A Framework for Privacy in the Internet of Things

Vom Fachbereich Informatik der
Rheinland-Pfälzischen Technischen Universität Kaiserslautern-Landau
zur Verleihung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von

Christopher Heinz

Datum der wissenschaftlichen Aussprache: 19.07.2023

Dekan: Prof. Dr. Christoph Garth
Gutachter: Prof. Dr. Christoph Grimm
Prof. Dr. Jens Schmitt

DE-386

Abstract

"There is an increasing recognition of the value of personal data, and the extent of damage that can be caused if personal information is interfered with, released or stolen. [...] The General Data Protection Regulation (GDPR) applies to anyone, or any organisation that handles personal data relating to a European citizen, irrespective of where in the world the data is held or used." [Hei+20]

While most approaches, such as e.g. Amazon Web Services (AWS) rely on a strong security to ensure privacy, this thesis goes one step further. Not all scenarios require data to be available to the data processor in clear text. Apart from legal obligations, there is no means to protect the data any further from being misused or sold. The GDPR helps in setting the legal boundaries on data processing and redistribution. Yet, malicious intent may not be hindered by law at all and one might have to rely on other means to protect it.

Furthermore, the Internet of Things (IoT) is dealing with a wide variety of application domains at the same time. Interoperability between these different domains with their isolated ecosystems is the second obstacle. The framework proposed in this thesis offers full control over IoT data generated, even after it has been transmitted to a third-party. Yet, the conceptual idea is applicable even as a standalone application.

This framework utilizes Homomorphic Encryption (HE) to encrypt and secure sensitive data. The special attributes of HE schemes enable data processors to work with sensitive data without decrypting it. The HE scheme is based on the Ring Learning with Error (RLWE) Problem, which is considered quantum safe and can therefore be considered future-proof. With this scheme, only the data owner holds the secret key for decryption. This can be used by a single-party, to outsource complex but confidential computations to the cloud.

In the case where multiple parties need to collaborate, an additional secure Multi-party computation (MPC) Protocol is implemented. For example allowing to calculate e.g. the sum or average value over all inputs provided by multiple parties. With basic single- and multi party scenarios enabled, this framework already covers a majority of typical IoT use-cases.

In this Thesis, the framework is implemented and evaluated in different use-cases, covering both single-party and multi-party scenarios. It will be shown to be functional and applicable in IoT applications.

Danksagung

Diese Arbeit bringt ein weiteres meiner Kapitel an der Technischen Universität Kaiserslautern / Rheinland-Pfälzischen Technischen Universität Kaiserslautern-Landau zu einem Abschluss. Bereits während meines Bachelorstudiums haben mich die eingebetteten Systeme am meisten interessiert. Während meines Masterstudiums durfte ich dieses Interesse weiter vertiefen, sodass meine Faszination für das Internet der Dinge nur der Gipfel einer langen Reise war. Das Thema birgt zahlreiche Facetten - sowohl Möglichkeiten wie auch Risiken. Ich bin dankbar dafür, während und mit meiner Forschung etwas Licht in diesen Themekomplex bringen zu dürfen.

An dieser Stelle möchte ich mich bei all jenen bedanken, die mich während meiner Forschung und im Laufe meiner Promotion unterstützt haben. Im Besonderen gilt mein Dank Herrn Prof. Dr. Christoph Grimm, der mein Interesse an der Informatik und im Speziellen an den Cyber-Physikalischen Systemen durch seine Lehre im Verlauf meines Studiums weiter entfacht hat und mir im Anschluss daran ermöglichte eine Promotion anzustreben. Ich möchte ihm danken, dass er mir stets mit Ratschlägen und einem offenem Ohr zur Seite stand und für die Möglichkeiten, die ich dank ihm wahrnehmen konnte.

Für die Zweitbegutachtung möchte ich mich bei Prof. Dr.-Ing. Jens Schmitt bedanken. Für die Leitung der Promotionskommission bedanke ich mich bei Prof. Dr. Paul Lukowicz.

Weiter gilt mein Dank meinen Kollegen für das angenehme Umfeld, die interessanten Diskussionen, denen so manche Idee entsprang und durch die die Ergebnisse meiner Forschung auch bereits Anwendung fanden. Besonders hervorheben möchte ich dabei Herrn M.Sc. Hagen Heermann, der während seiner Masterarbeit, die sich ebenfalls mit dem Thema Internet der Dinge und homomorpher Verschlüsselung beschäftigte, eng mit mir zusammen gearbeitet hat. Außerdem danke ich Herrn Dennis Naumann, der ebenfalls durch seine Arbeit meine Forschung vorangebracht und potentielle Stolperstellen aufgedeckt hat.

Nicht zuletzt möchte ich meiner Familie für die langjährige Unterstützung danken. Insbesondere Frau Alena Lencioni, die es niemals leid wurde mich auch in Zeiten zu motivieren, an denen ich meine Fortschritte gerade an einem Tiefpunkt sah. Außerdem gilt ein besonderer Dank meinem Vater, der mir bereits in jungen Jahren die Wunder der Informationsverarbeitung aufzeigte

und so mein Interesse an der Informatik weckte. Weiter möchte ich mich bei Frau Zoë Eckfelder für die Motivation und die gestalterische Unterstützung bedanken.

Zuletzt danke ich meinen Freunden, die in all den Jahren stets für mich da waren und die meinen Spielereien rund um das Internet der Dinge nie leid wurden.

14. Mai 2024, Christopher Heinz

Contents

List of Figures

Acronyms

AWS	Amazon Web Services
BFV	Brakerski/Fan-Vercauteren
BGV	Brakerski-Gentry-Vaikuntanathan
BHC	Beyond honest-but-curious
BLE	Bluetooth Low-Energy
BSI	British Standards Institution
CKKS	Cheon-Kim-Kim-Song
CPS	Cyber-Physical Systems
DC	Data Controllers
DGHV	Van Dijk, Gentry, Halevi and Vaikuntanathan
DP	Data Processors
DPO	Data protection officer
EDPB	European Data Protection Board
EV	Electric Vehicle
FHE	Fully Homomorphic Encryption
GDPR	General Data Protection Regulation
HC	Honest-but-curious
HE	Homomorphic Encryption
IoT	Internet of Things
ITU	International Telecommunication Union
JSON	JavaScript Object Notation
LHE	Leveled Fully Homomorphic Encryption
LWE	Learning with Error
M2M	Machine-to-Machine
MPC	Multi-party computation
OGWAPI	Open Gateway API
OSI	Open Systems Interconnection
PHE	Partially Homomorphic Encryption

REST	Representational State Transfer
RFID	Radio-Frequency Identification
RLWE	Ring Learning with Error
RSA	Rivest, Shamir and Adleman
SVM	Support Vector Machine
SVP	Shortest Vector Problem
UDHR	Universal Declaration of Human Rights
VAS	Value Added Service
XMPP	Extensible Messaging and Presence Protocol

Chapter 1

Introduction

"There is an increasing recognition of the value of personal data, and the extent of damage that can be caused if personal information is interfered with, released or stolen. Protection of personal data from unauthorised access or modification has been recognised in privacy regulations around the world. The General Data Protection Regulation (GDPR) applies to anyone, or any organization that handles personal data relating to a European citizen, irrespective of where in the world the data is held or used. Personal data includes information about an individual, their movements or assets that can be associated with them directly, or by reference to other data files. Basic protection of privacy is achieved by using strong encryption to provide security of information, that is not sufficient. Personal data may be passed securely to a database, but who can access that database and what limits can be placed on their usage of those data?" [Hei+20]

While compliance to GDPR is mandatory, there are some limitations and pitfalls, which will be further elaborated: For example, most frameworks and platforms nowadays, such as e.g. Amazon Web Services (AWS) [Ama] or Microsoft Azure [Mic] rely on a strong security to comply to the GDPR. Yet, once data is given to any one of these third-party platforms, these platforms need to be fully trusted as there is no control over what the private data is being used for or whether it is misused or even sold illegally without consent. While this is in clear violation to the GDPR and the victims can take legal measures afterwards, in the best case they will be compensated. Yet, damage is done already. The approach of this thesis goes one step further, giving the user control over his personal data even after it is handed over to a potentially untrusted third party. Similar approaches like [Sha+17] or even [Gen09] show the potential of (fully) Homomorphic Encryption (HE) schemes to achieve that goal.

Privacy and protection of personal data is an issue not only in traditional information systems or social media platforms. Even more so, with the emerging technology of smart home and the Internet of Things (IoT), as IoT devices often operate in a personal, private environment. Devices such as Smartwatches, Fitness trackers and medical devices being some obvious ones. Smart home

devices such as motion and temperature sensors or even Smart Speakers such as Amazon Alexa extend the range of smart objects. There are many more connected devices the users may be less aware of, all of which are generating, processing and transmitting personal data to potentially untrusted or malicious cloud services. User privacy remains an important issue and one of the key acceptance criteria when developing and deploying IoT Systems. (see [LK16; Hei+20])

Figure 1.1.: *Wide range of applications in the Internet of Things including e.g. mobility, health care or energy management.*

Applications for the IoT are however not limited to the domain of smart home. Figure 1.1 tries to show how manifold the field of potential applications is. Apart from smart home, some other examples include e.g. mobility, health care, energy management or industry 4.0.

For each application, there exist numerous solutions, each one offering newer, faster or more secure encryption schemes and approaches to privacy. Ultimately, as achieving privacy is desired by most users, at least one approach should be applied and implemented. However, end-users are mostly not able to or do not like to bother with encryption schemes and algorithms. They do not want to follow lengthy setup processes either. Most prefer a “plug and play” solution. Otherwise, security and also privacy are most of the time simply skipped and considered “not worth the trouble”.

The framework developed in this thesis does integrate into the larger IoT platform ”VICINITY”, allowing the utilization in a wide variety of use-cases

and thus grants more privacy-awareness or even enables completely new business models. Heart of the solutions presented in this thesis is the *privacy microservice*, in the following only referred to as privacy service. This service allows developers to take full advantage of the HE without the need of becoming a cryptography expert. This is similar to people being able to drive cars without detailed knowledge over the internal combustion engine. While most software developers are familiar with principles and technologies well-known in traditional software systems and web-based development, the IoT is still new to a majority of them. Additionally, the IoT is not yet standardized in any way. There rather exists a wide variety of standards, protocols and technologies, each suitable for a particular purpose, forming its own, isolated ecosystem. It is unfeasible to understand and know each single one. A layer of abstraction is required, enabling developers to handle this large landscape of ecosystems. This abstraction layer can also be seen as a perfect place to implement privacy, making it build in "by design".

This thesis proposes a solution on how to realize such an abstraction layer and how to enable privacy "by design" in a seamless, straightforward manner.

1.1. Use-Cases

Roughly speaking, potential use-cases in the IoT can be split into two categories: single- and multi-party computation. As the names suggest, single-party computation means use-cases or scenarios where data is generated only by a single party and is sent to or processed by a Value Added Service (VAS). This is particularly useful, if computations can and should be outsourced to a more powerful node. Multi-party computation on the other hand means, that multiple parties generate data and intend to cooperate in order to produce a higher value. Both categories come with their own challenges.

1.1.1. Use-Case: Health Data Monitoring

Typical single-party computation scenarios involve data collection at a constraint node, transmitting this data to a more powerful sink, where it is stored or can be further processed. In the context of IoT, this sink node could typically be offered as a VAS, where the VAS provider would charge a fee for data storage and processing in the cloud. In return, the data owner/User benefits from the cloud storage potential where his data is kept safe and can be accessed by him at any time and any place. The first use-case, which this thesis tackles is the domain of eHealth. In this scenario, the user trusts the VAS with very private, medical data. Think of a fitness tracker/Smartwatch, which continuously measures the users heart rate during a fitness training session. In order to conveniently access this information afterwards e.g. from a laptop or a mobile device, this data is transmitted to and stored by a cloud VAS. Furthermore, in order to analyse the user's training progress, the user's average heart rate is calculated and evaluated over multiple training sessions.

The dataflow of this use-case is shown in figure 1.2.

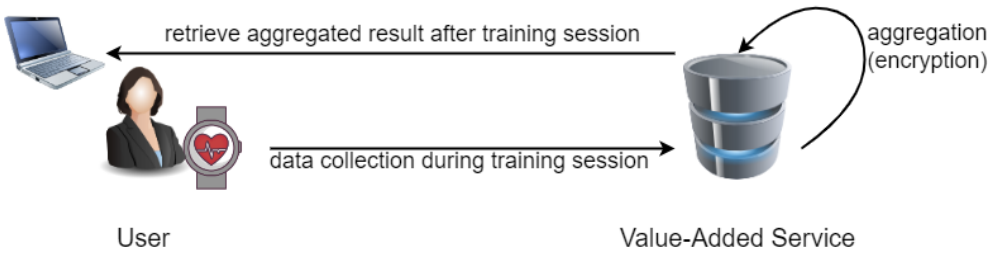

Figure 1.2.: *User generates heart rate data, which is stored on the VAS. Aggregated results can be retrieved afterwards.*

1.1.2. Use-case: Smart meter aggregation

"The smart electricity grid introduces new opportunities for fine-grained consumption monitoring. By integrating devices that provide electricity consumption data, utility providers can benefit from a balanced utilization of energy in an attempt to achieve a higher level of efficiency in provision for electricity. At the same time, consumers can benefit directly from the use of smart grid technologies by having access to cheaper sources of electricity with increased reliability and security. However, consumers worry that such intelligent monitoring devices, which can transmit power-usage information every few minutes, can make them vulnerable to privacy attacks." [DA16] Under these circumstances, a typical multi-party computation use-case is the collection of power usage information collected from so called "smart meters". These metering devices provide detailed information about a households electric energy consumption. Collecting this information of a whole district or even an entire city allows for an educated forecast of power consumption and profiling. This can help to provide to a more stable grid and hence prevent power shortages. Yet, this information can also cause serious risks to the user's privacy. Frequent collection of this information may reveal information about the consumer's daily activities and must hence be classified as private data.

Figure 1.3 shows, how energy consumption data of such smart metering devices is gathered and transmitted to the grid operator's value-added service. As the operator is only interested in the overall energy consumption and not so much in die individual consumers, it calculates the sum over all measurements first.

For this use-case, multiple distributed and independent parties need to cooperate. In order to keep all user's private data safe, a measure needs to be found to protect against semi-honest, also called honest-but-curious adversaries, who follow the execution protocol, but try to find out as much information about the other users as possible. Even worse, there could also be adversaries, that do not even follow the execution protocol at all but instead try to e.g. eavesdrop upon the communication of others or even modify protocol messages to try and abort or sabotage the protocol execution. This risk is multiplied with each unknown and untrusted party involved. While the energy consumption use-case outlined above is one example, the underlying problem of sharing

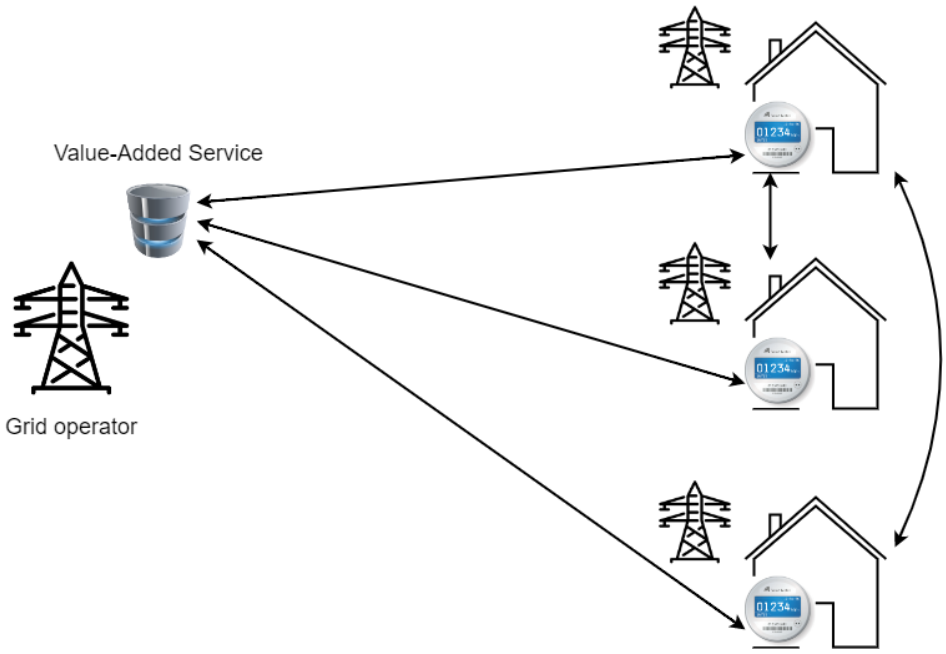

Figure 1.3.: *Multiple households need to cooperate with each other and the Grid operator's Value-Added Service*

and working together with multiple untrusted parties is manifold. Hence, a solution to the above use-case can be used in many other applications as well.

1.2. Contributions

In this thesis, a framework for privacy in the IoT is developed and its feasibility for different applications and use-cases explored and evaluated.

The underlying architecture for this framework was developed in the course of the Horizon 2020 project VICINITY, funded by the EU[1]. The VICINITY project will be explained in more details in 2.2. The results of this thesis can be used as an extension for the VICINITY infrastructure, beyond its original scope. However, this framework is not limited to just that.

Nonetheless, VICINITY plays a major role for this thesis. In [HG16], the first concepts and ideas of the project were outlined. VICINITY aims to offer "Interoperability as a service". One key problem in the IoT, apart from already outlined privacy concerns, it the lack of interoperability. Meaning numerous solutions emerge from different vendors, each providing isolated "islands" or "silos" without any means to interact with external systems. This drastically limits the potential, that IoT can offer. Hence, VICINITY aims to offer a platform to integrate arbitrary systems, creating a much larger, vendor-agnostic and cross-domain ecosystem.

[1]The VICINITY Project: https://vicinity2020.eu/

Naturally, to develop a platform integrating various different solutions by different vendors, research on currently available platforms and standards needs to be conducted first. In [Myn+17], dominant and important standards and standardization bodies were identified and explored. VICINITY integrates with any arbitrary platform and standard with the use of Adapters (see 2.2.3). Some of the most widely used open source platforms are analysed and evaluated as well. For most of them, a matching adapter was implemented in the later stages of the project.

In [Gua+17] VICINITYs concept of "Interoperability as a Service" is explained in more detail. In order to adapt to the existing standardization landscape, VICINITY provides an ontology, which is used to unify communication between different proprietary solutions. Instead of changing existing solutions, all that is necessary to connect to the VICINITY peer-to-peer network is to provide an adapter, which basically translates proprietary communication protocols to the VICINITY ontology and vice-versa.

During the development and lab-testing phases of VICINITY, major bugs and pitfalls were already identified in small lab setups. However, the VICINITY solution is also demonstrated in four large-scale deployment installations across Europe (see 2.2.6). In order to evaluate potential shortcomings in the VICINITY architecture on larger deployments, yet before deploying them in the field, a simulation of use-cases using network simulator Omnet++ is presented in [Kol+18].

In [Kol+19], the aforementioned network simulator approach was further developed and utilized to evaluate the feasibility of HE in the context of a use-case similar to what has been proposed in 1.1.2. On this simulated use-case, partially homomorphic encryption (Pallier encryption, see 3.2.2) and fully homomorphic encryption (Brakerski-Gentry-Vaikuntanathan (BGV) encryption, see 3.4.4) were evaluated in terms of their runtime behaviour and compared to a pure plain text approach.

While in [Kol+19] the use of HE in general was evaluated, [Köl+19] was also integrating into the VICINITY peer-to-peer network. The same network simulation tool was used to analyse runtime behaviour of the use-case described in 1.1.2. Yet, the test setup also included the VICINITY peer-to-peer network as well as a first version of the *Privacy Service* component as will be further discussed in this thesis.

To this point, security and hence privacy was only considered on the IP-layer. In [Lip+20] also a look into wireless network security on a physical layer was taken. Yet, this is out of scope of this thesis.

To summarize all experience and knowledge earned during the development of the VICINITY project, [Uwi+20] and [Hei+20] were published. [Uwi+20] was focusing on the standardization bodies and IoT platforms available, as introduced in [Myn+17].

The results of the research done towards integrating HE into an IoT platform such as VICINITY are evaluated in [Hei+20]. Also, the legal framework on privacy, the GDPR is described. The results of [Hei+20] are now further examined and described in this thesis.

1.3. Related Work

Secure Sharing of Partially Homomorphic Encrypted IoT Data

In the work of [Sha+17], the Framework of *pilatus* is described, which can be used to share private data with and via a cloud service. One key aspect of pilatus is, that data is only shared in encrypted form. More precisely, it is encrypted using Elliptic Curve ElGamal encryption, a Partially Homomorphic Encryption (PHE) scheme. Homomorphic encryption will be discussed in further details later in this thesis in Chapter 3. Shortly, partially homomorphic encryption allows some arithmetic operations to be carried out on ciphertexts, without disclosing any information on the original plaintext data. E.g. using Elliptic Curve ElGamal encryption, addition can be done on two ciphertexts with the result being equal to the two plaintexts being added first and encrypted afterwards. In Pilatus, this is used so Alice can store encrypted data at a cloud service. Later, Alice requests the cloud service to process (e.g. aggregate) some of her stored data, with the result still encrypted, being returned to Alice for decryption. Another feature of Pilatus is the mechanism of re-encryption and re-keying. Data, which Alice stored at the cloud service can be re-encrypted with Bob's public key, such that Bob can then decrypt this data with his own private key. This makes it, such that a particular set of Alice's data is shared with Bob and Bob alone. The general idea is shown in figure 1.4

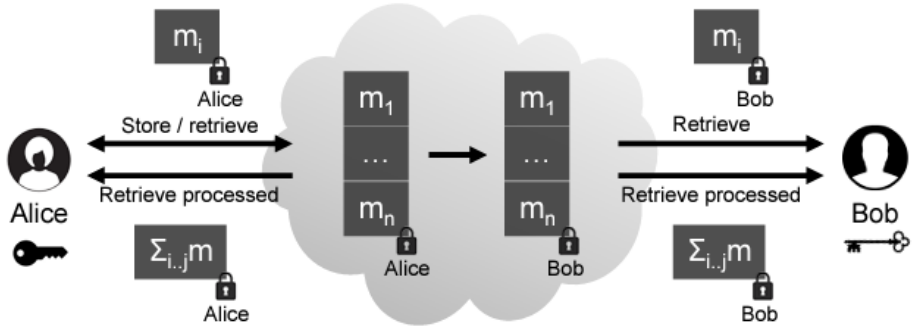

Figure 1.4.: *General dataflow in single-party computation. Alice sends private information to Bob. As Alice does not fully trust Bob, encrypted data is exchanged. [Sha+17]*

Using a cloud service just for storage und retrieving processed data back is one potential use-case as also described in 1.1.1. Also the ability to share ones data with another user like e.g. a doctor still falls into the category of single-party computation, as only one party acts as a data provider, while there can be more than one data consumer.

