvenient than opening a SSH-shell from the desktop PC in the office.

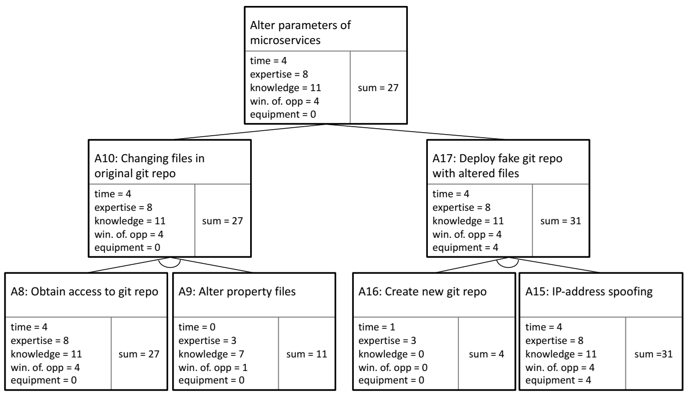


**Figure 6.5:** AtPT for threat intention B4. View: Root, Alter parameters of microservices.

To sum up, the attack path just described consists of A2, A3, A4, A5, A6, A7 then a decision has to be made, if the messages should be altered or deleted. However, in terms of likelihood the nodes do not differ, but practically spoken deletion is often easier. The path would continue via A8, A10.

To completely describe the AtPT for compromising the evidence of an intervention via a logbook attack, the alternative path via the database attack vector (A16) has to be described, as shown in Figure 6.4. For attacking the database, administrator privileges (A5) are needed combined with an attack against the database such as SQL injection (A15) or via command line interface (CLI). The path down to the leaves for A5 is already described in the previous paragraphs. Its TOE resistance depends on leaf A4, that describes the privilege escalation via a CVE. Attack Vector A15 is divided into dropping tables (A13) or modifying tables (A14).

The scores for A13, A14 are equal and subsequently A15 is identical as well. For both attacks, less than a day is assumed, only a proficient expertise level (3) is needed, no special equipment (0) is required and the window of opportunity is unlimited (0). In total, the database attacks combine to 7 points, which translate to no resistance at all (no rating). However, since A5 and A15 are connected via an AND-statement the parent node A16 receives the TOE resistance high, since the attacks depend on the privilege escalation to be carried out.

The most likely attack path would be via the database, since no special software is needed, thus less time is required for learning and incorporating the software. To compromise a database, no new software has to be deployed so that the effort on the attacker side is less than attacking the message queue, especially if the intention is to just compromise the integrity of the measuring instrument.

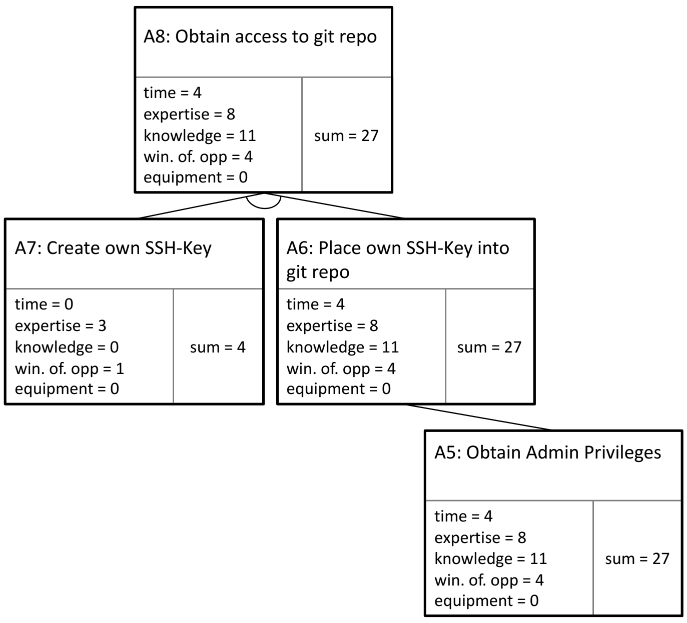


**Figure 6.6:** AtPT for threat intention B4. View: Subtree of attack vector A8.

### 6.2.5 Integrity of Parameters

Threat intention B4 aims to harm legally relevant software parameters violating security properties such as integrity and authenticity. In the following paragraphs the presented scenario offers an attacker a chance to alter persistent saved parameters of the logbook service by attacking the configuration service. Two possible attack scenarios are presented via an AtPT. The tree is compartmentalized into several subtrees, because of its size (see Figures 6.5-6.7). As already pointed out, the subtree consisting of the node A1-A5 could be reused for several attack scenarios without revaluation. Due to repetition it was decided not to map the whole subtree of A5 downwards in Figure 6.6. A complete subtree can be seen in Figure 6.3.

It is proposed that the attacker changes microservice property files in the original git repository to attack the microservice architecture (A10) and provides, for example, false message queue groups. That could lead to loss of messages in the legal relevant logbook. Aiming for the configuration basis can cause fundamental harm and chaos to the whole system.

To be able to carry out attack vector A10, it is assumed that the attacker has to obtain access to the original git repository (A8) and is able to alter the property files (A9). Nodes A8 and A9 are linked via an AND-connection to A10 (see Figure 6.5).

In order to obtain access to the git repository (A8), a SSH-key has to be created (A7) and placed into the specific folder for the git repository to be evaluated (A6). A7 and A6 are linked via an AND-statement to A8 (see Figure 6.6). Attack vector A6 is linked to subtree A5, that describes accomplishment of obtaining administrator privileges (a complete subtree is shown in Figure 6.3).

The score for subtree A5 has been described in the previous section, so that the evaluation starts with A6. All of the attributes stem from A5 and the difficulties to obtain administrator privileges, which are prerequisites for A6. To create an SSH-key (A7) takes less than a day (0) with proficient expertise (3). How to do this is public knowledge (0) and tutorials are easily to find on the Internet. Furthermore, assuming that an inside attacker is already in the system, the window of opportunity is easy to accomplish (1) and also no special equipment (0) is necessary. This yields a total sum of 4 points and a TOE resistance with no rating.

