Received: 31 January 2018 | Accepted: 7 March 2018

DOI: 10.1002/spy2.18

SPECIAL ISSUE ARTICLE

WILEY

Secure Cloud Computing: Reference Architecture for Measuring Instrument under Legal Control

Alexander Oppermann[1] | **Federico Grasso Toro**[1] | **Florian Thiel**[1] | **Jean-Pierre Seifert**[2]

[1]Department 8.5 Metrological IT, Physikalisch-Technische Bundesanstalt (PTB), Berlin, Germany

[2]Security in Telecommunications, Technische Universität Berlin, Berlin, Germany

Correspondence

Alexander Oppermann, Department 8.5 Metrological IT, Physikalisch-Technische Bundesanstalt (PTB), Abbestraße 2-12, 10587 Berlin, Germany.

Email: alexander.oppermann@ptb.de

In Europe, measuring instruments under legal control are responsible for an annual turnover of 500 billion Euros and contribute a significant part to the economy including establishing trust between all stakeholders. In this article, a secure cloud reference architecture for measuring instruments is presented, addressing both requirements and roles in the Legal Metrology framework. With the introduction of Cloud Computing in Legal Metrology, a new role of a Cloud Service Provider has to be established. The general approach of the reference architecture shall be evaluated to determine if Cloud Computing can be integrated into the legal framework. In a bottom-up approach, each layer of the cloud is addressed and carefully tested against the essential requirements for Legal Metrology. Splitting a well-contained measuring instrument into a distributed measuring system creates new challenges guaranteeing security and integrity of the measurements. Addressing these problems, technologies such as fully homomorphic encryption are evaluated, improved, and implemented to enable calculations on encrypted measurements. In addition, a secure communication protocol for encrypted data is presented to address the demand of integrity of encrypted measurements throughout their lifecycle. Lastly, a continuous monitoring approach is presented to detect anomalies and to classify the system behavior depending on their severity and impact into three categories: green, yellow, and red.

KEYWORDS

applied cryptography, cloud computing, distributed computing, fully homomorphic encryption (FHE), legal metrology, secure computing, smart meter gateway, trusted cloud

1 | INTRODUCTION

In the last decade, Cloud Computing has been developed constantly, overcoming different challenges to mature in the fields of security, stability, and reliability. Thus, there is a need for in-depth evaluation to determine if Cloud Computing is mature enough to meet the demands, requirements, and challenges of Legal Metrology. According to estimations about four to six percent of the gross national income is declared by Legal Metrology in European countries[1] (see Section 1.1).

Requests are increasing from manufacturers of measuring instruments 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 introduced 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. Costs will be reduced significantly in administration, service, and maintenance. In addition, in case of data processing, it will be stored and administrated centrally, and other software processes will be externalized, that is, moved to the cloud.

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 (see Figure 4).[2] The remainder of the measuring instrument with metrological relevance, like data processing and storing units, will be centralized in a secure Cloud Computing solution.

The interaction of all market participants, such as the manufacturer, user, Notified Bodies, and the market and user surveillance, will change with the introduction of Cloud Computing in Legal Metrology. Furthermore, new roles will be created, like Cloud Service Provider, and their duties and functions have to be determined and embedded into the legal framework of legal metrology.

In Europe, all measuring instruments that are liable to Legal Metrology have to undergo a conformity assessment in order to prove that all requirements of the European Measuring Instrument Directive (MID) are met.[3] In Germany, the MID is regulated via the German Measures and Verification Act (MessEG) and further imposes requirements for national regulated measuring instruments. Additional support for technical and regulatory issues, besides the MID, is provided by the WELMEC Software Guide[4] and the International Organization of Legal Metrology (OIML) Guide[5] (see Section 1.1). The development of new standards, validation, and recommendation for Cloud Computing in Legal Metrology is part of further investigation. This research will be part of the guidance documents for manufacturers in order to keep the state of the art.[2]

As part of an ongoing research project, a secure cloud reference architecture that fulfills the essential requirements of European norms and standards for the highest security level is presented in the following sections. This architecture is adaptable to instrument-specific needs as well as the security level.

The remainder of this article is organized as follows: Section 1 gives an overview of the legal requirements; Section 2 presents the secure Cloud Computing architecture in a bottom-up approach; Section 3 introduces homomorphic encryption and highlights our contribution to this field; Section 4 describes the application scenarios and presents primary results; Section 5 gives an outlook of the monitoring approach for the secure cloud architecture; and finally, Section 6 and 7 provide conclusions and further work, respectively.

1.1 | Legal Metrology

Legal Metrology covers a wide range of measuring instruments from commercial or administrative purposes to measurements with public interests. In Germany, over 100 million legally relevant meters are in use.[6] 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, for example, super markets. The traffic system, for example, requires meters for speed or alcohol meters in a large amount in order to guarantee safety. What all of these applications have in common is that neither the user nor the affected person can check the validity of the determined result; rather, they rely on the accuracy of the official 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 the trust in those measurements. In addition, Legal Metrology fulfills the purpose of ensuring the functioning of the economic system as well as protecting the consumer at the same time.

The 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[5] focuses especially on software requirements for legal measuring instruments.[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 (publicly or privately organized departments to verify measuring instruments) throughout Europe and manufacturers alike, who implement the MID (see Section 1.4).[7,8]

1.2 | Measuring Instrument Directive

The MID comprises ten types of measuring instruments that are of special interest and importance to the economy due to their widespread 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 1). The *PTB* is a Notified Body in Germany. Aside from that role,

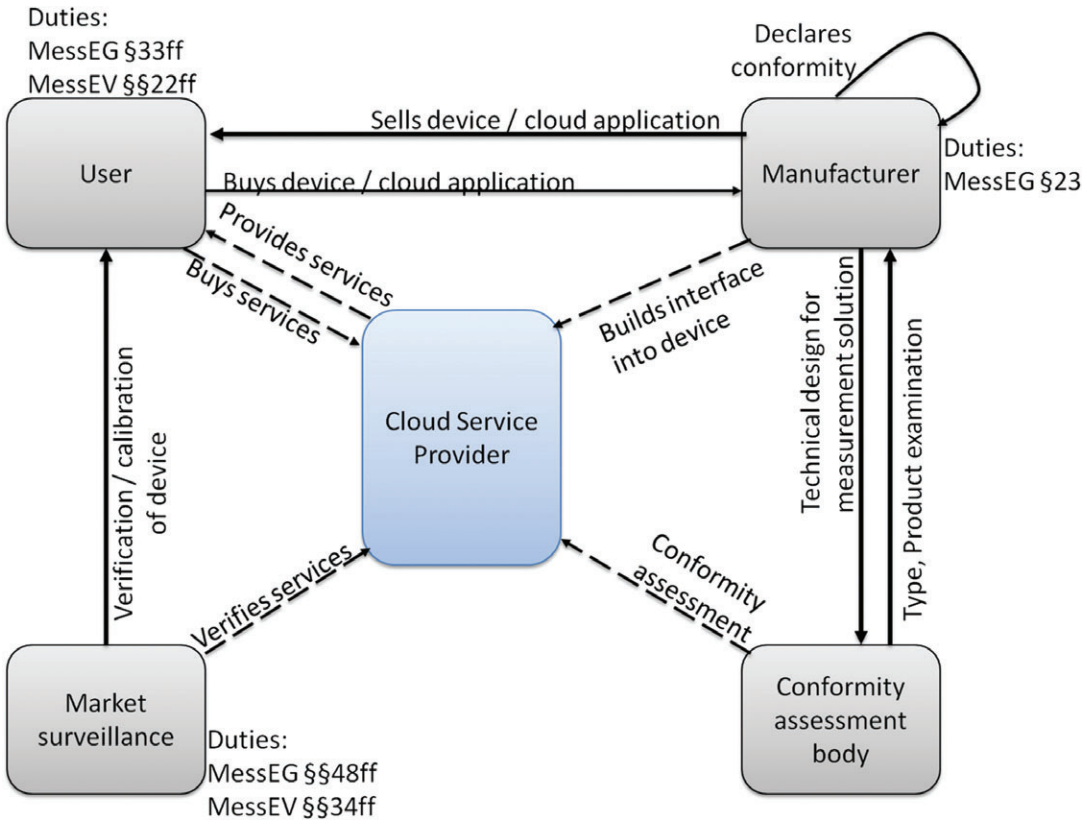

FIGURE 1 Overview of the different actors, their roles, and schedule of responsibilities and duties in Legal Metrology. The Cloud Service Provider role has to be determined

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 advising on European regulations. Given the scope of these responsibilities, it is crucial that the Notified Body be independent and neutral in order to be able to fulfill its duties impartially.

1.3 | 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; and 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 article, 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.

The most important MID software requirements are listed below, and in addition, Figure 2 displays platform requirements:

- Reproducibility of measurement results must be guaranteed, even if handled by different users.
- Durability of the measuring instrument's software over a period of time must be guaranteed. A measuring instrument shall be designed to reduce as far as possible the effect of a defect that would lead to an inaccurate measurement result, unless the presence of such a defect is obvious.
- A measuring instrument shall have no feature to facilitate fraudulent use, and possibilities for unintentional misuse shall be minimal. The latter points request that the impact of manipulations and defects are reduced as far as possible.
- A measuring instrument shall be designed to allow the control of the measuring tasks after the instrument has been placed on the market and put into use. Furthermore, software identification shall be easily provided by the measuring instrument.