Secure and Scalable Aggregation in the Smart Grid Resilient against Malicious Entities

On top of that, there are also scenarios, where there is more than one data provider and all data providers will need to cooperate in order to reach a higher goal. [DA16] addresses one typical example for such a multi-party computation scenario, as also described in 1.1.2. The authors of [DA16] address the issue of collection and aggregation of smart metering device data. Given a set of smart metering devices $S = \{S_1, S_2, ..., S_n\}$, an aggregator A, e.g. the energy provider is interested in the aggregation of all measurements m_i provided by the smart metering devices S_i. In order for any multi-party computation protocol to protect the given inputs, at least 2 participants need to be honest. If all $n - 1$ participants were to misbehave, they could potentially exchange all their given inputs and thus subtract their known inputs from the results, revealing the last, honest participants private input. Hence [DA16] introduces a set of *trustworthy* neighbors, who also participate in the protocol. This is a key aspect of the two protocols presented by the authors. Figure 1.5 shows the basic idea.

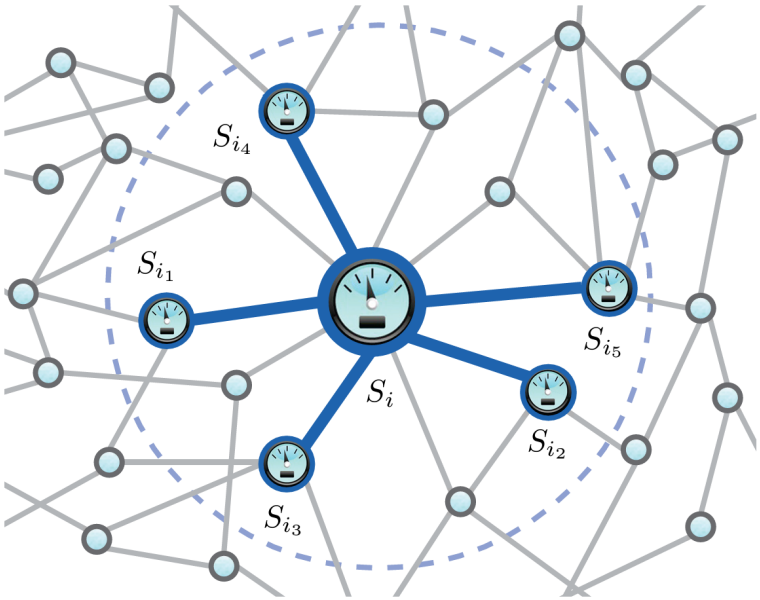

Figure 1.5.: *Among all participating smart metering devices, S_i only trusts a subset of $T = S_{i1}, S_{i2}, S_{i3}, S_{i4}, S_{i5}$. Any S_i has its own trusted set T_i. [DA16]*

In [DA16], two protocols are presented, addressing two different adversary models: A semi-honest adversary, also referred to as honest-but-curious adversary, who follow the execution of the protocol, but at the same time try to learn as much as possible about the smart meters' private data. Against such honest-but-curious (HC) adversaries, the HC Protocol is described. In practical applications one cannot assume just HC adversaries. Instead, the used

protocol needs to be extended in order to be resistant to adversaries, that also deviate from the protocols specifications. In [DA16], this extended protocol is called the BHC Protocol, as it also addresses beyond honest-but-curious (BHC) adversary behaviour. For the BHC Protocol, the authors also again rely on properties offered by using a homomorphic encryption scheme.

1.4. Outline

In Chapter 2 the fundamentals for this thesis are presented. As the kickoff for this thesis was the VICINITY IoT platform, first an overview on the ideas and challenges behind the IoT is given. Followed by an introduction to the VICINITY project and platform itself. As this thesis gives a framework for privacy in the IoT, the legal boundaries for privacy, more precisely the GDPR is reviewed.

Chapter 3 then gives a more in-depth dive into the topic of homomorphic encryption and secure Multi-party computation (MPC), two techniques used in order to build the framework of this thesis.

In Chapter 4, the conceptual idea behind this thesis is described. Each step towards the framework presented in this thesis is explained and evaluated.

Chapter 5 shows some analysis and runtime evaluations of the framework. Also possibilities to extend the framework as a developer are shown and explained.

Finally, in chapter 6 the thesis is summarized and ideas for future work are sketched and discussed.

Chapter 2

Background

Contents

In 1948 at the UN General Assembly in Paris, the United Nations proclaimed the Universal Declaration of Human Rights (UDHR) [Uni48]. It was a milestone document in the history of human rights. Drafted by representatives with different legal and cultural backgrounds from all over the world it continues to be seen as the standard to be achieved for all peoples and all nations. It set out fundamental human rights to be universally protected. However at that time, they would not predict how computers would become ubiquitous, with massive deployment of sensors and the creation of big data resources.[Hei+20]

Article 12 of the UDHR states: "No one shall be subjected to arbitrary interference with his *privacy*, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks." [Uni48] This objective still holds true today. The UDHR was further developed in Europe as the *Charter of fundamental rights of the European Union (2012/C 326/02)* [Eur12], which includes Article 7: Respect for private and family life: "Everyone has the

right to respect for his or her private and family life, home and communications". [Eur12]

Internet of Things (IoT) objects often operate in a personal, private environment. Devices such as Smartwatches, Fitness trackers and medical devices being some obvious ones. Smart home devices such as motion and temperature sensors or even Smart Speakers such as Amazon Alexa extend the range of smart objects. There are many more connected devices the users may be less aware of, all of which are generating, processing and transmitting personal data to potentially untrusted or malicious cloud services. User privacy remains an important issue and one of the key acceptance criteria when developing and deploying IoT Systems. (see [LK16; Hei+20]).

It is essential to adopt the principle of *privacy by design* and *privacy by default*. Privacy by design as a concept was introduced in 1995 and led to the identification of seven fundamental principles [Cav+09], which have been developed into evolving legislation, with the latest and most comprehensive legislation being the General Data Protection Regulation (GDPR) [Eur16; Hei+20].

Regulation requires privacy to be protected by restricting the usage of personal data. So first of, it needs to be defined what data sets will be considered to be personal data. Clarification over specific questions relating to privacy were considered by the Article 29 Data Privacy Working Party which was replaced by the European Data Protection Board (EDPB) [Eurb] on the day that the GDPR became law. However, the Article 29 WP archives [Eura] make interesting reading. [Hei+20]

Essentially, if there is a data set that contains personal information about an individual which does not in any way identify the individual that it relates to, then it ceases to be personal data. However, if it is possible to link this data set to the individual by applying another source of knowledge, then the data set must be considered to be personal data. For example, information about a car making a journey from a private residence to an office building at a certain time every morning. This information alone does not contain personal information. However, combining this with other information sources like for example knowing who lives at this residence and works at said building, then this daily journey would also be personal data. [Hei+20]

The easiest way to fully protect personal information is to not handle personal information at all. If personal information has to be processed, then the risk of unauthorized access to the personal data can be minimized by deleting it as soon as it is no longer needed.

It is important to understand the difference between privacy and security. The legal requirement is to protect privacy. One of the threats to privacy are cyber-attacks. While security is a critical requirement to protect privacy, it is not sufficient on its own. Privacy could be breached after encrypted data is passed securely to a database, but then the authenticated data processor proceeds to process data in ways that have not been agreed with the user. *Privacy by design* includes the need to protect the personal data from attack, even when the security systems have been breached. [Hei+20]

Some practical steps to protect personal data and satisfy the needs of GDPR rules when developing a new system or service are explained in Section 2.3.3.

2.1. Internet of Things (IoT)

The IoT is often seen as an evolution of the *traditional* Internet. This is true in many regards, as it utilizes existing IT infrastructure and is often build on top of or as an extension to existing IT systems. While these traditional systems connect networks of servers and offer their services to clients such as desktop PCs or Smartphones, IoT devices are much smaller. A single sensor node or even a connected light bulb can be seen as an IoT device. These small devices - mostly drastically limited in their computational power - are connected to one another, to new servers or existing IT infrastructure. Additionally, while in traditional IT systems, the clients can and often are operated by a human being (e.g. a person browsing the web on their PC or Smartphone), in the IoT, devices interact with one another. This direct communication between devices is oftentimes referred to as Machine-to-Machine (M2M) communication.

One final distinction between IoT and traditional IT systems is, that IoT devices interact with the real, physical world, either by measuring their surroundings e.g. temperature or light levels, or by actively altering their physical environment e.g. by raising the setpoint temperature of a thermostat or switching a light. With these devices and systems acting as a bridge or gateway between the IT-, also known as cyber-world on one hand and the real, physical world on the other, they are also referred to as Cyber-Physical Systems (CPS). Both terms IoT and CPS are used synonymously and depending on the application domain or geographical region, one is preferred over the other. In the remainder of this thesis, these systems will be referred to as IoT systems.

The IoT is used to describe the networking of smart objects via the internet to allow them to communicate with each other (M2M). This new type of communication enables new services, oftentimes without the need of human interaction. The term "Internet of Things" has been coined by Kevin Ashton [Ash09] in 1999, originally in the context of Radio-Frequency Identification (RFID) tags. Yet nowadays, it is defined by the International Telecommunication Union (ITU) [Uni12] as: "*A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving inter-operable information and communication technologies.*".

The IoT grew rapidly over the past decade and has not reached its peak yet. With a wide variety of applications, the number of connected devices is growing rapidly. It is predicted, that by 2025 roughly 75.4 billion devices will be interconnected [For16].

Compared to the Open Systems Interconnection (OSI) model, we add one layer *below* the *physical layer*, which represents the interaction with the physical environment of the devices themselves. In addition to the OSI layered model of the Internet, another reference architecture is dominant when talk-

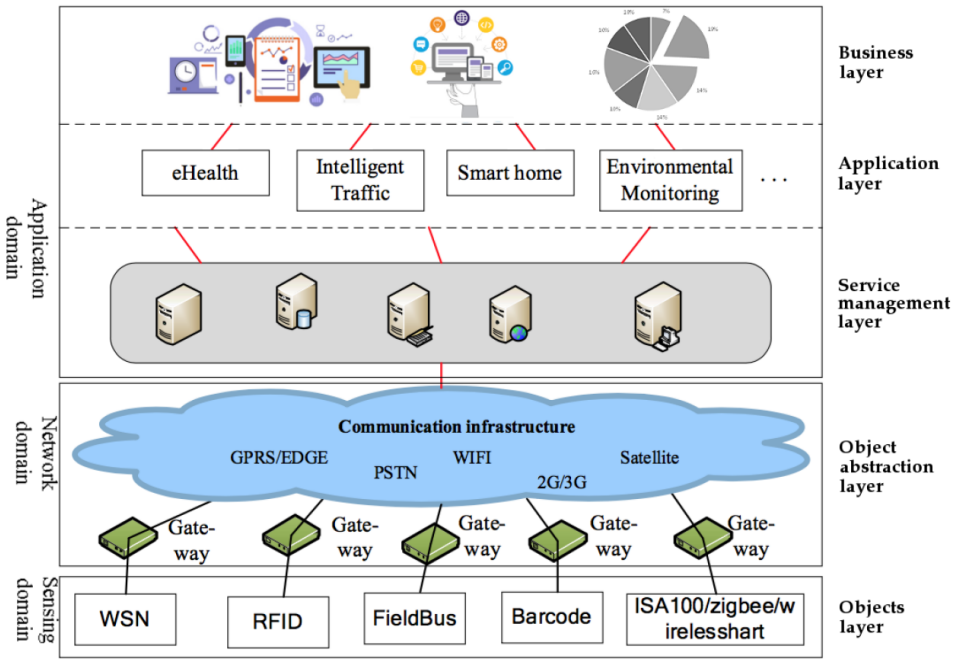

Figure 2.1.: *IoT Reference Architecture [Myn+17; CJL12; Zhu+10]*

ing about IoT Systems and devices. Figure 2.1 depicts this architecture. As pointed out above, this interaction could either be passive (sensors) or active (actuators). In Figure 2.1, this is all represented by the *Objects Layer*. Most of the time, the IoT devices at this layer are low-powered devices. Implementing the full TCP/IP communication stack is simply not feasible. Instead, this task it outsourced to so called *Gateway* Devices. These Gateways implement the full stack to connect to the Internet on one hand and communicate with the low-powered IoT devices via special, sometimes proprietary, low-energy communication channels such as ZigBee, Bluetooth Low-Energy (BLE) or RFID-based technology. At this stage, the IoT devices are to the outside world represented by their respective gateway. Their functionality is abstracted from their actual implementation or means to communicate with them. An outside object would simply interact with the gateway. In return this will relay the communication to the IoT object. This level of abstraction is referred to as *Object abstraction layer* in Figure 2.1. As we will see later, this layer will be the one primarily used to implement measures for adaptation to other, foreign systems and infrastructures, which in return will make it necessary to take precautions in terms of privacy and security. All will be implemented mainly on this layer of abstraction. The three layers of the Application domain above, namely *Service management layer*, *Application layer* and *Business layer* will need to adapt to the changes we introduce to the *Object abstraction layer* in a way, that they enable full control of the data owner, e.g. the owner of the IoT objects, over *who* is accessing his data and

for *what* purpose. Yet, with the base infrastructure selected for this purpose, only minor adaptations are necessary.

2.2. The VICINITY project

Many factors come into play, when designing an IoT platform. Aiming for a wide spread use of it, interoperability with other systems, isolated island solutions or *silos* as they are often called, is key. One potential solution could be to design a meta-standard, which aims to combine all or most of the common other standards or proprietary solutions currently out on the market. However, this endeavour is most certainly going to fail as humorously depicted in figure 2.2.

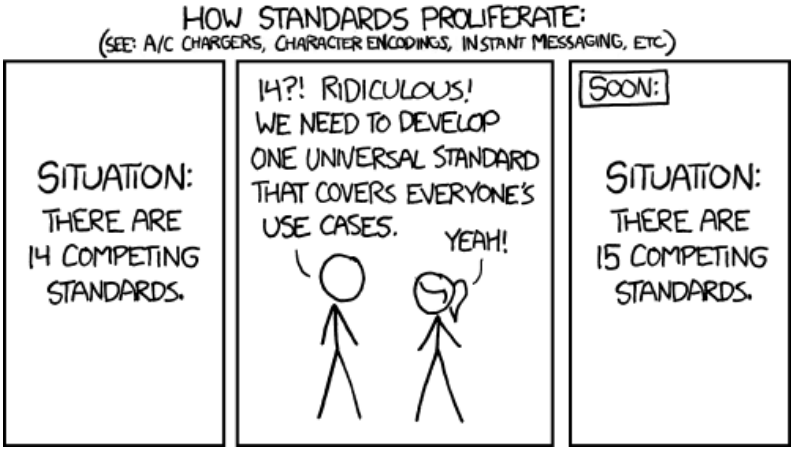

Figure 2.2.: *How Standards proliferate[XKC16]*

As part of the Horizon 2020 program, funded by the EU, the VICINITY IoT platform was implemented to tackle the problem of semantic interoperability. The VICINTIY platform aims to connect isolated IoT infrastructures into one global ecosystem called *virtual neighborhood*. Users can select other systems to which their smart objects should be connected. The platform automatically supports interoperability from technical up to semantic level. The technical interoperability is mainly delivered by the vendor specific and often proprietary gateways and adapters used to connect the smart objects to the network. Semantic interoperability of smart objects coming from different operators and using different standards however, is enabled with the semantic model, which is the core aspect of the VICINITY platform [Cas17].

A higher level overview of the VICINITY architecture is shown in figure 2.3. It is based on a decentralized, bottom-up and cross-domain approach that resembles a social network. Users can share their smart objects similiar to sharing pictures on social media. However, with VICINITY, users can configure their setups, integrate standards according to the services they want to use and fully control their desired level of privacy in a P2P (peer-to-peer) network. Furthermore, by combining services from different domains together

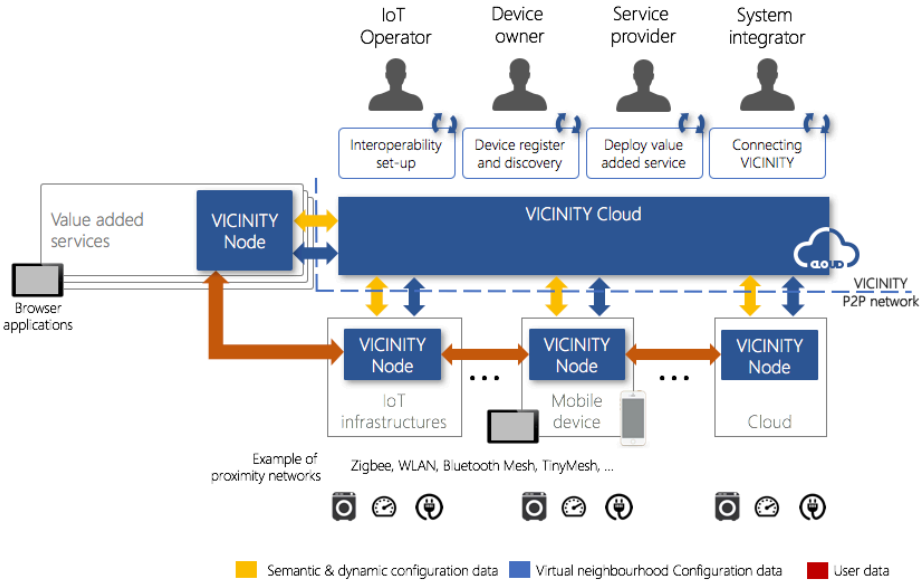

Figure 2.3.: *High-level VICINITY architecture [Ora17]*

with a privacy-respectful user-defined share of information, the VICINITY platform enables synergies among services from those domains and enables a new market of domain-crossing services.

The VICINITY platform consist of multiple individual, interconnected components, which is shown in Figure 2.4.

2.2.1. Open Gateway API (OGWAPI)

In order to connect to VICINITYs peer-to-peer network, the Open Gateway API (OGWAPI) [VIC20b] is used. Communication between connected VICINTIY nodes is done using the Extensible Messaging and Presence Protocol (XMPP). However, as this is not as wide spread and hence developers are not very customed to it, the OGWAPI abstracts from the XMPP protocol and instead offers a Representational State Transfer (REST) API for developers and other VICINITY components to interact with. Firstly, the OGWAPI is responsible for authenticating and authorizing communication partners and granting access into the peer-to-peer network itself. Once authorized, the OGWAPI component at a sending VICINITY node takes care of wrapping incoming REST requests into XMPP messages and transmitting them to the respective, receiving VICINITY node. On the receiving end, the OGWAPI unwraps these messages and delivers them via REST request forward into the attached IoT infrastructure. Next in line is another VICINITY component: The VICINITY Agent.

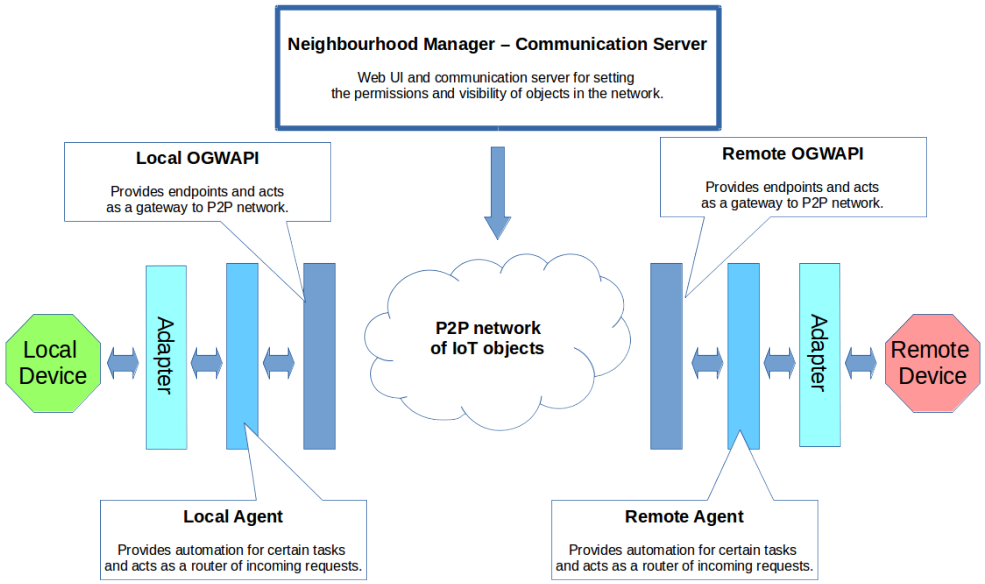

Figure 2.4.: *VICINITY architecture [VIC20b]*

2.2.2. VICINITY Agent

The VICINITY Agent is yet another component mostly there for convenience. While infrastructure operators could in theory use the OGWAPI directly in order to connect to and communicate via the VICINITY peer-to-peer network, they are more likely to utilize the VICINITY agent more conveniently. The Agent needs to be configured and given valid access credentials, as it takes care of the login process first and foremost. Furthermore, the Agent gathers information on the enabled smart objects from any of the attached Adapters (see 2.2.3). This information, called the Thing Description (see 2.2.4) is furthermore sent to the VICINITY Neighbourhood Manager (see 2.2.5), where these smart objects are registered and listed. Incoming messages, coming in via the peer-to-peer network and through the OGWAPI are distributed by the VICINITY Agent and forwarded to the destination object or Value Added Service (VAS).

