TU WIEN Informatics

Urban Sensing Environments: Exploiting the Adaptation Space for Data Protection

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

eingereicht von

Dipl.-Ing. Clemens Lachner, BSc.
Matrikelnummer 00403661

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dr. Schahram Dustdar

Diese Dissertation haben begutachtet:

Harald Gall

Omer Rana

Wien, 19. August 2022

Clemens Lachner

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Urban Sensing Environments: Exploiting the Adaptation Space for Data Protection

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Dipl.-Ing. Clemens Lachner, BSc.
Registration Number 00403661

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr. Schahram Dustdar

The dissertation has been reviewed by:

Harald Gall	Omer Rana

Vienna, 19th August, 2022

Clemens Lachner

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Dipl.-Ing. Clemens Lachner, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 19. August 2022

Clemens Lachner

Acknowledgements

I owe thanks to numerous people whose involvements in this thesis I want to acknowledge. I want to thank Schahram Dustdar who supervised my thesis and provided me with lots of valuable information and advice that where not only viable to my research but also to my own personal growth. Additionally, he gave me the flexibility needed to manage family, sports and the pursue of my PhD. Furthermore, I want to thank Thomas Rausch, Pantelis Frangoudis and Christos Tsigkanos for the many discussions and valuable feedback regarding my research. I also want to thank Harald Gall and Omer Rana for reviewing this thesis. Finally, I want to thank my family for their great support. Their patience and tolerance, as well as their support with my kids, heavily contributed to make this PhD happen. My greatest thanks go to my three kids, Fabian, Felix and Florian, who constantly remind me of what really matters in life.

The research presented in this thesis was funded by TU Wien research funds, as well as the European Union's Horizon 2020 research and innovation programme under grant agreement no. 871525 (FogProtect).

Kurzfassung

Urbane Sensorik ist die Grundlage für intelligente Städte. Es umfasst mehrere Domänen wie öffentliche Überwachung, intelligente Gebäude, Smart Health, Crowdsourcing/Sensing oder Umweltüberwachung. Aus jedem dieser Bereiche stammen digitale Dienste, die im Idealfall die allgemeine Lebensqualität der Bürger verbessern. Prominente Beispiele für solche Dienste sind z.B. die ferngesteuerte Klimatisierung eines intelligenten Zuhauses, Lichtsteuerung in Gebäuden auf der Grundlage von Belegungserkennung, effektive Navigation von Einsatzfahrzeugen durch Verkehrsüberwachungsdaten, automatische PH-Steuerung für städtische Wasserquellen oder groß angelegte Bewegungsmusteranalyse der Bürger, um nur einige zu nennen. Das Edge-Computing-Paradigma spielt eine wichtige Rolle in der urbanen Sensorik, da es die Entwicklung, den Betrieb und die Optimierung solcher Dienste erleichtert. Reduzierte Latenz, Skalierbarkeit, Bandbreitenentlastung oder hohe Zuverlässigkeit sind herausragende Vorteile von Edge-Computing-basierten Diensten. Viele dieser Dienste verarbeiten, übertragen oder speichern jedoch sensible Daten. Die oft begrenzten Ressourcen im Edge-Computing stellen kritische Herausforderungen an die Datenschutzaspekte solcher Dienste. Adaption ist ein wichtiger Faktor, um viele dieser Herausforderungen abzumildern und zu bewältigen. Basierend auf der klassischen Definition der Steuerungstheorie überwacht ein adaptives System seine eigene Leistung und passt seine Parameter in Richtung einer besseren Leistung an. Im Rahmen des Datenschutzes umfasst dies i) die Überwachung des Systems und seiner Umgebung, ii) die Analyse, ob Änderungen die Erfüllung von Sicherheits- und Datenschutzanforderungen gefährden, und iii) die Planung und Durchführung von Anpassungen, falls erforderlich, um den Fortbestand zu gewährleisten das diese Anforderungen befriedigt werden. In dieser Arbeit nutzen wir den Anpassungsraum, um den Datenschutz in ressourcenbeschränkten urbanen Sensorumgebungen zu verbessern oder zu erweitern. Zunächst stellen wir ein Systemmodell vor, das die Grundlage für ein adaptives urbanes Sensorsystem bildet, welches sich an irgendeine Form von Datenschutzbestimmungen halten muss. Das Modell konzentriert sich auf Datenschutzaspekte, wie z. B. eine feingranulare Zugriffskontrolle, und unterstützt die Definition von Datenschutzrichtlinien und deren Umsetzung innerhalb des Systems. Zweitens haben wir mehrere Datenschutzmechanismen evaluiert, indem wir ihre Leistung und ihren Energieverbrauch auf repräsentativen Edge-Geräten gemessen haben. Die evaluierten Datenschutzmechanismen umfassen kryptografische Block- und Stream-Chiffren, sichere Hash-Algorithmen, digitale Signaturalgorithmen und Algorithmen, die für Schlüsselaustauschprotokolle benötigt werden. Basierend auf

den Ergebnissen haben wir ein Source Location Privacy (SLP)-System entwickelt, das speziell für den Betrieb in Umgebungen mit eingeschränkten Ressourcen entwickelt wurde. Drittens untersuchten wir Datenschutzbedenken im Bereich der KI-gestützten öffentlichen Überwachung und wie man ihnen begegnet. Insbesondere interessieren wir uns für Video Analysis Pipeline (VAP) und die Herausforderungen an den Datenschutz, mit denen solche Systeme konfrontiert sind, i.e. das Durchsickern von personenbezogenen Daten aus aufgezeichneten und übertragenen Videodaten. Als letzten Punkt haben wir eine datenschutzorientierte Anpassungs-Engine für verteilte Videoanalyse-Pipelines entwickelt. Es verwendet ein erweitertes Systemmodell und Anpassungsregeln, um die Anforderungen von KI-unterstützten VAPs zu erfüllen. Darüber hinaus verfügt es über einen Optimierungsalgorithmus zur Verbesserung der Leistung, des Energieverbrauchs und des Datenschutzes eines verteilten VAP und seiner Funktionalitäten.

Abstract

Urban sensing is the foundation for cities to become *smart*. It comprises multiple domains, such as public surveillance, smart buildings, smart health, crowd sourcing/sensing or environment monitoring. From each of those domains stem digital services that, ideally, elevate the overall life quality of citizens. Prominent examples of such services are e.g., remote climate control of a smart home, light control in buildings based on occupancy sensing, effective routing of emergency vehicles through traffic monitoring data, automatic PH-control for urban water sources, or large scale movement pattern analysis of citizens, just to name a few. The edge computing paradigm plays a major role in urban sensing, as it facilitates the development, operation and optimization of such services. Reduced latency, scalability, bandwidth relief, or reliability are prominent advantages of edge computing based services. However, many of those services process, transmit or store sensitive data. The often constrained resources in edge computing pose critical challenges to data protection aspects of such services. Adaptation is a key enabler to attenuate and deal with many of those challenges. Based on the classical definition of control theory an adaptive system monitors its own performance and adjusts its parameters in the direction of better performance. In the context of protection of data, this involves i) monitoring the system and its environment, ii) analyzing whether changes threaten the satisfaction of security and privacy requirements, and iii) the planning and execution of adaptations, if needed, to ensure the continued satisfaction of these requirements. In this thesis, we exploit the adaptation space to improve or enhance data protection in resource constrained urban sensing environments. First, we present a system model that builds the foundation of an adaptive urban sensing system that has to adhere to some form of data protection regulation. The model focuses on data protection aspects, such as fine grained access control, and supports the definition of privacy policies and how to enact them inside the system. Second, we evaluated several data protection mechanisms by measuring their performance and energy consumption on representative edge devices. The evaluated data protection mechanism include cryptographic block and stream ciphers, secure hashing algorithms, digital signature algorithms, and algorithms needed for key exchange protocols. Based on the evaluation results, we developed a Source Location Privacy (SLP) system, specifically designed to operate in resource constraint environments. Third, we investigated data protection concerns, and how to address them, in the domain of AI-assisted public surveillance. Specifically, we are interested in Video Analysis Pipeline (VAP) and the challenges to data protection such systems are

confronted with, i.e., leakage of Personally Identifiable Information (PII) from recorded and transmitted video data. Lastly, we developed a data protection focused adaptation engine for distributed video analysis pipelines. It employs an extended system model and adaptation rules to meet the requirements of AI-assisted VAPs at the Edge. Furthermore, it features an optimization algorithm to improve performance, energy consumption and data protection of a distributed VAP and its functionalities.

Contents

Publications

The research presented in this thesis is partly based on the following peer-reviewed publications (i.e., journals, conferences, and book chapters). A full list of publications can be found on Google Scholar[1]

- C. Lachner, T. Rausch and S. Dustdar, "Context-Aware Enforcement of Privacy Policies in Edge Computing," *2019 IEEE International Congress on Big Data (BigDataCongress)*, 2019, pp. 1-6, doi: 10.1109/BigDataCongress.2019.00014.
- C. Lachner and S. Dustdar, "A Performance Evaluation of Data Protection Mechanisms for Resource Constrained IoT Devices," *2019 IEEE International Conference on Fog Computing (ICFC)*, 2019, pp. 47-52, doi: 10.1109/ICFC.2019.00015.
- C. Lachner, T. Rausch and S. Dustdar, "ORIOT: A Source Location Privacy System for Resource Constrained IoT Devices," *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1-6, doi: 10.1109/GLOBECOM38437.2019.9013348.
- C. Lachner, T. Rausch and S. Dustdar, "A Privacy Preserving System for AI-assisted Video Analytics," *2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC)*, 2021, pp. 74-78, doi: 10.1109/ICFEC51620.2021.00018.
- C. Lachner, Z. Á. Mann and S. Dustdar, "Towards Understanding the Adaptation Space of AI-Assisted Data Protection for Video Analytics at the Edge," *2021 IEEE 41st International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2021, pp. 7-12, doi: 10.1109/ICDCSW53096.2021.00009.
- C. Lachner, J. Laufer, S. Dustdar, K. Pohl, "A Data Protection Focused Adaptation Engine for Distributed Video Analytics Pipelines," *2022, IEEE Access 10, 68669-68685*

[1] https://scholar.google.at/citations?user=QL11BZ4AAAAJ&hl=en

CHAPTER 1

Introduction

Urban sensing is the key foundation for cities to become *smart*. It comprises multiple domains, such as public surveillance, environment monitoring, smart health, crowd sourcing/sensing or smart buildings, as depicted in Fig. 1.1. Each of those domains offers services that, ideally, elevate the overall life quality of citizens. Prominent examples of such services are e.g., remote climate control of a smart home, light control in buildings based on occupancy sensing, effective routing of emergency vehicles through traffic monitoring data, automatic PH-control for urban water sources, or large scale movement pattern analysis of citizens, just to name a few. The edge computing paradigm plays a major role in urban sensing. Depending on the definition, Internet of Things (IoT) and Wireless Sensor Network (WSN) can be seen as an infrastructural part of edge computing. In this thesis, IoT and WSN are treated as subsets of edge computing, thus allowing for finer grained categorization if needed.

Typically, edge computing comprises a variety of different connected devices with minimal to average computing power. Fig. 1.2 depicts exemplary compute hardware, which we classify into three different tiers, based on their capabilities. In this thesis, we focus on domains and respective use cases employing low tier to mid devices. These devices continue to permeate deeper into our personal environment as well as in commercial and industrial areas, by sensing, processing, and storing all kind of data [VF13a]. Releasing data to centralized services is especially problematic for systems that handle sensitive data, such as video data in public surveillance or patient data in e-Health systems, as this loss of control can hinder data management workflows from complying to privacy policies [DJP11, LRD19a] such as the General Data Protection Regulation (GDPR) or Health Insurance Portability and Accountability Act (HIPPA). For many applications, like in healthcare, home automation or infrastructure monitoring, these circumstances call for privacy and security protection [DWK15a]. Integrity, confidentiality, availability, undetectability, and unobserveability are the key elements of such protection mechanisms. The overall risk that such a system is confronted with could be assessed by looking on

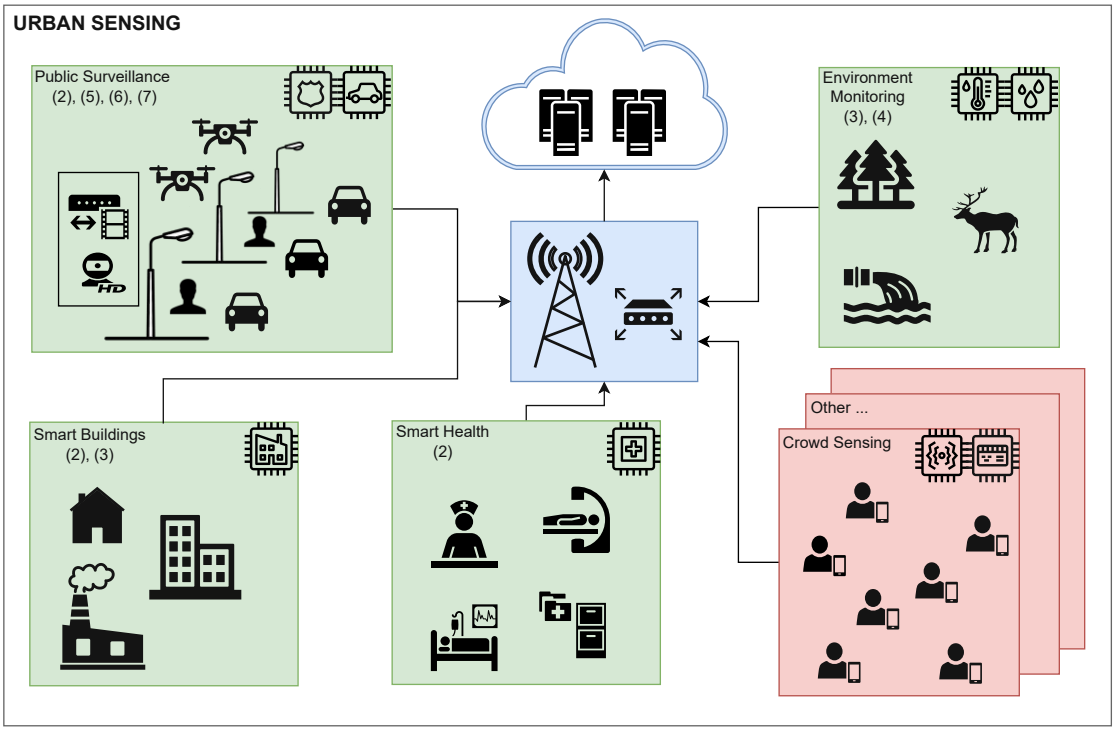

Figure 1.1: The Different Domains in Urban Sensing

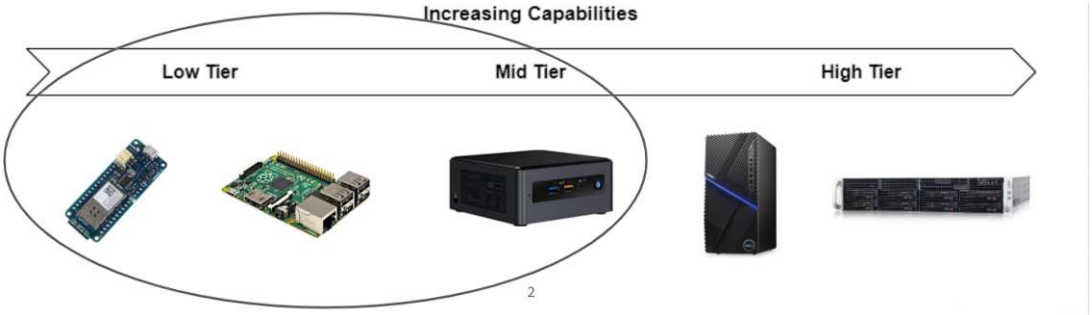

Figure 1.2: Device Categorization into Tiers based on its Capabilities

its various attack surfaces that are exposed to an adversary. For example, characteristic threats to the confidentiality of a system are spoofing attacks, message altering, replay or flooding attacks, just to name a few [FWKM15]. In general, data protection is a critical non-functional aspect of edge computing based systems, and remains an active area of research. It has to be noted, that the term data protection is not very well defined from a global perspective. For example, according to the GDPR, data protection mechanisms should prevent personal data breaches [Gen16]. However, throughout this thesis, the term data protection is often used in a broader sense, i.e., not limited to personal data only but sensitive data in general that needs to be protected.