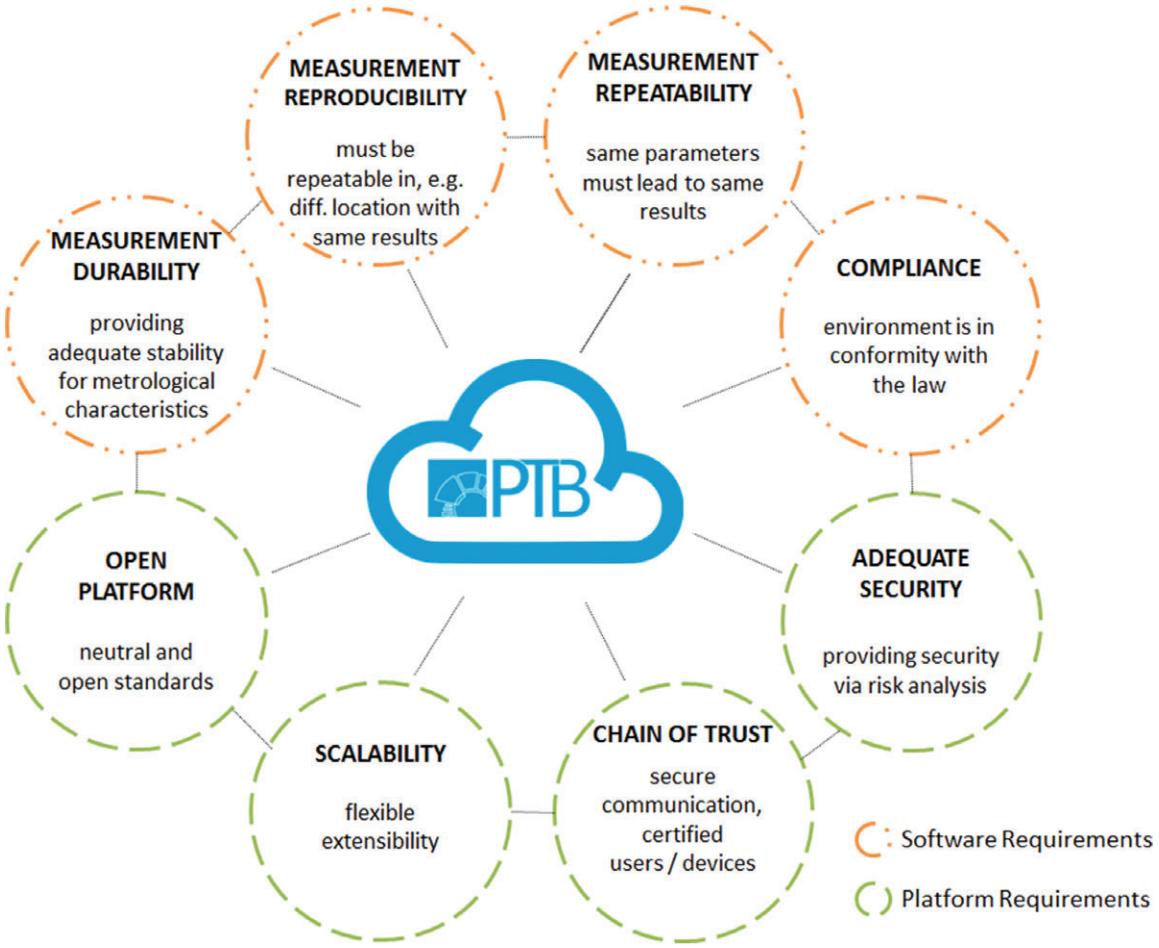

FIGURE 2 Overview of the different requirements in Legal Metrology for measuring systems. *Orange* represents the demands for handling measurement data. *Green* represents the platform requirements

1.4 | WELMEC

The European Free Trade Association (EFTA) and European Cooperation in Legal Metrology are responsible for WELMEC. 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 distributes the *WELMEC 7.2 Software Guide*. The current issue is WELMEC 7.2 version 2015.[4]

The WELMEC 7.2 Software Guide gives an overview on software security with special focus on measuring instruments. Furthermore, it helps manufacturer 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.

When designing software for secure measuring instruments, it is advisable to isolate the legally relevant part 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 protected interfaces. Moreover, if the software is built modular, it will ease the update process for manufacturers and Notified Bodies. Legally relevant modules are defined by WELMEC 7.2 Software Guides as modules that make a contribution to or influence measurement results, for example, displaying data, protecting data, saving data, identifying the software, executing downloads, transferring data, and checking of received or stored data.[4]

1.5 | 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. First, the manufacturer of an instrument is responsible for putting instruments that comply fully with the directive on the market and into use. To assure conformity with the MID, the manufacturer submits a prototype of the instrument to a Notified Body for conformity assessment.

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.

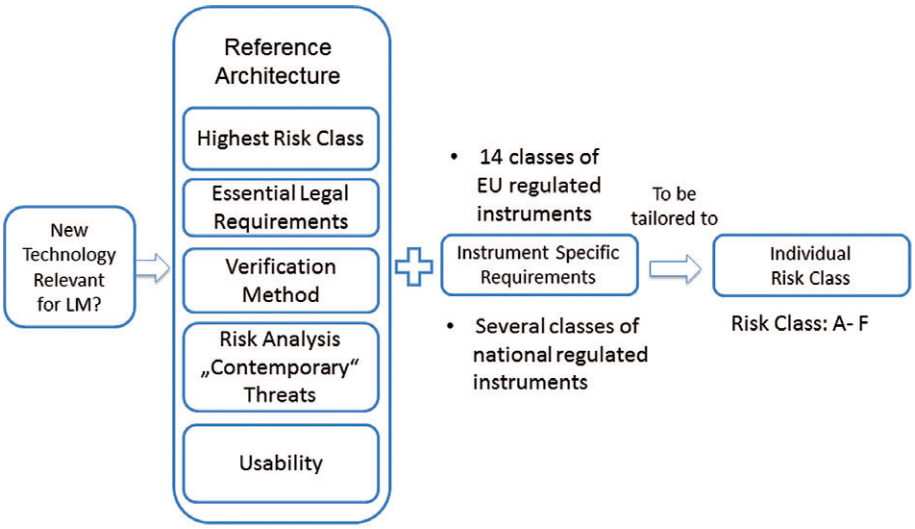

FIGURE 3 Overview of the requirements for a reference architecture in Legal Metrology. Furthermore, measuring instrument-specific requirements and the demand to be adaptable to different risk classes are displayed

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 ID 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 seen as potential attackers (see Figure 1).

1.6 | The role of Cloud Service Provider

In Legal Metrology, there are four important roles established: the Notified Body, the manufacturer of the measuring instrument, the user of the measuring instrument, and the market and user surveillance (see Section 1.5). 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 1). Either the role of the cloud service will be filled from the manufacturer as part of his responsibilities and 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 and the user, respectively, but under private law, the Cloud Service Provider can be held responsible via service-level agreements (SLAs).

1.7 | Reference Architecture

As with all fields, Legal Metrology is undergoing fundamental changes. The world and its technologies is changing in 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.

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, Cloud Computing, the technological knowledge is gained, and an architecture is built specifically 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.

In order to prove if new technologies are important and viable to the field of Legal Metrology, it has to fulfill a number of requirements (see Figure 3). A successful reference architecture has to fulfill the essential requirements of the European directives[3,10] and guidelines[4] (see Sections 1.3 and 1.4).

Furthermore, it has to provide an easy applicable verification method for the market and user surveillance (see Figure 10) as they do not have the time nor the financial resources to dig deep into the used technology stack. In addition, a risk analysis of contemporary threats and attacks has to be carried out and evaluated.[9] This will not be part of the present article. Finally, the

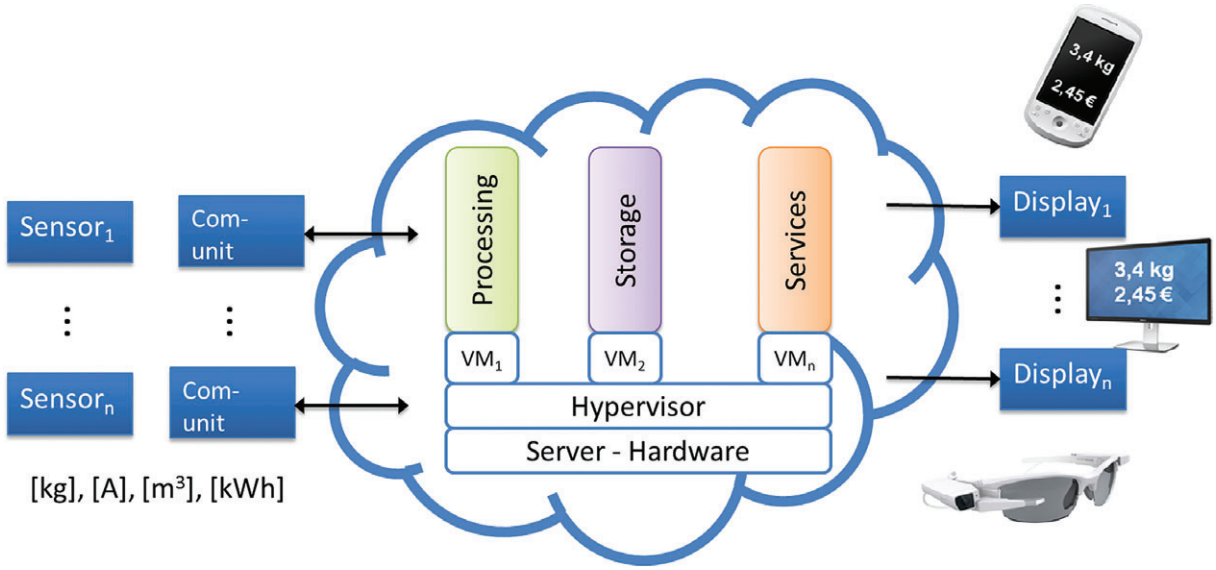

FIGURE 4 Overview of the reference architecture concept for a distributed measuring instrument with a sensor and a communication unit in the field. Processing, storage, and service units are moved to the cloud. The client is left with a secure display on a variety of devices

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 be able to adapt it to the 14 classes of European and several nationally regulated measuring instruments, which reach from gas, power, heat to water, oil, and speed meters.

2 | RELATED WORK

In an earlier version of the current article, presented at the 2017 International Conference on Intelligent, Secure and Dependable Systems in Distributed and Cloud Environments (ISDDC 2017), fully homomorphic encryption (FHE) was investigated for Legal Metrology applications. Solutions for the long periods of time needed to process encrypted data were found by multithreading the secure data processing and an extension of the FHE library LibScarab was developed to allow arithmetical operations between encrypted measurements. The evaluation of the proposed approach yielded encouraging results for the parallelizing algorithms, solving the time constraints, although the full secure Cloud Computing reference architecture was lacking a full frame to prove its applicability within the Legal Metrology framework. The current article extends the work presented in Oppermann et al "Secure Cloud Computing: Multithreaded Fully Homomorphic Encryption for Legal Metrology." International Conference on Intelligent, Secure, and Dependable Systems in Distributed, and Cloud Environments, Springer, Cham, 2017, by providing a full overview of the cloud-based architecture and proposing a condition monitoring approach, specifically metrics-based anomaly detection technique.

Similiar approaches like SensorCloud,[11] SealedCloud[12] and TRESOR[13] 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 patients 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 | SECURE CLOUD COMPUTING ARCHITECTURE

As a premise for this secure Cloud Computing architecture, a transition has to be accomplished from well-contained measuring instruments to a distributed measuring system (see Figure 4). The radical change that imposes the distribution of components, with only a physical sensor and communication unit left in the field, while processing and storage will be centralized, was already pointed out in the last section for the legal framework in Legal Metrology. In this section, the focus lies on a security approach for a secure cloud reference architecture that fulfills the essential requirements of the MID (see Section 1.3) while at the same time providing adequate security and preserving the flexibility and economical advantages of Cloud Computing.