2.2.3. VICINITY Adapter

The VICINITY Adapters are the translation units. They adapt the (proprietary) communication standards and protocols of the smart objects and map them onto the VICINITY ontology, constructing a Thing Description (see 2.2.4), which is used and understood by the Neighbourhood Manager and also other smart objects or VAS. That is any smart object has its properties, actions and events listed in its Thing Description. This Description can

be consumed by other VAS in order to understand how to interact with said object. For example, if a VAS is supposed to signal some external events by changing the light in a room, it would look for objects of the VICINITY ontology type *Lightbulb* and change its attached property affecting the ontology type *LightColor*. Further details about the VICINITY ontology can be found at [VIC20a]. Adapters abstract the higher level behaviour, described by the VICINITY ontology, from the lower level hardware details of the underlying IoT object. This means something like "make this lightbulb red" will be translated into whatever the IoT object representing the requested lightbulb understands. This could be sending IP packets of a particular format or sending out messages over wireless technologies like Bluetooth or Zigbee. As a user or VAS provider, there is no need to worry or even care about these details. All that is necessary is formulating the semantic meaning of what action should be performed e.g. "make this lightbulb red".

2.2.4. Thing Description

The Thing Description is a set of attributes describing an IoT object as a JavaScript Object Notation (JSON) document. Attributes include meta information such as a unique identification, a human readable name, a type, location information, properties, events and actions that this object can perform. The objects type and its object id are mandatory. Its properties, events and actions are all optional. An objects information in a thing description is based on the VICINITY ontology (see [VIC20a]) and hence enables semantic interoperability between different IoT objects and VAS. Properties can either allow read-only or read and write access. Typical examples for properties are e.g. the current brightness level of a lightbulb or heart rate readings of a sensor device. How this information can be accessed is described in the Thing Description as a set of REST API endpoints. Actions are longer lasting processes, which can be executed by an IoT object, similar to an asynchronous function call. A typical example would be an Electric Vehicle (EV) charger, which can be called to start or stop the charging process on the connected Vehicle. The current process of a running action can also be monitored. Once, e.g. the charging process is finished, an event triggers. VAS can subscribe to events in order to trigger some processing or action on their own. Once an event is triggered, all subscribers are notified via the VICINITY neighbourhood. A typical example of a Thing Description is shown in appendix A.1.

2.2.5. VICINITY Neighbourhood Manager

Finally, the VICINITY Neighbourhood Manager[1] is where all the above pieces come together. First off, The Neighbourhood Manager lets users control access of their own VICINITY nodes to the peer-to-peer network. Once a node connects to the network (via the VICINITY OGWAPI, see 2.2.1), the VICINITY Agent (see 2.2.2) will upload its known Thing Description to the Neighbourhood Manager. Devices described in this Thing Description are afterwards

[1]The Neighbourhood Manager is publicly available at https://vicinity.bavenir.eu/

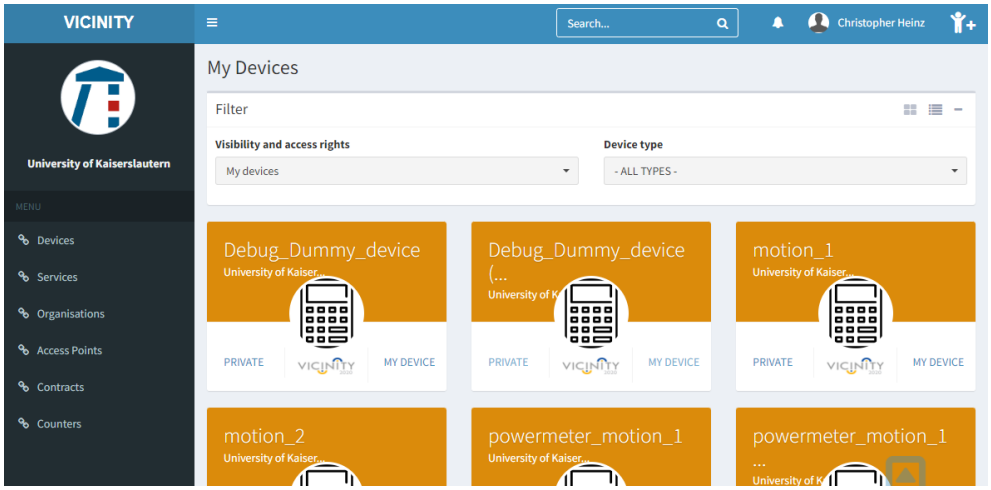

Figure 2.5.: *VICINITY Neighbourhood Manager is used to manage and control access to shared IoT Devices*

listed in the Neighbourhood Manager as shown in Figure 2.5. VICINITY offers the structure of organizations, which could either be a company offering access to their devices, value-added services or even a single household requesting and using one of these services. By default, access to a users own IoT devices is private, meaning only the user himself is allowed to access and see these devices. Further access levels can be configured per device and allow full public access to a device. That could for example be of interest for an outdoor temperature sensor. It could also be public visibility, yet still requires an explicit request and confirmation before read access is granted. The latter is the most common setting for devices, which should be part of a bigger IoT ecosystem. This setting allows the user to have full control over *who* can access *which* of his IoT devices. Access to these devices is then finally granted in the form of *contracts*. Such a contract needs to be agreed upon by both sides, that is by the device owner on one hand and the VAS operator on the other hand. The device owner can further control, which of his devices are part of said contract and whether the VAS is granted read-only or read and write access. Any contract can be revoked by either side at any point in time, leaving the shared devices private and inaccessible by third parties once again.

2.2.6. VICINITY pilot demonstrators

VICINITY's approach was demonstrated on different real-world scenarios and addressing cross-domain applications including Building Automation, Ambient Assisted Living, Smart Energy, Smart Parking and eHealth. In addition, VICINITY was tested in Lab installations before being deployed in the field. An overview on VICINITYs pilot locations is shown in Figure 2.6. VICINITY's potential to create new, cross-domain services is demonstrated by value added services, such as micro-trading of demand-side management capabilities, AI-driven optimization of smart urban districts and business intelligence

Figure 2.6.: *VICINITY pilot and lab installations*

over IoT. [Köl+19]

During the projects lifetime, some shortcomings of the initial architecture were identified. For example, the user is able to select and choose *who* is granted access to his IoT objects by the help of a virtual contract established on the VICINITY platform (see 2.2.5) Yet, there is only a read-only or full write access option available for each contract, drastically limiting choices on *what* can be done with this data. In the worst case, once a malicious party is given access to an object, all data can be stolen and even be distributed to unauthorized third-parties without the users consent. This problem gave rise to the research concluded in this thesis. The framework presented is intended to fully integrate into the VICINITY architecture and to enhance its functionality and overcome the above mentioned shortcomings but is not limited to this platform.

2.3. General Data Protection Regulation (GDPR)

The following section is adapted from my own publication in [Hei+20]. In this section, we step back from the technical view and review laws and regulations. There is a discrepancy between what is technically possible and what is legally allowed. As already mentioned, the current legal framework provided by the EU is *The General Data Protection Regulation (GDPR)*. The following text focuses on the changes in data protection law compared to former laws and also gives some insight into what should be taken into account when creating a new service that uses personal data.

2.3.1. Introduction

Regulations concerning data protection in the European Union have evolved slowly. The member states tended to have national laws and even different laws in their local regions. The regulations immediately prior to GDPR were the European Commission's directive DIR95 from 1995, which was complemented by the Directive on Privacy and Electronic Communications (EC Directive 2002/58/EC) [TT17]. Since this directive was made the ability and extent of automated data processing far exceeded what was intended to be regulated. Therefore, in 2012, the European Union decided on the development of the new *General Data Protection Regulation* in order to deal with the problems that have evolved due to too weak and varying regulations.

The very same regulation has to be applied in different scales: for a single individual using a personal computer at home, as well as for a global and multi-billion company like Facebook. Consequently, it is a very comprehensive legal document with a huge impact on EU-citizens and companies. It also applies to everybody outside of the EU, who handles personal data of citizens of the European Union. Furthermore, the fines for data breaches have increased significantly. This has motivated awareness and the attention of many people and legal entities have taken steps to be compliant to the GDPR.

2.3.2. Evolution of privacy laws to GDPR

General provisions, principles and definitions

The GDPR introduces world-wide obligations: the law applies wherever personal data relating to an EU citizen is processed. To help understand the legal requirements new definitions were introduced.

One of them is pseudonymization, which means that processed data is handled (processed and physically stored) in a way, that it cannot be related to the person it belongs to, without additional information. This is different from anonymization where the identity of the person that this relates to is removed in such a way that the link cannot be reinstated.

While pseudonymization reduces the ease with which personal data can be accessed, it does not prevent the decoding of the data. A single breach of security would be unlikely to result in any loss of personal data, without access to the pseudonymization tables. Pseudonymized data has still to be considered to be personal data, so it should be carefully protected.

New definitions regarding the type of data have also been introduced.

- *Personal Data* is any information that can be associated with and identify an individual. This clearly includes any data where the individual is named explicitly, but it also includes any data which can be associated with an individual by correlation with another database. For example: information about a privately-owned car is deemed to be Personal Data. Similarly, information from sensors installed in a private home would be deemed to relate to the home-owner.

- *Sensitive personal data*: Much of the data associated with an individual would not be considered to be sensitive personal data. However, information about health, membership of organizations, personal relationships, sexual orientation, religion, criminal activity and convictions etc. would be considered to be sensitive because much more damage would result from a data breach. For the purpose of normal operation they are treated as personal data.

The main practical differences are the need for greater limitation of processing sensitive personal data and the way a data breach has to be handled and reported. Sensitive personal data also includes: "Genetic data is defined as any data relating to an individual's characteristics that are inherited or acquired during early prenatal development. Biometric data denotes any data relating to an individuals' physical, physiological or behavioural characteristics and allowing a unique identification. Data concerning health means any information related to an individuals' physical or mental health or the provision of health services to the individual." [Bri17]

Medical data is already strictly confidential and must be kept secured. However, data like an individual's Body Mass Index (BMI) together with their name or date of birth becomes classified data. Even if the individual's height and weight are written on a piece of paper, one could calculate the BMI and thus this would be a data breach. Thinking of fitness clubs that provide plans for their members training, this implies a significant challenge that will inevitably result in changes of practice. New principles that can be found in the regulation are for example the transparency of data processing which grants the user access to information on how their data is processed. In practice this means that any process involving their personal data needs to be explained to a customer in the case of any concerns about the process.

Additionally, the new GDPR clarifies several points that can also be found in the Directive 95. These are in particular:

- *Data minimization principle* stating that the amount of data kept must be limited to the amount necessary to fulfil the task for which one has given the consent.

- *Processing of children's data.* If children are under 16 years of age, processing of data requires the consent and authorization of a parent or custodian. This implicitly limits any use of online content which is intended to be for minors, especially under the age of 16. So far it appears that providers of such content may need to put much effort on implementing this restriction.

Transparency and modalities

This category is closely related to the principle of transparency of data processing described above. It forces a data controller to determine the purposes of the processing of data. There must be an explanation of conditions for the consent. The consent must be given freely and explicitly, can be withdrawn

at any time and must not be expanded to any corresponding process without prior permission.

Information and access to personal data

This category expands the user's rights to get information from the data controller about transfers of data to other countries. It also requires a mechanism for the user to access his data and ask for corrections to be made. Furthermore, information on the processing and purpose of processing must also be provided on request. Information about the user's rights must be provided. This point, as well as the previous one, forces data processing entities to be aware of their own processes and to check their lawfulness since they may be required to explain what they are doing.

Rectification and erasure

There are two very important rules in the GDPR that need to be considered: *right to rectification* and *right to erasure*. The *right to rectification* obliges any data processing or storing entity to correct any data if requested by the user, as well as restricting the processing of the data to the amount authorized by the user. Additionally, the *right to erasure*, which is also known as the right to be forgotten, grants the user the right to instruct the deletion of their data. This erase has to happen in a cascading manner, particularly if the processor or the controller has handed over any data to a third party. These two rights oblige data processing or storing entities to be aware of any relation to any user and storage location(s) of all data in their possession.

Right to object and automated individual decision making

These two points refer back two corresponding articles in the DIR95. First, the *right to object* forces the processor of personal data to provide legitimate reasons for his work and the processing of the data. This must be stated explicitly. The second point protects the user from being evaluated and maybe even categorized automatically without human interaction. If consent to automated processing is not given explicitly, then the user should be provided with the opportunity to conclude their business through an interaction with a natural person. However, there is a real threat that service providers will simply decline to provide the required service if the user is unwilling to allow the use of their personal data.

General obligations

This category contains at first the principles of *data protection by design* and *by default*. *Data protection by design* implies that, for every new process implemented, data protection must be a major design criterion. This forces companies to be aware of data protection even at an early point and maybe avoid setting up processes due to their non-compliance. *Data protection by default* implies the highest standard of data protection settings must always be the

default. In practice, for example if one applies for a social network, the privacy settings must be the highest possible and any broadcast of personal data must be prevented by default. Further points of this category clarify responsibilities for the processing of personal data. Also an obligation to "maintain record activities [...] and co-operate with the supervisory authority" [Dix+13]. This clearly puts processors into responsibility and forces them to support supervision authorities, which might improve an open and more aware culture of data processing.

Security of personal data

The main point of this category is the obligation to report personal data breaches to a supervisor, respectively a supervisory authority and to the user. So, from now on, one must be informed if any information is leaked or anything contrary to the GDPR happens.

Data protection impact assessment and prior consultation

Mainly, this is highly related to the principles of *data protection by design* and *by default*. Controllers are obliged to evaluate the data protection impact prior to a process and only execute it, if there is a low or no risk as a result. If a high risk is detected, the supervisory authority must be taken into consideration and contacted in order to evaluate the process together and share the responsibility or even cancel the process.

Data protection officer (DPO)

A major new concept of the GDPR is the designation of a data protection officer (DPO). There are several preconditions that obligate an entity to designate someone as the DPO. Mainly, they are as follows: If an entity's core business or major activity is the processing of personal data or monitoring users, if special (in particular sensitive) data is processed or if the entity is a public authority. Any designation beyond these conditions is voluntary. The DPO has several tasks like monitoring the processes, supporting the implementation of new processes, analysing existing processes, informing and training the staff, checking on the general compliance and providing advice. Moreover he/she has to perform risk impact analyses, report to the management board, and also to report breaches to the supervisory authority. The DPO does not have to be a full time role, but the DPO must be selected carefully as someone who can fulfil his or her duty independently without the risk of any negative consequence regarding his or her other responsibilities.

Code of conduct and certification

The new regulation offers ways to approve the compliance to codes of conduct by supervisory authorities and general "certification, mechanisms, seals and marks" [TT17] to show that an organization complies with the regulation.

This is to simplify such processes without running the risks associated with self-certification, which limit the accountability of programs.

Transfer of personal data to Non-EU countries or international organizations

The articles related to this category aim to strictly limit the uncontrolled distribution of personal data, especially to other countries according to their level of data protection compliance.

Remedies, liability and penalties

The last category is divided into three parts.

- First, it is now clearly specified how a user can take action against an entity believed to be infringing the GDPR. Furthermore, it is clarified which entities are allowed to take action in addition to the user taking action. This point is a huge progress in the field of customer rights because it is now much easier to demand one's rights.
- Second, from now on, everybody involved in the processing of data is responsible and can be called to account if a data breach occurs. This makes it harder or even impossible for entities to pass on their liability by processing data through a network of processors.
- Last and most important, the fines that can be imposed, have risen significantly and are now at a maximum of 20 million euros or 4% of the total annual global turnover. Therefore, it has become more important for companies to comply to the GDPR in order to avoid massive, if significant breaches occur.

2.3.3. Practical steps to protect Personal Data

There are many consultancies who can advise organizations and projects on how to comply with GDPR. Alternatively, if a company takes care of it by itself, there are very useful resources available, such as the British Standards Institution (BSI) guide on 20 steps to GDPR compliance (see [Bri17]). A check list was used in the VICINITY project, that was adapted from the BSI guidelines. The steps developed for VICINITY are presented below in chronological order, with later activities needed as the design and implementation moves ahead:

1. There should be a project Ethics Board that will appoint a project DPO (to work with DPOs in each company involved in the collaboration). Appointing a DPO is a legal requirement if it is considered that a breach of personal data privacy could have a major impact. In any case, it is still worth having a DPO.
2. Identify and assign responsibilities to Data Controllers (DC) and Data Processors (DP) for the new value-added services.

3. DPO should create and maintain a register to record the achievement of all the actions identified below, as an aid to manage the process and evidence that good processes were followed - which could be vital evidence of mitigation measures taken, if there is ever a data breach that leads to legal action being taken against the organization.

4. Training should be provided to people with new responsibilities for protecting personal data. (Company top-team, DPO, DCs, DPs and Ethics Board as a minimum). Also, raising awareness among everybody being involved, leads to an increased acceptance and cooperation.

5. Data controllers create data registers to identify all the types of personal data and sensitive personal data that will be collected. Headings for information to be included in the register include:
   a) For each system that handles personal data, create a file that identifies the nature of personal data held.
   b) Record why it is held and how it will be processed.
   c) Explain how legal consent for this purpose will be obtained.
   d) Define the required data retention period & data removal date.
   e) Identify the set of people that will be granted access to this personal data, keeping this to the minimum number possible.

6. Data controller works with the system architect to create a data flow diagram - showing all the subsystems that the data passes to or through.

7. Ensure adequacy and non-excessiveness: review the data flow diagram and identify any steps that can be taken to reduce the number of subsystems that have sight of the personal data and also to reduce the overall collection of data.
   a) Remove personal identification as soon as possible leading to anonymization.
   b) If anonymization is not possible, use pseudonymization as soon as possible, keeping the pseudonymization table in a core database. This will mean that if (when!) a data breach does occur, it will only release pseudonymized data which the unauthorized third party cannot understand unless they can also hack the pseudonymization table.

8. Try to minimize the risk of wider distribution of personal data by avoiding use of third party services, using in-house systems by preference. Where third party suppliers must be used, choose these carefully based on an assessment of the risks of leakages occurring. Also, try to establish a contractual framework w.r.t. compliance.

9. DPOs should oversee the creation of briefing packs including a Privacy Notice and consent forms to confirm permission for current and intended

uses of the personal data. Both, for internal and external stakeholders' information.

   a) Fully explain what data will be collected, why the data is needed, all the uses that will be made of the personal data, who will have access to personal data, where data will be stored, how long it will be held, who to contact if the user has worries or wishes to change the permission that they have granted.

   b) Explain the rights that the user has: inspect data held; right to rectify; withdraw permission to use; decline profiling; instruct transfer of data to another service provider or to have all records deleted.

10. Data Retention: the data controller should check that personal data has been destroyed by the agreed destruction date.

11. DPO monitors that user rights are being observed.

12. Data breach response: Develop and install a plan for what to do when a data breach does occur: who plays which role in the company, who to inform, handling press enquiries, etc. The legal framework provides duties such as reporting within 72 hours to the supervising data protection authority and the user.

13. Review what has been implemented. Are there any locations where personal data is held that could be at risk of a physical attack, hack or leak.

14. Produce a baseline Data Protection Impact Assessment (DPIA). What would be the result of a failure? This needs to be fed into project and company risk registers.

    a) Note that the production of a DPIA is a legal requirement if the activity is deemed to be "Likely to produce a high risk", in other case it is just good practice.

15. Update the DPIA considering operational aspects.

16. Ensure that the project senior management continue to be committed to the actions needed, to ensure personal data are properly protected.

    a) Management Board should be made aware of and discuss: incidents, near misses, user requests, DPIA reviews, performance of third parties trusted with personal data

17. Review and revise (if necessary) the company Data Privacy Policy.

18. Communicate to, and continue to train system architects, designers, developers, integrators and operators on the approach that is to be taken, regarding handling of personal data, and the need to mitigate risks at all stages.

Chapter 3

Single- and Multiparty Computation

Contents

"Homomorphic Encryption (HE) is a breakthrough new technology which can enable private cloud storage and computation solutions."[Alb+19]

HE uses a public-key encryption scheme with homomorphic properties. This process relies on the concept of homomorphism, i.e. the ability to carry out the required operations on encrypted data with the result being equivalent to performing the same operation on the unencrypted data and then encrypting it afterwards. This is depicted in figure 3.1.

However, this representation of homomorphic encryption is quite simplified. It is based upon comprehensive number-theoretical considerations about ideals, lattices and ideal lattices. Figure 3.1 shows the basic functionality of homomorphic encryption. This thesis will tackle some of the most widespread encryption schemes only on the surface, where appropriate. For further reading

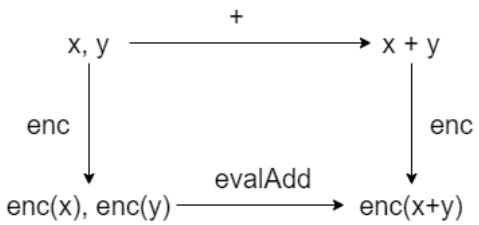

Figure 3.1.: *Evaluating $x + y$ on plaintext or ciphertext yields the same result*

on the theory below, please refer to the mathematical works of Paillier [Pai99] and Gentry [Gen09]. Further information on serveral improvements and new generations of HE schemes are introduced by Vaikuntanathan, Brakerski, van Dijk and Halevi [Dij+10; GH11; BV11; BGV11].

Formally, a HE scheme ϵ can be seen as $\epsilon = (KeyGen, Enc, Dec, Eval)$ with:

1. *KeyGen*, key generation
2. *Enc*, encryption
3. *Dec*, decryption
4. *Eval*, evaluation/computation

While the homomorphism of the encryption scheme is only relevant for evaluations and computations on ciphertext, it makes the scheme way more powerful:

$$Dec(Eval(Enc(x), Enc(y))) = Dec(Enc(x + y)) \tag{3.1}$$

This approach means that neither the input data nor the output data can be understood or decoded by the system that is carrying out the operation. As described in the jewelry store analogy (see section 3.1), this directly translates to a glove box, where operations could only be carried out inside the box, or in the context of HE, on encrypted data.