Attack vector A8 receives the point score from A6, since because of the AND-connection only the maximum of both attributes will be passed upwards. That leads to a total sum of 27 points for obtaining access to the original git repository and implies high resilience.

Altering property files can be done in less than a day (0), with only proficient expertise (3), the window of opportunity offers an easy access (1) with any text editor (equipment = 0). That totals in 11 points and matches basic resilience for this attack.

A10 receives the attributes in total from A8 and thus defines the total score of 27 points and a high resilience for this attack path (see Figure 6.5).

The alternative attack vector for B4 is to deploy a fake git repository with already altered property files (A17). This attack vector splits into first creating and deploying a new git repository (A7) and then tricking the system into trusting and pulling the files from the fake git repository via IP spoofing (A15). The spoofing attack itself is subdivided into carrying out a network attack such as ARP-poisoning (A13), in order to replace the IP address and then rebooting the configuration service (A14). To be able to carry out a network attack, the attacker is assumed to have full access to the internal network of the distributed measuring system (A11). To monitor the internal network traffic (A12), the attack vector A11 is necessary (see Figure 6.7). Once again subtree A5 is required to successfully implement A11.

The score of gaining full access to the internal network interface card (NIC) and thus to the internal network (A11) is inherited from A5 and the struggle of obtaining administrator privileges. To gain a full picture of the structure of the internal network with its services (A12) takes less than a month (4) with

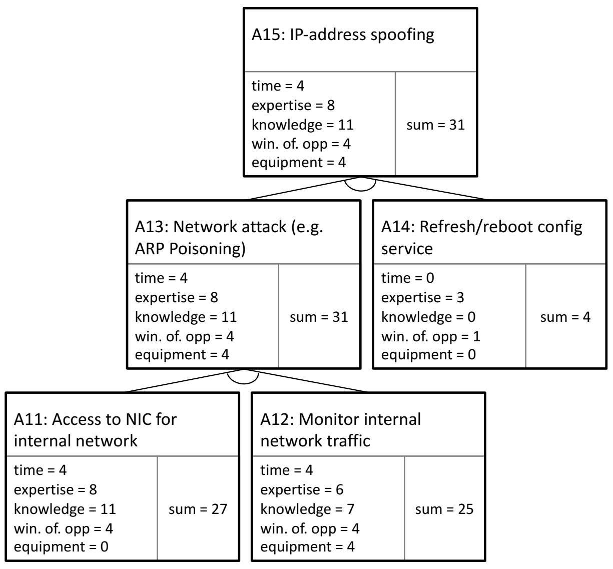


**Figure 6.7:** AtPT for threat intention B4. View: Subtree of attack vector IP-address spoofing.

an assumed expertise in networking (6) and a sensitive knowledge of the measuring system (7). A moderate window of opportunity (4) is predicted, because it is difficult to explore a supervised internal network undetected. Furthermore, specialized software is needed to monitor network traffic (4). This yields in total 25 points and maps to a high resistance to attacks with probability score of 1 and a risk of 1.

Carrying out network attacks, such as ARP-poisoning (A13), requires less than a month (4) for experts on several fields (8) with sensitive knowledge of the system (11), a moderate window of opportunity (4) and specialized software (4). For most network attacks, it is not necessary any more to write specialized software. There exists publicly available gray software, that can be used to detect vulnerabilities or can be misused to attack computer systems. This attack vector combines to 31 points and a high resistance factor. From here on, no significant changes to the resilience are contributed until the final attack vector A17. Minor actions are required to finally deploy a fake git repository, but both acquire only 4 points in total with a negligible threat resistance (see Figure 6.5).

The most probable attack path will be via A10, changing the properties files in the original repository, since it is the least complex one and without the hassle of deploying software and monitoring traffic etc. This is also reflected in the total score of 27 against 31 points.

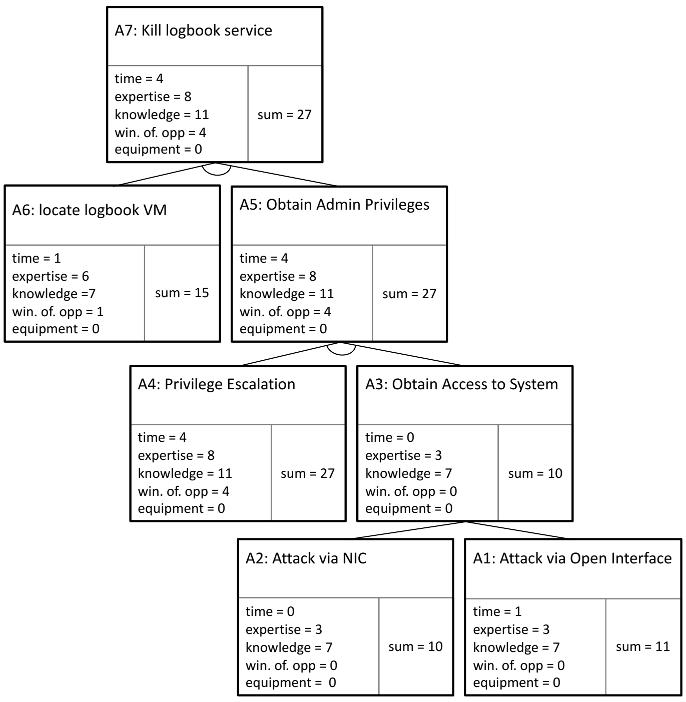


**Figure 6.8:** AtPT for threat intention B5 to violate the availability security property.

## 6.2.6 Availability of a Service

Threat intention B5 targets the availability of a legally relevant logbook service. In Figure 6.8 the complete AtPT is illustrated with the already introduced subtree A5, that enables access to the measuring system and comes along with an escalation of privileges. This subtree in conjunction with the localization of the logbook service's virtual machine (A6) enables the final attack vector that kills the logbook service (A7).

The final score stems from the difficulty to gain access to the system and to elevate the privilege level (subtree A5), which totals 27 points. With an associated risk level of 1 and a probability score of 1. Locating the logbook's virtual machine is with 15 points in total an enhanced basic TOE resistance level, but has no significant influence on the final score of killing the logbook service (A7).