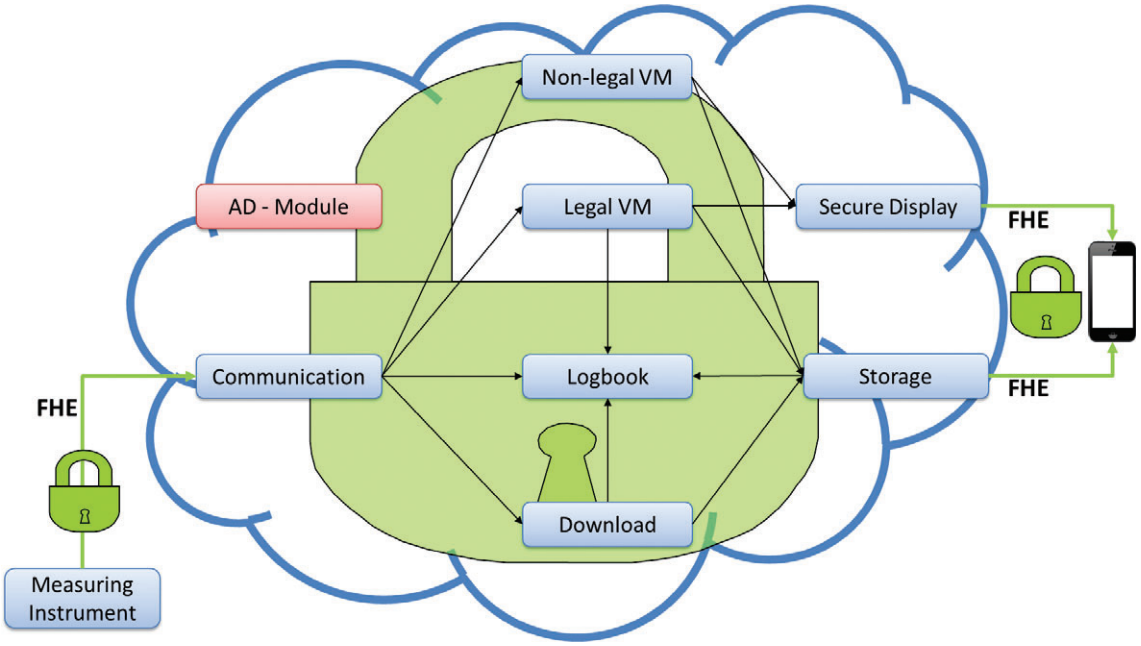

FIGURE 5 Architectural model overview of virtual machines and their communication paths

The different layers of this architecture are described in the bottom-up approach and are detailed in the following subsections. Furthermore, the communication and security measures of such an architecture will be discussed.

3.1 | Architectural Approach

The separation of legally relevant software processes and keeping them safe is the driving force to implement virtualization techniques. The most important functions are separated in different compartments. The legally relevant modules are listed below. A summarized view of the architectural model is visualized in Figure 5. The measurement results are encrypted via FHE within the measuring instrument and then received and processed by the cloud. The encryption endures the whole processing time in the cloud and will be only decrypted outside of the cloud at a secure end-user device.

After already separating the networks into logically smaller entities via subnetting (see Section 3.2), a decision was made to create, for the most important legally relevant tasks, separated virtual machines (VMs). These are:

Logbook: The Logbook VM hosts the software process responsible for logging all relevant activities around the measuring system, that is, arrival, processing, saving of measurement data, user activities, software updates, and so on.

Legal Processing: The legal VM is responsible for processing measurement data. This VM has the most CPU cores available as 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 update 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 (in Figure 5, it is the red anomaly detection module).

3.2 | 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 mock 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 VMs. The four subnetworks are divided into an ingress/entrance

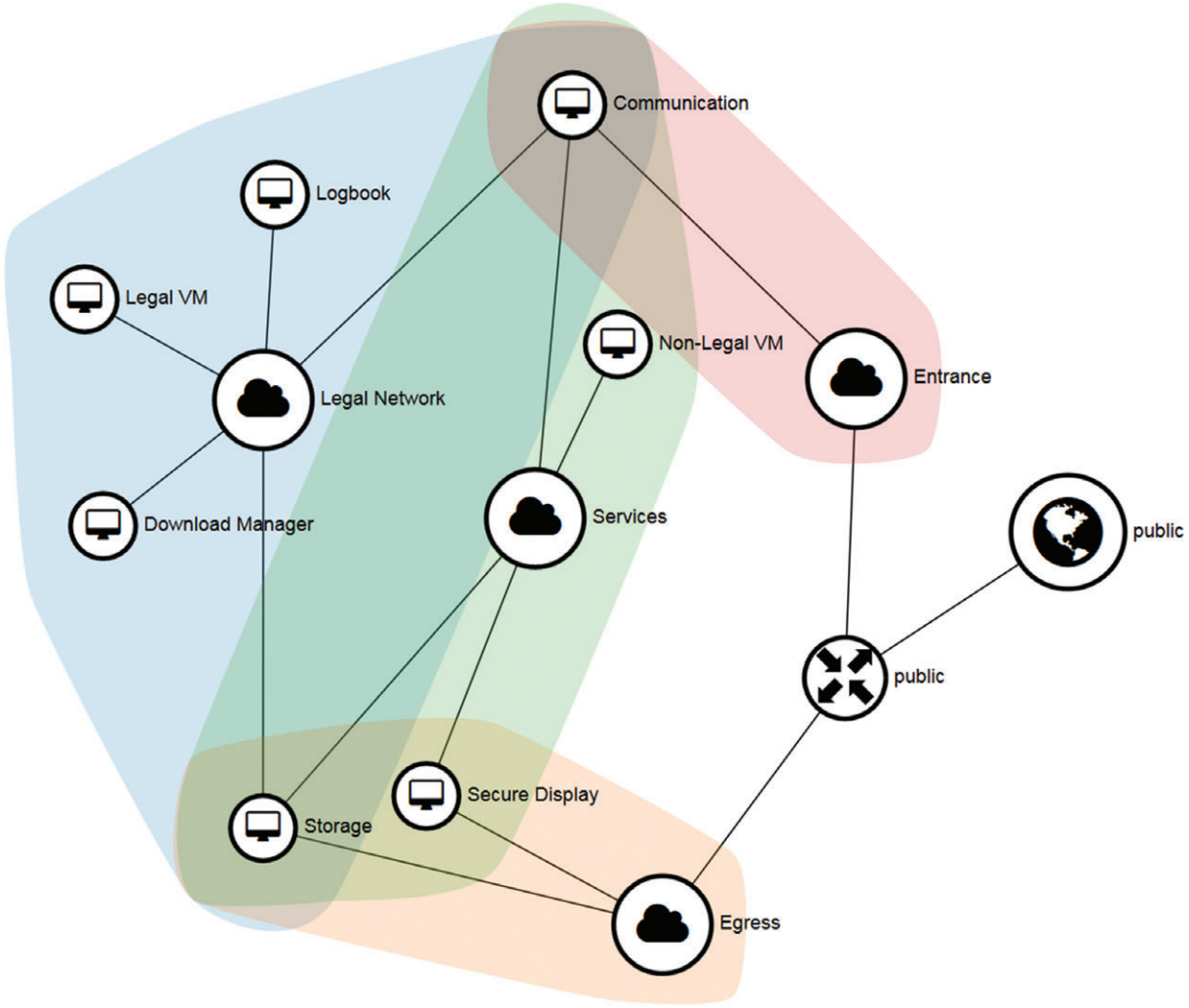

FIGURE 6 Overview of the different requirements in Legal Metrology for measuring systems. Orange represents the demands for handling measurement data. Green represents the platform requirements

subnetwork, a Legal Metrology subnetwork, a services subnetwork, and an egress/exit subnetwork. In case this concept should be distributed across different locations, for example, data centers, countries, and so on, then it would be sensible to use Virtual LAN and to aggregate different VMs across physical boundaries. In addition, it would enable a separation already on OSI layer 2. For the sake of simplicity, subnetting is sufficient for this prototype.

Ingress/entrance network: The ingress/entrance network hosts the communication VM (see Figure 6, marked red), which acts as a gateway to the Internet for the rest of the VMs. This design decision was made deliberately in order not to expose the legal network directly to the Internet and increase its security.

Legal Metrology network: The Legal Metrology network hosts three legally relevant VMs (see Figure 6, 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 6, marked green) that want to host legally non relevant VMs and services, for example, 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 6, marked yellow); thus, it hosts the secure display VM that 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 also enables the first low-level separation of legally relevant processes.

3.3 | Platform as a Service—Microservices

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 multitier architecture[14] as it places the developer tools at disposal to coordinate and handle processes and

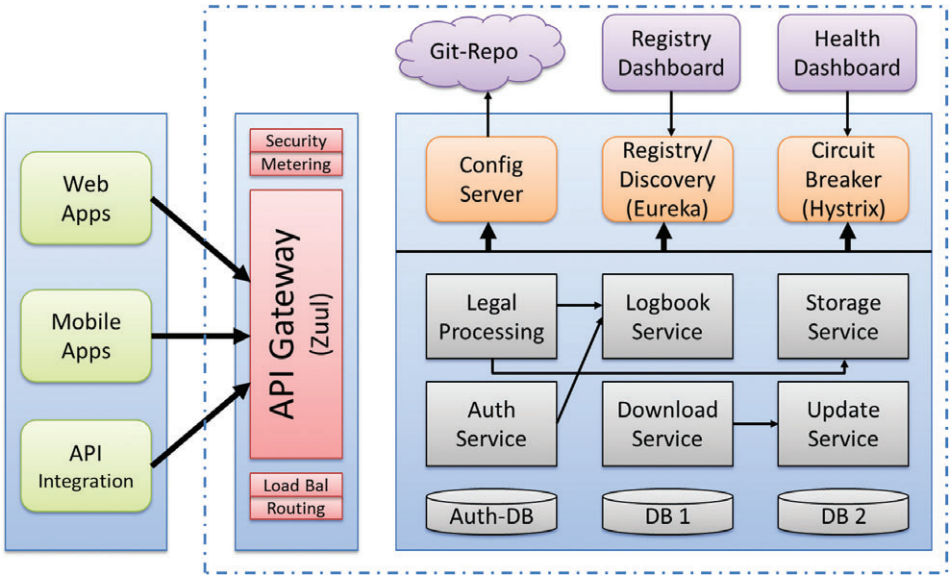

FIGURE 7 Overview of the platform concept using the open-source Netflix stack. Green represents the external devices that interact with the platform via a gateway (in red). Orange and purple represent the orchestration layer. Gray represents chosen microservices with an exemplary data flow

applications and further integrate them into the surrounding layers. A PaaS usually comprises execution runtimes, 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 perfectly fits the architectural premises of Cloud Computing for designing distributed service-oriented architecture (SOA). The pattern was implemented by using the open source Netflix-stack for microservices.