3.1. Homomorphic Encryption: An Analogy

An analogy is commonly used to illustrate the idea of homomorphic encryption: Alice's jewelry store (see [Gen09]):

Here, Alice owns a jewelry store and produces and crafts the finest jewelry out of valuable, raw materials. The demand for her art has grown high, Alice cannot produce every piece herself. Instead, she hires workers to help her. As she does not know the newly hired assistants, Alice does not fully trust them and is afraid one of them might try to steal some of the precious raw materials.

Alice is in a dilemma, that she needs help in crafting her jewelry and her workers will require access to the raw materials in order to do so. Yet, she still is afraid of theft by her workers.

Alice comes up with the idea to put all of the raw materials inside a locked glove box, for which only she keeps the key. Her workers can assemble the jewelry inside the box, but are unable to get any of the materials or the final jewelry out. Only once they are finished, Alice can use her key to unlock the box and retrieve the final jewelry.

While this story may sound quite far fetched, this very well translates to homomorphic encryption. First off, the underlying problem is quite similar. While Alice in the analogy is not able to produce all the jewelry herself, we can think of e.g. constrained low-energy devices, which do have quite limited computational power. So computations could be out-sourced to e.g. a value-added service in the cloud. In this case, Alice's raw materials would be the (private) data generated by the low-energy device and the finished jewelry would correspond to a refined, evaluated result by a Value Added Service (VAS). To give a concrete example, one could think of a wearable device tracking health related data of a user during a series of workouts. The VAS would be analyzing this data for e.g. performance improvements like the user becoming fitter with every new training session.

Naturally, this data is highly sensitive and simply sharing this data in plain text is not desirable. Instead, placing this data in something like Alice's glove box would be ideal. HE allows that by transmitting sensitive data encrypted and save, while still allowing the worker or the VAS to operate on this data. The VAS only get access to the encrypted data, which makes it essentially useless for it to "steal". Only the final "jewelry", that is the processed data is send back for decryption by the user.

In the following, this analogy is used to further explain certain concepts.

3.2. Partially Homomorphic Encryption

The basic concept that we want to exploit is to use a binary operation on plaintext while only manipulating the ciphertext without the knowledge of the encryption key. Schemes supporting homomorphic evaluation of a computable function define homomorphic encryption. A scheme like RSA (see 3.2.1) is homomorphic for multiplications only and is hence called a partially homomorphic encryption scheme as opposed to fully homomorphic encryption schemes (see 3.3), that to a given extend allow arbitrary operations. Generally speaking, HE can be split into three categories, as shown in figure 3.2: Partially homomorphic encryption allows only one operation on ciphertexts, e.g. just multiplication or just addition. Fully homomorphic encryption allows an arbitrary number of multiplications and additions. Leveled fully homomorphic encryption allows additions and multiplications, but only a predefined amount before decryption becomes impossible.

3.2.1. Rivest, Shamir and Adleman (RSA)

"The idea of homomorphic encryption is as old as public-key encryption. Homomorphic properties are given in the Rivest, Shamir and Adleman (RSA) en-

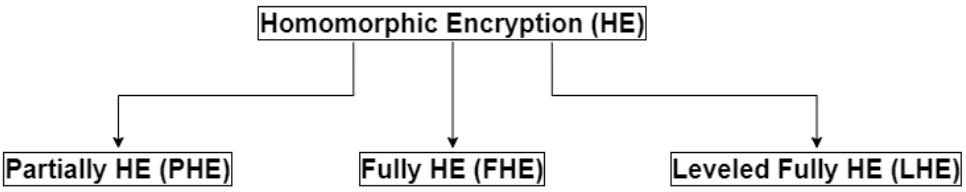

Figure 3.2.: *HE can be classified into three categories: Partially homomorphic encryption, fully homomorphic encryption and leveled fully homomorphic encryption.*

cryption, which was introduced in 1978 by Rivest, Shamir and Adleman [RSA78]." [Hei+20]
The RSA encryption scheme is homomorphic with respect to multiplication,
as given in algorithm 1.

Algorithm 1: RSA algorithm that supports multiplicative homomorphic encryption.

KeyGen:
- Select two prime numbers p and q with $p \neq q$
- $n = pq$
- ϕ(n) = (p - 1)(q - 1)
- Select e with $1 < e < \phi(n)$ and $gcd(e, \phi(n)) = 1$
- $d = e^{-1} \bmod \phi(n)$
- Public key: (e, n)
- Private key: d

Enc:
- $c = m^e \bmod n$

Dec:
- $c = m^d \bmod n$

EvalMult:
- given two ciphertexts c_1 and c_2, with encryption key (e, n):
- $c_1 = m_1^e \bmod n, c_2 = m_2^e \bmod n$
- then $c_1 \cdot c_2$ yields:
- $c_1 c_2 = Enc(m1)Enc(m2) = m_1^e m_2^e \bmod n$
- $= (m_1 m_2)^e \bmod n = Enc(m_1 m_2)$

"As the plain RSA or "Textbook-RSA", which was outlined above, is not considered secure anymore, a padding mechanism was introduced to avoid possible attack patterns. However, these padding functions break the homomorphic properties of RSA. Additionally, one might require not multiplication, but rather the addition of c_1 and c_2."[Hei+20]

RSA is an example of a partially homomorphic cryptosystem. Indeed, there are different types of homomorphic encryption schemes (see Figure 3.2. In general, they can be split into Fully Homomorphic Encryption (FHE), Leveled Fully Homomorphic Encryption (LHE) and Partially Homomorphic Encryption (PHE). PHE only allows the use of one operation (either addition or multiplication), for example the addition of ciphertexts. In contrast, FHE allows the calculation of all general polynomials in any arbitrary number. Finally, LHE or *somewhat homomorphic encryption* as it is also sometimes called, allows additions and multiplications, but only a predefined amount before decryption becomes impossible. This will be further explained in section 3.4. Some of the most frequently used encryption schemes (like RSA) are partially homomorphic with respect to multiplication but not additive homomorphic.

3.2.2. Paillier Encryption

"In 1999, Pascal Paillier published his work on a new "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes" [Pai99]. Paillier's encryption scheme allows for the two ciphers c_1 and c_2 to be added up rather than multiplied. This operation is much more common and will be required for the data protection approach, which is further described in Section 3.4.6. Paillier first introduced Composite Residuosity and classes of numbers that fulfil this property."[Hei+20] "As a link to RSA (see 3.2.1), Paillier elaborates the hardness of finding the n-th modulo n^2 residue and therefore states the intractability of Composite Residuosity."[Hei+20]

Algorithm 2: Paillier Encryption Scheme [Pai99]

KeyGen:
- With $n = pq$, $\phi(n) = (p-1)(q-1)$ and p and q are random primes of length k.
- Public Key: n
- Private Key: $(n, \phi(n))$

Enc:
- Given message m to be encrypted with $0 \leq m < n$
- Select random r with $0 < r < n$
- Compute ciphertext c as:
- $c = (n+1)^m r^n \bmod n^2$

Dec:
- Given ciphertext c
- Compute message m as:
- $m = \frac{(c^{\phi(n)} \bmod n^2)-1}{n} \phi(n)^{-1} \bmod n$

EvalAdd:
- The product of two ciphertexts corresponds to the encrypted sum of the plaintexts:
- $c_1 \cdot c_2 = g^{m_1} \cdot r^n \cdot g^{m_2} \cdot r^n \bmod n^2$
- $= g^{m_1+m_2} \cdot r^{2n} \bmod n^2 = Enc(m_1 + m_2) \bmod n$

It hence follows, that the property of homomorphic additivity holds for the encryption function and thus, the given encryption scheme is additive homomorphic.

3.2.3. ElGamal Encryption

In 1985, Taher Elgamal described an encryption scheme based on the Diffie-Hellman key exchange [Elg85]. The encryption scheme is defined over a cyclic group G. Its security is based on the difficulty of computing discrete logarithms in G. The encryption scheme is given as in algorithm 3.

Algorithm 3: ElGamal Encryption Scheme [Elg85]

KeyGen:
- Generate (G, p, g), where G is a cyclic group of order p and generator g.
- Choose x randomly with $1 \leq x \leq p-1$
- Compute $h = g^x$
- Public Key: (G; p; g; h)
- Secret Key: x

Enc:
- Given message $m \in G$
- Select random r with $0 < r < p-1$
- Compute $s = h^r$
- Compute $c_1 = g^r$
- Compute $c_2 = m \cdot s$
- Ciphertext is given as:
- $c = (c1; c2)$

Dec:
- Given ciphertext $c = (c1; c2)$ and secret key x
- Compute message m as:
- $m = c_1^{-} x \cdot c_2$

EvalMult:
- Compute $c \cdot c'$ as $(c_1, c_2) \cdot (c'_1, c'_2)$
- $= (c_1 \cdot c'_1, c_2 \cdot c'_2) = (g^r \cdot g'^r, h^r \cdot m \cdot h'^r \cdot m') = (g^{r+r'}, h^{r+r'} \cdot m \cdot m')$

Hence, it follows that ElGamal encryption is homomorphic with respect to multiplication.

3.3. Fully Homomorphic Encryption

The encryption schemes above all belong to the category of PHE. That is, they allow either multiplication of two ciphertexts or addition of them, but never both. Encryption schemes offering both operations at the same time make up another category of schemes, namely FHE schemes.

In his PhD thesis [Gen09], published 2009, Craig Gentry introduced "A fully homomorphic encryption scheme" by applying bootstrapping on ideal lattice based, *somewhat homomorphic encryption* schemes (see 3.4).

For *somewhat homomorphic encryption* schemes, both addition and multiplication are possible on ciphertexts. However, only up to a certain point, after which the noise introduced into the ciphertext grows too large and decryption becomes impossible. More on that will be discussed in 3.4. Additionally, some *somewhat homomorphic encryption* schemes allowed arbitrary circuits or functions being computed on homomorphically encrypted ciphertexts [SYY99]. However, these ciphertexts increase in size exponentially to the depth of the circuit evaluated, making it unfeasible for repetitive evaluations of ciphertexts.

In his thesis however, Gentry intended to develop an encryption scheme, which would allow an unlimited number of additions and multiplications. So the general idea of Gentry was:

- starting point is the *somewhat homomorphic encryption* scheme using ideal lattices
- then squash the decryption procedure so that it can be expressed as a low-degree polynomial
- and finally apply a bootstrapping transformation, through a recursive self-embedding, to obtain a fully homomorphic scheme.

With the results of Gentry's PhD thesis, it is possible to evaluate arbitrary functions on ciphertexts, however at the cost of computational demand and key size.

Going back to the analogy of Alice's jewelry store (see 3.1), think of lots of clutter stockpiling up inside the glove box, making it more and more difficult to continue working. Gentrys bootstrapping idea could mean, that first Alice's workers will need to place all the valuable materials into another box, inside of the glove box, and only then they themselves can unlock the surrounding box to retrieve the inner box. This will in return remove all clutter, which was laying around in the original box, so they could continue working inside this new, fresh box.

3.4. Somewhat/Leveled Homomorphic Encryption

The first fully homomorphic encryption scheme was developed by Boneh et. al. [BGN05]. This scheme allowed to perform unlimited additions but only

one multiplication of ciphertexts. Many other encryption schemes were developed since then, which also allow addition and multiplication of ciphertexts to a certain extend. Gentry (see [Gen09]) called these schemes *somewhat homomorphic*. However, as later schemes let the user control and fine-tune at which level decryption becomes impossible, a more fitting term for these schemes is LHE. Some of the most commonly used ones include:

- Brakerski-Gentry-Vaikuntanathan (BGV) [BGV14]
- Brakerski/Fan-Vercauteren (BFV) [Bra12; FV12]
- van Dijk, Gentry, Halevi and Vaikuntanathan (DGHV) [Dij+10]
- FHEW (Ducas and Micciancio) [DM15]
- Cheon-Kim-Kim-Song (CKKS) [Che+17] (aka HEAAN)
- TFHE (Chillotti, Gama, Georgieva and Izabachene) [Chi+20]

Most of the recent FHE or better LHE schemes are based on the (ring-)learning with errors problem (see 3.4.3). With these schemes, it is essential to add some error or noise e to the ciphertext, to make the underlying problem np-hard. A simplified structure of a ciphertext based on the Ring Learning with Error (RLWE) problem is shown in figure 3.3.

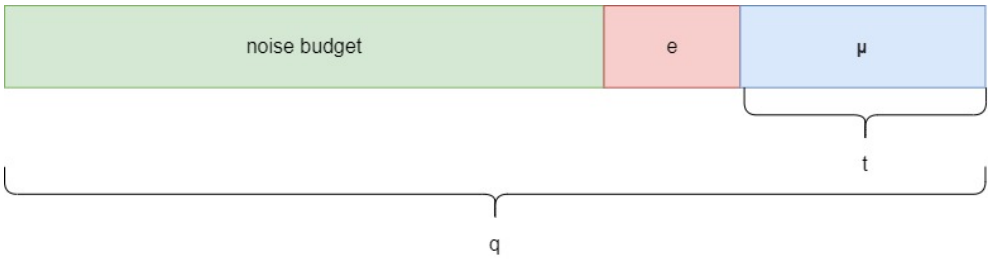

Figure 3.3.: *Illustration of ciphertext structure. The noise e grows with each homomorphic multiplication. If the size of e exceeds the noise budget, decryption is impossible and it is no longer possible to obtain the payload μ. t denotes the plaintext modulus, q denotes the ciphertext modulus.*

As can be seen, decryption is only possible as long as the error e does not exceed the available noise budget. For most schemes, this noise budget can be chosen by the user to match their intended application. However, choosing a larger noise budget directly translates to larger ciphertext sizes.

To tailor the encryption to the needs of the intended application, LHE schemes can be configured with additional, auxiliary parameters, such as:

- The security level describing the strength of the cipher.

- One coefficient modulus for each multiplicative operation, that should be performed
- The Polynomial Modulus Degree
- The Plaintext Modulus

3.4.1. Lattice based Encryption Schemes

As outlined above, most commonly used *somewhat homomorphic encryption* schemes today are based on lattices. To further understand and argue about these schemes, we first need to understand what a lattice is: "A lattice is a set of points in n-dimensional space with a periodic structure" [MR09]. A simple lattice is illustrated in Figure 3.4.

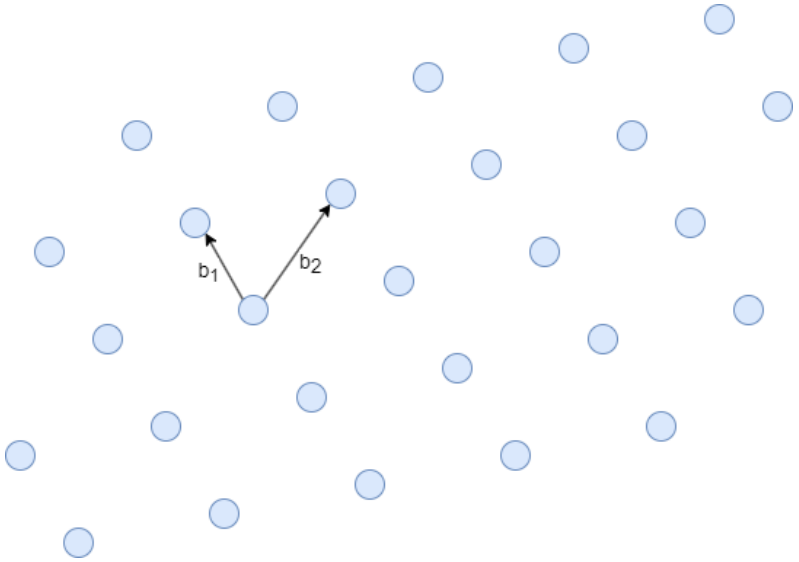

Figure 3.4.: *Simple lattice in $\mathbb{R}^2$ with basis vectors b_1 and b_2*

"More formally, given n-linearly independent vectors $b_1, ...b_n \in \mathbb{R}^n$, the lattice generated by them is the set of vectors

$$\mathcal{L}(b_1, ..., b_n) = \left\{ \sum_{i=1}^{n} x_i b_i : x_i \in \mathbb{Z} \right\}.$$

The vectors $b_1, ..., b_n$ are known as a basis of the lattice." [MR09]

In other words, given a set of basis vectors $B = b_1, ..., b_n$, a lattice is the set of all integer linear combinations B. Naturally, the same lattice can be generated by different sets of basis vectors. Several problems can be constructed based on lattices, while finding solutions to these presumed hard problems is used to build encryption schemes. E.g. one of the most basic lattice problems is the Shortest Vector Problem (SVP). For the SVP, a lattice is given represented

by an arbitrary basis and the goal is to find the shortest nonzero vector in it. [MR09] In fact, the *exact*SVP has been proven to be NP-hard (see [Ajt98]).

3.4.2. Learning With Errors

The Learning with Error (LWE) problem [Reg10] is, given a lattice in $\mathbb{Z}^n$ with base vectors $b_1, ...b_n$ and given a point sk in $\mathbb{Z}^n$ near a particular lattice point p, to find the exact lattice vector p. In other words, this translates to: Given a sequence of *approximate* linear equations on sk, find sk.

More formally, the LWE problem is given in algorithm 4

Algorithm 4: Learning with errors algorithm [Reg10]

1 Given a size parameter $n \geq 1$ and a modulus parameter $q \geq 2$
2 Specify a secret key $sk \in \mathbb{Z}_q^n$
3 Choose vector $a \in \mathbb{Z}_q^n$ uniformly at random
4 Choose noise $e \in \mathbb{Z}_q^n$ distributed normally with standard deviation $\chi \in \mathbb{Z}_q$
5 Output $\langle a, sk \rangle + e \bmod q$
6 Keep sk and repeat for a and e

The system of equations could be easy solved without the added error term e. However, with the added error, this problem is proven to be NP-hard (see [Ajt98]). In fact one of the best solution algorithms from Blum, Kalai and Wasserman [BKW00] still in $O(2^n)$. Until today, no quantum attacks against LWE are known [Reg10]. The LWE problem can be used to construct a variety of cryptographic algorithms. Yet, most encryption schemes do not use LWE itself but the derivation RLWE.

3.4.3. Ring Learning With Errors

The RLWE problem (see [LPR12]) is moving on from the LWE problem but dealing with polynomials instead of integer values. Also n and q, the size and modulus parameters from algorithm 4 are now subject to the following constraints:

- n is a power of 2
- q is prime and $q = 1 \bmod 2n$

Then a ring R is defined as:

$$R = \mathbb{Z}[X]/X^n + 1$$

Finally, the RLWE problem can be formulated as shown in algorithm 5.

Algorithm 5: Ring learning with errors algorithm [LPR12]

1 Given a size parameter n as a power of 2 and a modulus parameter q prime and $q = 1 \bmod 2n$
2 Specify a secret key $sk \in \mathbb{Z}_q^n$
3 Choose vector $a \in R_q$ uniformly at random
4 Choose noise e from $\chi \in R_q$
5 Output $\langle a, sk \rangle + emodq$
6 Keep sk and repeat for a and e

3.4.4. BGV Encryption Scheme

The BGV encryption scheme [BGV14] is based on the above RLWE problem (see 3.4.3), so consequently the security of this scheme also depends on the hardness of the RLWE problem (see [LPR12]), which was proven to be NP-hard (see [Ajt98]). BGV defines two polynomial rings. The plaintext and the ciphertext ring. During encryption, an element on the plaintext ring is mapped onto the ciphertext ring with an almost random mask. This mask is computed from the public-key plus some error component e, as can be seen in algorithm 6. In order to be able to later decrypt a given ciphertext again or in other words, to remove the mask with the random error e again, the ciphertext consists of two elements $c = (c_1, c_2)$ both elements of the ciphertext ring. The first element c_1 contains the masked plaintext. The second element c_2 contains additional information, which are required in order to be able to decrypt. This is similar to the ciphertexts of the ElGamal encryption scheme (see 3.2.3), where ciphertexts also contain an additional element required for decryption.

The error term e is crucial for any RLWE-based scheme such as BGV in order to make the underlying problem hard to solve. With each homomorphic operation (addition or multiplication) this error grows. Typically, a ciphertext starts at level L after encryption. With each operation on the ciphertext, this level decreases until level 1 is reached. At this point, no further operation except for decryption is possible. Otherwise, the error e, also called noise, exceeds the available *noise budget* (see Figure 3.3) and it is no longer possible to correctly recover the plaintext for the given ciphertext. Hence, the BGV encryption scheme falls into the category of LHE schemes. The encryption prodedure is shown in Algorithm 6.