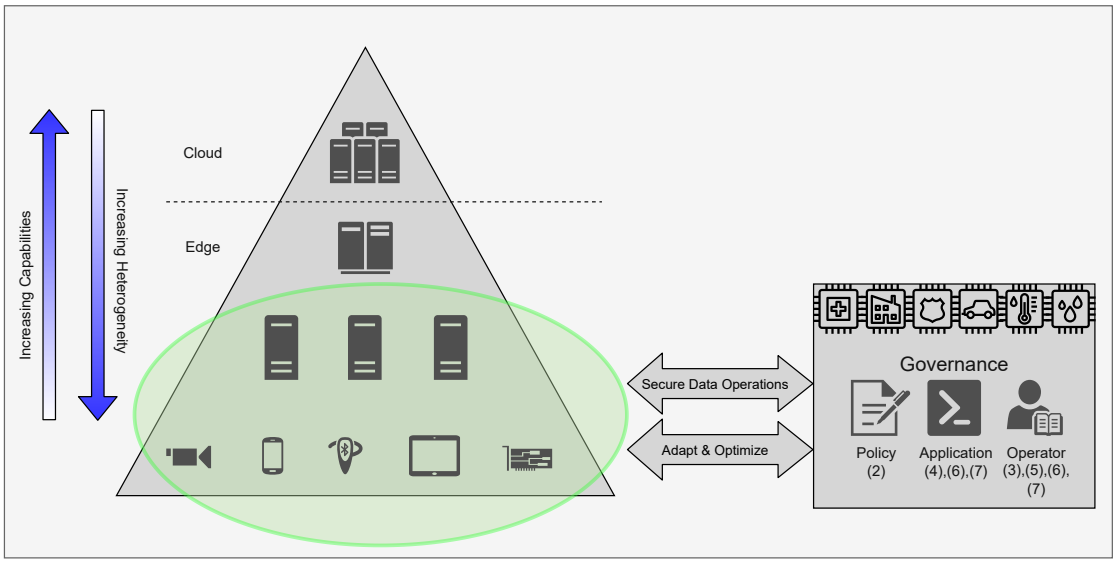

Figure 1.3: Secure Data Operations through Adaptation in Mid- and Low-Tier Edge Computing Systems

Adaptive Systems bear great potential to enable flexible and context-aware data protection for such systems, especially if resource constrained devices are incorporated. Based on the classical definition of control theory an adaptive system monitors its own performance and adjusts its parameters in the direction of better performance [NA12]. In the context of protection of data, this involves i) monitoring the system and its environment, ii) analyzing whether changes threaten the satisfaction of security and privacy requirements, and iii) the planning and execution of adaptations, if needed, to ensure the continued satisfaction of these requirements [KC03]. To enable such adaptations at run time, appropriate adaptation rules have to be defined at design time, specifying what adaptation to perform in which situation. While there are several existing approaches to creating self-adaptive systems, they assume that the designer is able to define the appropriate adaptation rules [ST09]. For defining adaptation rules, it is crucial to understand i) what changes in the environment may happen, ii) what self-adaptations the system may perform, and iii) how those changes and self-adaptations impact the relevant system properties. In this thesis, we investigate the adaptation space to improve or enhance data protection in resource constrained urban sensing environments. Specifically, as depicted in Fig. 1.1, we propose solutions to tackle several problems from four urban sensing domains. The numbers inside Fig. 1.1 reflect the actual chapters in this thesis. Each of those chapters deal with one or more particular problems in the respective urban sensing domain (highlighted in green), which is described in more detail in the following section.

1.1 Problem Statement

The improvement of performance related functional requirements for urban sensing applications, e.g., shorter computing times, optimizing data storage or reducing latency is a popular area of research in edge computing. However, adaptations to increase non-functional requirements, such as security and privacy, are often second to such performance related characteristics of a system [MAM15]. This circumstance can hinder data management workflows from complying to privacy policies [DJP11] or security provisions such as HIPPA [oHS96]. Current research falls short of providing concrete frameworks and solutions for modeling privacy constraints and enacting data processing rules to meet privacy requirements [SCZ+16]. Additionally, internal and external run-time parameters, reflecting the environmental context of a system, may heavily influence the decision making processes regarding the relevant data protection mechanism that need to be carried out in order for the system to adequately comply to a privacy policy. In chapter 2 we tackle this issue and present a novel model for defining and enacting privacy policies based on context-aware edge computing. Our proposed approach is motivated by use case of the smart health domain. Furthermore, well established countermeasures reducing or eliminating the attack surface on a system are often not designed to be executed on resource constrained devices or may perform poorly on those [SWZL12]. Many of widely used, commercially available, resource constrained devices come with built in *crypto chips*, i.e., specialized hardware to accelerate specific cryptographic applications. These chips usually feature methods used by the Secure Sockets Layer (SSL) or Transport Layer Security (TSL) standard, but they do not support the many other different algorithms available used for data confidentiality and data integrity that could be needed to facilitate the development of custom protocols, e.g., onion routing, relying on secure and privacy preserving data transfer [LRD19b]. As technology advances, generalized statements on performance of data protection mechanisms in smart home, such as: *"Asymmetric Cryptography is infeasible"*, should not be made. Therefore, it is necessary to continuously (re-)evaluate those various mechanisms and to show whether such statements still hold true or not. A thorough performance evaluation of such algorithms on representative devices can aid developers and system engineers in designing a system that adheres to specific security guidelines, yet maintains adequate performance[LD19a]. Additionally, systems may encounter changes of several parameters during runtime, such as sensor reading frequency, data size, or a specific means/levels of protection quality (e.g., encryption strength) concerning sensitive data. The level of data protection could manifest itself via changing security policies enforcing a minimal key size of an encryption algorithm or ensuring data integrity using a message authentication code. Generally speaking, strategies to protect data need to be appropriately designed and correctly implemented in order to mitigate the associated risks. Therefore, we need a better understanding of the main levers for performance and protection of data in edge computing systems incorporating resource constrained devices. In chapter 3, we evaluate several data protection mechanisms on a real world testbed, featuring typical resource constrained smart home devices. This evaluation is viable for developers and system engineers

to tackle the aforementioned problems. Based on our results, adaptation strategies and adequately defined respective privacy policies can be designed and implemented in various urban sensing domains, such as smart buildings or environment monitoring. Those domains typically feature resource constrained devices and face challenges as mentioned above. A concrete problem in urban sensing, e.g., in environment monitoring, is Source Location Privacy (SLP). SLP, as the name implies, aims to keep the location private, where data was originally collected. Referring to smart home, being an interconnected network, this would in most cases result in efforts to keep the IP-address of a device private. A well-known example in literature is the Panda-Hunter Game, where a WSN is deployed in a forest to monitor pandas. Hunters take the role of an adversary, trying to capture the panda. The goal is to prevent the hunter from locating the source, i.e., the sensor attached to a specific panda [Kea05]. In chapter 4 we present a SLP mechanism based on the onion routing principle specifically designed to meet the requirements of a system incorporating resource constrained devices.

In the public surveillance, another concrete problem in urban sensing is the protection of Personally Identifiable Information (PII), extracted from video data. For example, PII could be inferred from recording persons (faces) or cars (licence plates) by a traffic monitoring system. Recent research in particular has identified edge computing as a key enabler for privacy-sensitive systems that deal with real-time video processing [SSX+15, GHB18]. However, the increasing number of cameras in public spaces cause growing concerns about the abuse of mass surveillance systems and the implications on personal privacy and freedom [SMM+09]. Therefore, adequate protection of private data is an increasing concern in all kinds of domains making use of public video streams, such as health, financial, and social security. The most common straight forward generalized approach to protect sensitive data is the installation of access control mechanisms alongside with various encryption techniques, in order to protect data at rest and in transit. An exemplary video analytics implementation at the edge might incorporate a computing unit, connected to a camera, encrypting and transmitting a video feed via TSL to a cloud server, where some form of e.g., Role Based Access Control (RBAC) ensures that the decrypted video feed may only be processed by a entity with adequate permission or role. Most existing approaches focus on privacy and security related operations of the video stream itself or protecting its transmission. Instead of applying encryption or privacy preserving image transformation techniques to a recorded video feed, the relevant information from the feed could be extracted with the help of sophisticated machine learning techniques [LRD21]. This information only is then transmitted and made available, and, of course, also protected by similar mechanism as described in the previous example. The (raw) video is never transmitted or persisted/distributed permanently. This concept is presented in more detail in chapter 5. Recent advances in Artificial Intelligence (AI), particularly in Machine Learning (ML), enable effective automatic video processing [DA04, FBVC01, BZR+05, HBB+07]. The emerging Edge computing paradigm facilitates the deployment of distributed AI-applications and hardware. AI-assisted video analytics can provide valuable information and benefits for parties in various domains. Face recognition, object detection, or movement tracing are prominent

examples enabled by this technology. Hence, AI can help in satisfying requirements for adequate protection of data. For example, faces and license plates can be automatically detected by appropriate ML-based object detection algorithms, and then anonymized by further video manipulation methods. However, AI-assisted protection mechanisms are often coupled with heavy computational costs. Furthermore, the capabilities of connected devices as well as the properties of the videos to process can change over time. Usually, processing videos is very resource-intensive, but the end devices producing the videos are resource-constrained. Thus, video processing systems at the edge must be self-adaptive to be able to react to changes at run time while maintaining an adequate protection of data. In particular this becomes relevant for systems running a Video Analysis Pipeline (VAP) which needs the end devices to be flexible concerning protection methods. To some extent, video processing can be offloaded to the cloud to take advantage of the virtually unlimited computational capacity offered by the cloud. However, offloading to the cloud is associated with high latency and high network load. Hence, a better solution is to deploy devices with sufficient computational capability near the network edge. End devices can offload some video processing functionality to nearby edge nodes, thereby benefiting from low-latency access to computational capacity without overloading the core network. For video analytics at the edge, these questions are complicated, since there are many different types of possible environment changes and self-adaptations, and they have intricate implications on a variety of system properties [BFI19, BFGL20, VF13a]. In particular, environment changes may happen both at the infrastructure level and the application level, and also self-adaptations are possible on both levels. Optimization is either carried out during design-time or run-time of an application, by carrying out adaptations on either the infrastructure or application level. Accuracy plays a crucial role for applications incorporating data protection mechanisms, e.g., anonymization, that have to adhere to specific privacy regulations like the GDPR. An application may have to sacrifice potential performance benefits from optimization strategies to achieve the necessary level of accuracy. The heterogeneous and dynamic environment in edge computing greatly increases the complexity of such optimization strategies even more. Thus, video processing systems at the edge must be self-adaptive to be able to react to changes at run time.

First, in chapter 6 we explore the adaptation space for AI-assisted video analytics at the edge and how to exploit it to improve the data protection aspect of such systems. Then, in chapter 7, we present a concrete implementation for an adaptation engine that is capable of automatically resolving issues related to either performance or data protection (or both). We evaluated the adaptation engine based on an AI-assisted VAP in the public surveillance domain, specifically on a traffic monitoring use case.

1.1.1 Research Questions

To summarize, the research presented in this thesis is driven by the four following research questions:

($\mathbf{RQ}_1$) *How can edge computing based data protection policies be formulated and processed in order to build the foundation for an adaptive urban sensing application?*

($\mathbf{RQ}_2$) *What parameters need to be evaluated and considered for data protection focused adaptation strategies and how can those be applied in low tier resource constrained urban sensing environments?*

($\mathbf{RQ}_3$) *What are the implications on the adaptation space regarding AI-assisted data protection mechanisms in mid tier urban sensing applications?*

($\mathbf{RQ}_4$) *How can we exploit the adaptation space in an edge based urban sensing application?*

1.2 Methodology

In order to answer the proposed research questions, for each research question, we followed the following academic methodological approach:

1. The state-of-the-art is analyzed and related work identified.
2. Challenges and shortcomings will be investigated.
3. Novel concepts will be theoretically elaborated and practically evaluated and verified.

In the following, each of those steps is described in more detail.
Edge computing in general is seen as a potent enabler of urban sensing applications. The complexity of these applications ranges from processing single environment information, e.g., temperature readings, up to processing computationally heavy expensive machine learning based tasks such as video analysis. However, many edge devices executing those applications are limited in their capabilities, such as computing performance, energy consumption or storage. Additionally, many of those applications also face challenges related to data protection. In this thesis, it will be investigated how adaptation strategies could potentially mitigate risks to data protection while still maintaining adequate performance. The research conducted in this thesis will focus on low and mid tier edge computing devices operating in the scoped urban sensing domains. Survey and vision papers will provide a general overview on these challenges, while technical papers will describe a detailed in-depth view on the specific research areas. Challenges and shortcomings of these papers will then be investigated and novel solutions derived with respect to the specific urban sensing domains covered by this theses. First, state-of-the-art policy languages and definitions and how to apply them in edge based systems

will be researched in order to answer RQ_1. Second, special attention will be given to research concerning performance evaluations of protection mechanisms at the edge. This is particularly interesting for low tier devices, as well established protection mechanisms, such as encryption, may not perform well or not at all on those devices. Third, literature on adaptive systems in general will be investigated, followed by a thorough research on adaptive systems featuring AI-assisted applications. Understanding the limitations and specific capabilities and mechanisms of those areas is necessary to answer RQ_2 and RQ_3. Lastly, adaptive system modeling strategies, concepts and tools will be investigated in order to drive the research needed to answer RQ_4.
Practical experiments conducted on representative state-of-the-art low and mid tier edge hardware will on the one hand (partly) validate existing research and on other hand produce up-to-date results. Furthermore, the experiments will be deliberately designed to include the relevant data protection mechanisms that are needed to elaborate sophisticated and viable adaptation strategies for each of the urban sensing domains in scope of this theses.
Throughout the stages of this thesis, collaborations with students within and outside the department were conducted.

1.3 Scientific Contributions

This thesis provides significant contributions to the field of performance analysis for resource constrained devices and its adaptations space focusing on data protection for urban sensing applications. Novel possibilities and optimization strategies for adaptation management in AI-assisted edge computing paradigms were elaborated and their impact on data protection techniques validated.
Regarding RQ1, we proposed a privacy policy model leveraging edge computing techniques where sensitive data flow is handled closer to the user, because ensuring privacy is not just a matter of authentication and authorization but a more complex task which should take the environmental context in which data is managed into account. Regarding RQ2, precise measurements conducted on state-of-the-art representative edge hardware yield fundamental performance values that enable stakeholders to make accurate assumptions about data throughput and correlating energy consumption of an application that i) incorporates resource constrained devices, and ii) needs to implement some sort of data protection functionality. Additionally, systems may encounter changes of several parameters during runtime, such as sensor reading frequency, data size, or a specific *protection level* for data. The level of protection could manifest itself via changing security policies enforcing a minimal key size of an encryption algorithm or ensuring data integrity using a message authentication code. In particular this becomes relevant for adaptive systems, e.g., edge nodes running an risk-assessment engine which needs the end devices (e.g., resource constrained smart home/Edge devices) to be flexible concerning methods to protect data. Algorithms supporting confidentiality, authenticity, and integrity are of particular interest, whereas their limitations and throughput rates can be used to calculate protection/performance/energy consumption trade-offs for a specific hardware

configuration.
On top of such performance and energy consumption measurements, our results enable the design and implementation of specific optimization strategies that enhance application behavior, by monitoring and adapting critical system parameters that influence the protection of data and performance, thereby constantly reacting to potential environmental changes. Such an optimization strategy should aim to balance performance, quality of data-protection, and energy consumption is either performed during design-time or run-time of an application, by carrying out adaptations on either the infrastructure or application level. Furthermore, a concrete implementation of a source location privacy protocol strategy is proposed, which is based on the previously gathered results presented in this thesis. Regarding RQ3, the impact of artificial intelligence and machine learning on edge based systems and its implications on the protection of data is evaluated. The evaluation results presented in this thesis are particularly (but not exclusively) applicable to video analytics on the edge, which can be significantly enhanced by incorporating AI-based techniques and such systems are expected to be continuously deployed in e.g., smart cities. However, in many cases such systems face challenges like heterogeneous deployment infrastructure and dealing with dynamic contextual changes in a recorded videos nature such as varying number of people recorded over a given period of time. Lastly, regarding RQ4, first, a privacy system for AI-assisted video analytics at the edge is proposed. The system ensures that applications leveraging extracted data have no access to the video stream. It extracts relevant information from video data and governs the secure access to that information. An attribute-based authorization scheme allows applications to only query a predefined subset of extracted data. Second, an adaptation engine is proposed capable of modeling and exploiting the investigated adaptation space in the domain of AI-assisted video analytics. The proposed engine leverages the application- and infrastructure based adaptation space of a distributed VAP.
To briefly summarize, the presented results provide parameters, levers, guidelines, strategies and concrete implementations that are useful for either developers, requirement engineers, and software architects, but researchers as well They aid and facilitate the design, implementation and deployment of urban sensing systems dealing with resource constrained edge devices and that need to adhere to security and privacy rules.