Microservice architecture—A microservice architecture is a fine-grained SOA pattern that creates, for each task and process, a service. A service is focused only on one task, thus 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 as 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 at 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 7). This prevents the 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 also reduce 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 pseudoresilience as the downtime can only last as long as the size of the queue can handle the incoming messages.

Disadvantages: Despite all the advantages of a microservice pattern, there are some challenges. The flexibility and scalability comes at the 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 because of their distributed nature. They externalize their communication, and this can become a serious problem when network latency adds up.

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.

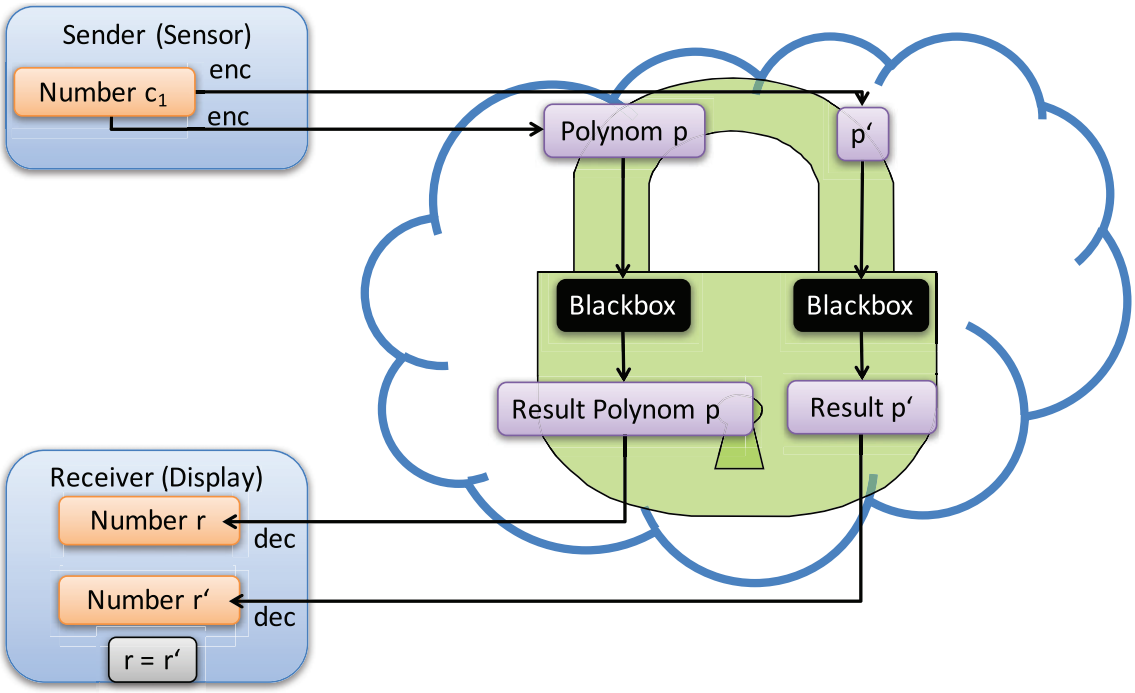

FIGURE 8 Overview of encrypted communication protocol

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 as 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 fallback can be implemented. This service increases the overall resilience of the distributed system.

3.4 | 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 it can be easily distributed within the Cloud Computing nodes.

In this reference architecture implementation approach, this tier offers the FHE application for smart meter gateway (SMGW) tariff applications, which take full advantage of the LSIM extension (see Section 4.4).

3.5 | Communication protocol

An overview of the proposed communication protocol is described in detail below (see Figure 8). 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 sent via a secure channel directly to the trustworthy receiver (steps 4-6) in order to be able to decrypt the encrypted measurements results (step 18) produced by the cloud (server). With the public key, the client encrypts the same measurement twice and sends them to the cloud. This procedure creates the simplest form of a signature to ensure authenticity and integrity of the measurement results.

Even though homomorphic encryption usually aims at ensuring data privacy,[6] it may be used in other areas and for other purposes as well. A potential 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 with 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 precalculated data will decrease the possibility of an attacker staying undetected for a long period.

If homomorphic encryption is carefully combined with testing a running algorithm via precalculated test data, such random effects may be detected. Subsequently, the scheme detailed here[15] may be used to achieve a certain degree of robustness toward algorithm and data manipulation as well.

Algorithm 1 LSIM client-server communication protocol (overview)

Input: Measurement data $m_1 \dots m_n$, Encryption Parameter eP, Public Key P_k, Secret Key S_k, Usable bandwidth upload/download (U,D), Index of elements $i = 1 \dots n$

Output: Measurment results $mr_1 \dots mr_n$

Client:

1: creates measurement data m_i
2: encrypts measurement data and index twice:
3: $c_i \leftarrow m_i$ and $c_i' \leftarrow m_i$,
4: index $v_i \leftarrow i$ and $v_i' \leftarrow i$
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, respectively

Server:

10: checks certificate and the origin of messages
11: **if** check is successful **then**
12: process data $\rightarrow$ c_i and c_i' will be treated the same
13: compute r_i and r_i', respectively
14: **else**
15: throw data away, $\rightarrow$ error message to logbook
16: **end if**

Receiver:

17: authenticates against server
18: pulls measurement results from server
19: decrypts measurement results and index with S_k
20: **if** $dec(r_i) = dec(r_i')$ **and** $dec(v_i) = dec(v_i')$ **then**
21: no manipulation and mr_i is correct
22: **else**
23: throw measurement data away,
24: *return* error message to logbook
25: $flag$ measuring system failure.
26: **end if**

3.6 | Security approaches and their limitations

Classical security approaches use encryption to ensure privacy and signatures for verifying the authenticity of messages. These attempts aim to secure the communication between two endpoints. In the Cloud Computing context, the attention shifted to secure computation (see Figure 9). 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, proof is needed for correct and trustworthy results. Verifiable computing demands that the evaluation of an interactive proof[16] needs to be more compact than the actual computation. FHE offers an elegant way to ensure privacy by design as it prevents clear-text information leaks. Although externalizing confidential processes to the cloud, the industry places

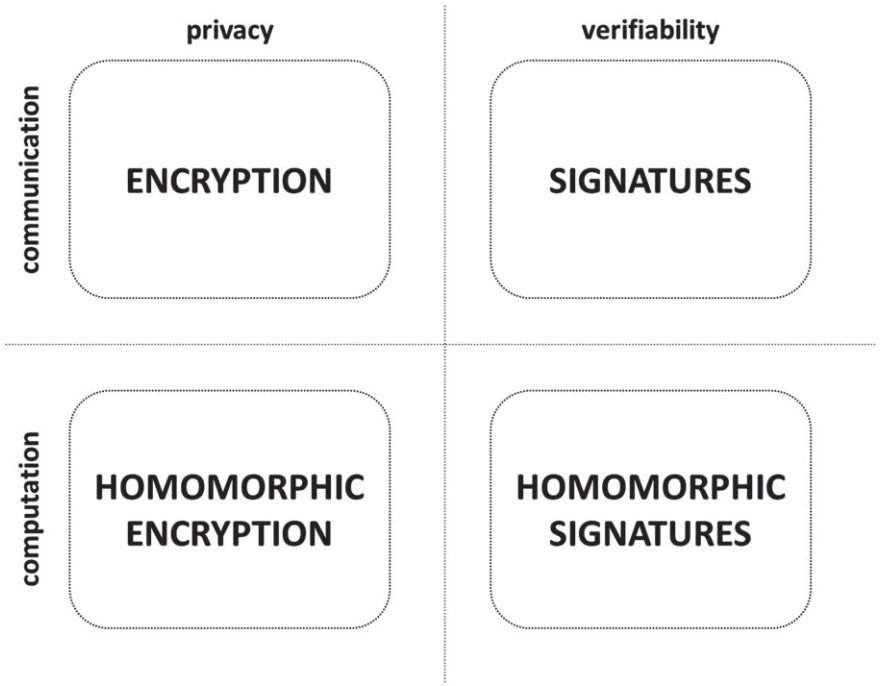

FIGURE 9 Overview of different security techniques and their area of application

little trust in external service providers. The authors chose FHE over a homomorphic signature scheme not only to help solve this conflict; 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[17,18] so that it is still feasible at a later time to integrate verifiable computation with this line of research.

3.6.1 | Measuring instrument and trustworthy components

In the context of the intelligent measuring system, all parties need to authenticate themselves before initiating a communication connection. To this end, the SMGW as well as partners communicating with it over the wide area network (WAN) are in possession of hardware security modules (HSMs), which are responsible for key handling, signature verification, and white listing of communication partners.

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 (BSI, 2013). The HSM performs various checks with the certificates before being able to continue with the respective procedure, that is, it checks the signature of the certificate, the lifetime span (maximum of 7 years) of the issued certificates, revocations lists, and the issuer of the certificate and reviews the mode of usage, such as key usage validation and extended key usage validation.

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

3.6.2 | Verification method for the market surveillance

As part of the envisioned reference architecture (see Figure 3), a verification method has to be developed in order to support the user and market surveillance to verify measuring systems in the field. As the used technology become more complex to validate for use in the field of Legal Metrology, the Notified Body provides support with a simple verification method the authorities in the field use to carry out their tasks.