## 6.2.7 Effect of Attacker Motivation

Esche et al. described in [70] possibilities to represent attacker motivation during risk assessment. The presented AtPTs are created for a highly moti-

**Table 6.7:** Mapping of expertise and motivation level according to [70]

| Expertise | Score | Motivation | Score |
|---|---|---|---|
| Layman | 0 | no motivation | 9 |
| Proficient | 3 | low | 6 |
| Expert | 6 | moderate | 3 |
| Multiple Expert | 8 | high | 0 |

vated attacker. In order to reconsider these trees with a low or medium motivated attacker, the expertise and equipment score have to be replaced with a higher motivation score according to Table 6.7 if they are originally smaller. This will result in a decreased probability score for a lower motivation and vice versa for a highly motivated attacker. It is noteworthy, that the likeliest attacker path can shift, when the motivation is adjusted.

### 6.2.8 Suitable Countermeasures

To find the best suitable place for countermeasures in an AtPT, it is recommended to locate an inverted subtree for mitigating attack vectors and increasing the impact of applied countermeasures. An inverted tree is usually any leaf that is connected to more than one node of the previous level. Subsequently, the size of an inverted tree matters, since the greater it is, more parent nodes are impacted. In the trees for B3 and B4, A7 and A16 depend on A5 as well as A6 and A11 depend on A5. Subtree A5 is of general importance, because it describes the unauthorized access to the measuring system and privilege escalation. A countermeasure specifically tailored for A5 will exacerbate obtaining administrator rights. This node will have the biggest impact on all three threat scenarios from B3-B5.

A suitable countermeasure is to strengthen the access rights and to enforce a least privilege policy. For example, one could implement Security Enhance Linux (SELinux) for virtual machines (VM), that provides a mandatory access control system and security policies. Instead of using a standard Linux, the kernel extension SELinux provides by default a least privilege policy that denies everything except if it is specifically allowed by access policies (enforcing mode). All violations against these rules are logged and an alarm can be triggered. To obtain administrator privileges by an escalation of access rights would need significantly more time (less than 2 months (7)) with SELinux in place. Furthermore, if the attacker is able to bypass SELinux via switching from enforcing to permissive mode it needs to be done on every VM with a bespoke software (7). However, rolling out SELinux to the measuring system would mean a lot of configuration overhead, but it would elevate the security score by 10 points to 37. This security enhancement would propagate via the inverted tree to the top of each AtPT.

## 6.3 Summary

In this Chapter, the secure cloud reference architecture for distributed measuring instruments under legal control was subjected to a specially tailored risk assessment method for software in Legal Metrology. After formally introducing the risk analysis, five threats for the reference architecture were described and evaluated extensively. The first two threats were assessed using the traditional method via tables. However, this approach seemed infeasible for more complex threats. Therefore, the Attack Probability Tree (AtPT), that eases the handling of more complex attacks, was introduced and applied. It was shown that adequate protection of the essential requirements formulated by the MID is provided by the secure cloud reference architecture. Therefore, the architecture is qualified to be implemented in measuring systems under legal control.

The detailed analysis of the threat intentions using AtPTs revealed for all formulated threats and attacked security properties a high resilience factor. Nevertheless, through the inverted subtree method for AtPTs the optimal entry point for countermeasures was identified. The implementation of countermeasures reduced the risk to the level provided by physical sealing and increases the resilience to attacks.

# 7 Conclusion and Future Work

*"In literature and in life we ultimately pursue, not conclusions, but beginnings."*

*—Sam Tanenhaus, Literature Unbound*

THE PARADIGM SHIFT from local, concentrated to distributed measuring instruments has a fundamental impact on society and its jurisdictional conception. It is obvious for all stakeholders that Cloud Computing is not just a technical trend that will pass but rather a door opener and key technology for future technological developments. Paving the way for a secure and legal Cloud Computing solution is necessary to be able to invest in a future-proofed infrastructure to keep pace with upcoming trends. Especially for small and medium-size enterprises, legal compliance is from utmost importance for penetrating the market successfully with their solutions. Responsibilities and liabilities among stakeholders with respect to cross-border use still have to be determined. The sooner this area of conflict is resolved the better technological progress can contribute to preserve competitiveness of European industry in the world.

The proposed secure cloud reference architecture offers an evaluated technological approach in conformance with the Legal Metrology framework. Furthermore, it demonstrates a technical solution to avoid inspection of server hardware by the market surveillance body when employing an off-premise Cloud Computing solutions. By introducing FHE, the measurements are encrypted throughout the metrology lifecycle. Moreover, two main security concerns, such as a malicious insider and integrity of measurements, are specifically addressed by FHE. Reducing concerns enormously on manufacturer side to employ Cloud Computing solutions. Also, the long periods of time processing encrypted data are significantly reduced. FHE offers a quantum secure, flexible and future-proofed approach for security conscious industries that protects the trust in measurements results throughout their entire lifecycle.

To sum up, the presented Secure Cloud Computing Reference Architecture fulfills the highest risk class, the essential requirements and offers a verification method for the market and user surveillance. Furthermore, contemporary threats were assessed, and attack vectors evaluated to fully comply with the MID and WELMEC requirements. The presented approach offers a practical solution to the challenges of a distributed measuring system within legal boundaries. The used encryption scheme offers a quantum secure foundation that does not restrict the Cloud Computing abilities in terms of flexibility and scalability.

Future work will comprise more extensive anomaly detection and further automation of this approach. In addition, the risk assessment part will focus on different attacker motivation as well as diverse attack paths. Furthermore, the formalization of creating AtPTs has to be optimized and standardized. Moreover, Kratzke presented an interesting approach for a reactive defense mechanism [14], that acts similar to an immune system. He suggest that rapid redeployment of integrity proofed virtual machine images helps fighting against unknown vulnerabilities. The main goal is to reduce the time of an undetected intruder in the system. In 2015, the discovery of an attacker in the system took in average 146 days. Even if the wrongdoer can use the same way back into the system, he will be shut out again by redeployment. This will have a noticeable effect on the attacker motivation and decreases the probability of persisting access to the system. It will be interesting to further pursue this approach and combine it with the secure cloud reference architecture.