1.4 Thesis Structure

The thesis is structured as follows: Chapter 2 presents a novel model for defining and enacting privacy policies based on context-aware edge computing. The proposed approach is motivated by use case of the smart health domain. The design choices of such policies are ideally grounded in fundamental research specific to the urban sensing domain. Hence, in Chapter 3 we present an evaluation of several data protection algorithms, measuring their impact on resource constraint devices with regarding performance and energy consumption. The presented results were conducted on a typical IoT devices and build the foundation of a concrete solution to the source location privacy problem in various urban sensing domains, such as environmental monitoring. This solution

is based on the well-established onion routing principle and presented in chapter 4. Chapters 5 - 7 shift the focus to another prominent domain of urban sensing, i.e., public surveillance. Concretely, those chapters focus on AI-assisted video analytics pipelines. Chapter 5 presents a privacy preserving system for AI-assisted video analytics, that extracts relevant information from video data and governs the secure access to that information leveraging the concepts of Attribute Based Encryption (ABE). In chapter 6 we explore the adaptation space of AI-assisted VAPs, i.e., we identify factors that can be adapted in AI-assisted data protection for video analytics using the example of a face blurring pipeline in a traffic monitoring system. A concrete implementation of an adaptation engine for such systems is presented in chapter 7. The engine is evaluated (but not limited to) based on the use case (i.e., traffic monitoring) as described in the previous chapter. Lastly, the thesis is concluded in chapter 8. Furthermore, this chapter provides an outlook on how the presented solutions will be integrated and applied in future work.

CHAPTER 2

Context-Aware Enforcement of Privacy Policies in Edge Computing

Privacy is a fundamental concern that confronts systems dealing with sensitive data. The lack of robust solutions for defining and enforcing privacy measures continues to hinder the general acceptance and adoption of these systems. Edge computing has been recognized as a key enabler for privacy enhanced applications, and has opened new opportunities. In this chapter, we propose a novel privacy model based on context-aware edge computing. Our model leverages the context of data to make decisions about how these data need to be processed and managed to achieve privacy. Based on a scenario from the Smart Health domain (aka eHealth), we show how our generalized model can be used to implement and enact complex domain-specific privacy policies. We illustrate our approach by constructing real world use cases involving a mobile Electronic Health Record that interacts with, and in different environments.

After a brief introduction, the remainder of this chapter is structured as follows. In Section 2.2 we present a motivational scenario from the eHealth domain. Section 2.3 gives an overview of related work. In Section 2.4 we present our privacy model, and how context-awareness is factored into this model. In Section 2.5 we discuss the enforcement of privacy policies by our model based on context-aware edge computing. Finally, Section 2.7 concludes the chapter and gives an outlook on future work.

2.1 Introduction

Cloud computing and the continued centralization of computation and data management has caused growing concern about data privacy [SD16]. Releasing data to centralized

services is especially problematic for systems that handle sensitive data, such as patient data in eHealth systems, as this loss of control can hinder data management workflows from complying to privacy policies [DJP11] such as the GDPR or security provisions such as HIPPA. Edge computing has been recognized as a key technology for enabling privacy-aware IoT applications. However, the complexity inherent to edge computing architectures makes it extremely difficult for application developers to implement mechanisms that can guarantee privacy policy compliance, especially in complex domains with high amounts of stakeholders, such as eHealth. Current research falls short of providing concrete frameworks and solutions for modeling privacy constraints and enacting data processing rules to meet privacy requirements [SCZ+16]. Data confidentiality, data integrity and data privacy are the key concepts to meet those requirements. To avoid information leakage, strict access policies must ensure the confidentiality and integrity of private data, as well as handling data locality. This is commonly realized by defining roles according to different stakeholders of a system, typically enforced by RBAC techniques [FCK95]. However, further data privacy agreements (e.g., data exchange between stakeholders), that should also be adequately defined, established and implemented via privacy policies, introduce additional architectural, conceptual, and performance related challenges. In this chapter, we present a novel model for defining and enacting privacy policies based on context-aware edge computing. By leveraging context-awareness of edge computers, we enable runtime decision making for applications on how to enforce privacy policies during data workflows. Compared to existing approaches, which are mostly tailored to a specific use case or domain [XSSC06, KAB09, CRJS13, Klo17], our model focuses on context-awareness and corresponding actions edge devices have to take. We thereby enable flexibility in implementation, and aid system architects or developers in designing and building decentralized systems that can make use of privacy-sensitive data, while complying to complex privacy policies of a given domain. As part of our privacy model, we define privacy levels to incorporate a finer formal description granularity. We demonstrate our approach based on a scenario from the eHealth domain, where privacy is considered a critical requirement.

2.2 Motivational Scenario

The National Committee for Vital and Health Statistics (NCVHS), a key advisory committee to the US Department of Health and Human Services, defines privacy in the context of health information as *"an individual's right to control the acquisition, uses, or disclosures of his or her identifiable health data"* [Coh06]. Hence, granting patients control over their medical records, even if the data are owned by another party (as is common in electronic health records) is fundamental for enabling privacy. The concept of mHealth [KST11] can facilitate this by leveraging mobile platforms to monitor, process, and store medical data in a so called Mobile Electronic Health Record (mEHR). However, this decentralization increases complexity of data management workflows, in particular when privacy policies need to be enacted by the system automatically at runtime. We consider a typical scenario from the medical domain, where a patient is

going into a hospital seeking medical consultation and is subsequently examined by a physician. The patient has a mEHR as an application installed on their smartphone. This record comprises personal data fields such as name, address, or gender; and medical data fields arranged in sections such as diagnoses, prescriptions or therapies. Multiple other stakeholders are also involved in further steps in this scenario. For example, consider a pharmacist who hands out prescribed medication, or a biotechnological specialist (who is, e.g., consulted for specialized artificial implants). These stakeholders can also be other machines such as a drug dispensers or smart medical imaging devices. Furthermore, in this scenario, diagnoses can also be done remotely, which requires a system to be interconnected across different networks. These examples all require the sharing or processing of sensitive data in different contexts between different stakeholders. A system that supports this type of complex scenario requires not only standard role-based access control mechanisms to ensure privacy, but has to *react and adapt* to different changes in environmental context. Such reactions are operations or processes that ensure privacy of data is preserved and carried out in a way that conforms to a corresponding policy.

2.3 Related Work

Securing the integrity and sharing of information is a critical requirement in complex distributed systems that deal with sensitive data. Data encryption is a common tool to hinder an unintended user to infer information of stored or transferred data. An overview of different well researched encryption approaches and techniques is presented in [AV16]. To protect data from alteration, modification or deletion in a distributed system several data integrity strategies have been developed and established. These include Provable Data Possession (PDP) [ABC+07], Proof of Retrievability (PoR) [JKJ07] and Third Party Auditing (TPA) [WWRL10]. Nowadays, interconnected edge devices like smart phones, sensors, Radio-Frequency Identification (RFID) tags, or smart home devices are producing a huge amount of data. As these devices become more and more integrated into our daily life, they significantly affect and change our way of living, social behavior and life style [SCZ+16]. These data are then used to generate context-aware information (e.g., tracking the commute duration from a person's home to their work location) [SKW15]. In this work we focus on data privacy per se, which deals with safeguarding personal information. For instance, patients private data, diagnoses or therapy plans may be misused by, e.g., insurance companies to adapt rates, and must therefore be well protected. Enforcing privacy in software systems has mostly been addressed by incorporating RBAC techniques. Standards like the eXtensible Access Control Markup Language (XACML) [OftAoSIS] can be used to define policies and handle access of data but do not take different environmental contexts into account nor describe how and where these policies should be enabled. Furthermore, most of those mechanisms are based on a centralized architecture with a complex constellation of roles and sensitive data is often duplicated or distributed. Our model describes how such well established RBAC approaches could be extended, by incorporating edge computing techniques, to facilitate the development of decentralized privacy preserving systems.

2.4 Privacy Model in Edge Computing

The privacy model we propose describes the circumstances under which an entity is allowed to access specific parts of sensitive data. Defining a consistent model enables a coherent definition of policies that ensure privacy of data, also handling data locality. Our privacy model combines and correlates certain levels of privacy (e.g., visibility constraints on specific data sets) with a given context. The determination of a specific context is best handled closest to the according environment. Therefore, edge computing is well suited for this task, where every edge device is exposed to a certain and specific context.

2.4.1 Privacy Policies

A common way to enable and enforce privacy in edge computing is to define policies that specify the handling of sensitive data [YYSL12, RGP15]. One use case may be the need for policies to correctly handle the collection, exchange and disclosure of patient data (e.g., in medical consultation scenarios). Retaining data quality and accessibility required for medical processing while respecting privacy aspects of all involved persons is one of the key challenges when defining privacy policies in eHealth. A simple concrete implementation of such a policy could describe which data fields of a given set of personal information should be persisted and which data fields should be handled transiently. The downside of such policies is that they are often tailored to specific use cases and therefore lack flexibility and generality. In this work we distinguish between two different types of policies. First, there are policies which define the data persistence modalities of private data, and second, policies which describe how and where the data is modified (e.g., anonymized or encrypted) for transport and inspection or further computation. Therefore, we distinguish between logical and physical privacy boundaries. While logical boundaries deal with role-based constraints, legal constraints, etc., physical boundaries comprise location-based constraints, network constraints, or other involved devices.
After policies are defined, only a subset of those need to be deployed and stored on relevant edge devices, based on the assumption that not every policy is applicable in every context.

2.4.2 Privacy Levels

To incorporate more granularity into policies, we suggest to define different levels of privacy. Regarding our example scenario, we divide privacy related data into (i) domain specific data like anamnesis data, medical diagnoses, pharmaceutical prescriptions, etc., and (ii) personal data like full name, address, and other data of personal domain. An example of a privacy policy in the eHealth domain could force a device or system to decide between visibility levels of patient data. One level could define that all data is visible (e.g., for personal usage), while a physician or pharmacist should only be allowed to access specific data sections, like open prescriptions or medical diagnosis. If a physician seeks consultation, another level could state that all personal information is invisible and only medical data is visible. At last, the most restrictive level would be that

all data is invisible to anybody, e.g., being encrypted for transmission. Our proposed model presupposes that for every different level contextual information is incorporated. Furthermore, these levels should be defined at a more granular level depending on given system dependencies like domain specific, political, legal or architectural constraints and operate either on holistic data sets, data sections or single data fields.

2.4.3 Context-Aware Decision Making

To achieve a higher degree of flexibility in implementing privacy policies we propose a context-aware decision making process to dynamically adapt to changing environment situations at the edge. In literature, the place where this decision making process is carried out is commonly referred to as a Policy Decision Point (PDP). This PDP typically relies on a so called Policy Administration Point (PAP), where your policies are unequivocally defined (as will be described in Section 2.5.1 and stored. Context in computer science can be interpreted in many different ways. In the focus of this chapter, we use the term *context* as environmental information recognizable by edge devices. This could be information about the network and its topology, connected devices, spatial information, proximity, location or time. A context-aware system is able to interpret changes in the environment and react to them in a predefined manner. After a decision on such a reaction is made, the concrete action typically takes place at a so called Policy Enforcement Point (PEP). In this work, the Policy Application Manager (PAM) (as described in Section 2.5) can be seen as such a PEP. The decisions made and actions carried out are typically monitored by a Policy Information Point (PIP), which can be seen as a checkpoint to validate the correctness of the policy enforcement steps. While a PIP is definitely a useful tool, it is basically a monitoring component and not closely tied to our model. Hence, it is out of scope of this work.
One way of telling the system how to react to certain context changes is by defining previously mentioned policies and enhance them with contextual parameters. Regarding privacy, such context-aware systems are on the one hand able to anticipate potential risks and provide recommendations to, e.g., a user which actions to take, and on the other hand automate certain adaptations in data management and processing. However, in real world scenarios such systems are not provided with holistic environment information all the time, and sometimes they have to make decisions based on incomplete data. This has to be taken into account when implementing privacy policies following either a conservative strategy, i.e., disable controls or hide sensitive data on a user interface, or an optimistic decision making strategy like allowing read access on partly obfuscated data.

2.5 Context-Aware Policy Enforcement on the Edge

By sticking to our motivating example, we illustrate how context-aware privacy policies are defined and describe the corresponding decision processes that enable/enforce those given a certain context using our model. The model comprises three essential encapsulated parts that work together in a coherent way but can also be separately implemented

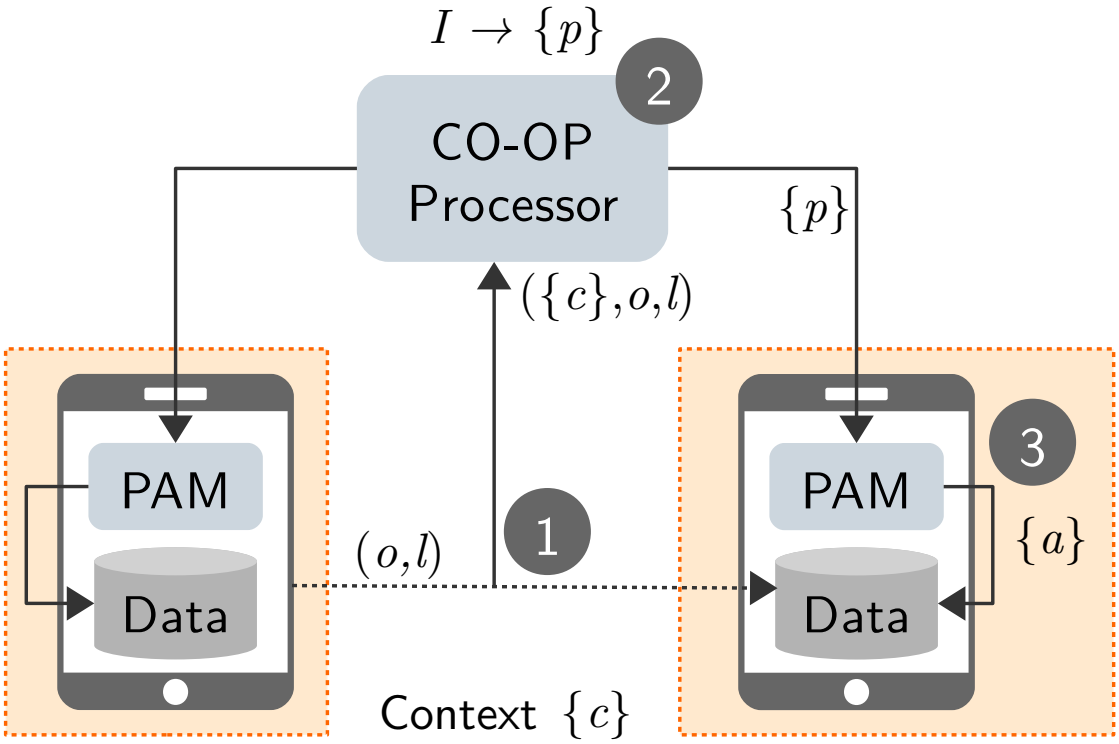

Figure 2.1: Interaction of model components and edge devices operating in different contexts

(e.g., using existing standards) if need be. Part one describes the process how to define context-aware privacy policies for a certain system and how to describe them using our privacy model. The second part defines an inference mechanism to obtain one or more policies (depending on the policy definitions granularity grade) and the third part describes a mechanism that determines the resulting actions after one or more policies were identified.