If an end user expresses doubts to the user and market surveillance of, for example, 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

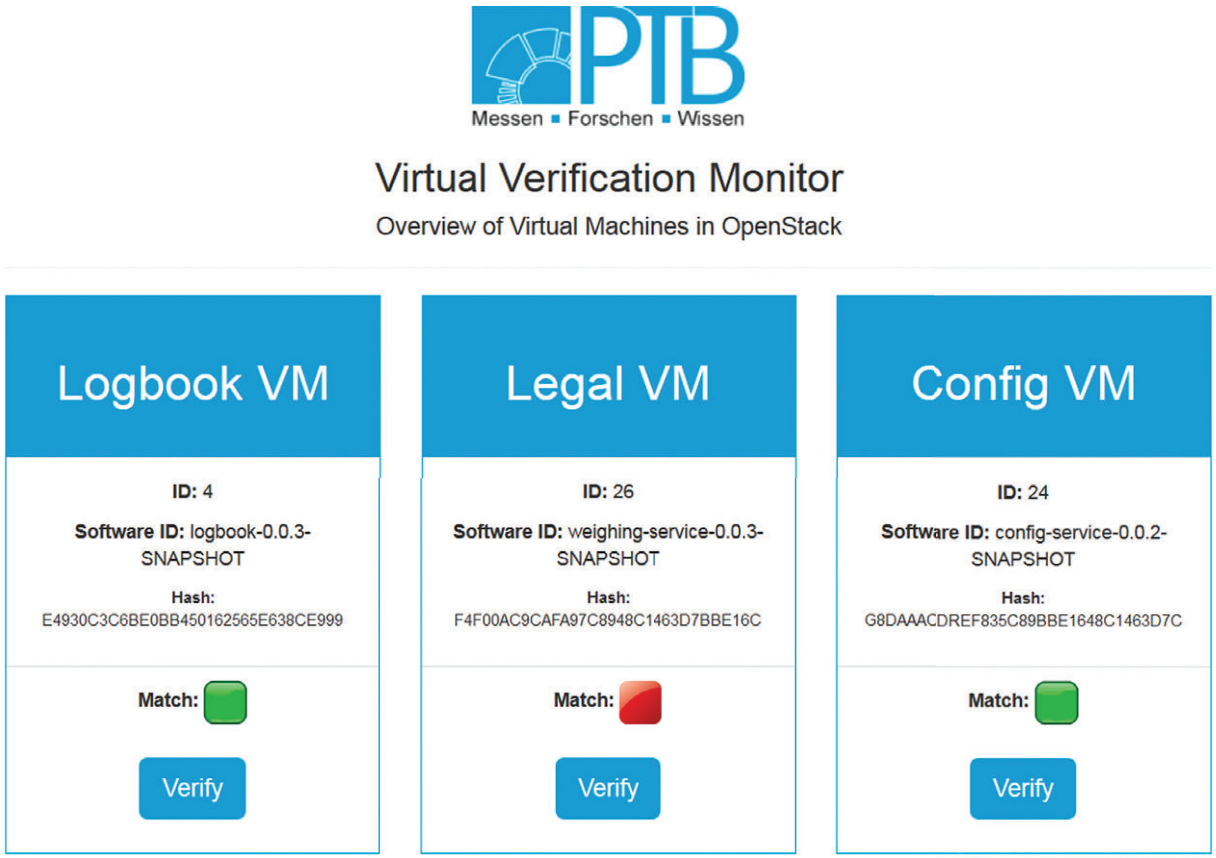

FIGURE 10 Screenshot of the prototype verification monitor for the market and user surveillance

logbook. The metrology software has to provide software identification. An acceptable solution according to the Welmec Guide would be:

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

In Figure 10, a simple web application is displayed, which shows three exemplary legally relevant VMs 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 VMs directly from the used cloud. Each VM 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).

3.7 | Summary

In this section, the envisioned distributed measuring instrument was introduced, and the foundation for the proposed secure cloud reference architecture was laid. As a centerpiece, virtualization on different levels, that is, infrastructure, platform, and services, enables software separation for legally relevant and non relevant processes. 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 was discussed. 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 the present article, the use of the FHE technique focuses on covering this security gap. Finally, a remote verification tool was presented to fulfill the demand reference architecture requirements.

4 | HOMOMORPHIC ENCRYPTION

This section gives the reader a condensed overview of the most important preliminaries of lattice-based cryptography,[22–24] that is, especially FHE[19–21,25] in Section 4.3. Furthermore, the authors' contributions are highlighted in Section 4.4.

4.1 | Principles of lattice-based cryptography

Homomorphism is a structure-preserving transformation in linear algebra between two types of sets (structures), A and O, with a function $f : A \rightarrow O$ and an inverse f', which preserves the operations of the original set (structure), with the result

$$\forall x, y \in A \; : f'(f(x) \oplus f(y)) = x \oplus y. \tag{1}$$

Furthermore, it must guarantee the closure of addition $\oplus$ and multiplication $\otimes$ operations, with the result

$$\forall x, y \in A \; : f'(f(x) \oplus f(y)) = x \oplus y, f'(f(x) \otimes f(y)) = x \otimes y. \tag{2}$$

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 an XOR- and AND-Gate (for an applied approach, see Section 4.5).

Lattice is a set of all integer linear combinations of basis vector $b_1, b_2, \ldots, b_n \in R^m$ so that they form a discrete subgroup (lattice) $\mathscr{L} \subset R^m$ that is defined as

$$\mathscr{L}(b_1, b_2, \ldots, b_n) := \sum_{i=1}^{n} b_i \mathbb{Z} = \left\{ \sum_{i=n}^{n} t_i t_b \mid t_1, \ldots, t_m \in \mathbb{Z} \right\}. \tag{3}$$

We need to further define 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—A matrix $B := [b_1, \ldots, b_n] \in \mathbb{R}^{m \times n}$ is a basis matrix of the lattice $\mathscr{L}$ if $\mathscr{L}(B) := \mathscr{L}(b_1, \ldots, b_n)$ holds true, i.e. B spans the lattice $\mathscr{L}$.

Area of a basis matrix—An area of a basis matrix $B := [b_1, \ldots, b_n] \in \mathbb{R}^{m \times n}$ is the parallelpiped

$$\mathscr{P}(B) := \left(\sum_{i=1}^{n} x_i b_i \mid 0 \leq x_i \leq 1 \right). \tag{4}$$

In the next subsection, the security of lattice-based cryptography will be briefly discussed and put into the perspective of "standard" cryptography.

4.2 | Security of lattice-based cryptography

The security of cryptosystems usually relies on hard-to-solve problems in order to guarantee the computational safety of such a system. Traditionally, one uses functions that are easy to evaluate in one direction, but reversing them by sheer brute force shall be very hard. A short comparison of lattice-based cryptography is summarized in Table 1. The following problems are often found when discussing lattice-based cryptography.

Shortest vector problem (*SVP*)—This problem addresses a set of smallest (shortest) lattice-based vectors $b_i \in \mathscr{L}$, $b_i \neq 0$, where the Euclidean norm of $\|b_i\|$ is minimal.

Closest vector problem (*CVP*)—This problem determines the distance of a point $x \in \mathbb{R}^m$ to a lattice-based vector $b \in \mathscr{L} \subset \mathbb{R}^m$, so that the distance $\|b - x\|$ is minimal.

TABLE 1 Comparison of lattice-based cryptography against "standard" cryptography

Lattice-based cryptography	"Standard" cryptography
Provable secure	Not always provable
Based on worst-case security	Based on average-case security
Based on hardness of lattice problems	Based on hardness of factoring discrete log, etc.
(Still) not broken by quantum algorithm	Broken by quantum algorithm
Very simple computation (efficient for weak hardware)	Requires modulus exponentiation, etc.

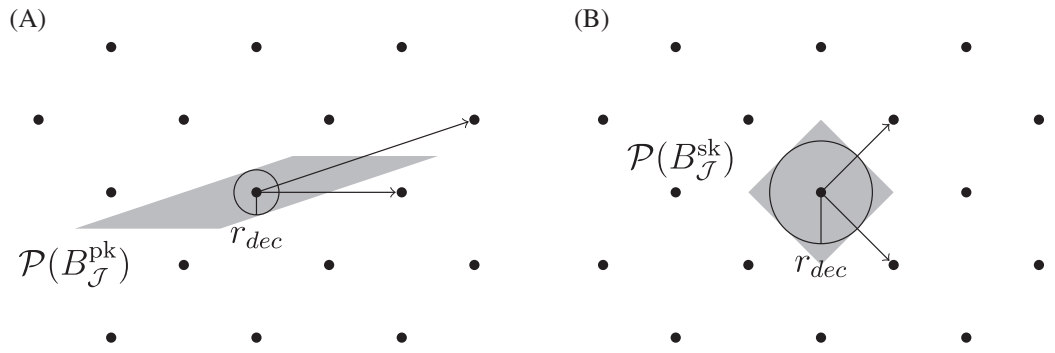

FIGURE 11 Comparison of public and secret basis B of a lattice $\mathscr{L}$[18] (A) Small radius r_{dec} for public basis $B^{\mathrm{pk}}_{\mathcal{J}}$. (B) Big radius r_{dec} for secret basis $B^{\mathrm{sk}}_{\mathcal{J}}$

4.3 | Gentry-based homomorphic cryptosystems

In 2009, Gentry et al[19] 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—given an isomorphic quotient ring $\mathbb{Z}[x]/\langle f\rangle$, where f is an irreducible monic polynomial to the integer lattice $\mathbb{Z}^n$, so that any ideal $\mathcal{I} \subseteq \mathbb{Z}[x]/\langle f\rangle$ creates an integer sublattice $\mathscr{L} \subseteq \mathbb{Z}^n$. Subsequently an ideal lattice is $\mathscr{L}(B) \subseteq \mathbb{Z}^n$, such that $B = \{g \ mod\ f : g \in \mathcal{I}\}$, for an irreducible monic polynomial f and an ideal $\mathcal{I} \subseteq \mathbb{Z}[x]/\langle f\rangle$.

Building a cryptosystem on ideal lattices with a quotient ring result in a homomorphic property as ideals guarantee closure of addition and multiplication (see homomorphism definition 4.1). Furthermore, Gentry et al are proposing that in order to generate an asymmetric key pair, two bases $B^{pk}_{\mathcal{J}}, B^{sk}_{\mathcal{J}}$ should be created, 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 11A). It will be used for encryption and can be easily reduced by Gaussian elimination procedure, that is, simple computation, no need for expensive hardware.

The second basis creates the secret key (see Figure 11B). The basis vectors of this secret basis are orthogonal to each other and thus are enabled to find a representative of a vector in this lattice. It is worth noting 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 $B^{pk}_{\mathcal{J}}$ provides no hint of the underlying CV problem (see closest vector problem in Section 4.2).

The biggest disadvantages of the original Gentry-based approach were the enormous key sizes, especially for the public key averaged in gigabytes, and the time-consuming recrypt operations. These problems are addressed among others by Smart and Vercauteren,[26] which lay the foundation for the cryptographic library LibScarab by Brenner et al[27] that is used for our contribution in the next section.

4.4 | LibScarab extended for integer arithmetic and multithreading (LSIM)

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

One possible way to implement a division algorithm is to represent numbers as fractions by saving the 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 without giving up security and privacy. This means that all algorithms should be either completely deterministic, or there should be enough computing power to calculate all possible results in parallel.