As it becomes obvious in algorithm 6, *EvalMult*, the multiplication of two ciphertexts increases the number of ring elements to 3. This is also referred to as *ciphertext expansion*. Hence, $C^{(3)}$ cannot be decrypted as it is. An additional step becomes necessary first: Relinearization. [Inf22a; Inf22b]

Relinearization

After *EvalMult*, the resulting ciphertext C^3 contains 3 ring elements $C^3 = (C_1^3, C_2^3, C_3^3)$. The goal of Relinearization is to find $\hat{C}^{(3)} = (\hat{C}_1^{(3)}, \hat{C}_2^{(3)})$, such

Algorithm 6: BGV Encryption Scheme [BGV14]

KeyGen:

Given two polynomial rings. The plaintext ring P is given as:

$P = R_t = \mathbb{Z}_t[x]/(x^n + 1)$ with t being the plaintext modulus and ring dimension n

The ciphertext space is defined at each level l with $\leq l \leq L$ as:

$C = R_{q_l} \times R_{q_l}$ with q_l being the ciphertext modulus at level l

Secret Key (SK): Sample random, ternary polynomial in R_2

Public Key:

$PK = (PK_1, PK_2)$ with

$PK_1 = [-1(a \cdot SK + t \cdot e)] \bmod q_L$

$PK_2 = a$, with a being random polynomial in R_{q_l}

Enc:

Given plaintext message m in P

Sample small, random polynomials u, e_1 and e_2 $C = (C_1, C_2)$ is given as:

$C_1 = [PK_1 \cdot u + t \cdot e_1 + m] \bmod q_l$

$C_2 = [PK_2 \cdot u + t \cdot e_2] \bmod q_l$

Dec:

$m = [[C_1 + C_2 \cdot SK] \bmod q_l] \bmod t$

EvalAdd:

$C^{(1)} + C^{(2)} = ([C_1^{(1)} + C_1^{(2)}] \bmod q_l, [C_2^{(1)} + C_2^{(2)}] \bmod q_l) = (C_1^{(3)}, C_2^{(3)}) = C^{(3)}$

EvalMult:

$C^{(1)} \cdot C^{(2)} = ([C_1^{(1)} \cdot C_1^{(2)}] \bmod q_l, [C_1^{(1)} \cdot C_2^{(2)} + C_2^{(1)} \cdot C_1^{(2)}] \bmod q_l, [C_2^{(1)} \cdot C_2^{(2)}] \bmod q_l)$

$= (C_1, C_2, C_3)$

that

$$C_1^3 + C_2^3 \cdot SK + C_3^3 \cdot SK^2 \bmod q_l \approx [\hat{C}_1^{(3)} + \hat{C}_2^{(3)} \cdot SK + r] \bmod q_l$$

For this, another Key apart from the already defined public key PK and secret key SK can be computed beforehand: the evaluation key EK:

$$EK = (-(a \cdot SK + e) + SK^2, a)$$

With this, also access to SK^2 is provided, since $EK_1 + EK_2 \cdot SK = SK^2 - e$. With EK, relinearization and hence $\hat{C}^{(3)}$ can be computed as

$$\hat{C}_1^{(3)} = [C_1^3 + EK_1 \cdot C_3^3] \bmod q_l$$
$$\hat{C}_2^{(3)} = [C_2^3 + EK_2 \cdot C_3^3] \bmod q_l$$

3.4.5. CKKS Encryption Scheme

The previous encryption schemes all work on integer values and polynomials. For some practical applications, this does not suffice. Rather working with real numbers is required for these applications. With the CKKS encryption scheme [Che+17], this is possible and even more. In CKKS, the plaintext space is in the complex number space $\mathbb{C}^{\frac{N}{2}}$ for a positive integer $N \in \mathbb{N}$. As the previous scheme BGV, CKKS is also based on the RLWE problem. With this being said, the RLWE problem works on polynomials, not complex numbers. Hence, the input complex number vectors first need to be scaled to the desired accuracy of real numbers and mapped onto single plaintext objects (polynomials). This step in CKKS is referred to as *encoding*. [Inf22c]

Encoding

Encoding in CKKS means, given an $\frac{n}{2}$-vector of complex numbers z and a scaling factor s, return a single plaintext element $a \in R$, with R being a polynomial ring as defined in 3.4.3. Decoding is the reverse operation, taking a plaintext element a and the same scaling factor s that was used during encoding and returning the mapped vector of complex numbers. Encoding and decoding can be done as

$$ENCODE(z, s) = \lfloor s \cdot \pi^{-1}(z) \rceil$$
$$DECODE(a, s) = \pi(\frac{1}{s} \cdot a)$$

The mapping function π in both $ENCODE$ and $DECODE$ functions is the complex canonical embedding, a variant of the Fourier Transform [Inf22c].

With the input vector encoded, the plaintext object can be encrypted. The CKKS encryption scheme works then similar to the BGV encryption scheme as described above. The CKKS encryption scheme is given in algorithm 7

Algorithm 7: CKKS Encryption Scheme [Che+17]

KeyGen:

Key Generation in CKKS is done similar to the BGV encryption scheme as described in algorithm 6 Secret Key (SK): Sample random polynomial of degree n in R_2

Public Key:

$PK = (PK_1, PK_2)$ with

$PK_1 = [-a \cdot SK + e)] \bmod q_L$

$PK_2 = a$, with a being random polynomial in R_{q_l}

Enc:

Given plaintext message m in R

Sample small, random polynomials u, e_1 and e_2 $C = (C_1, C_2)$ is given as:

$C_1 = [PK_1 \cdot u + e_1 + m] \bmod q_l$

$C_2 = [PK_2 \cdot u + e_2] \bmod q_l$

Dec:

$m = [C_1 + C_2 \cdot SK] \bmod q_l$

EvalAdd:

$C^{(1)} + C^{(2)} = ([C_1^{(1)} + C_1^{(2)}] \bmod q_l, [C_2^{(1)} + C_2^{(2)}] \bmod q_l) = (C_1^{(3)}, C_2^{(3)}) = C^{(3)}$

EvalMult:

$C^{(1)} \cdot C^{(2)} = ([C_1^{(1)} \cdot C_1^{(2)}] \bmod q_l, [C_1^{(1)} \cdot C_2^{(2)} + C_2^{(1)} \cdot C_1^{(2)}] \bmod q_l, [C_2^{(1)} \cdot C_2^{(2)}] \bmod q_l)$

$= (C_1, C_2, C_3)$

As BGV and CKKS work similar except for the additional encoding and decoding steps necessary in CKKS, it can also be seen in algorithm 7 again: *EvalMult*, the multiplication of two ciphertexts increases the number of ring elements to 3. This also requires the Relinearization step [Inf22a; Inf22b] for CKKS, as previously described in section 22.

3.4.6. How is data protected with Homomorphic Encryption?

This section is adapted from my own publication in [Hei+20]. We can utilise the homomorphic property of certain encryption schemes to allow for secure sharing of private data with potentially malicious or untrusted third parties. One example could be a VAS offered by someone else. If a user is expecting some particular benefits of the VAS or is even forced into using it, keeping his sensitive data private is a major concern. Missing trust can potentially even be a showstopper for the whole Internet of Things (IoT).

The following approach for secure data sharing is proposed: First and foremost, only encrypted data should be shared. Encrypted user data could be collected and stored in the cloud, so that storage and requests are performed in the cloud on a more powerful device.

This is an important mechanism against eavesdropping. Yet, it does not solve any privacy concerns in a scenario, where the receiving party is malicious, compromised or for other reasons simply cannot be trusted. Security and privacy always come down to a matter of trust. Who can be trusted? Do I really need to give out my sensitive data to this service? Do I need to give out any sensitive data at all? In many cases, the latter is not necessarily required.

An example utilising homomorphic encryption in combination with a fitness-tracking app is described in section 1.1.1: A fitness-tracker uploads encrypted fitness data onto the cloud and later asking for "average heart rate over the last 24 hours": in this case, decryption is done at the user's site, with no clear text visible to third party at all. This enables moving data storage and parts of the data processing onto a more powerful cloud server without disclosing any cleartext information to said server, as decryption is done at the user's site. If the user keeps his decryption key on e.g. his laptop and his smartphone, he is able to decrypt the data on both devices and can hence still take advantage of the benefits a cloud offers without trusting it.

In a multi-party or data sharing case, encrypted data is again uploaded to a VAS. Only this time the data is aggregated using the inputs of multiple parties. Once enough inputs have been collected, the encrypted data is immediately aggregated and then all the personal data deleted. With a sufficient number of sources the aggregated data can be considered to be anonymized. Anonymized data can be decrypted and made available to the VAS for further processing outside the constraints of General Data Protection Regulation

(GDPR).

A typical example for such a scenario would be an energy demand management as described in section 1.1.2. Participating households are required to send their current or planned energy consumption to a common grid operator. The operator is in general only interested in the overall consumption of a whole district or city, not the individual households. Yet, as energy consumption can lead to conclusions about personal behaviour or even simpler tell whether a user is at home or not, this as well has to be seen as private data, which should not be revealed. Using homomorphic encryption, encrypted data can be upload and aggregate (e.g. sum up) all reported energy consumption in encrypted form. The aggregated result can finally be decrypted with the consent of all participating parties. A more in depth protocol on how this kind of secure multi-party computation can be achieved utilizing homomorphic encryption is given in [Dam+11; DA16].

The use of FHE will not be required in all cases. In some cases PHE will be sufficient, if for example a service only requires an aggregation (e.g. the sum) to operate and will only collect the provided inputs just to calculate the aggregated value.[Hei+20]

3.4.7. Homomorphic Encryption and GDPR

This section is adapted from my own publication in [Hei+20]. GDPR places a number of requirements on the Data Controllers (DC) and the Data Processors (DP). The main requirements are that personal data must be kept secure, accurate and private, that it must only be used for the stated and agreed purposes and that the user can examine data records and insist on errors being corrected. If the user wishes all his personal data records must be transferable from one organization to another, and ultimately the user can insist that all his records are deleted.

It must be recognized that shared encrypted data remain personal data, so there are constraints on where the data may be stored. Specific permission would be required to allow the personal data to be aggregated for an agreed purpose. If the encrypted personal data could ever be stolen and decrypted, then there would be a major breach of the regulations, with consequential fines. Once the data elements have been aggregated with a large number of other data elements, then the resultant aggregated dataset would not be considered to be personal data and would be outside GDPR. It would be wise to ensure that as soon as the required aggregation had been completed, all the encrypted personal data should be deleted. No copies of the encrypted personal data should be kept. The danger with holding files of personal data that are strongly encrypted by today's standards, within a few years stronger code cracking schemes will be introduced and these data would no longer be secure.

The practical implication is that a third party could operate a database with encrypted personal data held in limited-time storage, such that "untrusted" organizations would be allowed to perform analysis of the data to the extent and for the purposes permitted by the user. They would be able to receive the results of their analysis without ever seeing any source data. This still leaves the challenge of how to ensure that the data analysis carried out falls within the range of usage that the Data Subjects have agreed. So, in practice all users would need to be trusted, but the process described here does reduce the attack surface and makes the whole process more secure. [Hei+20]

3.5. Secure Multiparty Computation

[DA16] describes a protocol to aggregate data of multiple parties securely. This is relevant for the use-case described in 1.1.2, where data of different smart meters should be aggregated. Given a set of smart meters $S = \{S_1, S_2, ..., S_n\}$, [DA16] introduces a set of trusted smart meters $T_i = \{S_{i_1}, S_{i_2}, ...\}$. This trusted set is different for every user and participant in the protocol. Every user can define, which other participant can be trusted. A straight forward way to tell whether a participant can be trusted is by adding smart meters of friends and neighbours to the trusted set and not add any unknown devices. The aggregator A can be the energy supplier. A wants to know the current power drain on the grid. Only A has to know the result of the aggregation. More so, A *only* has to know the result and non of the intermediate values. A protocol assuming honest-but-curious (HC) adversaries, the HC Protocol is described in algorithm 8. [DA16]

The protocol in algorithm 8 is safe against honest-but-curious behaviour of the participants [DA16]. This means, the protocol is safe as long as there is at least one more uncorrupted participant. If of n parties, $n - 1$ are corrupted, they can work together to learn the secret input of the last uncorrupted party and thus breaking the protocol. However, if at least two participants are honest, the corrupted adversaries can at best learn the combined input of these two participants. As part of the HC Protocol, secret inputs are not shared directly. Instead, data is blinded with random values. When the parties calculate the blinded measurement, they subtract all received random values. Additionally, all initially generated and distributed random shares are added.[Nau20] Finally, they also add their own input m_i. According to the protocol, the aggregator receives all blinded measurements, which are then added up. The HC protocol is based on what is known as additive secret sharing[1]. To reconstruct a shared value, all shares are needed for reconstruction. This is unlike Shamir's Secret Sharing, where also less than that can be sufficient if desired. [DA16; Sha79].

If an adversary is no longer honest and might deviate from the protocol, the HC protocol breaks. To overcome this issue, an extension to the HC protocol is described in [DA16]. This extension is safe beyond honest-but-curious

[1] https://mortendahl.github.io/2017/06/04/secret-sharing-part1/#additive-sharing

Algorithm 8: HC Protocol against honest-but-curious behaviour [DA16]

Data: $S = \{S_1, ..., S_n\}$ is the set of all smart meters. T_i denotes the trusted set of meter i.

foreach $S_i \in S$ **do**
  foreach $S_j \in T_i$ **do**
    S_i generates a random share $r_{i,j}$ and computes $r_i = \sum_{S_j \in T_i} r_{i,j}$
  end
  S_i sends S_j the share $r_{i,j}$
  S_i waits until it receives all shares destined to it and calculates the blinded measurement

$$b_i = m_i + r_i - \left(\sum_{S_i \in T_j} r_{j,i} \right)$$

  S_i sends b_i to aggregator A
end
Upon reception of all blinded measurements, A computes $\sum_{i=1}^{n} b_i$, which is equal to $\sum_{i=1}^{n} m_i$

adversaries and is hence called BHC protocol. It is shown in algorithm 9.

The BHC Protocol introduces two new concepts compared to the HC Protocol: using HE for encrypting the shares and a zero-knowledge proof to verify that every participant follows the protocol correctly. Algorithm 9 describes, that each random share is encrypted twice: with the sender's key and with the receiver's key. Both encrypted shares are sent to the respective party in the trusted set T. Using a zero-knowledge proof, it is verified that both shares encrypt the same value. If this holds, this check proves that the sender is actually in posession of the secret information and does hence follow the protocol. The crucial thing about this proof is the fact, that no information about the secret can be learned from it. To increase security, this check can be repeated multiple times to minimize the risk of a false-positive result. One way to implement such a zero-knowledge proof is to encrypt a known value randomly selected by the verifier with the given public key. It is then asked to decrypt this value, which is easily done if one actually has the secret key. Repeating this check reduces the likelihood of returning the correct result simply by chance. For each party, if all zero-knowledge proofs succeed and all shares are received, the combined blinded measurement is computed similar to the HC protocol. Except this time, the measurements are encrypted using HE. These measurements are again sent to aggregator A, who can again utilize the zero-knowledge proof to check that all measurements received were provided by honest participants. Finally, A can add all measurements encrypted with his own public key and retrieve the final result [DA16].

Algorithm 9: BHC Protocol against beyond honest-but-curious behaviour [DA16]

Data: $S = \{S_1, ..., S_n\}$ is the set of all smart meters. T_i denotes the trusted set of meter i.

foreach $S_i \in S$ **do**
  foreach $S_j \in T_i$ **do**
    S_i generates a random share $r_{i,j}$ and computes $r_i = \sum_{S_j \in T_i} r_{i,j}$
  end
  S_i computes $E_i(m_i)$ and $E_i(r_{i,j})$, $E_j(r_{i,j})$ for all $S_j \in T_i$
  S_i sends $E_i(r_{i,j})$ and $E_j(r_{i,j})$ to each S_j
  S_i proves that $E_i(r_{i,j})$, $E_j(r_{i,j})$ encrypt the same share using protocol $ZK_PEQ(E_i(r_{i,j}), E_j(r_{i,j}))$
  S_i waits until it receives all encrypted shares $E_i(r_{k,i})$ destined to it and acalculates the encrypted blinded value:

$$E_i(b_i) = E_i(m_i)\frac{\prod_{S_j \in T_i} E_i(r_{i,j})}{\prod_{S_i \in T_k} E_i(r_{k,i})} = E_i(m_i + r_i - \sum_{S_i \in T_k} r_{k,i})$$

  S_i computes $E_A(b_i)$ and sends to A the values $E_A(b_i)$, $E_i(b_i)$ along with $ZK_PEQ(E_A(b_i), E_i(b_i))$
end
Upon reception of all $E_A(b_i)$, A verifies that all shares were incorporated correctly
A decrypts $E_A(b_i)$ and computes $\sum_{i=1}^{n} b_i = \sum_{i=1}^{n} m_i$

With the BHC protocol, everyone can verify the shared values and detect misbehaviour. Additional steps can be taken to exclude a party from future computations if they misbehave. Using HE, communication channels are secure against eavesdropping and inputs cannot be modified due to verification [DA16].

Chapter 4

Approach

4.1. Threat Model

This thesis focuses on Internet of Things (IoT) applications. Private data from connected devices is processed by an unknown Value Added Service (VAS) and hence needs to be protected. For this thesis, the following parties are considered: the VAS, users, and an infrastructure used for communication between them.

4.1.1. Threats

For this thesis, the VAS is considered beyond honest-but-curious (BHC), as described in section 3.5. As the user should at any time be able to withdraw any access granted to the VAS once misbehaviour is detected, this assumption is valid. Once granted access, the VAS might be eager to learn more about the user's data without being noticed, yet still following the protocol to avoid exclusion. Additionally, external adversaries might also gain access to data, which was transmitted to the VAS, due to a compromised system. Potential threats to consider hence are due to system compromise (e.g., data theft), financial incentives (e.g., unauthorized trades with users data), or malicious insiders (e.g., curious admins). The infrastructure being used (VICINITY) is available open source and was developed under supervision of our chair. Despite it can in theory be trusted, one might still consider the possibility of the system being compromised by an attacker.

4.1.2. Motivation

This thesis aims to protect the confidentiality of user's data while still enabling business models and VAS as much as possible. Generally, no private data should leave the user's control if possible. Generally, no data is shared without consent at all. Only in special occasions where the user can trust the VAS and cleartext data is considered necessary. Only then, the user can consent to share this information in clear text. In any other case, the user has the possibility

and is strongly advised to only share this data under a strong Homomorphic Encryption (HE) scheme.

4.2. VICINITY

As already outlined in 2.2, the VICINITY project has built its infrastructure based upon the principle "privacy by design". Since this thesis was inspired by the work on this project, the infrastructure offered by VICINITY is used as a first layer for the framework proposed in this thesis.

VICINITY's Neighbourhood Manager (see 2.2.5) gives the user control over his/her own and shared IoT devices. It lists the device's properties like its type, geographical location, via which gateway and adapter it is connected to the VICINITY Neighbourhood, and gives an overview over the device's properties, actions that can be invoked on it and events that it will trigger. Figure 4.1 gives an idea on how the Neighbourhood Manager functions.

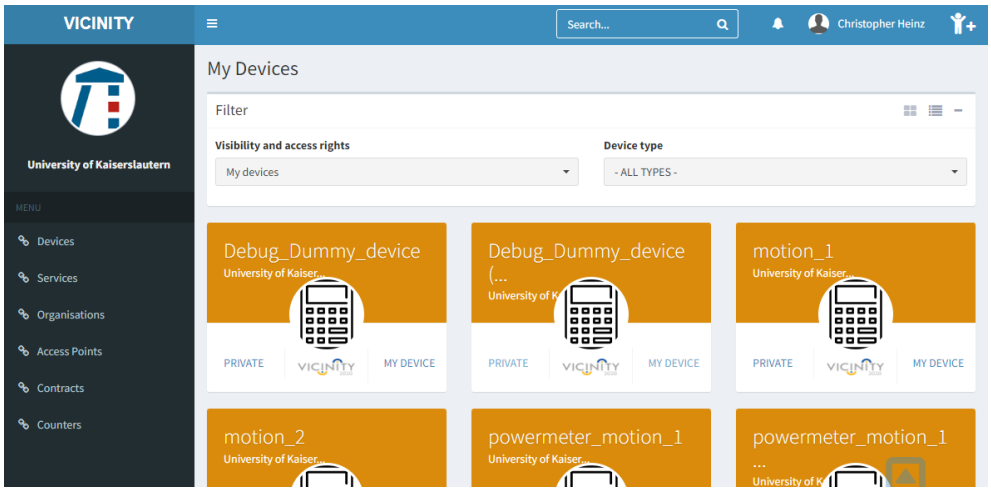

Figure 4.1.: *VICINITY Neighbourhood Manager is used to manage and control access to shared IoT devices*

To give a more concrete example, the motion sensor *motion_1*, as seen in figure 4.1, will list information about its location both in a human readable form like "Kaiserslautern", "Germany", "AG CPS Lab", as well as its GPS-coordinates. For properties, it will show whether it is currently armed and will trigger on detected motion events or not. If there is motion detected, it will also fire an event in VICINITY, which subscribed services will receive. In return, such a service can again trigger an action based on this event like "turn on some lights" or "send an alarm".

Access to these devices is granted in two steps. First off, the device's visibility is controlled. Meaning, the user is able to decide who can see, that he/she has a particular device in the first place. Say you have some sensitive, medical device attached to the VICINITY network. Then the user is able to only make this device visible for his doctor alone. Other users will not even

know, that this device even exists. The default visibility setting is “private”, so no one, except the user himself will see a particular device.

Being able to see the device only grants access to its meta-information like the aforementioned available properties, events or geographical locations. Accessing its “live” data like sensor readings is controlled separately and is granted in a second step. Any VAS also attached to the VICINITY network will need to request access to any devices “live” data. The device owner can then decide to accept or decline this request. If the device owner grants access to any of his/her devices for a particular VAS, a VICINITY contract is formed, granting the VAS either read-only or read/write access to this device. As long as neither side revokes this contract, the VAS can from this time onward access the devices live as well as - if available - historic data. As VICINITY itself only provides an infrastructure for peer-to-peer communication, afterwards, user data is only transmitted from the data subject directly to the consented VAS and never stored on any cloud on its way.

Now we take homomorphic encryption and aggregation into consideration. Once we have data that has been encrypted homomorphically, an aggregation is performed on the resulting data. The result now is anonymized and outside the scope of the General Data Protection Regulation (GDPR).

4.3. Homomorphic Encryption

The VICINITY mechanisms in place already leave the user in control over *who* can access *what* data *when*. However, it does not give any control over *how* this data is used.

To offer more control over *how* this data is used, a new component is introduced into the VICINITY infrastructure: The privacy microservice or privacy service for short. This component is added into the existing infrastructure, which is already in place at any given VICINITY deployment as shown in figure 4.2.

4.3.1. Privacy Service

This framework utilizes HE to encrypt and secure sensitive data. The special attributes of HE schemes enable data processors to work with sensitive data without decrypting it first. This leaves the data protected at all times. The selected HE scheme is based on the Ring Learning with Error (RLWE) Problem, which is considered quantum safe and is hence secure to use even in future applications (see 3.4.3). With this scheme, only the user holds the secret key to decrypt the data. This can be utilized in a single-party use-case, where e.g. the complex computation is outsourced to the cloud, but the result is only of interest to the user.