Figure 2.1 illustrates the general principle of our model and how the distinct components work together in three basic steps. First, a device requests access to (private) data on another device. Context, Operation and Privacy Levels are transmitted to a Context Operations (CO-OP) Processor. Second, the CO-OP Processor returns inferred policies to the PAM implemented on each device, similar to a traditional PDP. Third, the PAM triggers certain actions performed on corresponding data records to enforce the policies returned by the CO-OP Processor. As mentioned in the previous section, this process is similar to an action performed by a traditional PEP. Each of those elements will be discussed in detail in the following.

2.5.1 Policy Definition

Policies should be defined at an early stage during the system design phase, and comprise fine-grained domains-specific parameters as it reduces ambiguity in implementation details. The structural elements of our model are i) privacy levels, ii) data management operations, and iii) context. The first thing to ascertain is the characteristic of either logical or physical types of different contexts in the system. A typical example for a logical form of context is the *role* of a user or device. A physical form of context could be the current *location* of a user or the proximity of a device to an edge node. In practice, context should be defined as fine-granular as possible. In this work, we suggest different types or domains of context, as described in section 2.6. However, those types do not comprise an exhaustive description or manifestation for every possible real world scenario. Therefore, it is necessary that at design time, as good as possible, every context is identified as such and, moreover, each of those identified contexts is described and modeled as detailed as possible. The distinction between different types of context, allows for a more structured elicitation of requirements as well as in decoupling corresponding source code and re-usage capabilities of such code. The better the quality of this context description, the better will be the accuracy and effectiveness of the inferred actions taken as a response to e.g., change in context in the proposed system model. To achieve this, precise measurements (e.g., quality of a Global Positioning System (GPS) signal to determine the physical location of an object or user) have to be conducted as soon as possible and carefully validated in order to achieve stable results.
After identifying different types of contexts, their manifestations have to be mapped to certain data management operations (like classical CRUD operations, or persistence constraints) that respect predefined privacy levels.

Table 2.1 shows exemplary privacy policies defined in the context of our motivational scenario. These policies are based on typical use cases in the eHealth domain. In the textual descriptions of the policy, we highlight the structural elements as defined by our model. A policy defines one or more contexts c_i, a privacy level l, and a data operation o.

2.5.2 CO-OP Processor

After several privacy policies are defined, an edge device needs to know if and when a certain policy has to be applied. This task is implemented by the CO-OP Processor which infers one ore more policies given domain-specific context parameters, privacy levels, and a set of data operations. The corresponding inference function I can be formally expressed as:

$$I : C^n \times O \times L \rightarrow P^m \tag{2.1}$$

where C is the set of domain-specific context parameters $\{c_1, c_2, ...\}$, and O is a specific operations (such as CRUD or persist), and L is the corresponding privacy level on which should be operated on. The output is one or more policies $p \in P$ that need to be enacted. The CO-OP Processor should be implemented directly on an edge device.

Policy	Description
p_1	A physician (c) may read (o) the personal section (l) of an mEHR.
$p_{1.1}$	A physician (c) may not persist (o) personal data (l) from the mEHR.
$p_{1.2}$	A physician (c) may create, read, update or delete (o) the diagnoses and prescription part (l) of the mEHR.
$p_{1.3}$	A pharmacist (c) may read (o) the prescription section (l) of an mEHR.
p_2	Drug dispensers (c) may read (o) the prescription section (l) of the mEHR only if the user is in close proximity.
p_3	Data access (o) on the medical section (l) is allowed only after explicit user permission if a connection is established across different networks (c).
p_4	Data access (o) on the medical section (l) is allowed only if the involved devices (c_1) are located in the same geofence (c_2).
p_5	After the creation (c_1) of a prescription section (l) of the mEHR, there is a predefined time window (c_2) for a device of type drug dispenser (c_3) to access (o) this section (l).

Table 2.1: Example of data privacy policies of an eHealth application based on domain-specific context parameters, data management operations, and privacy levels

2.5.3 Policy Application Manager

The determination of a certain policy by the CO-OP Processor implies specific actions for a device to be executed. The Policy Application Manager takes one or more such policies $p \in P$ as an argument and returns one or more actions $a \in A$ that have to be taken to enact the privacy policy. This function A_f can be formally defined as:

$$A_f : P^n \rightarrow A^m \tag{2.2}$$

where P is the set of policies determined by I, and A is the set of possible actions.

Returned actions include basic create, read, update or delete (CRUD) operations or persistence tasks as well as more complex actions like triggering another decision process delegated to the CO-OP processor or transferring, decrypting, or further computing data. Formally described, an action operates within the privacy model of a given policy accomplishing the enforcement of its purpose. The PAM should be implemented on a per

device basis or on edge nodes, because triggered actions may differ between, e.g., device types. Delegating actions to the corresponding devices is also performed by the PAM.

2.6 Types of Context

The following scenarios illustrate the practical application of our model by covering several examples of context manifestations, data management tasks the resulting inference of corresponding privacy policies and their practical application.

2.6.1 Role Based Context

To perform specific tasks on a mEHR, policies are defined which describe *who* is allowed to perform these data management tasks. Our predefined policy P1.2 state that *a physician* is allowed to create, update or delete diagnoses and prescription parts of the mEHR and P1 that he is only allowed to read personal data fields. At a special point during the consultation process a physician (e.g., via their PC) wants to establish a connection to the patient's smartphone to access the mEHR. He sends a request to the CO-OP processor at the edge, providing the context in form of his *role*, the operation, in our case *read*, as well as the privacy level *personal and diagnoses section* of the mEHR. The CO-OP processor returns policy ID P1, P1.1 and P1.2 which will then be further processed by the Policy Application Manager. Because of P1.1 personal data won't be persisted on the physicians PC, therefore treated as transient data as long as the connection is established (if not stated otherwise by a policy). On the other hand the mEHR of the patient registers a connection request being made by *a physician* and therefore also requests policies from the corresponding edge node. It then applies the corresponding policy which allows the connection to be established and the physician to access the patients data. On the contrary, if a pharmacist wanted to do the same, no defined policy would allow him to do so (P1.3 only allows read access on the prescription section). This RBAC strategy is the foundation of our model. Exhaustive research have been conducted on this topic and could therefore facilitate the base implementation of our model.

2.6.2 Proximity Related Context

Sticking to our motivating example, we assume the physician prescribed the patient a certain medicament. The patient then connects his mEHR to a special drug dispenser [Mic95]. The patients mEHR application recognizes the drug dispenser as a *special device type* and infers a corresponding policy (P2) which allows any device of this certain type to read the prescription data part of the mEHR. Furthermore, the device running the mEHR has to be in close proximity to the drug dispenser. While a specific device could also be modeled as a role, proximity has to be *sensed* and processed by an edge device near by. If both constraints are satisfied, the drug dispenser then reads the prescription related data fields and hands out the patient his medicine.

2.6.3 Network Related Context

Another privacy policy (P3) defines that incoming connections are only handled automatically if the connecting device (e.g., physicians PC) is located in the same network as the device with the mEHR application installed. As long as this is the case, resulting actions are based on decisions made by defined policies as described above. However, if for instance the patient is at home and a physician tries to access the mEHR from the hospitals network, an edge node recognizes that the mEHR is not connected to its network and therefore sends a notification on the user stating that someone or something is trying to access his mEHR. Via permission dialog or a similar Graphical user Interface (GUI) mechanism the patient is then able to allow or deny the incoming connection and processing of data.

2.6.4 Location Based Context

Determining the physical location of the patient's device is a common task in mobile computing and especially on smartphones. Certain policies could be enforced based on the patients current location. As an example we extend our use case and refer to policy P4, that enforces the *presence* (e.g., at least inside the hospital) of the patient near a specific device, similar to proximity based context. This could be realized by defining so called geofences, which describe a virtual perimeter based on GPS data [NSS13]. This could become relevant for e.g., management tasks like dynamic bed allocation performed by nursing personnel, if patients are not restricted to stay in their room or hospital permanently. In contrast to a proximity related context, this type of context is best used if a location can be determined accurately. As an example, the drug dispenser problem as described in section 2.6.2 could also be modeled by using a location based context. However, in reality the accuracy of an indoor based geo-location used to determine if a user is in close proximity to the drug dispenser will be significantly more prone to errors than using a proximity related context based on e.g., RFID hardware or Bluetooth.

2.6.5 State Based Context

The last example of context-aware decision making uses an inferred *state* based on previous actions that were executed on the patients mEHR. For instance it is reasonable that after the physician prescribes the patient a medicine, the next logical step for the patient would be to get to the drug dispenser and collect his prescribed medication. One of our predefined policies (P5) defines, that after the prescription part of a mEHR is altered there is a (for example 72h) time window for the patients smartphone to establish a connection to a drug dispenser. However, those state based context processing could potentially lead to a system incorporating many exception conditions.

2.7 Summary

The Internet of Things with all its edge devices generate, process and store a huge amount of data. A lot of these data include privacy sensitive information and can be used to infer specific user behavior patterns or, in worst case scenarios, compromise a user. Holistic approaches are facing architectural, conceptual, as well as performance related challenges. Therefore, we propose a model leveraging edge computing techniques where sensitive data flow is handled closer to the user, because ensuring privacy is not just a matter of authentication and authorization but a more complex task which should take the environmental context in which data is managed into account. Armed with a variety of powerful sensors that are considered to recognize relevant environmental information, these edge nodes take away workload from traditional centralized, cloud based approaches while also aiding sensible data locality tasks. Our model can aid architects and developers to identify these contexts a system is confronted with. By defining policies at an early point in system design, privacy concerns can be mitigated or even eradicated. Hence, privacy policies have to be tied to several contexts in form of fine grained definitions. However, to assure the quality, accuracy and effectiveness of the actions taken to handle those risks, precise measurements and a high quality of those measurements have to be carried out. Thus, involved Edge Devices must be able to *sense* a certain context as precisely as possible and developers must implement the corresponding inference functions as fine granular as possible to enforce a certain policy. We suggest that this can be achieved by enriching policies, based on well established RBAC features, with contextual information. Therefore, the enforcement of such policies is not limited to be executed only after specific requests of a device to a corresponding edge node. Edge devices, being aware of the context, could anticipate potential risks beforehand and automate certain adaptions in data management and processing.

CHAPTER 3

A Performance Evaluation of Data Protection Mechanisms for Resource Constrained IoT Devices

Data Protection is a major research topic concerning the Internet of Things. IoT systems continue to permeate deeper into our personal lives, where devices sense, process, and store all kinds of data. This poses various challenges to security and privacy aspects, especially to applications running on resource constrained devices. In this chapter we evaluate selected, well established data protection mechanisms that enable confidentiality, integrity and authenticity of data, as well as secure key exchange mechanisms. Specifically, we look into the performance and energy consumption of different cryptographic block and stream ciphers, secure hashing algorithms, digital signature mechanisms, and key exchange protocols executed on state-of-the-art resource constrained devices. By providing limitations and data throughput values and their respective energy consumption, our obtained results ease the calculation of performance/data protection/energy consumption thresholds and facilitate the design and development of secure self-adaptive IoT systems in urban sensing environments. The results presented in this chapter are based on research presented in [LD19a] Those results were then updated and enriched with energy consumption measurements within a collaboration with Sebastian Kaindl and also used in his diploma thesis [Kai21]. The support of Sebastian in gathering these results is thankfully acknowledged.

After a brief introduction, the rest of the chapter is structured as follows: Section 3.2 provides an overview of related work. Relevant information on cryptographic background, methodology, and experimental setup is given in Section 3.3. In Section 3.4 we present the experimental results. Finally, in Section 3.6, we conclude the chapter and give an outlook on future research.

3.1 Introduction

Different interconnected devices with minimal to average computing power make up the majority of the Internet of Things. They sense, process, and store various data from different domains and have become an integral part of our daily lives [VF13a]. The specific nature of data related to critical domains, such as healthcare, public surveillance or home automation, requires tailored data protection mechanisms concerning security and privacy aspects. Data Integrity, confidentiality, authenticity, anonymization and source location privacy pose different, partly overlapping challenges for the design and implementation of an IoT system. Those challenges become even more demanding when dealing with resource constrained devices. In this chapter we refer to devices that lack computing power, i.e., Microcontroller Unit (MCU)s with low CPU power and limited memory. Furthermore, we distinguish between devices typically found in Wireless Sensor Network, which could be seen as an infrastructural subset on the IoT, and typical resource constrained IoT devices. Most of the devices found in WSN are sensors and actuators mostly adhering to the IEEE 802.15.4 protocol, using communication frameworks like ZigBee. We define a resource constrained IoT device as a device with limited computational power but integrated Ethernet or WiFi capabilities and USB connectivity for facilitated programming.

The design of an IoT system, dealing with sensitive data, is often accompanied with the definition or followed upon a set of distinct data protection rules, e.g., in the form of policies. The overall risk that such a system is exposed to could be assessed by looking on its various attack vectors of an adversary. Concerning privacy, protecting the system from attacks on confidentiality of the data is crucial. Other characteristic threats to those system are spoofing attacks, message altering, replay, and flooding attacks [FWKM15]. Well established countermeasures reducing or eliminating the attack surface on a system are often not designed to be executed on resource constrained devices or may perform poorly on those [SWZL12]. However, as technology advances, generalized statements on performance of data protection mechanisms in IoT, such as: *"Asymmetric Cryptography is infeasible"*, should not be made. Therefore, it is necessary to continuously evaluate those various mechanisms and to show whether such statements still hold true or not. In this work we evaluate the performance of different algorithms concerning data integrity, authenticity, and confidentiality as well as their respective energy consumption. Our testset of algorithms comprises well established standard implementations of encryption, key exchange, signing, and hashing methods, as well as lightweight implementations, respectively. Many of widely used, commercially available, resource constrained IoT devices come with built in *crypto chips*, i.e., specialized hardware to accelerate specific cryptographic applications. These chips usually feature methods used by the SSL or TSL standard to facilitate secure data transmission, but they do not support the many other different algorithms available used for data confidentiality and data integrity that could be needed to facilitate the development of custom protocols, e.g., onion routing, relying on secure and privacy preserving data transfer. Systems may encounter changes of several parameters during runtime, such as sensor reading frequency, data size, or a specific *level*

of data protection. The results gathered by our evaluation can aid developers and system engineers in designing a system that adheres to specific security guidelines and privacy policies, but also balances performance and energy consumption. The quality of data protection could manifest itself via changing security policies enforcing a minimal key size of an encryption algorithm or ensuring data integrity using a secure hashing algorithm. In particular this becomes relevant for adaptive systems, e.g., edge nodes running an risk-assessment engine which needs the end devices (i.e., our resource constrained IoT devices) to be flexible concerning data protection methods.

3.2 Related Work

As IoT devices continue to permeate deeper into our personal lives, security and privacy aspects have become a major research topic in this field. Suo et al. [SWZL12] divide an IoT system into four layers. Bottom-up these are: Perception Layer, Network Layer, Middleware Layer, and Application Layer. An overview of possible threats and adverse scenarios on each of those layers is provided by Farooq et al. [FWKM15], concluding that the majority of threats are located in the Network Layer. To mitigate or at least lower the risk of these threats, trade-offs have to be made either at design time or runtime of a system. Those trade-offs involve parameters like energy consumption, strength of encryption or data throughput. Altering one of these parameters will most likely affect one or many of the others. In the area of WSN, where energy consumption and management is particularly of interest, security evaluations have been done by several researchers. The majority of related work in this fields deal with lightweight implementations of cryptographic algorithms. In [Den14], various security solutions were analyzed. They provide an overview of the individual security characteristics, requirements, attacks, and encryption algorithms, considered to be useful for designers of a secure WSN. Alharby et al. [AHWR18] studied the security costs in terms of energy consumption focusing on the IEEE 802.15.4 transmission protocol definition. In their simulation they demonstrate the impact of security message overhead on data latency and throughput, followed by an evaluation of the effects those overheads have on energy consumption and their memory footprint. In this work we consider such network overheads as negligible, because of the many others factors that affect network performance, like signal strength or available bandwidth. Also evaluating the costs of security in WSN, but focusing on energy consumption, Lee et al. [LKS10] measured and compared four lightweight encryption algorithms on MicaZ and TelosB sensor motes. Those devices reside at the lowest end of the WSN spectrum, regarding computing power. In typical IoT environments, such as Smart Buildings, energy consumption may not be of primary concern, while data protection and throughput is considered more important. Therefore, in this work we solely focus on the evaluation of data protection mechanisms and their performance in terms of computing power, based on measurements gathered from, what we classify as, typical resource constrained IoT devices. While our classification is based on a devices' computing capabilities, other classifications can be found in literature, e.g., Bormann et al. [B+12] suggest a classification scheme based on the memory capacity of an IoT device.