A final decision should always be made in the unencrypted domain for the targeted algorithms. As they neither have a completely deterministic structure nor the option of the decryption of the data in an insecure environment, a different approach was pursued, replacing integers with binary numbers.

The LibScarab library is a good choice for the prototype as it provides all the 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 multithreading out of the box as 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.

4.5 | Implementation of arithmetic operators

Using binary numbers, only two operations are left: the modulo-2-addition that corresponds to an XOR-gate and a modulo-2-multiplication that corresponds to an AND-gate. Another operation is the (unencrypted) binary complement of the encrypted number, which corresponds to toggling bits in the encrypted domain. With these two operations (see Equations 5 and 6), 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. The addition, subtraction, and multiplication operations for integers had to be reimplemented according to the chosen binary word sizes (e.g., 32-bit and 64-bit length).

$$A \vee B = \neg(\neg A \wedge \neg B) \tag{5}$$

$$((1 \oplus A) = \neg A). \tag{6}$$

4.6 | Zero-test of encrypted integers

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 input for further encrypted arithmetic and logical operations. To verify the computed result, it has to be decrypted in the end.

4.7 | Comparison of encrypted integers

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 as the result of the subtraction itself is not important.

4.8 | Simple decisions on encrypted integers

It is significant to highlight that the result of a decision should remain in the encrypted domain. Therefore, 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:

$$Y = (condition)?A : B \tag{7}$$

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

$$Y = \neg C \cdot A + C \cdot B \tag{8}$$

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.

4.9 | Addition of encrypted integers

The implementation of arithmetic circuits is performed with special regard to multithreading. A particular challenge was to find a solution with a small amount of gates and a small circuit depth at the same time. An analogue problem is known in

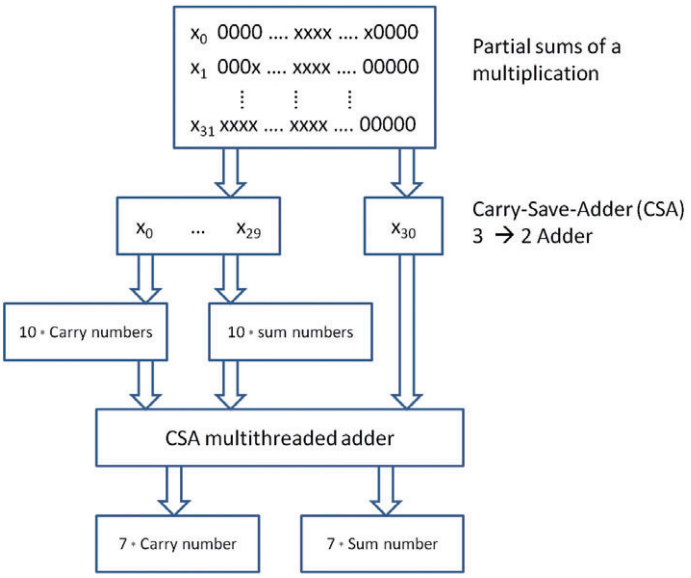

FIGURE 12 Tree structure of the optimized adder part

the field of digital design as "Area-Speed-Tradeof" as a lower circuit depth usually results in a higher number of gates used. Each additional gate needs computing time; thus, fast digital circuits often have 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)[28] with block size of 4 and 8 bits, also known as "fast adder" proved to be the slowest adder within the constraints of this article. The Carry Select Adder (CSA)[28] rendered mid-range performance. The modified Ripple Carry Adder (RCA)[28] revealed the shortest execution time as the operations of the first half adder was 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, subtraction was outsourced to a separate routine in order to save time during addition operations.

4.10 | Multiplication of encrypted integers

Fast multiplications using higher radix solutions[28,29] 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 and (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 multilevel bit-wise tree adder is used here (see Figure 12).

The main principle of a tree adder is that a full adder is used to add 3 bits. At the output, 1 sum bit and 1 carry bit are generated. The resulting number, consisting of carry bits, is shifted 1 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 that 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 multithreaded operations.

4.11 | Division of encrypted integers

The division operation is the most difficult one to parallelize because of interstep 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 also increases in this case. Advanced higher-radix algorithms like Sweeney-Robertson-Tocher (SRT) division uses lookup tables and/or a lot of multiplexers.[28,29] These are very expensive in case of a FHE software implementation, so 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.[28] 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

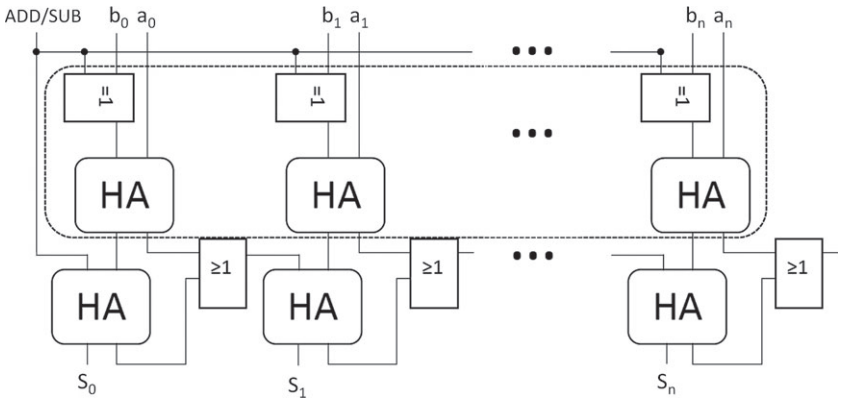

FIGURE 13 Multithreaded combined adder and subtracter

use of a multiplexer in each iteration. In contrast to the restoring division the nonrestoring 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 1 subtraction, the divisor (D) should be added back to the remainder (R) and the result (S) shifted to the left, that is, $S = 2 \cdot (R + D)$. 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 13). The sign-bit of the result is used to select the operation (addition/subtraction) for the next iteration.

4.12 | Summary

In this section, the field of FHE was introduced and its mathematical groundwork briefly explained. The base problems such as SVP and CVP for lattice-based cryptography were introduced and their significance emphasized. Advantages of lattice-based cryptography in comparison to traditional cryptography are summed up in Table 1. The contribution of the authors was highlighted in extending the FHE library LibScarab for zero-test, comparison, and simple decisions for encryption integers. Moreover, basic arithmetical operations, such as addition and multiplication, were extended for 32 and 64 bit usage, and division and subtraction were added to the library content. These contributions were necessary to allow implementation of encrypted tariff models for SMGWs.

5 | APPLICATION SCENARIOS

While the last section explained the foundation of FHE and highlighted the authors' contributions, this section focuses on the applied application scenarios for SMGW tariffs according to the Federal Office for Information Security (BSI). Four tariffs will be briefly explained along with the algorithmic requirements that are needed to implement these scenarios. The results of the homomorphic operations along with the yielded results of the application scenarios are presented.

5.1 | Tariff models according to TR03109-1

Even though homomorphic encryption usually aims at ensuring data privacy,[16] 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, only 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 toward algorithm and data manipulation as well. 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.[30] 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)[28] with an associated protection profile.[29] While the first mainly covers compatibility requirements and secure communication protocols for the SMGW, the latter is only focused on the SMGW's integrated HSM. Apart from defining communication protocols and the general environment of an intelligent measuring system, the TR also lists all approved tariff application scenarios (TAFs) 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.

Of the 12 TAFs defined in the TR, four will be examined and implemented in a secure cloud solution in this article. 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 Section 4 will be derived.

Tariffs with low data usage - TAF I

Tariffs with low data usage are tariffs 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 are 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 are tariffs where the price does not depend on the total energy consumed but on the current power consumption (energy per time interval). The process of switching to different price categories is therefore performed according to the value of the current measurement result.

Consumption-dependent tariff - TAF IV

Consumption-dependent tariffs are 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 algorithm described in Section 4.4 aims to realize 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:

- addition, multiplication
- 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.

5.3 | Results 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. In the first subsection, key generation, recrypt operation, and single arithmetic operations of FHE are compared after improving and extending LibScarab as described in Section 4.3. Section 5.4 comprises results of the application scenarios TAF 1-4 that were already outlined in Section 5. The speedup and efficiency are calculated for both single operations as well as for the tariff applications.

TABLE 2 Overview of key geometry and performance gain in comparison to Brenner et al[27]

Key geometry	Key size [kB]	KeyGen [s]	Speedup	Recrypt [ms]	Speedup
384/16/05	1.8/30	1.1	15.5	34	7.7
384/32/16	1.8/60	1.17	11.9	90	3.4
384/64/16	1.8/120	1.24	12	181	3.7
2048/64/16	9.6/626	516	6.1	670	2
4096/64/16	18/1250	5204	2.7	1660	1.9

The relative performance gain achieved by parallelization (speedup) can be measured and is defined in Grama et al[33] as a metric. This can be written as:

$$S(n) = \frac{T_s(n)}{T_p(n)} \tag{9}$$

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 Grama et al[30] as a metric that measures the fraction of time in which a processing unit is usefully employed. This can be asserted as:

$$E(n) = \frac{S(n)}{n} \tag{10}$$

where S is the speedup for the algorithm (see Equation 9), and n marks the number of processing units.

5.3.1 | Key generation and recrypt operations

While parallelizing the basic arithmetic operations, for example, 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[26]) in a multithreaded environment compared to a single-threaded one, that is, no relevant time gain for key generation is realized. The authors 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 2 for key generation and recrypt operation are single threaded. Nevertheless, in comparison with Brenner's reference implementation[27] and the stated times for key generation for 384-bit key length, a speed up of 15.5 could be yielded, that is, instead of 17 s, it only took 1.1 s to generate a key. The main factors are optimizations in the implementation, modernized libraries as already explained in Section 4.4, and faster hardware than in 2012. The recrypt operation achieved a speedup factor of 7.7 for 384-bit key length, that is, a recrypt costs only 34 ms instead of 263 ms. The amount of disk space for public and private keys are not different 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 14A, one can see the asymptotic characteristic of the optimization for the time usage. While executing a single-thread utilization took 124 s for a single multiplication, a 48-thread utilization only took 17 s for a single multiplication. Considering memory usage, the optimum is reached by utilizing 16 threads with 19 s for a single multiplication with a calculated efficiency gain of 41% (see Equation 10) 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 2 s for one addition for a single thread, while it could be pushed down to 1 s for a single addition utilizing 48 threads. The efficiency optimum is reached here by using only 2 threads with a 50% efficiency and a speedup of 1 needing 2 s. 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 took 196 s, while this was reduced to 121 s utilizing 48 threads. The optimum for this operation is reached using 16 threads with a speedup of 1.6 and a 10% efficiency. Similar results could be yielded for 64-bit arithmetic operations as seen in Figure 14B. 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 min (470 s) for a single-threaded multiplication compared to 69 s using 48 threads. The optimum is reached using 16 threads, with a speedup of 6.3 and an efficiency gain of 39% needing 75 s. Additions in the 64-bit space are optimized by a factor of 1.6, that is, needing 5 s for one addition using one thread compared to 48 threads lasting 3 s. The optimum is reached for four threads with a speedup of 1.3 and 31% efficiency. Again, addition is the fastest operation.