The privacy service acts as an additional *VICINITY adapter* (see 2.2.3), which will duplicate all existing adapters in return offering access to each of

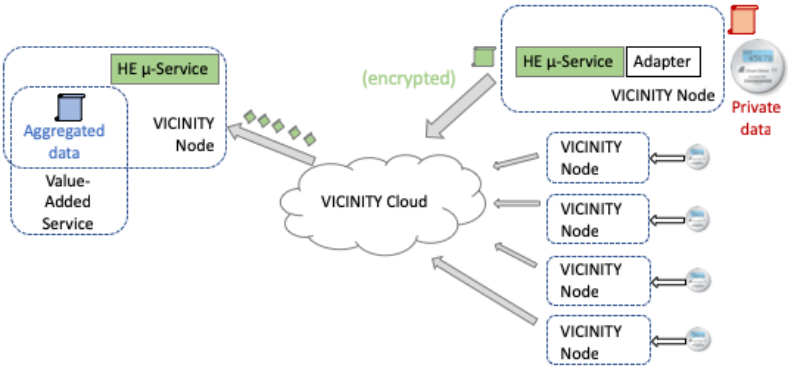

Figure 4.2.: *Homomorphic encryption microservice integrated into the VICINITY infrastructure.*

their attached devices. Yet, data of these duplicated or virtual devices is only available in an encrypted form. To be precise, each data retrieved from these virtual devices is encrypted using a homomorphic encryption scheme. These virtual devices then are made available to the VICINITY Neighbourhood and can be handled and dealt with as any other physical device. Its visibility level can be set and controlled and access to these devices is granted with the existing VICINITY contracts mechanism. Just, the data retrieved from these devices is encrypted and on its own useless to anyone except the owner of the privacy service. Only the owner holds the secret key needed for decryption.

The privacy service is implemented in C++ for multiple reasons. The first and most important one being the available libraries for homomorphic encryption being used. While implementing the different encryption schemes is certainly an option, there exist quite elaborate libraries ready to use, which are used in related research work and which already implement a number of performance improvements. For this reason, re-implementing these schemes was skipped in favor of more elaborate, ready-to-use implementations and libraries. Another reason for a C++ based implementation is the runtime performance. HE introduces an overhead to the computation, which results in increased runtimes. To compensate this as much as possible, C++ was used to generate native and efficient code for any platform used. The privacy service is published and available open source[1].

The chosen approach has multiple benefits. It protects the users private data from malicious intents, as the data itself is encrypted and hence useless to anyone except the user itself. On top of that, it leaves the user in full control. All devices are duplicated so the user can for each device and each VAS decide, whether this particular VAS should have access to the plaintext data (trusted services) or only to the encrypted data (semi-trusted services). All untrusted services will not have access to any data at all. Another added benefit of this approach is, that the user does not need to bother with any complex cryptographic operations. This is taken care of by the privacy service

[1] https://github.com/heinzc/PrivacyService

transparently. This greatly increases the acceptance by the users. Finally, many VAS operate by aggregating given input data first and processing the aggregated results afterwards. Yet, aggregating data can perfectly be done on homomorphically encryted ciphertexts. There is no need for accessing the plaintext information at all. This also increases the acceptance by both the users as well as the VAS providers, since their services are more likely be used if privacy is of no concern.

A couple of different encryption schemes were taken into consideration and further analysed.

4.3.2. Paillier encryption scheme

As a first look into the feasibility of homomorphic encryption for the given use-cases, one of the more basic homomorphic encryption schemes was analysed: The paillier encryption scheme [Pai99]. As described in more detail in section 3.2.2. Paillier is a Partially Homomorphic Encryption (PHE) scheme allowing addition of ciphertexts and thus to calculate e.g. the overall sum of provided input data by any particular device. For this purpose alone, Paillier encryption and hence PHE is sufficient. However, after close discussion with the VICINITY partners and large-scale demo site operators, this was quickly considered not flexible enough. For example partners wanted to scale/weight certain data. So multiplication of ciphertexts with constants and hence at least *somewhat homomorphic encryption* was necessary.

When integrating the Paillier encryption scheme into the privacy service, the *libhcs* library[2] was used.

[2]https://github.com/Tiehuis/libhcs

4.3.3. Brakerski-Gentry-Vaikuntanathan (BGV) scheme

In these use-cases, addition does not suffice. Hence, further investigation was done towards the use of Brakerski-Gentry-Vaikuntanathan (BGV) scheme (see section 3.4.4), which is a *somewhat homomorphic*, RLWE-based, encryption scheme. BGV is implementing performance improvements over the original fully homomorphic encryption by Craig Gentry (see above in section 3.4.4 [BGV11]). This approach gave the flexibility required by the VICINITY partners and demo site operators. Addition and multiplication (to an extend) of ciphertexts is possible using the BGV scheme. Yet, most partners requested working with fixed-point arithmetic. As all encryption schemes up to this point were purely integer based, a workaround was necessary in order to fulfill this request: Simply multiplying any given ciphertext with a scaling factor of e.g. 1000. While this might have worked for some of the use-cases, it was still just a workaround. However, another encryption scheme is available, that roughly speaking utilizes this exact trick: the Cheon-Kim-Kim-Song (CKKS) encryption scheme.

Two different implementations were evaluated and integrated into the privacy service for testing and runtime analysis. At first the *HElib* library[3] was used. As *HElib* did not offer support for CKKS encryption (see next section) at that time, another implementation was done. This time using the library developed and actively maintained by the Microsoft Research Group: Seal (see [22]).

4.3.4. CKKS Encryption scheme

Finally, the demand for an encryption scheme, which enables addition and (to some extent) multiplication of floating point numbers, is clear. This step was taken to make this work feasible in practice. One of the most promising encryption schemes, offering all this is the CKKS encryption scheme (see section 3.4.5). CKKS is yet another scheme based on the hardness of solving RLWE problems. As with BGV, CKKS ciphertexts also no longer encrypt single datapoints. Instead, as with most RLWE schemes, ciphertexts encrypt polynomials of data. This makes this scheme also particularly useful, when implementing more complex functions such as e.g. machine learning algorithms. This will further be explained in [Hee21]

4.4. Accessing and working with Ciphertexts

The general idea of this framework is, to encapsulate all the encryption away from the user. Similiar to a user accessing a website via an encrypted channel without necessarily knowing anything about the encryption taking place, this framework aims to provide similar functionality and abstraction. A user should simply be able to choose between sharing plaintext or encrypted data. Likewise, a VAS-Provider should not be required to know all too many details

[3] https://github.com/homenc/HElib

about the underlying encryption itself. The general idea is to provide means to do basic calculations on ciphertexts via an API of the privacy service. While most VAS developers are very used to working with an REST-Api, not so many of them are experts in the field of cryptography. Still, they should be able to work with the provided API in order to implement their desired VAS.

Yet, in some cases, basic arithmetic might not suffice. For those cases, the privacy service should provide access to the lower level instances of encrypted data in order to perform more complex operations on them. This was e.g. utilized to even do basic machine learning in the form of Support Vector Machine (SVM) classification (see [Hee21]) or add the functionality for Secure Multi-party computation (MPC) (see [Nau20]).

4.4.1. Available API Endpoints

The following is an exhaustive list of Representational State Transfer (REST) API endpoints, which are available on two visibility levels. To the local network, all private and public (/vicinity) endpoints are accessible. This is useful for the implementation of VAS, which need to aggregate or decrypt received data. The public (/vicinity) endpoints are available to any other VICINITY node, which has been granted access to it via the VICINITY Neighbourhood Manager. The private endpoints are not available via VICINITY at all.

- public
  - GET /vicinity/objects:
    Get the thing description of the privacy service. Used by the VICINITY agent. This will return the list of all attached VICINITY objects, as well as an entry for the privacy service itself for accessing its properties.
  - GET /vicinity/objects/<arg>/properties/<arg>:
    Read a property of either an attached VICINITY object or the privacy service itself. Attached objects will offer reading any of the properties of the original objects but with the payload encrypted. The privacy service itself can return either of the following properties: publickey, relinkeys, galoiskeys, trustedparties
  - PUT /vicinity/objects/<arg>/properties/<arg>:
    Write a property of an attached VICINITY object or either of the following properties of the privacy service itself: recrypt, recrypt_svm, hasaccess
  - POST /vicinity/objects/<arg>/actions/<arg>:
    Start an action on either an attached VICINITY object or the privacy service itself
- private
  - POST /encrypt:
    Encrypt a given input value and return the ciphertext

- POST /aggregate:
  Add all provided, encrypted input values and return the ciphertext
- POST /decrypt:
  Decrypt a given ciphertext and return the plaintext value

4.5. Adding support for Multiparty Computation

For the presented framework to support MPC, or better to support the BHC protocol described in section 3.5, multiple requirements need to be met. First and foremost, a direct communication between participants of the BHC protocol needs to be established. VICINITY offers these communication channels via the peer-to-peer network already. Next, a way for each participant to access the public keys of all other participants in the trusted set is mandatory. For all participants to take part in the MPC, the privacy service is required to be present on all participant's VICINITY nodes. Hence, the privacy service needs to provide a public endpoint where the public key can be accessed. Finally, the aggregating VAS needs to be part of the VICINITY network as well and smart contracts with all participants need to be formed via the VICINITY Neighbourhood Manager.

With these prerequisites met, the BHC protocol (see algorithm 9) can be implemented. Details on this implementation can be found in [Nau20]. The BHC protocol is implemented as a plugin which can be added to the privacy service on demand. The privacy service offers to extend its functionality in the form of plugins. These plugins can interface directly with the C++ data structures of the privacy service and can further enhance the functionality of the service. For the BHC protocol, what was necessary was a way to manage the trusted set of each privacy service. As the BHC protocol requires access to the public keys of each trusted participant and a way to send the blinded measurements to this participant, this needs to be done in two steps. First, the trusted privacy services need to be granted access in the neighbourhood manager in order to communicate and send the blinded measurements. Second, the privacy services in return need to know their trusted set in order to correctly encrypt and send the blinded measurements to these respective participants. This is certainly not the most convenient way to solve the issue at hand. However, for the scope of this thesis, this is considered feasible as this process is a one-time setup, which can be done in advance. The privacy services in the trusted set need to be found in the Service section in the VICINITY Neighbourhood Manager and added to the user's own privacy service.

4.6. Setup and Run

One other major acceptance criterion is, that the framework should not only be easy to use, but also easy to setup. As multiple, independent parts need to be run together, starting each of them individually is not the most convenient

way. Additionally, these parts may require additional tools or libraries to be installed in order to execute properly. Hence, the VICINITY framework offers a containerized version of the necessary tools in order to run. All of those together can be executed by a single command and will also restart after each system reboot. It is a logical conclusion to also have a containerized version of the privacy service available, which could also be integrated into the single setup script, so not only the original VICINITY components would start together, but also the privacy service would boot up together with them. For the initial setup, the user or infrastructure operator will be required to give the configuration files. After the initial setup, all further configuration can be done using VICINITYs Neighborhood Manager and hence via a web interface.

Chapter 5

Results

5.1. Experiments

In order to further evaluate the proposed solution, a proof-of-concept was implemented and evaluated with respect to feasibility and runtime behaviour.

VICINITY components alongside the proposed privacy service component were distributed and deployed across different networked nodes. The use-case scenarios were implemented on each node and tested against the real, live deployment over the VICINITY peer-to-peer network. For the following experiments, the privacy service and the used CKKS encryption were configured as shown in listing 5.1

```
void seal_he_handler::initialize()
{
    m_poly_modulus_degree = 8192;
    m_pParms = EncryptionParameters(scheme_type::ckks);
    m_pParms.set_poly_modulus_degree(
                m_poly_modulus_degree
    );
    m_pParms.set_coeff_modulus(CoeffModulus::Create(
                m_poly_modulus_degree,
                { 60, 40, 40, 60 }
    ));
    m_scale = pow(2.0, 40);

    m_pContext = new SEALContext(m_pParms);
    ...
}
```

Listing 5.1: *Privacy service CKKS encryption parameters*

5.2. Runtime analysis of Use-Cases

The proposed solution was tested and evaluated for the two use-cases given in 1.1. The experimental runtime analysis was conducted under lab conditions, yet utilizing the productive and real world VICINITY network deployment. More precisely, data is generated at client nodes, transmitted via the VICINITY network to a VAS-node, where it is evaluated (e.g. aggregated) and the resulting data is transmitted back to the origin for decryption.

5.2.1. Single party computation

In 1.1.1 a use-case is described, where during a workout session, the heart rate of a user is measured by a fitness tracker or smartwatch and sent to a VAS for storage and later evaluation at regular intervals. As this is medical data, it needs to be treated as highly sensitive, private data. For comparison only, measurements were also conducted where cleartext information is transmitted and stored. However, for the sake of keeping such kind of data secure and private, this is not advised.

Online Phase

For each setup and encryption scheme, 10 experiment runs were conducted, each collecting 100 data points from a remote source. In the given scenario, the remote source is a dummy device representing a fitness tracker as described in the use-case 1.1.1. Every 2 seconds, another data point was collected to avoid capturing data in too quick succession, and avoid potential side-effects one query might have on the next. In a real world scenario, this delay may even be longer.

The measurement and data collection code executed on the VAS side is shown in appendix A.2.

Figure 5.1 shows the runtime it took to collect one individual data point, averaged over 100 data points collected, per each experiment run. The first set of experiments is comparing the online time differences of the different approaches and encryption schemes. This does consider the overhead involved in the additional step introduced by adding the privacy service in between and the time it takes to encrypt the data. However, this does not consider the overhead when the collected data is being processed and the results being decrypted at the data owner again.

As can be seen in figure 5.1, there is a clear gap between the average runtimes on cleartext data compared to encrypted data. Figure 5.2 also shows this jump in average runtime. However, the standard deviation is around 3% for all scenarios, so for the online phase evaluated during this experiment, where data is being gathered, the presented approach scales well. For a more detailed look on the individual runtimes, please refer to appendix A.3.

It is also worth mentioning the effect the indiviual components have on the runtime. As seen in Figure 5.2, communicating via the VICINITY network and hence, taking advantage of the data sharing and control VICINITY offers, does

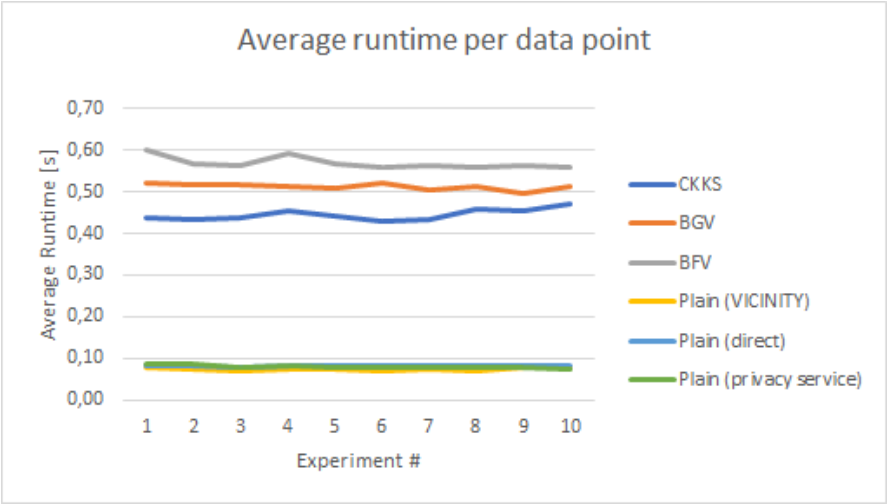

Figure 5.1.: *Average runtime per data point, comparing CKKS, BGV, Brakerski/Fan-Vercauteren (BFV), plaintext (via VICINITY), plaintext (direct access) and plaintext (via privacy service + VICINITY) over 10 experiment runs*

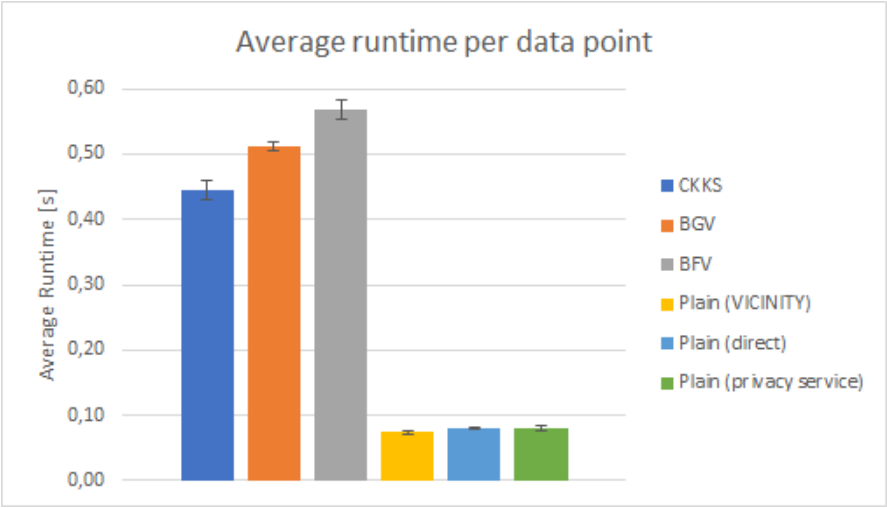

Figure 5.2.: *Average runtime per data point, comparing CKKS, BGV, BFV, plaintext (via VICINITY), plaintext (direct access) and plaintext (via privacy service + VICINITY) on average*

not affect the runtime behaviour at all. In fact, the measurement data shows that using the VICINITY peer-to-peer network for communication is actually faster than accessing the REST-API endpoint of the data provider directly. Given by the measurements, the runtime for plaintext via VICINITY has a standard deviation of around $\approx 3.5\%$, compared to the rather stable plaintext runtime with a standard deviation of only around $\approx 1.08\%$. For the runtime of gathering plaintext information via VICINITY and the privacy service, the standard deviation is even higher with around $\approx 4.45\%$. Generally speaking, these deviations are to be expected when communicating via the internet and all three measurements for gathering plaintext information are fairly equal compared to the big gap when gathering ciphertext information.

The increase in runtime when accessing encrypted data, is by a factor of ≈ 5.5 (CKKS), ≈ 6.42 (BGV) and ≈ 7.13 (BFV) respectively, compared to accessing plaintext data via VICINITY and the privacy service. This increase is mainly due to the overhead for encrypting the plaintext information first and second due to the large increase in payload size, which is transmitted. More on ciphertext sizes is covered in 5.3.2.

Offline Phase

Data collection naturally requires all participants, that is data provider and VAS to be online and connected. Data processing and decryption on the other hand can take place with only one participant being active at the same time. For the given use-case, let us assume that during a workout session, heart rate data is being recorded, transmitted and stored at a VAS (online phase, see 5.2.1).

After the training session, the user is now interested in his/her average heart rate. For the sake of practicality, let us further assume that the data values are stored encrypted, while metadata like timestamps are stored in clear text. Formulating a query in the form of e.g. “give me the mean of all measurements between 6pm and 7pm” can be resolved purely on these metadata and are the same across all scenarios and encryption schemes.

Additionally, the data aggregation and decryption is only necessary for encrypted data. For plaintext data, this is resolved by most databases during the actual query itself and does not require any further processing.

Hence, Figure 5.3 only compares the three fully homomorphic encryption schemes CKKS, BGV and BFV.

Figure 5.3 compares aggregation and decryption of ciphertexts encrypted with CKKS, BGV and BFV with inputs of 10, 20, 50 and 100 data points respectively. As BGV and BFV only allow integer arithmetic on encrypted data, calculating the mean value has to be resolved in clear text. The encrypted sum over all inputs, as well as the number of inputs are sent back to the data owner, who can then decrypt and will have to calculate the mean value as a last calculation on plain text. With CKKS this processing step can also be done on encrypted data. Yet, for better runtime comparison, all experiments were only calculating the encrypted sum over 10, 20, 50 and 100 data points, respectively.

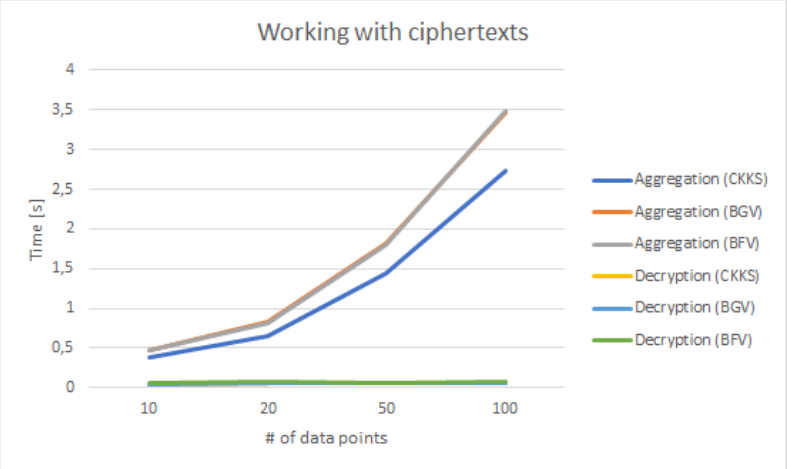

Figure 5.3.: *Comparing aggregation and decryption runtime of CKKS, BGV and BFV*

As expected, no noise is added to the cipher texts by only performing additions. Consequently, the decryption time is constant across all input sizes for all encryption schemes. As was also to be expected, aggregation time increases with increasing input size. All input data is sent from the VAS to the privacy service in one single request. Naturally, the payload size grows linear with the input size, as does the number of homomorphic additions. However, the effect the remaining overhead has on the overall runtime decreases and the aggregation time grows less than linear. While 10 data points took ≈ 0.37 seconds with CKKS encryption, 100 data points only took ≈ 2.73 seconds, increasing only by a factor 7.37. Also, BGV and BFV perform fairly even, while CKKS, the most recent scheme of these three schemes is around ≈ 0.74 seconds faster for an input size of 100.

For a more detailed overview on the measured runtimes, please refer to appendix A.4, appendix A.5, appendix A.6 and appendix A.7, respectively.

5.2.2. Multi party computation

In 1.1.2, a use-case is described, where an energy supplier queries the readings of multiple smart metering devices at regular intervals. As this is private data, it should not be simply collected in plain text. Instead, the MPC protocol as described in 4.5 is used. For this use-case, "the measurements were taken from the aggregation example running on a KDE Neon 5.18 virtual machine using Oracle VM VirtualBox, which ran on six threads of a 2.6 GHz Intel Core i7-9750H and 8 (out of 32) GB 2666 Mhz DDR4 RAM. The host system was Microsoft Windows 10 Education." [Nau20]

These evaluations of the MPC use-case were done in a bachelor thesis conducted in 2021 (see [Nau20]).