Rani et al. [RR16] studied literature on the performance of various lightweight encryption algorithms, focusing on the medical domain. In their work they provide an overview of addressed problems, methodology and outcome of different research articles. However, the addressed papers deal with evaluating the performance of lightweight cryptographic algorithms only, mostly implemented on a Field Programmable Gate Array (FPGA). An overview of various security mechanisms in the IP-based IoT is provided by Cirani et al. [CFV13]. In their work they discuss different algorithms located at the network, transport, and application layer. They focus on a detailed description of each algorithm and its properties like key size, code size, block size, etc., but do not evaluate their performance in terms of processing speed or computational effort. Closely related to our approach is the work of Ertaul et al. [EW17]. In their paper the performance of lightweight stream ciphers is evaluated. Implementation of three different algorithms were deployed on a NodeMCU development kit and their performance evaluated. Their measurements include i) stream cipher throughput, ii) power consumption, iii) memory utilization and iv) WiFi Round Trip Throughput. However, they solely focused on lightweight stream ciphers and their most powerful device corresponds to the least powerful one of our testbed. Another publication, closely related to our work is provided by Sethi et al. [SAKR12] in RFC8387. They evaluate various symmetric and asymmetric encryption algorithms with different key sizes on a 8-bit microcontroller. In their work they state that: *"It is important to state that 32-bit microcontrollers are much more easily available, at lower costs, and are more power efficient. Therefore, real deployments are better off using 32-bit microcontrollers that allow developers to include the necessary cryptographic libraries. There is no good reason to choose platforms that do not provide sufficient computing power to run the necessary cryptographic operations."* In our work the performance of algorithms related to both, integrity and confidentiality, is evaluated using a testbed comprising only 32-bit microcontrollers.

3.3 Methodology

In this section we discuss our evaluation process. First, we present the constellation and an architectural overview of the used testbed. Second, we briefly introduce selected, well established data protection mechanisms which we chose for our test scenarios, and how we conduct our experiments.
We remark that entropy generation for random number generation or specific system watchdog timer restart implementations are out of scope of this evaluation.

3.3.1 Testbed

Our testbed comprises four representative resource constrained IoT devices. We chose only commercially available and well established microcontrollers mounted on development boards which integrate our desired capabilities. This decision is based on our goal to facilitate the development of secure IoT systems, in a way that our obtained results can be easily adjusted to real world use cases. Our main criteria for selecting appropriate

Device	CPU	Architecture	Clock Frequency	SRAM
MKR1000	SAMD21 Cortex-M0+	32 Bit	48 MHz	32 KB
Due	AT91SAM3X8E	32 Bit	84 MHz	96 KB
ESP32	Xtensa® LX6	32 Bit	240 MHz	520 KB
Pi Zero	BCM 2835	32 Bit	1 GHz	512 MB

Table 3.1: Testbed devices used to evaluate data protection mechanisms for Resource Constraint IoT Environments

Algorithm	Type	Purpose
Advanced Encryption Standard (AES)	Block Cipher	Confidentiality
ChaCha	Stream Cipher	Confidentiality
P521	Elliptic Curve Operations	Signatures
Secure Hash Algorithm (SHA)	Hashing Algorithm Family	Integrity
SHA-HMAC	Message Authentication Codes	Integrity
curve25519	Elliptic Curve Operations	Key Exchange

Table 3.2: Overview on evaluated data protection mechanisms

devices, is i) limited CPU power and memory, ii) an integrated WiFi chip and iii) USB connectivity. We see such devices residing at the lower end of the IoT spectrum, close to WSN devices which we position at the lowest end. Out of this device set, we deliberately pick only MCUs that differ from each other in terms of computing power and memory, and which are prevalent on IoT development boards. Though there are a lot more devices commercially available, the majority does not differ in terms of computing power, because most of them come in the form of development boards, designed for various purposes, but featuring the same MCU mounted on the board. Table 3.1 provides an overview of the selected devices. All of those devices are programmed using either *C* or *C++*, while the sourcecode is compiled and deployed using the Arduino IDE.

3.3.2 Test Scenarios

This subsection provides an overview on the various data protection mechanisms evaluated. We are particularly interested in algorithms supporting confidentiality, authenticity, and integrity. Confidentiality refers to mechanisms enabling the protection of data from disclosure to parties not authorized to read or act on this data. Integrity mechanisms, on the other hand, deal with the prevention of altering data, either stored or during transfer, by unauthorized parties [Sch03]. Authenticity refers to authentication mechanisms used to verify a data source. Table 3.2 gives an overview on the evaluated algorithms, their type, and which purpose they fulfill.

Encryption

Encryption is a fundamental technique to establish confidentiality, which can be divided into two main categories, namely i) symmetric encryption and ii) asymmetric encryption. Substitution and transposition are the key features of encryption algorithms. Those algorithms are commonly referred to as *ciphers*. Symmetric ciphers use a (pre-)shared key for both encryption and decryption, and are further classified into i) stream ciphers and ii) block ciphers. A stream cipher algorithm encrypts one bit or byte of plaintext at a time and uses an infinite stream of pseudorandom bits as the key. Block cipher algorithms, on the other hand, encrypt plaintext with fixed blocks of n-bits size at one time [Sta17]. While block ciphers seem to be applicable to a broader range of applications and are therefore much more researched and implemented, it has been shown that stream ciphers in general perform better than block ciphers regarding CPU time [SM10] [Sta17]. In this work, we are focusing on the Advanced Encryption Standard (AES) block cipher, and the ChaCha stream cipher. There are also specific MCU variants of some well established encryption algorithm, that are optimized towards execution on such resource constrained devices, such as (multiple implementations of) tinyAES. A TinyAES algorithm written in C is also considered in this work, in order to find out if there are benefits in terms of energy consumption compared to its standard variant.
Asymmetric cryptography uses a pair of keys that are generated based on one-way functions. A one-way function easily computes any input, e.g., calculate $f(x)$, but it is hard to compute its inverse function, i.e., given the value of $f(x)$ it is hard to calculate x. Such a key pair consists of a private key which should be known to its owner only, and a public key which can be distributed and made freely available to anybody [KMVOV96]. Asymmetric cryptography can be used for both, encryption/decryption and digital signatures. Encrypting a message can be done by anybody using the receiver's public key, but decryption of that message is only possible with the corresponding receiver's private key. Signatures work the opposite way. First, a message is hashed and the resulting hash then encrypted with the senders' private key which will be send to one or more recipients alongside with the plaintext message. Authenticity and integrity of the message can then be verified by everyone in possession of the senders' public key [Lam79]. In recent years, a special form of asymmetric cryptography, namely Elliptic Curve Cryptography (ECC), has gained increased attention. It is based on the algebraic structure of elliptic curves over finite fields. The main advantage of ECC is that it provides equivalent security, compared to non-ECC techniques, but uses smaller keys [AS11]. Therefore, ECC in general becomes particularly interesting in IoT scenarios, as corresponding applications often operate on resource constrained devices. In this work we focus on signing and verifying a message using ECC, specifically Ed25519. This technique uses SHA-512 and the elliptic curve 25519.

Integrity

The concept of hashing functions is to transform a message of arbitrary length into an output sequence of fixed length, usually much shorter than the input message. The

ultimate goal for cryptographic hash functions is to create a unique value characteristic for a specific message, commonly referred to as hash of the message. Hashing algorithms are also based on previously described one-way functions, commonly used by data protection mechanisms concerning integrity of data, or for efficient data persistence mechanisms, i.e., usage of hash tables [NY89]. In our experiments we look into the performance of the latest, well established SHA family, i.e., SHA2 and SHA3. To preserve message integrity and authenticity during transfer, simply hashing a message and appending the corresponding hash is not sufficient. An adversary with knowledge about the used hashing algorithm could perform an active man-in-the-middle attack, by altering the message and calculating a new hash for that message, being then forwarded to the intended recipient. A Message Authentication Code (MAC) uses specific techniques that offer protection against such attacks, by incorporating a (pre-)shared secret key into its generation process. The values generated from a MAC are then verified by applying the same secret key used to calculate the MAC, which poses the main difference to digital signatures [RRG+91]. MAC algorithms can be implemented using either cryptographic hash functions or cipher algorithms presented in this work. In this work we are focusing on first one, by evaluating the SHA2- and SHA3-HMAC algorithms.

Key Exchange

For at least two parties to exchange confidential messages, a secret key needed for encryption and decryption must be shared among all participants. This key exchange can be seen as the weakest point, in terms of security, of symmetric cryptography. To tackle this issue, Whitfield Diffie and Martin Hellman proposed a cryptographic protocol, namely the Diffie-Hellman Key Exchange (DHKE). It leverages techniques based on asymmetric cryptography, enabling communication parties to securely exchange a common key, even if an adversary is eavesdropping the communication channel. A modern ECC variant of the DHKE is called Elliptic-Curve Diffie-Hellman (ECDH). The ECDH protocol consists of three basic steps for two parties that want to establish a shared secret key via an insecure channel. First, each party must generate a private and public key pair based on a common base point on the same elliptic curve on the same finite field. Second, the public keys are exchanged between the two parties. Third, a common secret key is derived from calculations that take the secret key of a communication partner and the previously exchanged public key of the other partner as input [HM11]. In this work we do not generate ephemeral key pairs, i.e., temporarily used key pairs, and evaluate the key generation as a separate step, because a static key generation is usually done only once when setting up a device. Furthermore, we do not evaluate step two to avoid incorrect measurements which take network latency into account that could be affected by environmental specific factors like signal interferences.

3.3.3 Experimental Setup

All measurements are based on open source implementations of each algorithm, separately compiled and deployed for each device, by using the Arduino IDE v.1.8.7 running on

64bit Linux Mint 19.1. We determine the performance and energy consumption of each algorithm by taking the arithmetic mean of results obtained after 100 runs, respectively. Performance results are expressed as *Throughput*. For block- and stream ciphers, as well as for the SHA hashing algorithms, throughput describes how many kilobytes can be processed in one second, denoted as (kB/s). For AES we measure the performance of the algorithm with different keysizes (128bit, 192bit, and 256bit) and different block modes of operation (CBC and CTR), respectively. Furthermore, three specific versions (based on encryption rounds) of the ChaCha algorithm are evaluated, namely ChaCha8, ChaCha12 and ChaCha20, both using a keysize of 256bit. Hence, for AES, we define a benchmark configuration as a combination of a specific mode of operation and key size. For the optimized tinyAES algorithm only the CTR mode of operation is evaluated, but with different keysizes, i.e., 128 bit and 256 bit. For each SHA family, we evaluate two different versions: The SHA2-256 and SHA3-256, which produce a hash with 256bit in size, and second, the SHA2-512 and SHA3-512 algorithm, which produce a hash with 512bit in size.

Key generation for ECC asymmetric algorithms were measured separately, as their throughput is measured in milliseconds, denoted as ms. ED25519 and P521 were evaluated by performing corresponding operations on a base64 encoded JSON file, 1 Megabyte in size, which acts as typical information structure an IoT device would send to the Edge or Cloud. Throughput measurements are all sent and processed and stored by a connected dedicated benchmark module. In order to measure power and current draw from the devices, we set up an additional dedicated energy measurement device. This device comprises an INA219 current sensor chip that is connected to an Arduino Uno Rev2 Wi-Fi and to the resource constrained device that is going to be evaluated. The Arduino processes the reading from the current sensor by using an I2C connection and sends the power values in watts to the benchmark-module.

For each benchmark run, a single data protection mechanism was evaluated. Each run consists of an idle and a load phase. During the idle phase, current and power of the device after startup and without processing anything were measured. In contrary, the load phase represents the device while executing the respective data protection mechanism. Power was measured in a timeframe of 120 seconds with a frequency of 100 milliseconds, resulting in a total of 1200 values. The average of the measured performance and energy consumption values were calculated accordingly. To ensure proper comparability of the results across all devices and respective security methods, the energy needed to process 1 Megabyte was calculated using the following formula:

$$E_1[mJ] = PowerIdle[mW] - PowerLoad[mW] * \frac{1MB}{Performance[B/s]}$$

Naturally occurring fluctuations were observed in base power consumption of the devices. To account for that, we calculated the difference between the idle and load phase.

MKR1000				
Type	Performance	Power Idle	Power Load	Energy 1 MB
AES				
CBC-128	93.13 kB/s	113.62 mW	113.84 mW	1222 mJ
CBC-192	78.14 kB/s	113.62 mW	113.53 mW	1453 mJ
CBC-256	67.17 kB/s	113.62 mW	113.64 mW	1692 mJ
CTR-128	90.51 kB/s	113.43 mW	113.51 mW	1253 mJ
CTR-192	75.18 kB/s	113.43 mW	113.51 mW	1510 mJ
CTR-256	65.77 kB/s	113.43 mW	113.44 mW	1725 mJ
tiny128-CTR	77.30 kB/s	112.17 mW	112.08 mW	1450 mJ
tiny256-CTR	56.48 kB/s	112.17 mW	112.05 mW	1984 mJ
ChaCha				
chacha8	897.27 kB/s	113.55 mW	113.58 mW	127 mJ
chacha12	764.40 kB/s	113.55 mW	113.61 mW	149 mJ
chacha20	589.75 kB/s	113.55 mW	113.61 mW	193 mJ

Table 3.3: Performance and Energy Consumption Results for Confidentiality Algorithms on the Arduino MKR1000

3.4 Results

This section provides our experimental results of evaluated data protection mechanisms regarding performance and energy consumption. It is followed by a discussion on the obtained values and how they can be interpreted. The results are put into corresponding categories, namely i) confidentiality, ii) integrity, iii) authenticity, and iv) key exchange. To increase readability we converted throughput rates to kB/s. A graphical representation of the results for confidentiality, integrity and elliptic curve operations for all devices can be found in Appendix A and are taken with permission from [Kai21].

3.4.1 Confidentiality

The following tables 3.3 - 3.6 display the results of evaluated configurations of AES algorithms (block cipher), ChaCha algorithms (Stream Cipher) and the optimized tinyAES (MCU optimized) variants for each device. Performance values in each row correspond to throughput, i.e., how many kilobytes of data can be encrypted/decrypted within 1 second.