# Bibliography

[1] Marko Esche, Federico Grasso Toro, and Florian Thiel. “Representation of Attacker Motivation in Software Risk Assessment Using Attack Probability Trees”. In: *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS), IEEE.* (2017), pp. 763–771.

[2] European Parliament and Council. “Directive 2014/32/EU of the European Parliament and of the Council”. In: *Official Journal of the European Union* (2014).

[3] “WELMEC 7.2 2015 Software Guide”. In: *WELMEC European cooperation in legal metrology* (2015).

[4] Florian Thiel et al. “Cloud Computing in Legal Metrology”. In: *17th International Congress of Metrology*. EDP Sciences. 2015, p. 16001.

[5] Norbert Leffler and Florian Thiel. “Im Geschäftsverkehr das richtige Maß - Das neue Mess und Eichgesetz, Schlaglichter der Wirtschaftspolitik”. In: *Monatsbericht; Bundesministerium für Wirtschaft und Technologie (BMWi)* (2013).

[6] “Organisation Internationale de Métrologie Légale”. In: *OIML, General requirements for software controlled measuring instruments* (2008).

[7] Manfred Kochsiek and Andreas Odin. “Towards a global measurement system: Contributions of international organizations”. In: *OIML Bulletin* 42.2 (2001), pp. 14–19.

[8] Daniel Peters et al. “Achieving Software Security for Measuring Instruments under Legal Control”. In: *In Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS)* (2014), pp. 123–130.

[9] Marko Esche and Florian Thiel. “Software risk assessment for measuring instruments in legal metrology”. In: *Computer Science and Information Systems (FedCSIS), IEEE.* (2015), pp. 1113–1123.

[10] DIRECTIVE 2004/22/EC. “Directive 2004/22/EC of the European Parliament and of the Council”. In: *Official Journal of the European Union* (2004).

[11] Ramnath Chellappa. “Intermediaries in cloud-computing: A new computing paradigm”. In: *INFORMS Annual Meeting, Dallas*. 1997, pp. 26–29.

[12] Lutz Schubert, Keith Jeffery, and Burkard Neidecker-Lutz. “The future of cloud computing: Opportunities for European cloud computing beyond 2010”. In: *Expert Group report, public version* 1 (2010).

[13] Peter Mell and Timothy Grance. “The NIST definition of cloud computing. National Institute of Standards and Technology Special Publication 800-145”. In: *Gaithersburg: US Department of Commerce. Google Scholar* (2011).

[14] Nane Kratzke. “A Brief History of Cloud Application Architectures: From Deployment Monoliths via Microservices to Serverless Architectures and Possible Roads Ahead”. In: *Applied Sciences* (July 2018). DOI: 10.20944/preprints201807.0276.v1.

[15] Harald E. Weiss. “Umfrage: Cloud kann teuer werden”. In: `https://heise.de/-4072912` (June 2018). Accessed: 2018-07-20.

[16] Tim Mather, Subra Kumaraswamy, and Shahed Latif. *Cloud security and privacy: an enterprise perspective on risks and compliance*. " O’Reilly Media, Inc.", 2009.

[17] Joint Task Force and Transformation Initiative. “Security and privacy controls for federal information systems and organizations”. In: *NIST Special Publication* 800.53 (2013), pp. 8–13.

[18] Stefan Krempl. “Microsoft-Fall: US-Justizministerium bringt Streit ueber Datenzugriff in der EU vor den Supreme Court”. In: `https://heise.de/-3755865` (June 2018). Accessed: 2018-07-23.

[19] Florian Thiel. “Digital transformation of legal metrology-The European Metrology Cloud”. In: *OIML Bulletin* 59.1 (2018), pp. 10–21.

[20] Tobias Haar. “Wolkenbruch”. In: `https://www.heise.de/-4089925` (July 2018). Accessed: 2018-07-23.

[21] Philipp Wieder et al. *Service level agreements for cloud computing*. Springer Science & Business Media, 2011.

[22] Dr. Marnix Dekker et al. “Auditing Security Measures: An Overview of schemes for auditing security measures”. In: *European Union Agency for Network and Information Security (ENISA)* (2013).

[23] ENISA. “Cloud Computing Certification - CCSL and CCSM”. In: `https://resilience.enisa.europa.eu/cloud-computing-certification` (June 2013). Accessed: 2018-07-24.

[24] European Commission. “Unleashing the Potential of Cloud Computing in Europe”. In: *Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions* (2012).

[25] Jeffrey Mogul et al. “Internet standard subnetting procedure”. In: (1985).

[26] Frank Buschmann et al. “A system of patterns: Pattern-oriented software architecture”. In: *Wiley New York* (1996).

[27] Craig Gentry. “Fully homomorphic encryption using ideal lattices.” In: *Annual ACM Symposium on Theory of Computing. Proceedings of the 41st annual ACM symposium on Theory of computing.* (2009).

[28] Alexander Oppermann, Jean-Pierre Seifert, and Florian Thiel. “Secure Cloud Reference Architectures for Measuring Instruments under Legal Control.” In: *6th International Conference on Cloud Computing and Services Science (Closer)* (2016), pp. 289–294.

[29] Renzo E Navas et al. “Nonce-based authenticated key establishment over OAuth 2.0 IoT proof-of-possession architecture”. In: *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE. 2016, pp. 317–322.

[30] Sheetal Kalra and Sandeep Sood. “Advanced remote user authentication protocol for multi-server architecture based on ECC”. In: *journal of information security and applications* 18.2-3 (2013), pp. 98–107.

[31] Joe Kilian. “A note on efficient zero-knowledge proofs and arguments”. In: *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*. ACM. 1992, pp. 723–732.

[32] Rosario Gennaro, Craig Gentry, and Bryan Parno. “Non-interactive verifiable computing: Outsourcing computation to untrusted workers”. In: *Annual Cryptology Conference*. Springer. 2010, pp. 465–482.