The given use-case was evaluated in a setup as shown in Figure 5.4.

There are 3 participating parties, each having their own smart metering

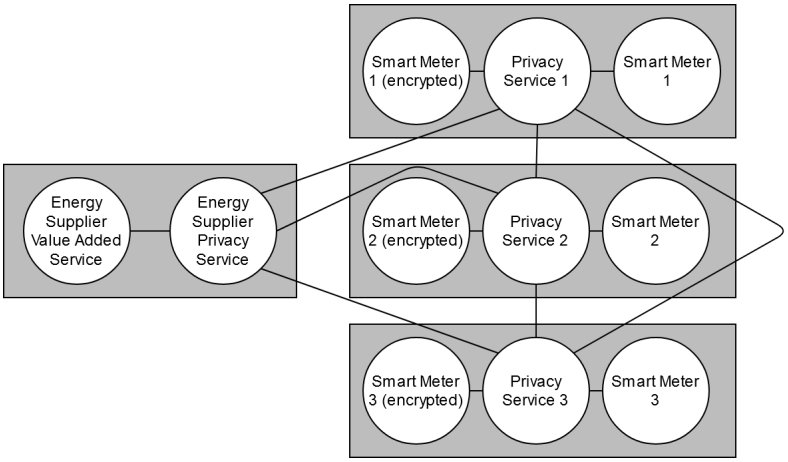

Figure 5.4.: *Test setup used for the MPC use-case experiment [Nau20]*

device, connected to their own privacy service instance and hence having their own encrypted replica of said smart meter, which is shared via VICINITY. The receiving VAS of the Energy supplier has its own instance of the privacy service set up and running. All smart metering devices are shared with the Energy Suppliers VAS via VICINITY. To evaluate the worst-case runtime behaviour, in a sense that the most amount of messages need to be transmitted, all participating parties trust each other.

Figure 5.5 shows the total aggregation time of the described use-case, while using the privacy service integrated into the VICINITY peer-to-peer network.

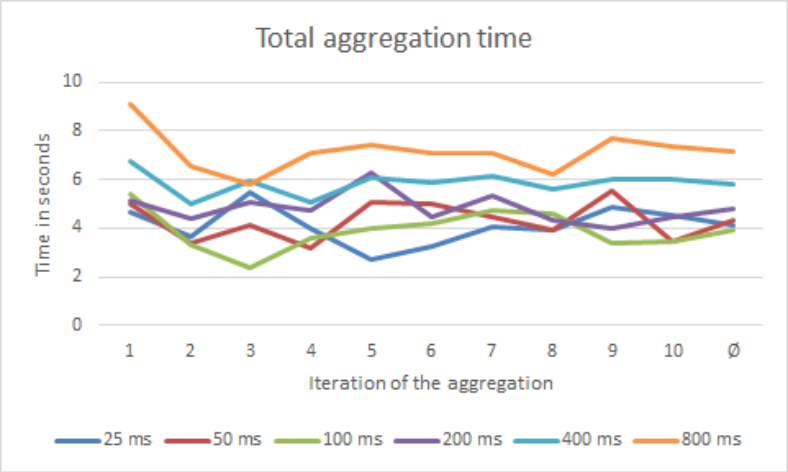

Figure 5.5.: *Total aggregation time of BHC protocol. Comparing different polling times [Nau20]*

The given use-case was implementing a polling mechanism in order to regularly check if a participant has new data available. As for the implemented BHC protocol (see 4.5), a direct communication between parties is manda-

tory to ensure privacy, the event mechanism of VICINITY cannot be used, as this would simply broadcast any available data to all subscribers, regardless if they are trusted for the given protocol or not. After each unsuccessful poll for new data, the requesting party waits for a given sleep duration. Figure 5.5 shows the effect different sleep durations have on the overall aggregation time. Naturally, longer sleep durations result in longer runtimes. With an average of $\approx$ 4 seconds in total runtime, the 100ms sleep approach performs best, as this seems to best match the time it takes to provide the calculated, blinded measurements.

This can be compared directly with Figure 5.6, where the runtime of an unsafe approach is shown.

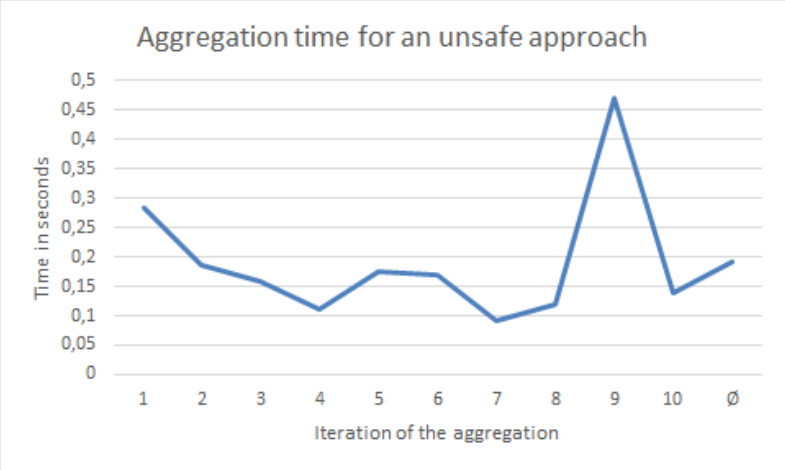

Figure 5.6.: *Aggregation time of unsafe approach [Nau20]*

In this approach, the sensor readings of the smart metering devices can be directly retrieved from the devices themselves. No encryption and no BHC protocol is implemented in this approach. As this reveals private data directly, this approach is not advised. Instead, it is only shown to compare the overhead introduced by the BHC protocol.

With one clear outlier during the experiments, the runtime for this unsafe approach averages around $\approx$ 0.2 seconds. The difference in runtime, and hence the overhead introduced by the BHC protocol is by a factor of 20. This is of course a trade-off between privacy and performance. For sure, this renders the implemented BHC approach infeasible for applications where quick responses are required. However, for the given use-case, a runtime of around $\approx$ 4 seconds is tolerable, as the use-case described in section 1.1.2 is supposed to run this aggregation at intervals of around 15 minutes and the energy grid is not able to switch loads quite as fast.

5.3. System evaluation

5.3.1. Runtime impact

In the following, the effect which using HE has on the overall runtime behaviour is evaluated and compared to the runtime on pure plaintext data.

First looking at the single-party use-case when utilizing the CKKS encryption scheme, the overhead per data point is roughly by a factor of 5.5, when accessing data. This is mainly due to the increase in ciphertext size (see 5.3.2). Retrieving one single data point takes on average around 0.44 seconds if the data is encrypted. For comparison, accessing the same data but in plain text takes on average around 0.08 seconds. If one were to retrieve a full set of for example 100 data points one after the other, this overhead would multiply. Encrypted data would take 44 seconds to transmit, while plaintext data would take around 8 seconds. However, the power of using HE can be utilized here. If all 100 data points are stored by the same VAS, then this VAS could aggregate this data before transmission. On average, aggregation of 100 data points roughly takes 2.73 seconds + 0.44 seconds for the transmission of the aggregated data point. Using this approach, the user would have his result after 3.17 seconds. The latter approach is equally secure, as only aggregated data is transmitted. Yet, compared to the previous approach, this method allows for a speedup of factor 13.8. To be fair, retrieving the 100 data points and only transmitting one aggregated result in plain text would only take 0.08 seconds. This is, if one were to assume the overhead when aggregating plaintext data is negligible. In this case, the privacy aware approach using HE would take around 40 times as long as the unsafe plaintext approach.

Looking at the multi-party use-case shows a similar result. The secure approach performs by a factor of 20 slower than the unsafe approach. As multiple transmissions, encryptions and zero-knowledge proofs are involved in the process, it is hard to tell, which has the largest impact. The unsafe approach does neither of these and simply collects all data in plain text. Still, a total runtime of around 4 seconds in the context of the use-case described in section 1.1.2 can be considered acceptable. Further, physical processes involved when controlling the power grid are way slower. However, for a more real-time oriented application, this approach is not feasible at all.

In summary, there is an overhead larger than one order of magnitude introduced by using the privacy service and utilizing HE. However, understanding the power of HE allows for a significant speedup for equally secure and privacy aware results.

5.3.2. Ciphertext sizes

In the following, the ciphertext sizes will be compared in order to better understand the implications and overhead introduced by using HE over pure plaintext data.

Accessing plaintext data directly, without any additional metadata, obviously yields the smallest payload size. An individual double (floating point)

value in C++ for example makes up for 8 bytes of data. Yet, accessing it via the VICINITY Adapter encapsulates this value into a JSON document, increasing the payload size to 35 bytes. The respective payload is shown below:

```
{
  "value": 97.89672329601301
}
```

This difference in payload size drastically increases, when transmitting values encrypted using HE schemes. Figure 5.7 shows this growth. Note the logarithmic scale.

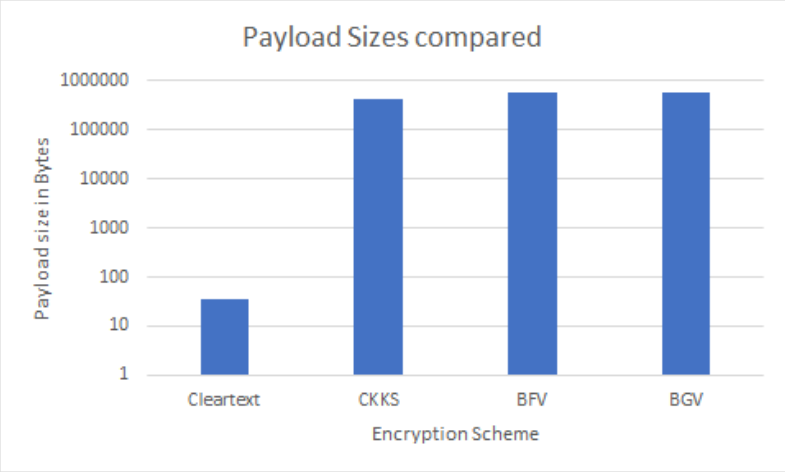

Figure 5.7.: *Payload sizes of HE schemes CKKS, BFV, BGV compared to cleartext*

All HE schemes were implemented using the SEAL library [22]. The respective schemes were set up using the recommended parameters as described in 5.1. Compared to the plain text payload, with all HE schemes, the payload size grows by around 4 orders of magnitude. BFV and BGV, two rather similar and with respect to their Encryption Function rather identical encryption schemes naturally yield exact same payload sizes of around 563.06 KB. CKKS payloads are overall smaller with 435.28 KB, yet compared to the plain text values still 4 orders of magnitude larger. So the overhead from using HE is not only relevant when computing with these ciphertexts. The overhead also plays a role when transmitting data them from source to sink. Figure 5.8 shows the effect different encryption parameters have on the payload sizes under the CKKS encryption scheme.

As already discussed in 3.4.5, changing the poly modulus degree directly affects multiple properties of the encryption. It directly affects the number of coefficients in the plaintext space, the computational performance, the security level and as can be seen in Figure 5.8, also the size of the ciphertext elements. Of course, these parameters can and should be tailored to the respective needs one has towards the intended application. However, the authors of [22] recommend a poly modulus degree of 8192 for most applications as this offers

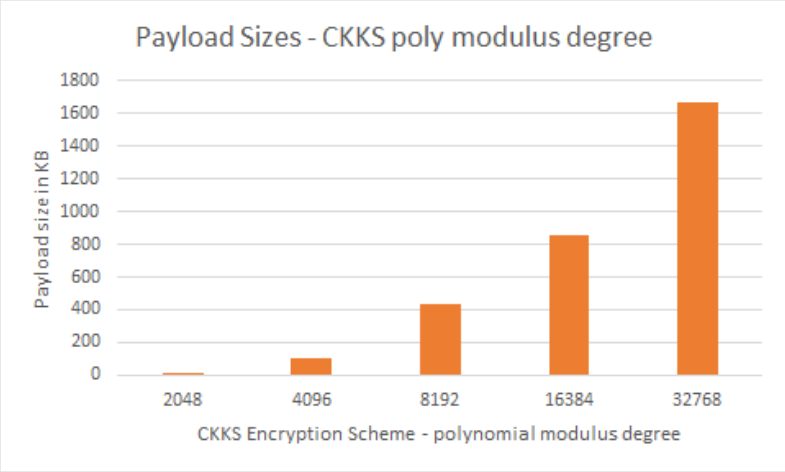

Figure 5.8.: *Payload sizes of CKKS encrypted ciphertexts. Encrypted with varying sizes of poly modulus degrees*

the best trade-off between performance and security. The size comparison supports this claim. Hence, the default value for the privacy service was also selected to be 8192.

5.4. Further Applications and Business Models

The privacy service is set up to be modular. Plugins can easily be integrated to extend its functionality. Without any extensions it covers most common use-cases and applications like collecting primitive data such as plain numbers. Applications for MPC were already developed as a plugin to the privacy service, to show its functionality. The intended use for the privacy service was to simply "plug and play", thus make the existing VICINITY communication even more secure and privacy-aware. No further knowledge of the underlying encryption or HE in general should be necessary. However, developers more experienced in cryptography should still be able to add and implement their own algorithms and functionality. In [Hee21], yet another plugin was developed, this time offering a more sophisticated application for HE. In this work, a proof-of-concept implementation was developed to perform machine learning on HE data. More precisely, the Support Vector Machine classifier was trained on encrypted data and could be used to classify data encrypted under the same encryption context. This approach directly translates to potential business models. While many users might not be willing to share private data with unknown or untrusted VAS, they might very well be willing to, when only encrypted data is shared. For many applications, either an aggregated value will suffice (see 1.1.2) or computation is outsourced onto a cloud server, which is not interested in any plaintext information at all (see 1.1.1). Furthermore, with applications such as machine learning, VAS providers could train their models or offer their proprietary algorithms on encrypted data, while at

the same time not having a hard time complying to the GDPR, due to not accessing unencrypted private data.

Chapter 6

Conclusions

In this thesis, a framework for Privacy in the Internet of Things was described and developed. This framework is using the EU funded project VICINITY (see 2.2) peer-to-peer network for communication. VICINITY was implemented to tackle the problem of semantic interoperability and was built with privacy in mind. The VICINITY infrastructure allows users from different domains to share their IoT assets and information in a privacy-respectful, user-defined way. This enables synergies among services from those domains and enables a new market of domain-crossing services. Of course, when it comes to privacy, the legal boundaries need to be addressed as well. For the european union, these boundaries are defined in the GDPR, which were further elaborated in Section 2.3. Apart from the communication, data handling is yet another important aspect of the framework presented in this thesis. Not only does the data need to be transferred securely. Also, the user wants full control over what is happening to his data, even after it was delivered to e.g. a VAS. HE is being utilized to achieve this goal (see 3). With HE, private data is only transferred fully encrypted. Users can still select to share their data in plain text. Either if this data is not private or if they fully trust the receiving VAS. In any other case, they have the option to only share their private data under a HE scheme. This leaves the user in full control of their private, plaintext data, while still allowing them to e.g. off load expensive computations to the cloud or take advantage of cloud resources or e.g. trained machine learning models offered by a VAS. The concept on how to combine these ideas into one framework is further elaborated in Chapter 4. As not only one HE scheme exists, some of the most recent and popular schemes were analyzed, implemented and evaluated. Additionally, a plugin mechanism is introduced, as it was found necessary to extend the privacy framework in order to add additional features, which require direct access to the ciphertext data structures. One example for such a plugin is the multiparty computation required to achieve the second use-case as described in 1.1.2. Finally, experiments were conducted in 5, proving the feasibility of the developed framework with regards to both use-cases described in 1.1 as well as many other potential applications in the field of IoT. While the application of HE to achieve these

use-cases is possible and feasible in order to protect the users private data, using HE introduces a considerable overhead both in terms of computation and communication. With further knowledge on the underlying encryption scheme, the scheme can be further tweaked and configured to match the intended application. However, the general idea behind the privacy service is, to encapsulate all the encryption and technicality inside the microservice and away from the basic user. More advanced users can however take advantage of the plugin mechanism to further improve the performance of the privacy service for their application.

6.1. Future Works

While the presented framework works as a proof-of-concept, there is room for improvement, some of which is outlined in the following:

- The foundation of the framework presented in this thesis is the VICINITY peer-to-peer network. It has been developed as part of the Horizon 2020 program, funded by the EU. It can be used as it currently is, however it will not be developed and improved any further, as the funding ended in 2020. The peer-to-peer network is hosted and maintained by the VICINITY partners at bAvenir, s.r.o[1]. bAvenir continues to improve the conceptual idea behind VICINITY and continues to update the peer-to-peer network as part of a new EU funded project called "AURORAL"[2]. bAvenir has reached out to me, asking to adapt to and support the AURORAL network as well. While the conceptual idea stays the same, this will result in a breaking change in the API. Hence, this was not yet done as part of this thesis work but is instead considered as a future work to be done.

- One key component of this framework is the privacy service developed in this thesis, allowing the seemless use of HE. HE is still one of the "hot topics" in research, but has not found many practical applications yet. One reason for this is, that homomorphic processing requires orders of magnitude more processing time and hence processing power, which may or may not be feasible depending on the application. The privacy service needs to constantly adapt to advances in these fields in order to be more feasible for everyday applications.

- The privacy service component is meant to work "out of the box" and especially should not require any deeper knowledge on the underlying encryption scheme used. Hence, it comes configured with a set of basic parameters as recommended by the authors of [22]. While this is already suitable for a variety of applications, including the ones presented in 1.1, more advanced users may want to fine-tune these parameters specific to their needs. This might even vary from one VAS to another. The privacy

[1] https://www.bavenir.eu/
[2] https://www.auroral.eu/

service can already handle multiple encryption contexts with different encryption keys. It can hence, also be extended to support different sets of parameters such as e.g. different polynomial modulus degrees or support more levels of multiplication as for example required by the SVM application (see 5.4, [Hee21]). These parameters can either be set when implementing new plugins or online as another set of parameters offered via VICINITY.

- For the same reason as above, currently the privacy service only supports the CKKS encryption scheme, as it covers all applications outlined in 1.1 and offers the best tradeoff between versatility and performance. However, more advanced users might again be interested in selecting more suitable encryption schemes to begin with. E.g. if input and output data are known to be integer values only, the BGV or BFV encryption scheme could be used instead, as they offer exact arithmetic instead of approximate results. Additionally, if it is known in advance, that simply adding or only multiplying ciphertexts will suffice, one could also opt for a PHE scheme to get a better performance as well.

Bibliography

[22] *Microsoft SEAL (release 4.0)*. https://github.com/Microsoft/SEAL. Mar. 2022.

[Ajt98] Miklos Ajtai. "Shortest vector problem in L2 is NP-hard for randomized reductions". In: *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*. STOC '98. New York, NY, USA: Association for Computing Machinery, 1998, pp. 10–19. ISBN: 0897919629. DOI: 10.1145/276698.276705. URL: https://doi.org/10.1145/276698.276705.

[Alb+19] Martin R Albrecht, Melissa Chase, Hao Chen, Jintai Ding, et al. "Homomorphic Encryption Standard". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 939.

[Ama] Amazon. *Amazon Web Services Developer Guide*. URL: https://docs.aws.amazon.com/en_en/iot/latest/developerguide/iot-dg.pdf.

[Ash09] Kevin Ashton. "That 'Internet of Things' Thing". In: *RFID Journal* (2009).

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. "Evaluating 2-DNF Formulas on Ciphertexts". In: *Theory of Cryptography Conference*. 2005.

[BGV11] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. *Fully Homomorphic Encryption without Bootstrapping*. Cryptology ePrint Archive, Report 2011/277. 2011.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) Fully Homomorphic Encryption without Bootstrapping". In: *ACM Trans. Comput. Theory* 6.3 (July 2014). ISSN: 1942-3454. DOI: 10.1145/2633600. URL: https://doi.org/10.1145/2633600.

[BKW00] Avrim Blum, Adam Kalai, and Hal Wasserman. "Noise-Tolerant Learning, the Parity Problem, and the Statistical Query Model". In: *CoRR* (2000). URL: https://arxiv.org/abs/cs/0010022.

[Bra12] Zvika Brakerski. "Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP". In: *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2012 - Volume 7417*. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 868–886. ISBN: 9783642320088. DOI: 10.1007/978-3-642-32009-5{_}50. URL: https://doi.org/10.1007/978-3-642-32009-5_50.

[Bri17] British Standards Institution. *EU General Data Protection Regulation 20 Steps to GDPR compliance - A methodical, systematic and logical approach - a whitepaper*. Tech. rep. British Standards Institution, 2017. URL: https://www.bsigroup.com/LocalFiles/en-GB/CSIR/Resources/Whitepaper/UK-ENGB-CSIR-WP-20-steps-to-GDPR-PDF.pdf.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. "Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages". In: *Annual Cryptology Conference CRYPTO 2011: Advances in Cryptology - CRYPTO 2011*. 2011, pp. 505–524.

[Cas17] Raúl Castro García. *VICINITY D2.2: Detailed Specification of the Semantic Model*. Tech. rep. UPM, 2017. URL: https://www.vicinity2020.eu/vicinity/sites/default/files/documents/vicinity_d2.2_vicinitysemanticmodel_v1.0.pdf.

[Cav+09] Ann Cavoukian et al. "Privacy by design: The 7 foundational principles". In: *Information and Privacy Commissioner of Ontario, Canada* 5 (2009). URL: https://www.iab.org/wp-content/IAB-uploads/2011/03/fred_carter.pdf.

[Che+17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. "Homomorphic Encryption for Arithmetic of Approximate Numbers". In: *Advances in Cryptology – ASIACRYPT 2017*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Cham: Springer International Publishing, 2017, pp. 409–437. ISBN: 978-3-319-70694-8.

[Chi+20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. "TFHE: Fast Fully Homomorphic Encryption Over the Torus". In: *J. Cryptol.* 33.1 (Jan. 2020), pp. 34–91. ISSN: 0933-2790. DOI: 10.1007/s00145-019-09319-x. URL: https://doi.org/10.1007/s00145-019-09319-x.