As expected, performance of encryption and decryption scales linearly with CPU frequency. The immense increase in performance of AES ciphers on the ESP32 is due to the built-in hardware acceleration for AES. The used tinyAES version is not supported by the ESP32 and therefore not measured. For the ESP32, CPU frequency could be adjusted easily via the Arduino IDE if desired. This becomes particularly interesting if energy consumption

ESP32				
Type	Performance	Power Idle	Power Load	Energy 1 MB
AES				
CBC-128	1813.04 kB/s	308.68 mW	323.67 mW	8.27 mJ
CBC-192	1736.13 kB/s	308.68 mW	324.58 mW	9.16 mJ
CBC-256	1667.84 kB/s	308.68 mW	324.57 mW	9.53 mJ
CTR-128	1133.36 kB/s	309.08 mW	324.85 mW	10.41 mJ
CTR-192	1255.52 kB/s	309.08 mW	324.66 mW	10.67 mJ
CTR-256	1411.05 kB/s	309.08 mW	325.22 mW	11.44 mJ
ChaCha				
chacha8	5984.45 kB/s	308.22 mW	349.95 mW	6.97 mJ
chacha12	5218.02 kB/s	308.22 mW	350.63 mW	8.13 mJ
chacha20	4154.18 kB/s	308.22 mW	351.63 mW	10.45 mJ

Table 3.4: Performance and Energy Consumption Results for Confidentiality Algorithms on the ESP32

Raspberry Pi Zero				
Type	Performance	Power Idle	Power Load	Energy 1 MB
AES				
CBC-128	727.61 kB/s	585.39 mW	915.14 mW	453 mJ
CBC-192	609.21 kB/s	585.39 mW	916.22 mW	543 mJ
CBC-256	522.42 kB/s	585.39 mW	916.72 mW	634 mJ
CTR-128	706.07 kB/s	574.08 mW	913.86 mW	481 mJ
CTR-192	591.21 kB/s	574.08 mW	918.70 mW	583 mJ
CTR-256	511.80 kB/s	574.08 mW	918.43 mW	673 mJ
tiny128-ctr	641.25 kB/s	591.77 mW	917.76 mW	508 mJ
tiny256-ctr	462.72 kB/s	591.77 mW	919.73 mW	709 mJ
ChaCha				
chacha8	7193.61 kB/s	580.67 mW	938.51 mW	50 mJ
chacha12	5854.71 kB/s	580.67 mW	941.45 mW	62 mJ
chacha20	4271.43 kB/s	580.67 mW	943.23 mW	85 mJ

Table 3.5: Performance and Energy Consumption Results for Confidentiality Algorithms on the Raspberry Pi Zero

Arduino Due				
Type	Performance	Power Idle	Power Load	Energy 1 MB
AES				
CBC-128	145.16 kB/s	718.92 mW	725.73 mW	47 mJ
CBC-192	121.71 kB/s	718.92 mW	725.26 mW	52 mJ
CBC-256	104.13 kB/s	718.92 mW	724.78 mW	56 mJ
CTR-128	138.48 kB/s	719.31 mW	724.33 mW	36 mJ
CTR-192	116.56 kB/s	719.31 mW	724.36 mW	43 mJ
CTR-256	100.63 kB/s	719.31 mW	724.35 mW	50 mJ
tiny128-ctr	127.08 kB/s	720.16 mW	725.81 mW	44 mJ
tiny256-ctr	85.90 kB/s	720.16 mW	725.86 mW	66 mJ
ChaCha				
chacha8	1556.27 kB/s	718.91 mW	741.82 mW	15 mJ
chacha12	1376.72 kB/s	718.91 mW	743.47 mW	18 mJ
chacha20	1118.60 kB/s	718.91 mW	747.03 mW	25 mJ

Table 3.6: Performance and Energy Consumption Results for Confidentiality Algorithms on the Arduino Due

is of major concern in system design.

3.4.2 Integrity

Tables 3.7 - 3.10 show the results of integrity related data protection mechanisms. Similar to AES configurations, we evaluated different configurations (i.e., hash sizes of 256bit and 512bit) for SHA2 and SHA3 algorithms and their respective HMAC variants.

As for encryption and decryption speed of previously discussed algorithms regarding confidentiality, hashing performance also scales linearly with CPU frequency. However, the ESP32 is again remarkably faster than the other Arduino MCUs, which is also due to its hardware acceleration for SHA.

3.4.3 Authenticity and Key Exchange

Tables 3.11 - 3.14 provide results regarding the performance and energy consumption of authenticity and key exchange algorithms. For the purpose of better readability, results of the P521-512 based digital signature (authenticity) and elliptic curve operations needed for secure key exchange (curve25519 and P521) are combined into one table. The result values do not incorporate measurements for timings of the actual exchange process of a key, e.g., transmission over LAN or WLAN, as such timings would be prone to fluctuations caused by many different environmental factors, such as interferences

MKR1000				
Type	Performance	Power Idle	Power Load	Energy 1 MB
SHA				
SHA2-256	109.45 kB/s	114.04 mW	114.08 mW	1024 mJ
SHA2-512	67.83 kB/s	114.04 mW	114.01 mW	1681 mJ
SHA3-256	40.61 kB/s	113.88 mW	113.85 mW	2803 mJ
SHA3-512	40.42 kB/s	113.88 mW	113.89 mW	2818 mJ
SHA2-hmac-256	43.49 kB/s	113.04 mW	113.06 mW	2600 mJ
SHA2-hmac-512	17.21 kB/s	113.04 mW	113.15 mW	6574 mJ
SHA3-hmac-256	10.24 kB/s	112.20 mW	112.06 mW	10944 mJ
SHA3-hmac-512	10.26 kB/s	112.20 mW	112.07 mW	10927 mJ

Table 3.7: Performance and Energy Consumption Results for Integrity Algorithms on the Arduino MKR1000

ESP32				
Type	Performance	Power Idle	Power Load	Energy 1 MB
SHA				
SHA2-256	1190.44 kB/s	309.47 mW	345.15 mW	29.97 mJ
SHA2-512	621.36 kB/s	309.47 mW	347.25 mW	60.80 mJ
SHA3-256	304.63 kB/s	309.43 mW	350.95 mW	136.30 mJ
SHA3-512	304.50 kB/s	309.43 mW	350.04 mW	133.37 mJ
SHA2-hmac-256	469.38 kB/s	308.41 mW	344.99 mW	77.93 mJ
SHA2-hmac-512	155.86 kB/s	308.41 mW	347.60 mW	251.45 mJ
SHA3-hmac-256	75.73 kB/s	308.71 mW	350.14 mW	551.04 mJ
SHA3-hmac-512	76.08 kB/s	308.71 mW	349.70 mW	542.70 mJ

Table 3.8: Performance and Energy Consumption Results for Integrity Algorithms on the ESP32

or physical bandwidth limitations. Similarly, for the digital signature timings only the actual sign operation on the device was measured.

As for confidentiality and integrity related results, Ed25519 and P521 operations also scale linearly with CPU frequency. Also, hardware acceleration for ECC on the ESP32 and Raspberry Zero are causing the immense performance increase.

3.5 Discussion

In section, we briefly discuss the results of the conducted experiments. First, we discuss the implications of *more secure* variants of the evaluated algorithms on energy consumption.

Raspberry Pi Zero				
Type	Performance	Power Idle	Power Load	Energy 1 MB
SHA				
SHA2-256	1545.69 kB/s	571.65 mW	915.41 mW	222 mJ
SHA2-512	872.31 kB/s	571.65 mW	937.44 mW	419 mJ
SHA3-256	351.47 kB/s	574.96 mW	891.84 mW	902 mJ
SHA3-512	352.10 kB/s	574.96 mW	917.94 mW	974 mJ
SHA2-hmac-256	618.45 kB/s	569.38 mW	905.67 mW	543 mJ
SHA2-hmac-512	221.35 kB/s	569.38 mW	917.84 mW	1572 mJ
SHA3-hmac-256	87.88 kB/s	579.30 mW	909.42 mW	3757 mJ
SHA3-hmac-512	88.25 kB/s	579.30 mW	910.59 mW	3754 mJ

Table 3.9: Performance and Energy Consumption Results for Integrity Algorithms on the Raspberry Pi Zero

Arduino Due				
Type	Performance	Power Idle	Power Load	Energy 1 MB
SHA				
SHA2-256	364.45 kB/s	720.08 mW	736.83 mW	46 mJ
SHA2-512	145.09 kB/s	720.08 mW	742.69 mW	156 mJ
SHA3-256	71.17 kB/s	716.39 mW	734.97 mW	261 mJ
SHA3-512	71.16 kB/s	716.39 mW	734.43 mW	254 mJ
SHA2-hmac-256	138.87 kB/s	722.28 mW	737,14 mW	107 mJ
SHA2-hmac-512	36.77 kB/s	722.28 mW	743,55 mW	579 mJ
SHA3-hmac-256	18.20 kB/s	719.23 mW	734.89 mW	861 mJ
SHA3-hmac-512	18.33 kB/s	719.23 mW	734.27 mW	820 mJ

Table 3.10: Performance and Energy Consumption Results for Integrity Algorithms on the Arduino Due

Second, we discuss the implications on performance of those variants. Then, for both, energy consumption and performance, we further discuss their potential impact on design and development of a self-adaptive urban sensing system.

3.5.1 Implications on Energy Consumption

As expected, an increase in quality of the evaluated data protection mechanism (e.g., by increasing AES key-size from 192bit to 256bit) also results in significant higher energy demands of the device. For example, AES in Cipher Block Chaining (CBC) mode of operation with a key size of 256bit leads to an overall increase of energy consumption of 38% on the Arduino MKR1000. In general, for all evaluated data protection algorithms

MKR1000			
Type	Performance	Power Idle	Power Load
Key-Exchange			
curve25519	1113 ms	112.19 mW	112.23 mW
P521	16494 ms	113.73 mW	113.57 mW
Digital-Signature			
P521-512	10444 ms	112.96 mW	114.84 mW

Table 3.11: Performance and Energy Consumption Results for Authenticity and Key Exchange Algorithms on the Arduino MKR1000

ESP32			
Type	Performance	Power Idle	Power Load
Key-Exchange			
curve25519	52 ms	307.15 mW	345.34 mW
P521	675 ms	330.04 mW	404.29 mW
Digital-Signature			
P521-512	437 ms	308.77 mW	310.01 mW

Table 3.12: Performance and Energy Consumption Results for Authenticity and Key Exchange Algorithms on the ESP32

energy consumption did not vary a lot. This may well be due to the fact that the evaluated MCUs most of the time execute the different data protection algorithms with near 100% CPU usage. Most interestingly, the tinyAES variant did not offer the expected savings in terms of energy consumption. However, we have to state that there are other implementation variants of this optimized AES algorithm, hence such variants should also be evaluated if energy consumption would be of top-most priority in system design. Additionally, in a typical urban sensing system, there are often hundreds of such MCUs deployed, gathering all kind of data. Hence, also tiny differences in energy consumption will add up in the end and become increasingly significant.

3.5.2 Implications on Performance

In general, our results support the claim that the performance of stream ciphers is superior to the performance of block ciphers. Interestingly, on the devices with lowest computing power (i.e., Arduino Due and Arduino MKR1000), the tinyAES algorithm provides only minor benefits to performance compared to standard AES implementations. However, for increasingly powerful devices this difference becomes more significant, e.g., on the Raspberry Zero the tinyAES variant is about 10% faster than its standard counterpart.

Raspberry Pi Zero			
Type	Performance	Power Idle	Power Load
Key-Exchange			
curve25519	64.65 ms	586.08 mW	940 mW
P521	585 ms	546.36 mW	883.66 mW
Digital-Signature			
P521-512	387 ms	417.34 mW	932.55 mW

Table 3.13: Performance and Energy Consumption Results for Authenticity and Key Exchange Algorithms on the Raspberry Pi Zero

Arduino Due			
Type	Performance	Power Idle	Power Load
Key-Exchange			
curve25519	202 ms	730.32 mW	737.86 mW
P521	2714 ms	716.82 mW	728.27 mW
Digital-Signature			
P521-512	1674 ms	640.20 mW	638.96 mW

Table 3.14: Performance and Energy Consumption Results for Authenticity and Key Exchange Algorithms on the Arduino Due

Hardware acceleration plays a major role when interpreting the measured performance values. On devices without hardware acceleration for a specific data protection algorithm, performance significantly decreases if the data protection quality of the algorithm is increased. For example, on the Arduino MKR1000, the performance decrease from SHA2 with a hash size of 256bit to a hash size of 512bit is about 40%. On the ESP32, the relative decrease in performance is roughly the same as for the Arduino MKR1000. However, on the ESP32, due to it hardware acceleration for SHA2 the absolute difference compared to the Arduino MKR1000 is close to 1 order of magnitude. This absolute difference is even higher for AES variants, also due to the hardware acceleration support for AES on the ESP32.

3.5.3 Implications on Self-Adaptive Urban Sensing Systems

In terms of security quality, i.e., how *safely* is sensitive data protected by a specific data protection algorithm, we have to state that for most urban sensing systems that incorporate MCUs like the ones we evaluated in this thesis, a key size of 128bit for AES would be more than sufficient. A key size of 256bit mostly becomes increasingly relevant for the so called post-quantum era, where quantum computers are used to *break* such

a security algorithm. In general, the quality of popular security algorithms is based on one of three hard mathematical problems: the integer factorization problem, the discrete logarithm problem or the elliptic-curve discrete logarithm problem. However, all of these problems can be easily solved on a sufficiently powerful quantum computer running Shor's algorithm [Ber09]. This concern has to be accounted for when defining the criticality of data leakage in an urban sensing system and compared against the potentially significant increases in energy consumption; assuming there may be hundreds of MCUs deployed in such a system. As expected, data protection tasks based on asymmetric cryptography take up a lot of execution time. However, prominent tasks, such as symmetric key exchange, typically not occur frequently, e.g., once if a new connection to another node is established. If the authenticity of data is of importance in an urban sensing system, i.e., many signing tasks are executed frequently, our results strongly indicate using devices with dedicated hardware acceleration, such as the ESP32, for those systems. In general, an urban sensing system engineer should opt for MCUs with dedicated hardware acceleration if data is processed frequently and data protection is prioritized over energy consumption. However, our results also suggest that there is great potential for the development and implementation of adaptation mechanisms for such systems, even if they incorporate resource constrained devices, such as MCUs. This becomes especially relevant for systems that deal with irregular data readings, e.g., more frequent readings at day but less frequent readings at night. In such cases, data protection, for example, could be increased during night, or, on the other hand, energy consumption could be reduced (e.g., on the ESP32) by throttling CPU speed. The recently emerged *tinyML* paradigm would be a concrete example that could benefit from such adaptation mechanisms. In tinyML resource constraint devices, e.g., microcontrollers as described in this chapter, are used to analyze sensor data based on machine learning models executed on the device itself. Depending on the particular use case, the transmission of either sensed or processed data to e.g., a centralized edge computing node, for instance to aggregate data, should be protected as well. Prominent urban sensing domains incorporating the tinyML paradigm are for example:

- Manufacturing: Predictive maintenance potentially reduces downtime and costs associated with equipment failure.
- Agriculture: tinyML devices can be used to get real-time data by monitoring crops or livestock.
- Healthcare: Real-time health monitoring enabled by tinyML devices can deliver better and more personalized patient care, for example in tele-medicine. A typical example would be patient monitoring, where analyzed data is transmitted to e.g., a doctor. Although this would not be considered a typical urban sensing application, hearing aids would be a another concrete example.

In those examples it is quite likely that either sensed data (Machine to Machine (M2M) communication) or analyzed data transmitted to a centralized compute node needs to

protected. Hence, such systems would benefit by balancing their energy consumption and performance.

3.6 Summary

In this chapter we presented an evaluation of selected, well established data protection mechanisms, including cryptographic block and stream ciphers, secure hashing algorithms, digital signature algorithms and algorithms needed for key exchange protocols, i.e., elliptic curve operations. Specifically, we measured how those algorithms performed, in terms of computational effort and energy consumption, on representative state-of-the-art resource constrained IoT devices. Our measurements provide results that can aid the design and development of secure IoT systems incorporating resource constrained devices. We provide values for different limitations regarding the configuration and execution of several algorithms on certain devices. Furthermore, we provide results on data throughput rates and energy consumption for each device. Those results can be used to calculate data protection/performance/energy consumption trade-offs for a specific hardware configuration. Our evaluation shows that an IoT system running applications relying on certain data protection mechanisms will largely benefit from incorporating microcontrollers that come with built-in hardware acceleration for several cryptographic tasks. Especially for ECC related tasks, like signatures and key exchanges, hardware acceleration has the most significant impact. The provided measurements and potentially subsequently implemented adaptation mechanisms based on those measurements could potentially increase the data protection aspects of many urban sensing applications. TinyML is a concrete domain in which many use cases would most likely benefit from such mechanisms, because most of them typically incorporate the exact same devices used in the evaluation presented in this chapter.

CHAPTER 4

ORIOT: A Source Location Privacy System for Resource Constrained IoT Devices

Privacy and Security are one of the major research topics regarding the Internet of Things and urban sensing in general. Due to the vast amount of devices collecting and processing sensitive data, anonymity and privacy mechanism are needed. Source Location Privacy plays a key role in prohibiting adversaries from tracing back this kind of data to its origin. In this chapter we propose a SLP preserving system that leverages techniques from the well established Onion Routing paradigm. The system is specifically designed for resource constrained IoT devices, i.e., devices lacking computing power. It features combined encryption schemes and symmetric key exchanges via ECDH. Our performance measurements, conducted on typical resource constrained IoT devices, show the feasibility of ORIOT and facilitate the integration into existing or planned urban sensing systems, depending on SLP features.