[33] Kai-Min Chung, Yael Kalai, and Salil Vadhan. “Improved delegation of computation using fully homomorphic encryption”. In: *Annual Cryptology Conference*. Springer. 2010, pp. 483–501.

[34] BSI. “Technische Richtlinie BSI TR-03109-1 Anforderungen an die Interoperabilität der Kommunikationseinheit eines intelligenten Messsystems”. In: *Bundesamt für Sicherheit in der Informationstechnik, Bonn* (2013).

[35] Alexander Oppermann et al. “Secure Cloud Computing: Multithreaded Fully Homomorphic Encryption for Legal Metrology”. In: *International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments* (2017), pp. 35–54.

[36] Sakil Barbhuiya et al. “A Lightweight Tool for Anomaly Detection in Cloud Data Centres.” In: *5th International Conference on Cloud Computing and Services Science (Closer)* (2015), pp. 343–351.

[37] Alexander Oppermann et al. “Anomaly Detection Approaches for Secure Cloud Reference Architectures in Legal Metrology.” In: *8th International Conference on Cloud Computing and Services Science (Closer)*. 2018, pp. 549–556.

[38] Performance Co-Pilot. “Performance Co-Pilot is a system performance analysis toolkit.” In: `http://www.pcp.io/` (2016).

[39] Netflix Inc. “An on-host performance monitoring framework”. In: `http://getvector.io/` (2016).

[40] Daniel Catrein and Cologne QSC AG. “Maintaining User Control While Storing and Processing Sensor Data in the Cloud”. In: *International Journal of Grid and High Performance Computing* 5.4 (2013), pp. 97–112.

[41] Hubert A. Jäger et al. “A Novel Set of Measures against Insider Attacks-Sealed Cloud”. In: *Open Identity Summit* (2013), p. 187.

[42] Mathias Slawik et al. “Innovative Architektur für sicheres Cloud Computing: Beispiel eines Cloud-Ecosystems im Gesundheitswesen”. In: *Informatik* (2012).

[43] Daniele Micciancio and Oded Regev. “Lattice-based cryptography”. In: *Post-quantum cryptography* (2009), pp. 147–191.

[44] Joop van de Pol. “Lattice-based cryptography”. MA thesis. Eindhoven University of Technology, 2011.

[45] Federico Bergami. “Lattice-Based Cryptography”. MA thesis. Universita di Padova, 2016.

[46] Marten Van Dijk et al. “Fully homomorphic encryption over the integers”. In: *Advances in cryptology–EUROCRYPT 2010* (2010), pp. 24–43.

[47] Michael Brenner. “Rechnen mit verschlüsselten Programmen und Daten”. PhD thesis. Technische Informationsbibliothek und Universitätsbibliothek Hannover (TIB), 2012.

[48] Xun Yi, Russell Paulet, and Elisa Bertino. *Homomorphic encryption and applications*. 2014.

[49] Frederik Armknecht et al. “A Guide to Fully Homomorphic Encryption”. In: *IACR Cryptology ePrint Archive* 2015 (2015), p. 1192.

[50] Zvika Brakerski and Vinod Vaikuntanathan. “Efficient fully homomorphic encryption from (standard) LWE”. In: *SIAM Journal on Computing* 43.2 (2014), pp. 831–871.

[51] Craig Gentry. "Computing on the edge of chaos: Structure and randomness in encrypted computation." In: *Electronic Colloquium on Computational Complexity (ECCC)*. Vol. 21. Citeseer. 2014, p. 106.

[52] Nigel P Smart and Frederik Vercauteren. "Fully homomorphic encryption with relatively small key and ciphertext sizes". In: *International Workshop on Public Key Cryptography* (2010), pp. 420–443.

[53] Henning Perl, Michael Brenner, and Matthew Smith. "Poster: an implementation of the fully homomorphic Smart-Vercauteren crypto-system". In: *Proceedings of the 18th ACM conference on Computer and communications security*. ACM. 2011, pp. 837–840.

[54] Shai Halevi and Victor Shoup. "Algorithms in helib". In: *International Cryptology Conference*. Springer. 2014, pp. 554–571.

[55] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) fully homomorphic encryption without bootstrapping". In: *ACM Transactions on Computation Theory (TOCT)* 6.3 (2014), p. 13.

[56] Nigel P Smart and Frederik Vercauteren. "Fully homomorphic SIMD operations". In: *Designs, codes and cryptography* (2014), pp. 1–25.

[57] Kim Laine and Rachel Player. *Simple Encrypted Arithmetic Library-SEAL (v2. 0)*. Tech. rep. Technical report, September, 2016.

[58] Junfeng Fan and Frederik Vercauteren. "Somewhat Practical Fully Homomorphic Encryption." In: *IACR Cryptology ePrint Archive* 2012 (2012), p. 144.

[59] Léo Ducas and Daniele Micciancio. "FHEW: bootstrapping homomorphic encryption in less than a second". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2015, pp. 617–640.

[60] L. J. M. Aslett, P. M. Esperança, and C. C. Holmes. *A review of homomorphic encryption and software tools for encrypted statistical machine learning*. Tech. rep. University of Oxford, 2015.

[61] Craig Gentry and Shai Halevi. "Implementing gentrys fully-homomorphic encryption scheme". In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2011, pp. 129–148.

[62] Israel Koren. *Computer arithmetic algorithms*. Universities Press, 2002.

[63] Mi Lu. "Modular Structure of Large Multiplier". In: *Arithmetic and Logic in Computer Systems, 1st ed, New Jersey: John Wiley & Sons, Inc* (2004), pp. 120–122.

[64] Diederik Verkest, Luc Claesen, and Hugo De Man. "A proof of the nonrestoring division algorithm and its implementation on an ALU". In: *Formal Methods in System Design* 4.1 (1994), pp. 5–31.

[65] "Système international d'unités, The International System of Units (SI)". In: *Bureau International des Poides et Mesures (BIPM)* 8th edition (2006).

[66] BSI. "Schutzprofil für die Kommunikationseinheit eines intelligenten Messsystems für Stoff- und Energiemengen (Smart Meter Gateway PP), Certification-ID: BSI-CC-PP-0073". In: *Bundesamt für Sicherheit in der Informationstechnik, Bonn* (2014).