[CJL12] Hao Chen, Xueqin Jia, and Heng Li. "A brief introduction to iot gateway". In: *IET Conference Publications*. 2012. ISBN: 9781849194709. DOI: 10.1049/cp.2011.0740.

[DA16] Tassos Dimitriou and Mohamad Khattar Awad. "Secure and scalable aggregation in the smart grid resilient against malicious entities". In: *Ad Hoc Networks* 50 (2016), pp. 58–67. ISSN: 1570-8705. DOI: https://doi.org/10.1016/j.adhoc.2016.06.014. URL: https://www.sciencedirect.com/science/article/pii/S1570870516301615.

[Dam+11] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. "Multiparty Computation from Somewhat Homomorphic Encryption". In: *IACR Cryptology ePrint Archive* 2011 (2011), p. 535. DOI: 10.1007/978-3-642-32009-5{_}38.

[Dij+10] van Marten Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. "Fully Homomorphic Encryption over the Integers". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques EUROCRYPT 2010: Advances in Cryptology - EUROCRYPT 2010*. 2010, pp. 24–43.

[Dix+13] Alexander Dix, Gregor Thüsing, Johannes Traut, Laurits Christensen, Federico Etro, Susan Ariel Aaronson, and Rob Maxim. "EU data protection reform: Opportunities and concerns". In: *Intereconomics* 48.5 (2013), pp. 268–285. DOI: 10.1007/s10272-013-0470-y. URL: https://doi.org/10.1007/s10272-013-0470-y.

[DM15] Léo Ducas and Daniele Micciancio. "FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second". In: *Advances in Cryptology – EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 617–640. ISBN: 978-3-662-46800-5.

[Elg85] T Elgamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *IEEE Transactions on Information Theory* 31.4 (1985), pp. 469–472. DOI: 10.1109/TIT.1985.1057074.

[Eura] European Union. *Article 29 Working Party Archives*. URL: http://data.europa.eu/eli/reg/2016/679/2016-05-04.

[Eurb] European Union. *European Data Protection Board*. URL: https://edpb.europa.eu/.

[Eur12] European Union. *CHARTER OF FUNDAMENTAL RIGHTS OF THE EUROPEAN UNION (2012/C 326/02)*. 2012.

[Eur16] European Commission. "REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016". In: *Official Journal of the European Union L119/1* (2016). URL: http://data.europa.eu/eli/reg/2016/679/2016-05-04.

[For16] Forbes. *Roundup Of Internet Of Things Forecasts And Market Estimates*. 2016. URL: https://www.forbes.com/sites/louiscolumbus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2016/#514956e1292d.

[FV12] Junfeng Fan and Frederik Vercauteren. *Somewhat Practical Fully Homomorphic Encryption*. Cryptology ePrint Archive, Paper 2012/144. 2012. URL: https://eprint.iacr.org/2012/144.

[Gen09] Craig Gentry. "A fully homomorphic encryption scheme". PhD thesis. https://crypto.stanford.edu/craig: Stanford University, 2009.

[GH11] Craig Gentry and Shai Halevi. "Implementing Gentry's Fully-Homomorphic Encryption Scheme". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques EUROCRYPT 2011: Advances in Cryptology - EUROCRYPT 2011*. 2011, pp. 129–148.

[Gua+17] Yajuan Guan, Juan C. Vasquez, Josep M. Guerrero, Natalie Samovich, et al. "An open virtual neighbourhood network to connect IoT infrastructures and smart objects - Vicinity: IoT enables interoperability as a service". In: *GIoTS 2017 - Global Internet of Things Summit, Proceedings*. IEEE, June 2017, pp. 1–6. ISBN: 9781509058730. DOI: 10.1109/GIOTS.2017.8016233. URL: http://ieeexplore.ieee.org/document/8016233/.

[Hee21] Hagen Heermann. "Homomorphic Encryption of Machine Learning: A Case Study Application". PhD thesis. Jan. 2021.

[Hei+20] Christopher Heinz, Nigel Wall, Alexander H Wansch, and Christoph Grimm. "Privacy, GDPR, and Homomorphic Encryption". In: *IoT Platforms, Use Cases, Privacy, and Business Models: With Hands-on Examples Based on the VICINITY Platform*. Ed. by Carna Zivkovic, Yajuan Guan, and Christoph Grimm. Cham: Springer International Publishing, 2020, pp. 165–184. ISBN: 978-3-030-45316-9. DOI: 10.1007/978-3-030-45316-9{_}8. URL: https://doi.org/10.1007/978-3-030-45316-9_8.

[HG16] Christopher Heinz and Christoph Grimm. "Vicinity". In: *Digitising the Industry Internet of Things Connecting the Physical, Digital and Virtual Worlds*. Ed. by Ovidiu Vermesan and Peter Friess. River Publishers, 2016. Chap. IoT Platfo, pp. 285–288. ISBN: 9788793379824. DOI: 10.13052/rp-9788793379824. URL: http://riverpublishers.com/dissertations_xml/9788793379824/9788793379824.xml.

[Inf22a] Inferati Inc. *Introduction to the BFV FHE Scheme*. Washington, 2022. URL: https://inferati.azureedge.net/docs/inferati-fhe-bfv.pdf.

[Inf22b] Inferati Inc. *Introduction to the BGV FHE Scheme*. Washington, 2022. URL: https://inferati.azureedge.net/docs/inferati-fhe-bgv.pdf.

[Inf22c] Inferati Inc. *Introduction to the CKKS/HEANN FHE Scheme*. Washington, 2022. URL: https://inferati.azureedge.net/docs/inferati-fhe-ckks.pdf.

[Kol+18] Johannes Kolsch, Christopher Heinz, Sebastian Schumb, and Christoph Grimm. "Hardware-in-the-loop simulation for Internet of Things scenarios". In: *2018 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*. IEEE, Apr. 2018, pp. 1–6. ISBN: 978-1-5386-4103-3. DOI: 10.1109/MSCPES.2018.8405399. URL: https://ieeexplore.ieee.org/document/8405399/.

[Kol+19] J. Kolsch, A. Ratzke, C. Grimm, C. Heinz, and G. Nandagopal. "Simulation based validation of a smart energy use case with homomorphic encryption". In: *Proceedings - 15th Annual International Conference on Distributed Computing in Sensor Systems, DCOSS 2019*. 2019. ISBN: 9781728105703. DOI: 10.1109/DCOSS.2019.00063.

[Köl+19] Johannes Kölsch, Christopher Heinz, Axel Ratzke, and Christoph Grimm. "Simulation-Based Performance Validation of Homomorphic Encryption Algorithms in the Internet of Things". In: *Future Internet* 11.10 (2019). ISSN: 1999-5903. DOI: 10.3390/fi11100218. URL: https://www.mdpi.com/1999-5903/11/10/218.

[Lip+20] C. Lipps, S.B. Mallikarjun, M. Strufe, C. Heinz, C. Grimm, and H.D. Schotten. "Keep private networks private: Secure channel-pufs, and physical layer security by linear regression enhanced channel profiles". In: *Proceedings - 2020 3rd International Conference on Data Intelligence and Security, ICDIS 2020*. 2020. ISBN: 9781728193793. DOI: 10.1109/ICDIS50059.2020.00019.

[LK16] H Lee and A Kobsa. "Understanding user privacy in Internet of Things environments". In: *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. 2016, pp. 407–412. DOI: 10.1109/WF-IoT.2016.7845392.

[LPR12] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. *On Ideal Lattices and Learning with Errors Over Rings*. Cryptology ePrint Archive, Paper 2012/230. 2012. URL: https://eprint.iacr.org/2012/230.

[Mic] Microsoft. *Microsoft Azure IoT Hub*. URL: https://docs.microsoft.com/de-de/azure/iot-hub/about-iot-hub.

[MR09] Daniele Micciancio and Oded Regev. "Lattice-based Cryptography". In: *Post-Quantum Cryptography*. Ed. by Daniel J Bernstein, Johannes Buchmann, and Erik Dahmen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 147–191. ISBN: 978-3-540-88702-7. DOI: 10.1007/978-3-540-88702-7{_}5. URL: https://doi.org/10.1007/978-3-540-88702-7_5.

[Myn+17] A Mynzhasova, C Radojicic, C Heinz, J Kölsch, C Grimm, J Rico, K Dickerson, R García-Castro, and V Oravec. "Drivers, standards and platforms for the IoT: Towards a digital VICINITY". In: *2017 Intelligent Systems Conference (IntelliSys)*. Sept. 2017, pp. 170–176. DOI: 10.1109/IntelliSys.2017.8324287.

[Nau20] Dennis Naumann. "Automatic Key Distribution and Management in Secure Single- and Multi-Party Computation". PhD thesis. July 2020.

[Ora17] Viktor Oravec. *D1.6: VICINITY Architectural Design*. Tech. rep. 2017. URL: https://www.vicinity2020.eu/vicinity/content/d16-vicinityd16architecturaldesign10.

[Pai99] Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". In: *Advances in Cryptology — EUROCRYPT '99*. Ed. by Jacques Stern. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238. ISBN: 978-3-540-48910-8.

[Reg10] Oded Regev. "The Learning with Errors Problem (Invited Survey)". In: *2010 IEEE 25th Annual Conference on Computational Complexity*. 2010, pp. 191–204. DOI: 10.1109/CCC.2010.26.

[RSA78] R L Rivest, A Shamir, and L Adleman. "A Method for Obtaining Digital Signatures and Public-key Cryptosystems". In: *Commun. ACM* 21.2 (1978), pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342. URL: http://doi.acm.org/10.1145/359340.359342.

[Sha+17] Hossein Shafagh, Anwar Hithnawi, Lukas Burkhalter, Pascal Fischli, and Simon Duquennoy. "Secure Sharing of Partially Homomorphic Encrypted IoT Data". In: *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. SenSys '17. New York, NY, USA: ACM, 2017, 29:1–29:14. ISBN: 978-1-4503-5459-2. DOI: 10.1145/3131672.3131697. URL: http://doi.acm.org/10.1145/3131672.3131697.

[Sha79] Adi Shamir. "How to Share a Secret". In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782. DOI: 10.1145/359168.359176. URL: https://doi.org/10.1145/359168.359176.

[SYY99] Tomas Sander, Adam Young, and Moti Yung. "Non-Interactive CryptoComputing For NC1". In: *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*. FOCS '99. USA: IEEE Computer Society, 1999, p. 554. ISBN: 0769504094.

[TT17] Tikkinen-Piri Rohunen and Markkula Tikkinen-Piri. *The EU General Data Protection Regulation: Changes and implications for personal data collecting companies*. 2017.

[Uni12] International Telecommunication Union. "Overview of the Internet of Things". In: *Recommendation ITU-T Y.2060* (2012).

[Uni48] United Nations. *Universal Declaration on Human Rights (UDHR)*. 1948. URL: http://www.ohchr.org/EN/UDHR/Documents/UDHR_Translations/eng.pdf.

[Uwi+20] Marie Uwiringiyimana, Gomathi Nandagopal, Yajuan Guan, Saso Vinkovic, Johannes Kölsch, and Christopher Heinz. "IoT Platforms". In: *IoT Platforms, Use Cases, Privacy, and Business Models: With Hands-on Examples Based on the VICINITY Platform*. Ed. by Carna Zivkovic, Yajuan Guan, and Christoph Grimm. Cham: Springer International Publishing, 2020, pp. 21–49. ISBN: 978-3-030-45316-9. DOI: 10.1007/978-3-030-45316-9{_}8. URL: https://doi.org/10.1007/978-3-030-45316-9_8.

[VIC20a] VICINITY. "VICINITY ontology model". In: (2020). URL: http://vicinity.iot.linkeddata.es/vicinity/.

[VIC20b] VICINITY. *VICINITY Open Gateway API — VICINITY Get Started 0.6.2 documentation*. 2020. URL: https://vicinity-get-started.readthedocs.io/en/latest/open-gateway-api.html.

[XKC16] XKCD.com. *How standards proliferate*. 2016. URL: http://xkcd.com/927/.

[Zhu+10] Qian Zhu, Ruicong Wang, Qi Chen, Yan Liu, and Weijun Qin. "IOT gateway: Bridging wireless sensor networks into Internet of Things". In: *Proceedings - IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, EUC 2010*. 2010. ISBN: 9780769543222. DOI: 10.1109/EUC.2010.58.

Appendix A

Additional Information

Contents

A.1. VICINITY Adapter Thing Description (plaintext)

```
{
  "adapter-id": "debugging-adapter",
  "thing-descriptions": [
    {
      "oid": "debug",
      "name": "Debug_Dummy_device",
      "type": "adapters:ActivityTracker",
      "properties": [
        {
          "pid": "test",
          "monitors": "adapters:HeartRate",
          "read_link": {
   "href": "/objects/{oid}/properties/{pid}",
   "output": {
```

```
 "type": "object",
 "field": [
   {
     "name": "value",
     "schema": {
   "type": "integer"
     }
   }
 ]
   }
           }
         }
       ],
       "actions": [],
       "events": []
    }
  ]
}
```

Listing A.1: *VICINITY Adapter Thing Description (plaintext)*

A.2. Measurement code for runtime analysis

```
if __name__ == '__main__':
  print("starting measurement against api: "
    + getLocalDataUrl)
  startTime = time.time()

  ciphertexts = []

  for x in range(iterations):
    lastExecutionTime = time.time()
    ciphertexts.append(get_data_request())
    currentRequestTime = time.time()
      - lastExecutionTime

    print("execution #" + str(x) + " took: "
      + str(currentRequestTime) + " seconds.")
    runtimeExecutionTime += currentRequestTime
    mintime = min(mintime, currentRequestTime)
    maxtime = max(maxtime, currentRequestTime)
    print(str(mintime) + ", " + str(maxtime))

    time.sleep(sleepTime)

  executionTime = (time.time() - startTime)

  print("execution time over " + str(iterations)
    + " iterations took:")
  print("sum of execution times: "
    + str(runtimeExecutionTime))
  print("maxtime: " + str(maxtime))
  print("mintime: " + str(mintime))

  print("got " + str(len(ciphertexts))
    + " elements")

  for x in range(10):
    aggregationStartTime = time.time()
    aggregatedResult = aggregate_request(ciphertexts)
    aggregationTime = (time.time()
      - aggregationStartTime)
    print("aggregation took: " + str(aggregationTime))

    decryptionStartTime = time.time()
    decryptedResult = decrypt_request(aggregatedResult)
    decryptionTime = (time.time()
```

```
    - decryptionStartTime)
print("decryption took: "
    + str(decryptionTime))
print("decryption result is: "
    + str(decryptedResult))
```

Listing A.2: *Measurement code for runtime analysis*

A.3. Measured runtimes by encryption scheme (online)

CKKS	BGV	BFV	Plain (VICINITY)	Plain (direct)	Plain (privacy service)
0,4365896988	0,519484041	0,600261598	0,076858182	0,08001631	0,086971231
0,4312808347	0,518105934	0,566894836	0,074668088	0,080316658	0,084945705
0,4389970350	0,515084598	0,561817896	0,071236942	0,079759324	0,079856346
0,4552111006	0,513821862	0,592413757	0,073943882	0,08119441	0,080637906
0,4423433018	0,50899437	0,566130021	0,072020667	0,080639954	0,078151245
0,4276945114	0,519773726	0,558341527	0,070831175	0,081733365	0,079483669
0,4315199780	0,504955039	0,561238112	0,07327507	0,081805043	0,078082263
0,4584593773	0,513616617	0,559093053	0,071111636	0,081253612	0,076893263
0,4551087451	0,497216341	0,561158741	0,077086868	0,080828061	0,078463225
0,4726392674	0,51206682	0,559859285	0,078336244	0,082810292	0,074153371

Measured runtimes by encryption scheme (online)

A.4. Measured runtimes for aggregation and decryption by encryption scheme - 10 data points

Aggregation (CKKS)	Aggregation (BGV)	Aggregation (BFV)	Decryption (CKKS)	Decryption (BGV)	Decryption (BFV)
0,3827362061	0,4745726585	0,4640419483	0,0579936504	0,0655126572	0,0665109158
0,3799862862	0,4735765457	0,4807050228	0,0555074215	0,0645096302	0,0660035610
0,3795316219	0,4729459286	0,4807422161	0,0565137863	0,0650000572	0,0685124397
0,3840501308	0,4835779667	0,4794740677	0,0569911003	0,0655236244	0,0669991970
0,3710627556	0,4815897942	0,4722588062	0,0565071106	0,0660037994	0,0685172081
0,3870725632	0,4755690098	0,4716520309	0,0569851398	0,0675187111	0,0685040951
0,3720898628	0,4821469784	0,4703989029	0,0550103188	0,0660035610	0,0705194473
0,3781108856	0,4765751362	0,4686412811	0,0565080643	0,0665156841	0,0675139427
0,3860423565	0,4845759869	0,4712419510	0,0559971333	0,0649986267	0,0689871311
0,3710434437	0,4696309566	0,4680991173	0,0560050011	0,0665237904	0,0655117035

Measured runtimes for aggregation and decryption by encryption scheme - 10 data points

A.5. Measured runtimes for aggregation and decryption by encryption scheme - 20 data points

Aggregation (CKKS)	Aggregation (BGV)	Aggregation (BFV)	Decryption (CKKS)	Decryption (BGV)	Decryption (BFV)
0,6416051388	0,8092658520	0,8126604557	0,0579993725	0,0655038357	0,0699810982
0,6472182274	0,8301353455	0,8431894779	0,0559976101	0,0670950413	0,0690033436
0,6737632751	0,8559010029	0,8107964993	0,0565032959	0,0660016537	0,0690088272
0,6649792194	0,8260633945	0,8037495613	0,0585153103	0,0660061836	0,0665047169
0,6466591358	0,8299620152	0,8061382771	0,0575172901	0,0655038357	0,0675044060
0,6536154747	0,8013324738	0,8085455894	0,0580675602	0,0655016899	0,0670101643
0,6531352997	0,8041160107	0,8333351612	0,0569989681	0,0649788380	0,0669941902
0,6432332993	0,8494830132	0,8371577263	0,0559985638	0,0675179958	0,0690047741
0,6457927227	0,7946033478	0,8056628704	0,0559997559	0,0655124187	0,0669887066
0,6398415565	0,8219733238	0,8302993774	0,0580053329	0,0665106773	0,0705683231

Measured runtimes for aggregation and decryption by encryption scheme - 20 data points

A.6. Measured runtimes for aggregation and decryption by encryption scheme - 50 data points

Aggregation (CKKS)	Aggregation (BGV)	Aggregation (BFV)	Decryption (CKKS)	Decryption (BGV)	Decryption (BFV)
1,4042599201	1,8651158810	1,8254659176	0,0570003986	0,0675184727	0,0655097961
1,4617352486	1,7967920303	1,8022961617	0,0580036640	0,0660068989	0,0675261021
1,4088478088	1,8236346245	1,8121743202	0,0580596924	0,0650031567	0,0665152073
1,4174094200	1,7898352146	1,7861704826	0,0585188866	0,0655171871	0,0665137768
1,4486465454	1,8121368885	1,7973458767	0,0595188141	0,0665295124	0,0665168762
1,4360585213	1,8019125462	1,8003411293	0,0585269928	0,0660047531	0,0670030117
1,4875986576	1,8390345573	1,7821390629	0,0589990616	0,0650029182	0,0665328503
1,4727325439	1,8074474335	1,7883889675	0,0585300922	0,0675177574	0,0675168037
1,4577770233	1,8082153797	1,7797145844	0,0575144291	0,0655138493	0,0661487579
1,4990565777	1,7966315746	1,8495554924	0,0579984188	0,0639994144	0,0685074329

Measured runtimes for aggregation and decryption by encryption scheme - 50 data points

A.7. Measured runtimes for aggregation and decryption by encryption scheme - 100 data points

Aggregation (CKKS)	Aggregation (BGV)	Aggregation (BFV)	Decryption (CKKS)	Decryption (BGV)	Decryption (BFV)
2,720429897 3	3,448805093 8	3,512964248 7	0,058597803 1	0,067516565 3	0,072003603 0
2,758008003 2	3,477398157 1	3,529965639 1	0,062629938 1	0,068508148 2	0,068004131 3
2,742797374 7	3,485898256 3	3,491895437 2	0,059662342 1	0,069514989 9	0,070403337 5
2,728089332 6	3,486355781 6	3,493015050 9	0,060609102 2	0,067514419 6	0,068518161 8
2,715174913 4	3,451318740 8	3,486349582 7	0,062119960 8	0,067995309 8	0,070516824 7
2,769376277 9	3,477206468 6	3,507486581 8	0,061074495 3	0,067513465 9	0,067518949 5
2,705862999 0	3,475531339 6	3,499540567 4	0,061003923 4	0,068005561 8	0,070506811 1
2,718818903 0	3,483399629 6	3,464580297 5	0,058525562 3	0,068505048 8	0,069509029 4
2,765114545 8	3,479212284 1	3,488577365 9	0,060003757 5	0,066995143 9	0,067519187 9
2,674051284 8	3,466830730 4	3,468148231 5	0,059514761 0	0,068523407 0	0,066514968 9

Measured runtimes for aggregation and decryption by encryption scheme - 100 data points

Appendix B

Curriculum Vitae

Work Experience

03/2024 until today	**SKS Welding Systems GmbH** Software Engineer
04/2021 – 02/2024	**frogblue Technology GmbH** Software Developer (Application Development)
10/2015 – 03/2021	**Technische Universität Kaiserslautern** Researcher, Department of Computer Science, Chair for *Development of Cyber-Physical Systems*
11/2012 – 08/2015	**Fraunhofer Institut für Techno- und Wirtschaftsmathematik (ITWM)** Research Assistant
05/2010 – 06/2012	**Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)** Research Assistant

Education

10/2012 – 07/2015	**Master of Science (Applied Computer Science)** Ø2,0 Technischen Universität Kaiserslautern Thesis: *Simulation Data Collection Configuration and Processing.*
04/2009 – 06/2012	**Bachelor of Science (Applied Computer Science)** Ø2,7 Technischen Universität Kaiserslautern Thesis: *Implementation of a Feature Extraction Routine for Local Image Features on the Graphics Computing Device.*
04/2008 – 03/2009	**Bachelor Studies Applied Computer Science** Technischen Universität Kaiserslautern