After a brief introduction, the rest of the chapter is organized as follows. Section 4.2 provides an overview about related work on IoT security and privacy. In Section 4.3, we introduce our proposed implementation. We evaluate our approach and discuss the results in Section 4.4. Finally, in Section 4.5 we conclude the chapter and give an outlook on future research.

4.1 Introduction

The Internet of Thingss describes a heterogeneous network comprising a variety of different connected devices with minimal to average computing power. These devices continue to permeate deeper in our personal environment as well as in commercial and

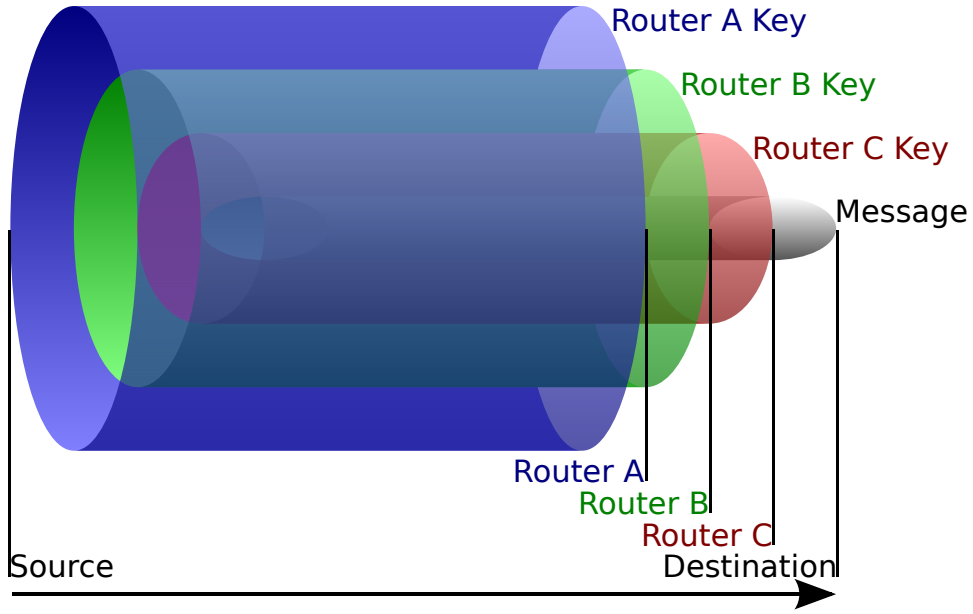

Figure 4.1: Layered encryption principle of Onion Routing[1]

industrial areas, by sensing, processing, and storing all kind of data [VF13b]. For many applications, like in healthcare, home automation or infrastructure monitoring, these circumstances call for privacy and security protection [DWK15b]. Integrity, confidentiality, availability, undetectability, and unobserveability are the key elements of such protection mechanisms. Though features of these elements overlap, according to [Fea15b], we place integrity, confidentiality, and availability into the security domain; undetectability and unobserveability into the privacy domain. Regarding privacy, we further distinguish between data-anonymity and Source Location Privacy. Broadly speaking, (personal) data provided by IoT devices can be used by adversaries to obtain or derive sensitive information that could compromise users. Data-anonymization techniques offer a solution to mitigate such privacy breaches. SLP, as the name implies, aims to keep the location private, where data was originally collected. Referring to IoT, being an interconnected network, this would in most cases result in efforts to keep the IP-address of a device private. A well established approach to achieve this goal is Single Path Routing (SPR). Data packets are routed to their final destination following a specific path inside a network to make it harder for an adversary to trace back their origin. Mix-cascades and onion routing are prominent concepts for SPR, where DC-nets and Tor are its most popular implementations. In this work we leverage principles of onion routing, where data is encapsulated in multiple layers of encryption, hence the analogy to an onion. Fig. 4.1

[1] https://en.wikipedia.org/wiki/Onion_routing#/media/File:Onion_diagram.svg

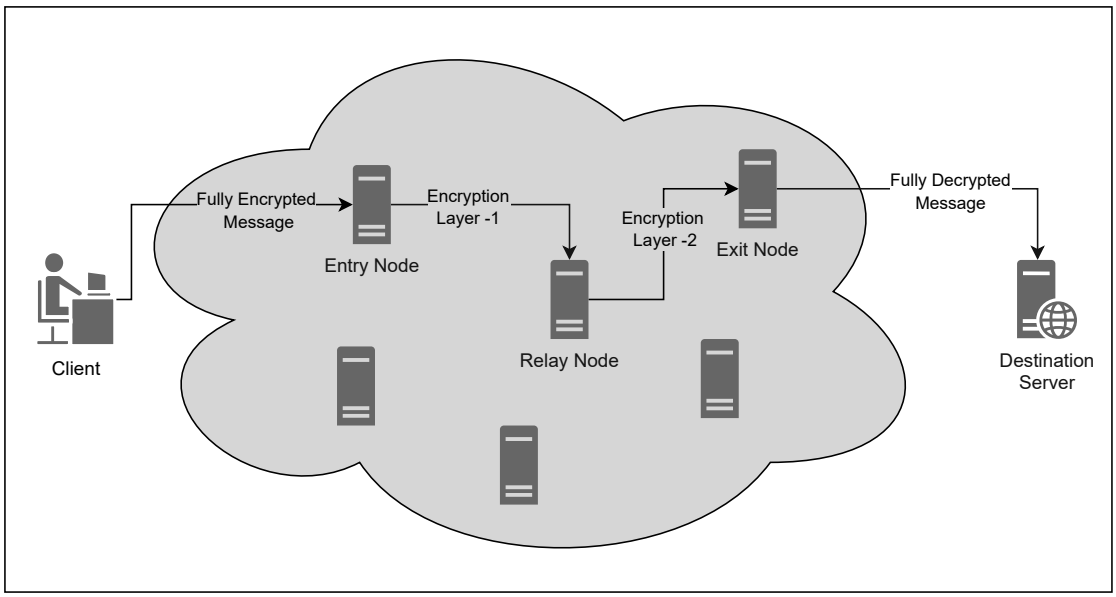

Figure 4.2: Onion Routing path with decreasing encryption layers from client to destination server

displays the layered encryption principle of Onion Routing. The multiple times encrypted data is then routed along a predefined path to its destination. Such a path consists of various nodes called *onion routers* or *relay nodes*. Each intermediary node removes one encryption layer and thereby only reveals the address of the next node in the route. Therefore, each node only knows the location of its predecessor and successor node. This mechanism facilitates sender anonymity. Fig. 4.2 displays a high level representation of such a path, where a single encryption layer is removed at each intermediary server along the path from client to the destination server. However, several weaknesses have been found to break this anonymity, like Timing or Traffic Analysis. The design of ORIOT is based on typical IoT systems, e.g., Amazon AWS IoT, where data is generated at various nodes inside a network and sent to the cloud for further processing.
Many IoT devices are constrained by their available resources, i.e., in most cases computing power, like microcontrollers, that are not running any operating system. Therefore, well established implementations of Onion Routing, e.g., Tor, are infeasible for such devices. The main contribution of this work is a SLP system, specifically designed for resource constrained IoT devices, to address this issue. It is implemented in C, and therefore highly compatible and portable to most IoT devices. Furthermore, we provide performance results on different cryptographic mechanisms that are integral parts of our system. Experiments have been conducted on typical resource constrained IoT devices, therefore our results facilitate the design and development of IoT systems that rely on SLP features, e.g., by implementing ORIOT.

4.2 Related Work

With the increasing spread and usage of IoT devices, security and privacy aspects have become a major research topic in the area of IoT data protection. Suo et al. [Sea12] state that there are four abstraction layers in IoT. Bottom-up these are: Perception Layer, Network Layer, Middleware Layer and Application Layer. In their summarizing paper, Farooq et al. [Fea15a] give an overview about possible threats and scenarios on these different layers, where the majority of threats are located in the Network Layer. Energy consumption and management, as well as efficient computing algorithms (e.g., share of workload among multiple devices) play a key role for resource constrained IoT devices. Therefore, for the majority of use cases concerning privacy and security aspects, trade-offs have to be made either at design time or runtime. An example of such a trade-off could be a stronger encryption scheme resulting in lower data throughput. In this work we want to tackle those issues and minimize such trade-offs. In the domain of Wireless Sensor Network various solutions to problems dealing with SLP or anonymization have been proposed. Most of the devices in WSN reside at the lowest end regarding computing power and represent a subset of IoT devices. Commonly those devices adhere to the IEEE 802.15.4 protocol, using communication frameworks like ZigBee. Besides being part of a WSN, they are also integrated in smart objects such as smart phones, tablets, smart watches, and many others gadgets [VF13b]. Security and privacy mechanism often require considerable computing power that cannot be provided by such devices. A typical pragmatic solution is the usage of *IoT gateways* that are placed between (sensor) networks and the Internet, powerful enough to facilitate more compute-intensive security and privacy mechanisms [Fea15b, HP15]. However, there may be situations where such gateways are not desired or possible. A well-known example in literature is the Panda-Hunter Game, where a WSN is deployed in a forest to monitor pandas. Hunters take the role of an adversary, trying to capture the panda. The goal is to prevent the hunter from locating the source, i.e., the sensor attached to a specific panda [Kea05]. Generally speaking, privacy can be either achieved by leveraging data-anonymization techniques, SLP-mechanisms, or a combination of both. Researchers have investigated several anonymization techniques, such as simple pseudonymization, attribute suppression, or more sophisticated approaches like the k-Anonymization model [oS19, LL18, OPD16, Cea11, ZO11]. However, most of these techniques do not incorporate SLP features, especially not for resource constrained devices. Jebri et al. [JAB15] propose a generic security and privacy model for IoT and WSN that includes SLP. Their work is based on a lightweight key agreement protocol, Identity Based Encryption (IBE), and Pseudonym Based Cryptography (PBC). To make use of an IBE system the authors had to incorporate a Private Key Generator (PKG) that acts as a trusted central key authority. PBC is a technique based on IBE, and is generally used to protect the identity of an entity. The architecture comprises a base station, a sink node, and a set of nodes. The PKG is integrated into the base station which stores the identities of the nodes. Before any data is sent over the network, a setup phase takes place in which several encryption and privacy mechanisms, e.g., generation of private and public keys, are configured. In

order to transmit information, each node sends its data, protected by calculated session keys, directly to the Sink Node (SN). Due to the use of PBC, the identity of the source node stays anonymous. Another concept which requires less encryption overhead is Anonymous Routing (AR). AR is a well suited concept to achieve SLP. There are many ways to implement, integrate, and extend this concept for applications that operate with sensitive data where the source has to stay anonymous. Palmieri et al. [PCM17] proposed a protocol for AR between different interconnected networks. It is designed specifically for IoT applications and is based on the Spatial Bloom Filter data structure. Furthermore, all routing information is encrypted using an additive and multiplicative homomorphic encryption scheme. However, as stated by the authors, this cryptographic system may not be suitable for computationally constrained devices. Another protocol that specifically targets resource-constrained mobile ad hoc networks, was proposed by Moldovan et al [MIG13]. Their group-based anonymous on-demand routing protocol works in a similar way to Tor. After detecting all nodes in network, a secret handshake with all nodes is performed by a dedicated trusted network administrator. Afterwards, for two neighboring nodes a secure common key is computed. Further cryptographic processes ensure resistance against different attacks, e.g., Message Coding Attacks. Specific request and response messages, which are partially broadcasted inside the network, are used to establish a communication path between source and target nodes, comprising pairs of securely linked nodes. Each node is known to others under a pseudonym, which is used to forward a message along the path, while keeping private both source and target. Referring to SLP in IoT, we assume that the source location relates to an IP-Address of a device. Especially in WSN, SLP problems are closely tied to real geolocation privacy of an entity, but the used techniques are similar and related to IP-Address privacy. To achieve geolocation privacy, Mutalemwa et al. [MS18] proposed a strategic location-based random routing approach. According to their scheme, data packets are encrypted and routed over the network according to the physical location of a source node. To determine a routing path, intermediary strategically positioned diversion nodes are randomly selected based on their distance to the source node. Successive packets are routed through different routing paths. Simulation results demonstrate that their approach makes it difficult and confusing for an adversary to trace back the origin of such data packets. In IoT, especially when dealing with resource constrained devices, such security and privacy mechanisms require several trade-off decisions to be made, as stated earlier. Techniques that leverage broadcasting mechanisms or rely on heavy network traffic in general will automatically cause a higher energy consumption of all devices. Computational intensive encryption mechanisms on the other hand are infeasible for scenarios where data is sent over the network with high frequency.

In this work we combine several above mentioned security and privacy techniques and incorporate them into an onion routing system, similar to Tor, targeted for resource-constrained IoT devices, to address the above mentioned trade-offs. Compared to other approaches, our system does not rely on heavy cryptographic algorithms to provide anonymity. On the other hand, ORIOT avoids network broadcasting strategies as used by different proposed SLP systems. By setting a specific path length for our message

transfer, a well balanced trade-off between network load and SLP level is achieved.

4.3 Onion Routing System

In this section we describe our proposed onion routing system called ORIOT. First we present an overview of the systems architecture. Second we explain the setup phase that is performed by a device when integrated into the system. Third we describe the path assembling strategy that is performed before sending data. Finally we present the encryption and actual routing processes.

4.3.1 Overview

The architecture comprises multiple devices (nodes) which are constrained in their computing power, and a single more powerful device acting as a central form of registry, therefore simply called *Registry*. Figure 4.3 shows the overall architecture. We proceed to explain each step, as marked by the corresponding number in blue circles.

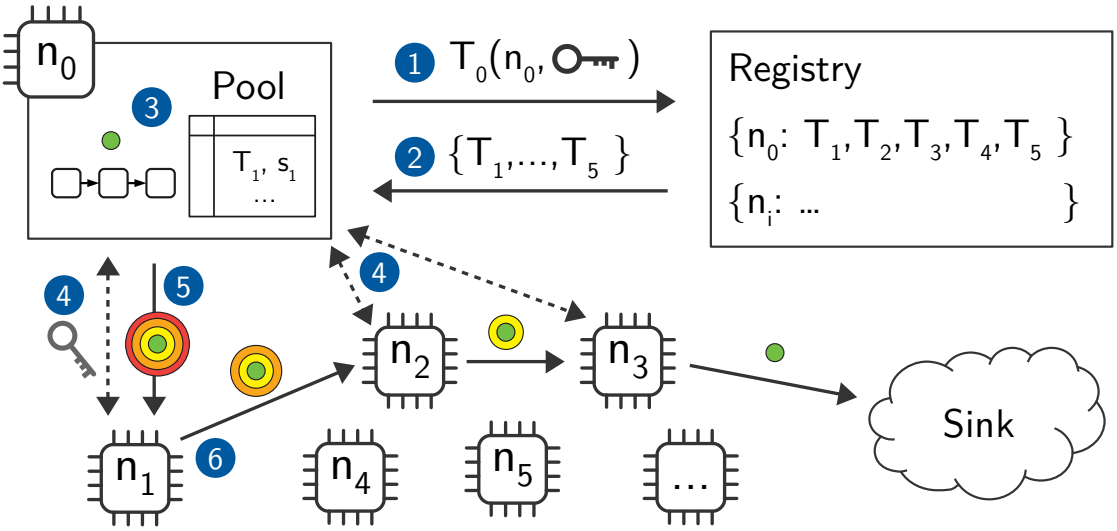

Figure 4.3: Overview of the ORIOT architecture

Devices that are part of a network are called nodes (depicted as $n_0, n_1, ...$ in Figure 4.3), which collect or process data provided by various sources such as sensors or system events. In our experiments we use various microcontroller development boards to act as our typical resource constrained IoT devices. In our experimental setup, all devices are part of a local network and are fully connected to each other. The systems' architecture focuses on mechanisms implemented for and operating on the Transportation Layer of the OSI model. The idea is, that if a node wants to send a message to a specific destination (e.g., the cloud), the data is encrypted several times (depicted as layered circles in Figure 4.3) and sent to the destination via multiple hops (depicted as continuous arrows in Figure 4.3) across the network. In our setup, each node sends its data to a

specific destination, e.g., a cloud server or sink. In WSN terminology this would refer to a *sink node*. Preliminary, nodes are provided with the public key of the cloud server. All messages are initially encrypted in a way so that only the cloud server can read the message (end-to-end encryption). However, our system design is not limited to sending messages to only one particular destination. The first step for a node that wants to send a message, is to ask the Registry for a designated *IP-Pool*. This IP-Pool comprises tuples of IP-Address and corresponding public key of nodes in the network. After receiving the IP-Pool, the node randomly selects a subset of the given tuples, which will represent the path along data is subsequently routed and transferred to its final destination. Because each node needs to be able to connect to any other (random) node in the network, it is necessary that each node is fully connected to each other, as mentioned previously. There may be deployment scenarios this may not be feasible or even possible. Technically, it would be possible to implement ORIOT for not fully connected nodes also. However, this would need specific implementation aspects to be considered and also decreases security aspects (as the randomness of the path is constrained by this limitation) and is out of scope of this work. A more detailed description of this IP-Pooling mechanism will be provided in the path assembling section later on. Similar to Tor, the first node of a routing path acts as *Entry Node* (the only node which is actually able to see the source IP-Address) and the last node acts as *Exit Node* which sends the data to the final destination. A layered encryption strategy ensures that every node along a routing path only knows the IP-Address of its predecessor node and its successor node.