[67] Ananth Grama et al. *Introduction to parallel computing*. Second. Pearson Education, 2003. ISBN: 0-201-64865-2.

[68] ISO27005:2011(e). "Information technology - Security techniques - Information security risk management." In: *International Organisation for Standardisation, Geneva, CH* Standard (June 2011).

[69] "Welmec 5.3 Risk Assessment Guide for Market Surveillance: Weigh and Measuring Instrument". In: *WELMEC European cooperation in legal metrology, WELMEC Secretariat, Ljubljana* (May 2011).

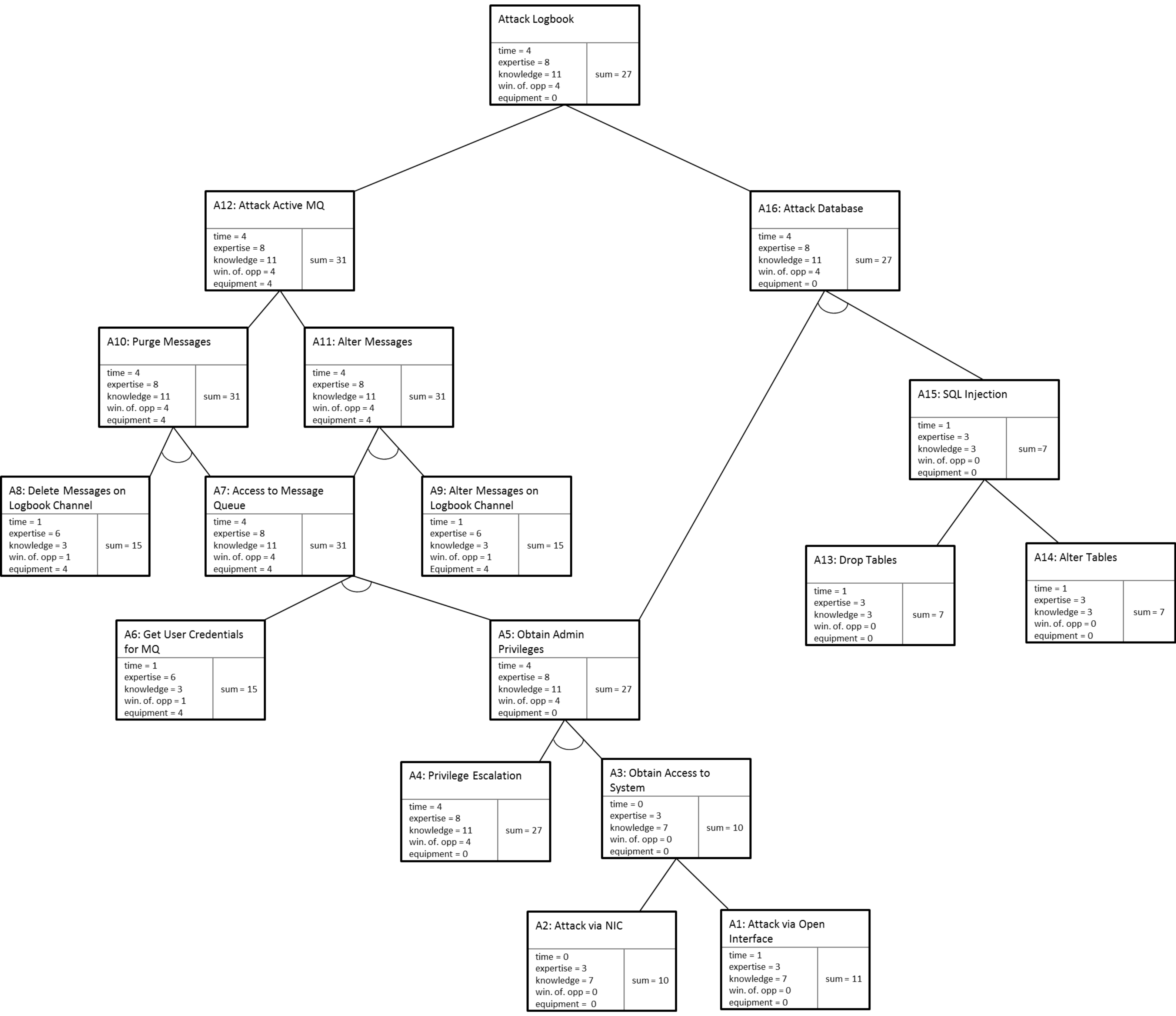
[70] Marko Esche and Florian Thiel. "Incorporating a Measure for Attacker Motivation into Software Risk Assessment for Measuring Instruments in Legal Metrology". In: *18. GMA/ITG-Fachtagung Sensoren und Messsysteme 2016,Nürnberg, Germany* 1.1 (May 2016), pp. 735–742.

[71] ISO/IEC18045:2012. "Common Methodology for Information Technology Security Evaluation". In: *International Organisation for Standardisation, Geneva, CH* (Sept. 2012).