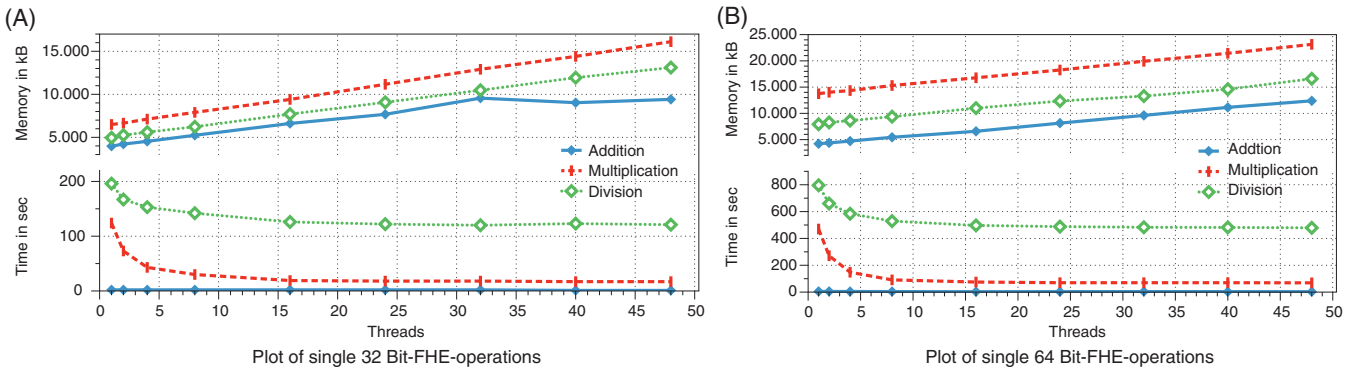

FIGURE 14 Plot of FHE operations (Add, Mult, Div) on a Server with an Intel Xeon CPU E5-2620 v3 @ 2.40 GHz and 64GB RAM

For 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 min (796 s) for a single-threaded division compared to about 8 min (479 s) using 48 threads. The optimum is reached using 16 threads with a speedup of 1.6 and 10% efficiency needing 8.5 min (497 s). A full overview for the 32- and 64-bit arithmetic operation results can be seen in Table 3.

5.4 | Results of application scenarios

In contrast to the tests performed in Section 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 Section 5. 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 15A) and summing them up on demand (see Figure 15B), for example, 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 15A) with respect to time and utilizing threads, the gain is 1.3, needing about 5 s for a single thread to only 3.8 s 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 4.3 s.

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

TAF 3 is the only scenario where a lot of comparisons and decisions were performed additionally, hence the time difference. A gain of factor 2.4 was yielded for a single thread needing 33.4 s compared to 48 threads consuming only 14 s. The optimum is

TABLE 3 Overview of 32- and 64-bit operation results

	32-bit operation			64-bit operation results		
Threads (n)	***t* [s]**	***S*(*n*)**	***E*(*n*) [%]**	***t* [s]**	***S*(*n*)**	***E*(*n*) [%]**
Add						
1	2	—	—	5	—	—
2/4†	2	1	50	4	1.3	31
48	1	2	4	3	1.7	3
Mult						
1	124	—	—	470	—	—
16	19	6.5	41	75	6.3	39
48	17	7.3	15	69	6.8	14
Div						
1	196	—	—	796	—	—
16	126	1.6	10	497	1.6	10
48	121	1.6	3	467	1.7	3

†In case of 64-bit operations, 4 threads were employed.

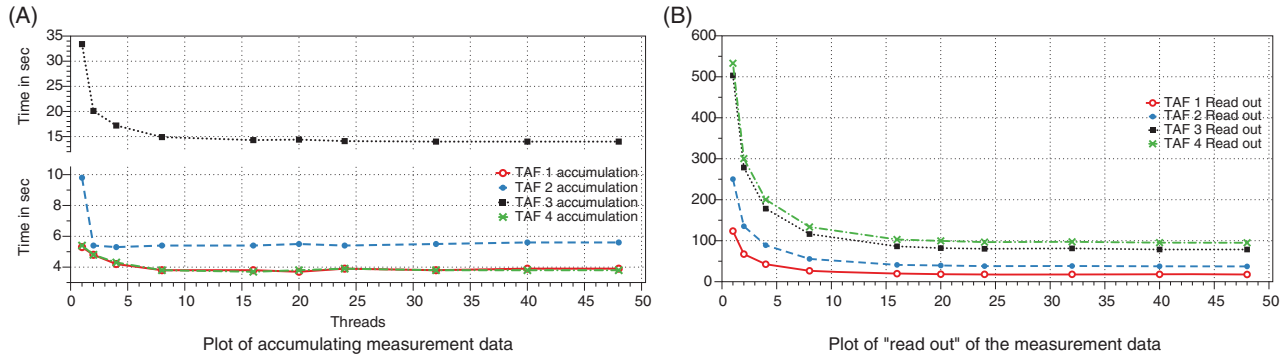

FIGURE 15 Plot application results on a Server with an Intel Xeon CPU E5–2620 v3 @ 2.40 GHz and 64 GB RAM

reached using 8 threads with a speedup of 2.2 and 28% efficiency, needing 14.9 s to finish. For all TAFs, parallelization helped to push execution time below the required boundary of 28 s.

For the monthly "read out" of the summed up measurement data, the application scenarios (see Figure 15B) differ more than for the accumulation process. TAF 1, low data usage, is the fastest and simplest one, gaining factor 7 for a single thread using 123.3 s compared to 48 threads consuming only 17.6 s. The optimum is reached using 16 threads with a speedup of 6.3 and a 39% efficiency, needing 19.6 s to finish.

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

The TAF 3, power-dependent tariff, and TAF 4, consumption-dependent tariff, are more complex and thus take around 9 min (503.3/533 s) for a single thread and around 1.5 min for 48 threads (78.6/95 s). 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 32% efficiency, needing 86 and 103 s to finish, respectively.

5.5 | Summary

This section covered the SMGW tariffs (TAF) according to the BSI and their TR[31] with an associated protection profile.[32] 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 were improved. Key generation improved by a factor of 15.5, needing only 1.1 instead of 17 s, and recryption was pushed down from 263 ms to only 34 ms, as seen in Table 2.

An overview of the arithmetic operations can be seen in Table 3. By parallelizing the FHE library, the time requirements can be pushed down for all TAFs below the required boundary of 28 s.[34] This proves that FHE 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. In the following section, the basis for future implementation for anomaly detection modules is established.

6 | CONTINUOUS MONITORING APPROACH

Through technologies such as Cloud Computing manufacturers can provide customers a modernized and flexible way to access their meters, for example, 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 poses many challenges for Legal Metrology. Through FHE,[31] the authors addressed the main threats, such as an insider attack and data manipulation in the cloud, in order to create an acceptable solution that provides adequate security.

In this section, a continuous anomaly detection approach is described for the prior detailed secure cloud architecture (see Section 3). The behavior and the tariff application pattern of the platform will be constantly monitored by a software module residing within the PaaS level as part of a monitoring service (see Figure 5). 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 anomalous, and its stability is affected.

6.1 | Anomaly detection approaches

For Cloud Computing architectures, two main approaches are worth further investigation: *consolge log-based* anomaly detection and *system metrics-based* anomaly detection.[35]

Console log-based anomaly detection is a set of rules to verify the system behavior. 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 (AD). However, creating 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 fewer privileges, and has a lesser impact on the monitored system. This approach is more suitable, especially for a distributed system when scalability, elasticity, flexibility, and migration of VMs 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.

6.2 | Distributed data collection

In a distributed environment, it is of utmost interest to keep the employed modules and services flexible as well as easily deployable to meet the demands of VMs that are spread across various time zones, networks, and teams. It is also 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 VMs, the highly distributable and modularized open-source monitoring software Performance Co-Pilot (PCP) (http://www.pcp.io/) is used. Over 300 metrics, for example, CPU utilization, network bandwidth, disk I/O, and memory usage, are 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.

Algorithm 2 Legal Metrology Application Lifecycle (Overview)

Input: Measurement data $m_1 \dots m_n$, Price data $p_1 \dots p_n$, index of elements $i = 1 \dots n$
Output: Measurement results $mr_1 \dots mr_n$

Server:
checks certificate and the origin of messages
if check is successful **then**
    notify logbook of received measurement data
    process data
    sum m_i with previous measurement data m_{i-1}
    if additions are successful **then**
        compute price p_i
        save price p_i
        notify logbook of successful price calculation
    end if
    save data mr_i
    notify logbook of successful addition
else
    purge data, → error message to logbook
end if

To analyze system anomalies, PCP provides collection archives to replay and analyze the system behavior step by step. These archives can be visualized with the *pmchart*-module. Furthermore, PCP supplies real-time monitoring to visualize the current state of the system. Vector (http://vectoross.io/) 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, that is, on the service level of each microservice, metrics can be collected and can 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 Algorithm 2), the normal and anomalous behavior of the cloud environment can be represented.

6.3 | Summary

In this section, the importance of an anomaly detection module for the secure Cloud Computing architecture was pointed out to prove the integrity of such a distributed system. Two approaches were investigated. The system metrics-based approach was selected due to its suitability to distributed system architecture. Moreover, this approach proves to be more flexible with a lesser impact on the monitored system. Furthermore, tools were described for the Legal Metrology application lifecycle (see Algorithm 2 for future distributed data collection.

7 | CONCLUSIONS

In this article, an introduction to the field of Legal Metrology was given, and its importance was demonstrated for a reliable working economy as it touches all important fields of daily life while establishing trust between the economic partners. Furthermore, the roles and requirements for a measuring instrument in Legal Metrology were discussed, and an overview of platform and software requirements was given. Using Cloud Computing in Legal Metrology, a new role of a Cloud Service Provider has been introduced. The responsibilities have to be determined in the near future as it depends on who is fulfilling this role in the former constellation (see Figure 1); thus, the responsibilities differ. The general approach of reference architecture was evaluated. This allows new technologies to be integrated into the Legal Metrology framework. Furthermore, new technology, such as Cloud Computing, needs to fulfill the essential requirements seen in Figure 3.