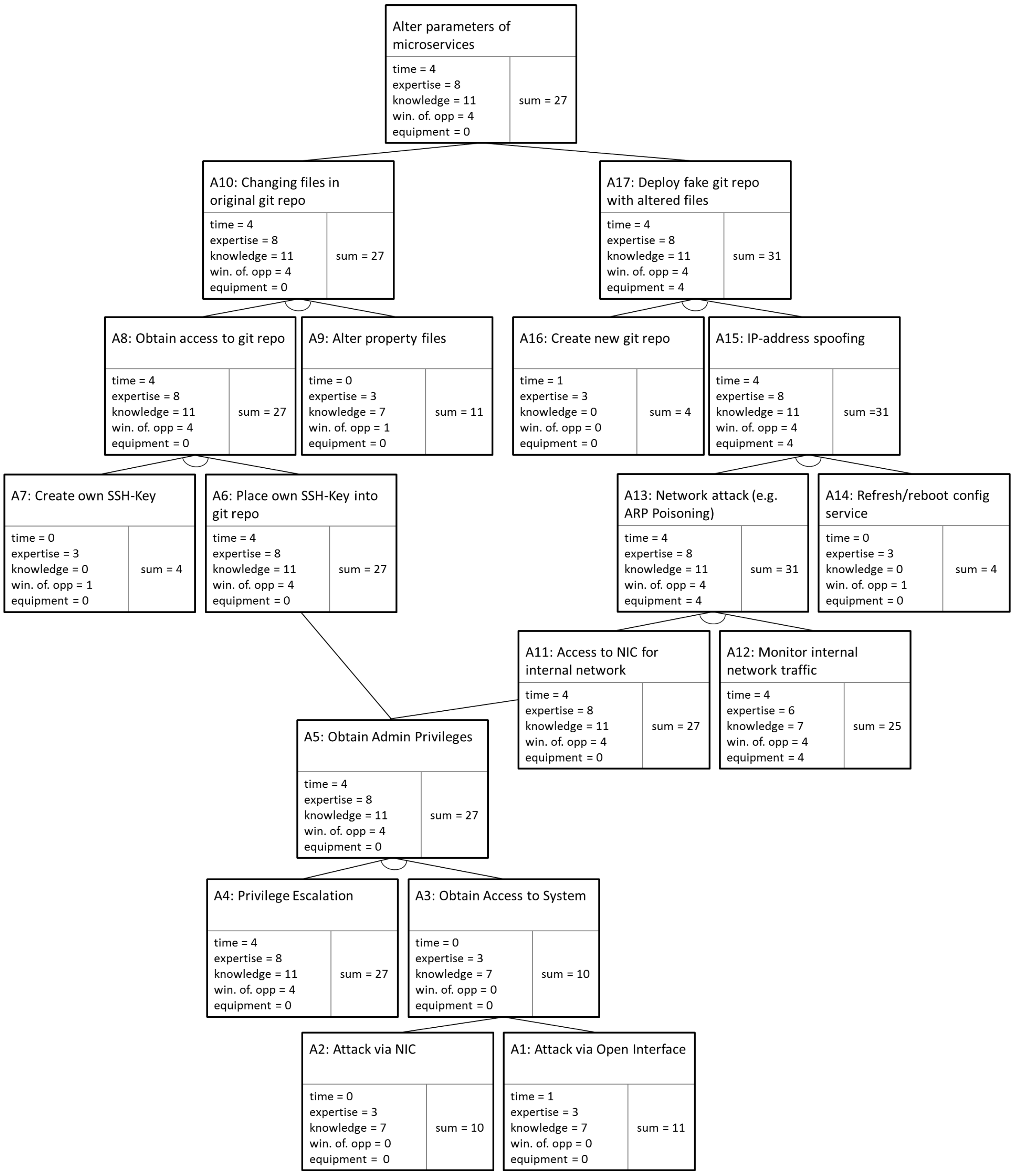
[72] Sjouke Mauw and Martijn Oostdijk. "Foundations of attack trees". In: *International Conference on Information Security and Cryptology*. Springer. 2005, pp. 186–198.

# Appendices

## AtPT for threat intention B3



## AtPT for threat intention B4



# Tables Overview

### B1 Integrity of transmitted measurement data:

A1 and A2 are prerequisites for A3.

| **Attack-ID** | **Attack Vector** | Time | Expertise | Knowledge | Window of opportunity | Equipment | Sum | Damage |
|---|---|---|---|---|---|---|---|---|
| A1 | Attacker tries to drive a MITM-attack | 1 | 6 | 11 | 10* | 0 | 28 | 1 |
| A2 | If MITM is successful decrypt encrypted data | 19 | 8 | 11 | 0 | 0 | 38 | 1 |

A1: MITM (e.g. mitmproxy) is practically very difficult to succeed, since the transmission is TLS secured and the endpoint has a certificate for authentication. Needs to happen during transmission.

A2: Even if an MITM is successful and the TLS-Layer should be broken, the data itself is encrypted. Lattice based cryptography is provable secure and provides worst-case security that is still not broken by quantum algorithm.

Insider: role → administrator;   View: Measurement System (sensor unit + cloud services)

| **Attack-ID** | **Attack Vector** | Time | Expertise | Knowledge | Window of opportunity | Equipment | Sum | Damage |
|---|---|---|---|---|---|---|---|---|
| A3 | Manipulate data in transit and send in time to cloud | 19 | 8 | 11 | 10 | 0 | 48 | |
| A4 | Exchange FHE-processing unit to manipulate data | 7 | 6 | 11 | 4 | 0 | 29 | |

A3: Even if the decryption is broken in A2, the data must be manipulated in a time critical range, so that the manipulated measurement data is not too much delayed and raises suspicion. Further has all data to be changed, since they are secured by a simple signature, that has to be taken into account. Furthermore, the attacker must separate unannounced test data from the measurement data to stay undetected.

A4: Replacing the fhe-processing unit to manipulate the data within the cloud service is difficult, since the attacker has to manipulate the hash of that module to stay consistent with the comparative hash. Further the same reasoning as in A3 is applicable.

## B2: Authenticity of transmitted measurement data

Insider: role → administrator, View: Measurement System (sensor unit + cloud services)

| **Attack-ID** | **Attack Vector** | Time | Expertise | Knowledge | Window of opportunity | Equipment | Sum | Damage |
|---|---|---|---|---|---|---|---|---|
| A1 | Manipulate sensor unit | 4 | 8 | 11 | 0 | 7* | 30** | 1 |
| A2 | Obtain security features of sensor unit and replace it by a tampered sensor unit | 4 | 8 | 11 | 0 | 7 | 30 | 1 |
| A3 | Spoof identity of sensor unit (IP spoofing) and send manipulated data into the cloud | 4 | 6 | 11 | 10 | 0 | 31 | 1 |

Subtask to complete A3.

| **Attack-ID** | **Attack Vector** | Time | Expertise | Knowledge | Window of opportunity | Equipment | Sum | Damage |
|---|---|---|---|---|---|---|---|---|
| A3.1 | Obtain protected key from secret vault | 1 | 6 | 11 | 0 | 0 | 18 | 1 |
| A3.2 | Obtain X.509 certificate (i.e. private Key) | 19 | 6 | 0 | 0 | 0 | 25 | 1 |
| A3.3 | Generate false measurement data | 19 | 6 | 11 | 0 | 0 | 36 | 1 |

A1: Implicates breaking seals and forging as well as replacing physical sensor, to carry out any manipulation. Needs special equipment to forge a seal.

A2: An attacker obtains security feature (protected key [security features of Key generation]) from a sealed instrument by breaking the seal and taking it apart. Then he stores this in an identical unit and seals unit with a forged seal and replaces the original.

A3: This attack focus on spoofing the identity of a sensor unit, in order to undermine the authentication feature of the cloud service. The original sensor is still active. Obtain security feature (private key + certificate)

* forge seal with specialized equipment

** highest protection level achievable => all other threats should have the same.

## B3: Integrity of Software

Insider: Role → administrator View: cloud services

| **Attack-ID** | **Attack Vector** | Time | Expertise | Knowledge | Window of opportunity | Equipment | Sum | Damage |
|---|---|---|---|---|---|---|---|---|
| A1 | Attacker uses open interface to plant code into MI | 1 | 3 | 7 | 0 | 0 | 11 | 1 |
| A2 | Attacker uses NIC to plant code into MI | 0 | 3 | 7 | 0 | 0 | 10 | 1 |
| A3 | Obtain access to system | 0 | 3 | 7 | 0 | 0 | 10 | 1 |
| A4 | Attacker uses CVE for privilege escalation | 4 | 8 | 11 | 4 | 0 | 27 | 1 |
| A5 | Obtain admin privileges | 4 | 8 | 11 | 4 | 0 | 27 | 1 |
| A6 | Get user credential for message queue | 1 | 6 | 3 | 1 | 4 | 15 | 1 |
| A7 | Access to Message-Queue | 4 | 8 | 11 | 4 | 0 | 27 | 1 |
| A8 | Delete Messages on Logbook-Channel | 1 | 6 | 3 | 1* | 4 | 15 | 1 |
| A9 | Alter Messages on Logbook-Channel | 1 | 6 | 3 | 1* | 4 | 15 | 1/3 |
| A10 | Purge Logbook-Messages | 4 | 8 | 11 | 4 | 0 | 27 | 1 |
| A11 | Alter Messages | 4 | 8 | 11 | 4 | 0 | 27 | 1 |
| A12 | Attack Active MQ | 4 | 8 | 11 | 4 | 0 | 27 | 1 |
| A13 | Drop database tables via SQL Injection | 1 | 3 | 3 | 0 | 0 | 7 | 1/3 |
| A14 | Alter database tables via SQL Injection | 1 | 3 | 3 | 0 | 0 | 7 | 1/3 |
| A15 | SQL Injection | 1 | 3 | 3 | 0 | 0 | 7 | 1/3 |
| A16 | Attack Database | 4 | 8 | 11 | 4 | 0 | 27 | 1/3 |

https://github.com/cr0hn/enteletaor

* deletes specific event

## B4: Integrity of Parameters

Insider: role → administrator View: cloud services

| **Attack-ID** | **Attack Vector** | Time | Expertise | Knowledge | Window of opportunity | Equipment | Sum | Damage |
|---|---|---|---|---|---|---|---|---|
| A1 | Replace property files in original git repo | 1 | 3 | 7 | 1 | 0 | 12 | 1 |
| A2 | Obtain access to git repo | 1 | 3 | 3 | 1 | 0 | 8 | 1 |
| A3 | Altering property files | 0 | 3 | 7 | 1 | 0 | 11 | 1 |
| A4 | Obtain access to git repo via own ssh key | 1 | 3 | 3 | 1 | 0 | 8 | 1 |
| A5 | Obtain access to internal network | 2 | 6 | 3 | 4 | 0 | 15 | 1 |
| A6 | Privilege Escalation | 2 | 6 | 3 | 4 | 0 | 15 | 1 |
| A7 | Persist access to system | 1 | 6 | 3 | 3 | 0 | 13 | 1 |
| A8 | Attack via NIC | 0 | 3 | 3 | 0 | 0 | 6 | 1 |
| A9 | Attack via open interface | 0 | 3 | 3 | 0 | 0 | 6 | 1 |
| A10 | Deploy fake git w/ altered files | 1 | 6 | 3 | 1 | 0 | 11 | 1 |
| A11 | Create own git and spoof IP address | 2 | 8 | 7 | 1 | 0 | 18 | 1 |
| A12 | Network Attack (ARP Poisening/spoofing) | 2 | 8 | 7 | 1 | 0 | 18 | 1 |
| A13 | Refresh/Reboot Config-Service | 0 | 3 | 3 | 1 | 0 | 7 | 1 |
| A14 | Monitor internal network traffic | 0 | 3 | 3 | 1 | 0 | 7 | 1 |
| A15 | | | | | | | | |

## B5: Availability of Services

Insider: role → administrator View: cloud services

| **Attack-ID** | **Attack Vector** | Time | Expertise | Knowledge | Window of opportunity | Equipment | Sum | Damage |
|---|---|---|---|---|---|---|---|---|
| A1 | Attacker uses open interface to plant code into MI | 1 | 3 | 7 | 0 | 0 | 11 | 1 |
| A2 | Attacker uses NIC to breach security measures | 0 | 3 | 7 | 0 | 0 | 11 | 1 |
| A3 | Attacker persists access to system | 1 | 6 | 7 | 1 | 4 | 19 | 1 |
| A4 | Privilege Escalation | 2 | 6 | 3 | 4 | 0 | 15 | 1 |
| A5 | Obtain access to internal network and scan it | 2 | 6 | 3 | 4 | 0 | 15 | 1 |
| A6 | Take over other internal VMs | 1 | 6 | 3 | 0 | 0 | 10 | 1 |
| A7 | Run DDoS attack against Service | 4 | 6 | 7 | 4 | 0 | 21 | 1 |
| A8 | Kill service directly on VM | 0 | 3 | 3 | 0 | 0 | 6 | 1 |