The presented secure cloud reference architecture fulfills all essential requirements in Legal Metrology, although further work is required to complete its implementation. This architectural structure was described in depth in a bottom-up approach in Section 3. The Cloud Computing framework is realized by the open-source software OpenStack and realizes the IaaS layer. The first separation of legally relevant software is realized by virtualization, dividing the network layer into different compartments. The PaaS layer is realized using the microservice pattern. The main advantages using this pattern lie in the possibilities of easily scaling the platform across heterogeneous machines around different domains while being resilient at the same time. Lastly, in the SaaS layer, the grounds for cloud-native applications were presented, aiming for the envisioned distributed measuring instrument.

In this envisioned future, the security and integrity of the measurements are at stake. Furthermore, a trustworthy system administrator role does not currently exist in Legal Metrology. Thus, 1 of the main research goals presented here is to prevent attacks that alter measurements, putting the integrity of the whole system at risk. Tackling this problem, FHE was evaluated, improved, and implemented to enable calculations on encrypted measurements, keeping them encrypted throughout their lifecycle. In Section 4.1, an overview of FHE is given, and the results for the application scenarios were presented in Section 5.4, proving that FHE is mature enough to be implemented nowadays (see Section 5.3).

Securing the measurement results by FHE and employing a special communication protocol to guarantee integrity (see Section 3.5) is not enough to secure the whole cloud architecture. An anomaly detection module (AD) was proposed, continuously scanning the cloud architecture for anomalous behavior. Depending on the severity of the anomaly, a classification can be performed automatically into 3 categories: green, the system is in normal state; yellow, an anomaly is detected but system stability not affected; red, anomaly recognized and instable system identified. In Section 6, the basis for future implementations for AD module were established. By proving the integrity of the overall system, the proposed approach will be sufficient to detect intentional manipulation of the measuring results, and by detecting anomalies, attacks can be prevented or mitigated for measuring instruments under Legal Metrology. Evaluating the presented secure Cloud Computing reference architecture will prove the maturity of this new technology that can be used within the Legal Metrology framework. This architecture will fulfill the highest risk class, the essential requirements, and a verification method for the market and user surveillance while taking into consideration contemporary threats, as presented in the overview in Figure 3.

8 | FUTURE WORK

The promising monitoring approach of the secure architecture must be further pursued to conclude the proof of maturity of Cloud Computing technology in Legal Metrology. By finding correlations between metrics, different kinds of anomalies in distributed environments shall be detected. Therefore, part of the future work will be evaluating metrics, such as CPU-load, CPU-runnables, network-sockstat, and memory-cached, in depth to create a test pipeline for automated testing and AD.

This metrics evaluation will be further improved by statistical methods. Moreover, categorizing incidents and detecting anomalies shall be automatized and implemented in a future version of the AD-module proposed in this article. This will provide sufficient information to understand the full behavior of the secure Cloud Computing reference architecture to prove the integrity and reliability of the overall system. A full risk analysis, as required by the European Directive 2014/32/EU, must be provided by the manufacturer. An adequate analysis and assessment of the risk shall be performed following the proposed method in Esche and Thiel.[9] This risk assessment method has been formalized and developed into a graphical representation in

Esche et al[1] that will be used in further work. Carrying out a risk assessment for contemporary threats will satisfy the remaining requirement presented in the overview in Figure 3.

Conflict of interest

The authors declare no potential conflict of interests.

ORCID

Alexander Oppermann http://orcid.org/0000-0002-0879-5325

REFERENCES

1. Esche M, Grasso TF, Thiel F. *Representation of attacker motivation in software risk assessment using attack probability trees*. *2017 Federated Conference on Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE; 2017:763-771.
2. Thiel F, Esche M, Peters D, Grottker U. *Cloud computing in legal metrology*. *17th International Congress of Metrology*. EDP Sciences; 2015:16001.
3. European Parliament and Council. Directive 2014/32/EU of the European Parliament and of the council. *Official Journal of the European Union*. 2014;L 96/149:149-250.
4. WG7. *WELMEC 7.2 2015 Software Guide. WELMEC European Cooperation in Legal Metrology*. Braunschweig, Germany: WELMEC Secretariat; 2015.
5. Métrologie Légale Organisation Internationale. *General Requirements for Software Controlled Measuring Instruments*. Paris, France: International Organization of Legal Metrology; 2008.
6. Leffler N, Thiel F. *Im Geschäftsverkehr das richtige Maß—Das neue Mess und Eichgesetz, Schlaglichter der Wirtschaftspolitik*. Berlin, Germany: Monatsbericht. Bundesministerium für Wirtschaft und Technologie (BMWi); 2013.
7. Kochsiek M, Odin A. Towards a global measurement system: contributions of international organizations. *OIML Bulletin*. 2001;42(2):14-19.
8. Peters D, Grottker U, Thiel F, Peter M, Seifert J-P. *Achieving software security for measuring instruments under legal control*. *Federated Conference on Computer Science and Information Systems*; 2014:123-130.
9. Esche M, Thiel F. *Software risk assessment for measuring instruments in legal metrology*. *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*; 2015:1113-1123.
10. 2004/22/EC Directive. Directive 2004/22/EC of the European Parliament and of the council. *Official Journal of the European Union*. 2004;L 135/1:1-80.
11. Henze M, Hummen R, Matzutt R, Catrein D, Wehrle K. Maintaining user control while storing and processing sensor data in the cloud. *Int J Grid High Perform Comput*. 2013;5(4):97-112.
12. Jäger, HA, Monitzer A, Rieken ROG, Ernst E. A novel set of measures against insider attacks – Sealed Cloud. In: *Lecture Notes in Informatics (LNI), Proceedings – Series of the Gesellschaft fur Informatik (GI)*; 2013:187.
13. Slawik M, Zickau S, Thatmann D, Repschläger J, Ermakova T, Küpper A, Zarnekow R. *Innovative Architektur fur sicheres Cloud Computing: Beispiel eines Cloud-Ecosystems im Gesundheitswesen*; 2012
14. Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M. *A System of Patterns: Pattern-oriented Software Architecture*. New York: Wiley; 1996.
15. Oppermann A, Seifert J-P, Thiel F. Secure cloud reference architectures for measuring instruments under legal control. *Closer*. 2016;1:289-294.
16. Joe Kilian. *A note on efficient zero-knowledge proofs and arguments*. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732. ACM, 1992.
17. Rosario Gennaro, Craig Gentry, Bryan Parno. *Non-interactive verifiable computing: Outsourcing computation to untrusted workers*. In *Annual Cryptology Conference*, pages 465–482. Springer, 2010.
18. Kai-Min Chung, Yael Kalai, Salil Vadhan. *Improved delegation of computation using fully homomorphic encryption*. In *Annual Cryptology Conference*, pages 483–501. Springer, 2010.
19. Gentry C. Fully homomorphic encryption using ideal lattices. Proceedings of the 41st Annual ACM Symposium on Theory of Computing. New York, NY: ACM; 2009.
20. Van Dijk M, Gentry C, Halevi S, Vaikuntanathan V. Fully homomorphic encryption over the integers. Advances in Cryptology–EUROCRYPT 2010. Springer; 2010:24–43.
21. Brenner M. Rechnen mit verschlüsselten Programmen und Daten [Ph.D. thesis]. Technische Informationsbibliothek und Universitätsbibliothek Hannover (TIB); 2012.
22. Micciancio D, Regev O. *Lattice-based cryptography*. *Post-quantum Cryptography*. Springer; 2009:147-191.
23. Pol J. Lattice-based Cryptography [master's thesis]. Eindhoven University of Technology; 2011.
24. Bergami F. Lattice-based Cryptography [master's thesis]. Universita di Padova; 2016.
25. Yi X, Paulet R, Bertino E. *Homomorphic Encryption and Applications*. Springer; 2014.
26. Smart NP, Vercauteren F. *Fully homomorphic encryption with relatively small key and ciphertext sizes*. *International Workshop on Public Key Cryptography*. Springer; 2010:420-443.
27. Perl H, Brenner M, Smith M. *Poster: an implementation of the fully homomorphic Smart-Vercauteren crypto-system*. *Proceedings of the 18th ACM Conference on Computer and Communications Security*. New York, NY: ACM; 2011:837-840.
28. Koren I. *Computer Arithmetic Algorithms*. Universities Press; 2002.
29. Lu M. *Modular structure of large multiplier*. *Arithmetic and Logic in Computer Systems*. 1st ed. Hoboken, NJ: John Wiley & Sons, Inc.; 2004:120-122.
30. BIPM. *Système International D'unités, The International System of Units (SI)*. 8th ed. Bureau International des Poides et Mesures (BIPM); 2006.
31. BSI. *Technische Richtlinie BSI TR-03109-1 Anforderungen an die Interoperabilität der Kommunikationseinheit eines intelligenten Messsystems*. Bonn, Germany: Bundesamt für Sicherheit in der Informationstechnik; 2013.
32. BSI. *Schutzprofil für die Kommunikationseinheit eines intelligenten Messsystems für Stoff- und Energiemengen (Smart Meter Gateway PP)* . Certification-ID: BSI-CC-PP-0073. Bonn, Germany: Bundesamt für Sicherheit in der Informationstechnik; 2014.
33. Grama A, Gupta A, Karypis G, Kumar V. *Introduction to Parallel Computing*. 2nd ed. Pearson Education; 2003.
34. Oppermann A, Yurchenko A, Esche M, Seifert J-P. *Secure cloud computing: multithreaded fully homomorphic encryption for legal metrology*. *International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*. Springer; 2017:35-54.
35. Barbhuiya S, Papazachos ZC, Kilpatrick P, Nikolopoulos DS. A lightweight tool for anomaly detection in cloud data centres. *Closer*. 2015;343-351.

AUTHORS' BIOGRAPHIES

Alexander Oppermann received his Diploma in Computer Science from the Technical University of Berlin (TU-Berlin), Germany, and is now a Ph.D. student at the TU-Berlin under the guidance of Prof. Dr. J.-P. Seifert. He is a fellow of the Helmholtz Research School on Security Technologies. Furthermore, he is a research assistant at the PTB, the national metrology institute of Germany. His current research interests focus on secure Cloud Computing in Legal Metrology, homomorphic encryption, and condition monitoring.

How to cite this article: Oppermann A, Toro FG, Thiel F, Seifert J-P. Secure Cloud Computing: Reference Architecture for Measuring Instrument under Legal Control. *Security and Privacy* 2018;e18. https://doi.org/10.1002/spy2.18