4.3.2 Setup Phase

The setup phase is a specific piece of code that is executed only once when a device is started. It can be divided into two essential steps:

Encryption Setup

In our setup we incorporate an end-to-end encryption scheme, i.e., only our designated destination cloud server is able to read a message in plaintext. Similar to TSL we use a combination of asymmetric and symmetric encryption. First, a node generates a random secret key which is later on used to encrypt the message. Second, the generated key is encrypted with the public key of the cloud server. With its private key, the cloud server is able to decrypt the shared secret which is then used to decrypt the actual message.
The next step in setup phase is the generation of a public and private key pair. As described later, those are used for our layered encryption scheme.

Network Setup

After the encryption setup has finished, every node publishes its previously generated public key to the Registry, which saves this information as a tuple of $< PublicKey_{Node}, IP\text{-}Address_{Node} >$ in a list. The Registry then randomly adds those tuples to a specifically sized pool (e.g., 10 tuples stored in one pool), depending on the

size of the network, and the available storing capacities of the device hosting the registry. However, a pool must contain at least a minimum of three nodes. Three nodes is also the standard used by the well established TOR project[2] and stems from the fact that with an n-relay onion routing path, every byte transmitted is encrypted n times by your entry node, and every byte coming back from the sink is decrypted n times. While this would increase security (better chance to compensate for rogue relay nodes) it also comes with a hefty cost in terms of performance and energy consumption, which is both typically limited in the domain tackled in this work. To further increase the level of anonymity we recommend using one specific pool for each node in the network, although pools could be reused if storage on the registry node is limited. Pools should be randomly reorganized, based on a configurable *staleness factor*. The actual value of this staleness factor (e.g., 10 minutes) is determined by the expected traffic over the network, i.e., how frequently data will be sent from a node. After pools are created, every node in the network will be assigned one pool.
After the encryption and pooling steps are completed, a node starts listening for incoming data. To that end, the node opens a predefined common port and waits for specific instructions. All communication (except for the key exchange described in the path assembling section, which is based on TCP) is ideally based on, but not limited to, UDP because of three major reasons:

- A node only sends data and never expects an answer (except during the key exchange).
- UDP has a noticeable network performance advantage over TCP.
- UDP uses less energy than TCP mostly due to the decreased amount of data that need to be transmitted.

However, the usage of UDP also comes with downsides, most notably it is less reliable as the delivery of data to the destination is not guaranteed and UDP also only comes with basic error checking mechanisms using checksums. Moreover, there is no sequencing of data in UDP, hence reordering data (if required) has to be done by the application. Regarding energy consumption, adding a security for both protocols (TSL for TCP and DTSL for UDP) increases overall energy consumption, where DTSL requires a more significant amount of additional energy compared to TSL. However, there are also lightweight alternatives proposed in literature, such as [BWJ+19, HASW17, MEB16], that could be used as transport layer for ORIOT.

4.3.3 Path Assembling

A routing path comprises five nodes in total. We refer to the first node as the source node and the last node as the cloud server. Intermediary nodes are called relay nodes. Though

[2] https://www.torproject.org/

it would be possible to add more than three relay nodes to a path, we advise against it, as this increases network load without providing any more security or anonymity [Pro19]. In our example the IP-Address of the cloud server is known to each node, therefore it is not a part of the path assembling scheme. If a node wants to send a message, it opportunistically starts building a transfer route. Opportunistically means, that to this end, a node building a path will not know if another node, that will be selected as part of this path, is actually online or working properly. When assembling a routing path, the node randomly selects exactly three tuples out of its stored IP-Pool. These tuples correspond to our relay nodes and will form the intermediary path to the cloud server. For each relay node, the source node starts a key exchange (depicted by dashed arrows in Figure 4.3) by sending a *InitiateKeyExchange* request. If an addressable relay node receives this request, the process of generating a common symmetric key is initiated. If no response is received by the source node after a predefined timeout, it removes the corresponding tuple out of its IP-Pool and notifies the Registry about the faulty node. If there are too many faulty nodes (this threshold can be adjusted at design time), a node requests a new pool from the registry. However, if the node receives a response, the common symmetric key generation is established via the ECDH key exchange protocol [DH76]. All calculated symmetric keys are stored for each relay node for the layered encryption process later on. It is up to the developer if and how long previously negotiated symmetric keys are cached on the source node and the relay nodes for later reuse or not. This becomes particularly interesting if a node in the network is replaced, possibly resulting in an IP-Key (either asymmetric or shared) mismatch. The corresponding pseudocode is shown in Algorithm 4.1, but does not cover any caching mechanism or requests for a new entire pool.

Algorithm 4.1: Path Assembling

Input: pool $[T_0, T_1, T_2, ...T_n]$
Result: path[3]$< T, sharedKey >$ of tuples T_x where $x \in [0..n]$, and corresponding shared keys, respectively

```
1  while count(path) < 3 do
2      rT = getRandomTupleFromPool;
3      if path !contains(rT) then
           // perform ECDH key exchange
4          sharedKey = InitiateKeyExchange(rT);
5          if sharedKey then
               // save established key to corresponding tuple
6              keyAdd(path, sharedKey);
7          else
               // handle timeout and maxFaults
8          end
9      end
10 end
```

4.3.4 Encryption and Routing

After a routing path has been determined, the message encryption process is initiated. The corresponding pseudocode, denoted in Algorithm 4.2 covers the encryption layering process. First the message M is encrypted with the key K generated in the setup phase resulting in the message M_K.

The encrypted message for the cloud server, i.e., the inner-most layer of the onion L_0, will be in the form of:

$$L_0 = \{M_K, K_{EC}\} \tag{4.1}$$

where K_{EC} is K encrypted with the public key of the cloud previously done in the setup phase.

After that the first layer L_1 of encryption is added to L_0 in Form of:

$$L_1 = Enc_{KN_2}\{L_0, IP_{CS}\} \tag{4.2}$$

Subsequently, for each remaining node, an additional corresponding encryption layer will be added in form of

$$\begin{aligned} L_2 &= Enc_{KN_1}\{L_1, IP_{N_2}\} \\ L_3 &= Enc_{KN_0}\{L_2, IP_{N_1}\} \end{aligned} \tag{4.3}$$

where Enc_{KN_i} is an encryption function with the symmetric key previously exchanged with node N_i and IP_{N_i} is the IP-Address of the successor relay node, with $i \in [0, 1, 2]$.

Algorithm 4.2: Layered Encryption

```
   Input: msg, path[3], cloudPubKey, rndKey, cloudIP
   Result: Multiple Encrypted Message (Onion)
   // encrypt message for cloud server
 1 mk = encryptCloudMessage(msg, rndKey);
 2 kec = encryptKey(rndKey, cloudPubKey);
 3 lyr = composeCloudMessage(mk, kec);
 4 ip = cloudIP;
   // add encryption layers
 5 for i = count(path) - 1 .. 0 do
 6 |   key = getKeyFromNodeInPath(path, i);
 7 |   if i != 2 then
 8 |   |   ip = getIPFromNodeInPath(path, i+1);
 9 |   end
10 |   addLayer(lyr, ip, key);
11 end
```

Device Name	Processor	CPU Speed	SRAM	Flash Memory
Arduino MKR1000	Cortex-M0+ 32-Bit	48 MHz	32 KB	256 KB
Wemos ESP8266	Xtensa LX106 32-Bit	80-160 MHz	160 KB	4 MB
Espressif ESP32	Xtensa LX6 32-Bit	160-240 MHz	512 KB	4 MB

Table 4.1: Testbed Overview of Resource Constrained IoT Devices

	ECDH_1	ECDH_3	AES-256	ChaCha20-256
MKR-1000	0.492s	0.501s	68.04kB/s (14.35μs/B)	543.94kB/s (1.80μs/B)
ESP8266	0.082s	0.097s	211.56kB/s (4.62μs/B)	3,149.09kB/s (0.31μs/B)
ESP32	0.026s	0.027s	1,859.84kB/s (0,53μs/B)	4,078.84kB/s (0,24μs/B)

Table 4.2: Performance Results of Cryptographic Algorithms on a single Node

The final multiple encrypted message will be sent from the source node to the first relay node via a specific *ForwardMessageRequest*. If a relay node receives such a request it will then be able to decrypt one layer with its symmetric key and will forward the message to the next relay node, respectively, or in case of the last relay node, to the cloud server.

4.4 Performance and Boundaries

This section presents performance measurements and resulting boundaries of ORIOT. Our testbed comprises three microcontroller development boards with integrated WiFi capabilities, acting as typical resource constrained IoT devices. Table 4.1 provides an overview of the selected hardware.

A prototypical implementation was developed in *C* using the Arduino IDE v.1.8.7 running on 64bit Linux Mint 19.1. Code execution on such microcontrollers is divided in a setup phase (where code runs only once), and a loop phase. We are particularly interested in the performance of cryptographic functions, rather than round trip times or data transfer (e.g., pools) which are heavily prone to network latency affected by various unstable environmental factors like signal strength or interferences. In the setup phase, we investigate key generation, specifically a 256bit private and public key pair via Ed25519 elliptic-curve cryptography. Symmetric Key exchange (ECDH) and encryption layering is done in the loop phase. The first phase of ECDH is the generation of a public and private key (ECDH_1), already done in the setup phase. In the second phase of ECDH the public keys are exchanged between the two parties. The performance of this step relies solely on network traffic, therefore it is not covered by our measurements. In the

third phase (ECDH_3), a common secret key is derived from calculations that take the secret key of a communication partner and the previously exchanged public key of the other partner as input. Due to the nature of the algorithm, the third phase of ECDH performs similar to the first phase, but for better readability we measure and present it separately.
All symmetric encryption, i.e., creating the onion, is done using the AES block cipher with a 256bit key in Counter (CTR) mode of operation. Additionally, we measured the layering process using the ChaCha20 stream cipher with a 256bit key, as a lightweight alternative. However, we remark that modern microprocessors, like the ESP32, come with built in hardware acceleration capabilities for AES and ECC.
Table 4.2 displays the obtained results. The performance of symmetric ciphers is expressed as *limitation* and *throughput*. Limitation corresponds to the time it takes for an algorithm to process one byte of data, given in μs per byte (µs/B), while throughput describes how many bytes can be processed in one second (B/s). With provided throughput and limitation values, boundaries for a specific network can be then calculated individually. If energy consumption is a concern, many corresponding values can be obtained from the results presented in chapter 3, section 3.4. For symmetric ciphers, values outside brackets correspond to limitation, while values in brackets correspond to throughput. Each of those values correspond to adding one layer of encryption. To get a close approximation of the time needed to create all layers of encryption, i.e., the whole onion, the results need to be multiplied by the number of layers. Due to the nature of the used symmetric ciphers, time needed for encryption is almost exactly the same as for decryption. Values of asymmetric operations represent the time needed in seconds for the whole operation to finish, be it either key generation or deriving a common symmetric key.
The execution time of the cryptographic algorithms scale linearly with CPU frequency. This becomes particularly interesting if energy consumption is a critical design aspect of an IoT system using ORIOT. Devices like the ESP8266 and ESP32 can easily be underclocked, i.e., running the CPU at a lower frequency, hence consuming less energy.

4.5 Summary

This chapter presents a source location privacy preserving network system and its architectural concepts, specifically designed to operate on resource constrained IoT devices. Similar to Tor, it leverages techniques of the onion routing principle. Symmetric and asymmetric cryptography are combined with a path assembling strategy to realize anonymity of a node transmitting messages in a network to a specific destination. In contrast to other proposed solutions, ORIOT does not rely on either heavy cryptographic operations (that may not even be supported by a device) or broadcasting strategies, as stated in section 4.2. However, ORIOT (while not strictly limited to) relies on a fully connected network of nodes, which could be a limitation for some applications. Depending on the specific nature of an application, a developer may also opt to use TCP over UDP and/or its security layers TSL and DTSL respectively. However, this also significantly influences energy consumption and performance as the amount of data that will be transmitted

increases or decreases. We evaluated the performance of incorporated cryptographic algorithms on a set of typical resource constrained IoT devices. Similar to Tor, performance mainly depends on the latency and computing power of relay nodes. Our results provide insights on the performance limitations and throughput, which can facilitate the design and development of an IoT system implementing ORIOT. Additionally, from those results it is clearly visible that devices with hardware acceleration for security operations are best suited for such applications. As seen in chapter 3, section 3.4, hardware acceleration is also beneficial regarding energy consumption. However, as most security algorithms scale with CPU frequency, if the transmission rate of data required by an application is low (e.g., every other hour), devices that do not feature hardware acceleration or come with lower CPU performance may be better suited for such applications if they also consume less overall energy either in idle mode and during operation. This can also be achieved by underclocking specific devices. We deliberately chose to not measure overall round trip time or individual data transfer timings, because those values are typically heavily prone to network latency affected by various unstable environmental factors like signal strength or interferences. Furthermore, such values would also heavily differ depending on the connection type used, for example wired versus bluetooth versus wireless.

CHAPTER 5

A Privacy Preserving System for AI-assisted Video Analytics

The emerging Edge computing paradigm facilitates the deployment of distributed AI-applications and hardware, capable of processing video data in real time. AI-assisted video analytics can provide valuable information and benefits for parties in various domains. Face recognition, object detection, or movement tracing are prominent examples enabled by this technology. However, the widespread deployment of such mechanisms in public areas is a growing cause of privacy and security concerns. Data protection strategies need to be appropriately designed and correctly implemented in order to mitigate the associated risks. Most existing approaches focus on privacy and security related operations of the video stream itself or protecting its transmission. In this chapter, we propose a privacy preserving system for AI-assisted video analytics, that extracts relevant information from video data and governs the secure access to that information. The system ensures that applications leveraging extracted data have no access to the video stream. An attribute-based authorization scheme allows applications to only query a predefined subset of extracted data. We demonstrate the feasibility of our approach by evaluating an application motivated by the recent COVID-19 pandemic, deployed on typical edge computing infrastructure.

After a brief introduction, the rest of the chapter is structured as follows: In Section 5.2 we investigate and compare our approach to related work in the field, and describe the major differences to proposed solutions. Section 5.3 describes the exemplary scenario motivating our approach. The following Section 5.4 gives a detailed overview of our system. An exemplary implementation showcasing and evaluating the system is presented in Section 5.4.1. Finally, Section 5.5 concludes the chapter.

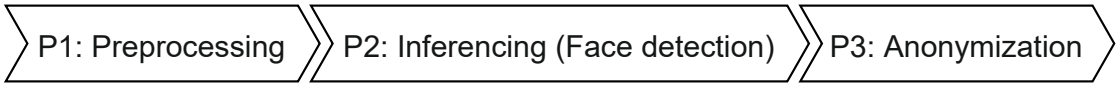

Figure 5.1: Parts of a Video Analysis Pipeline featuring a face-anonymization application

5.1 Introduction

Real-time video feeds from urban areas in combination with AI-based processing techniques provide exciting opportunities for novel smart-city applications [SSX+15, RD19]. The emerging edge computing paradigm empowers these applications even more, where high-resolution cameras deployed in public spaces are complemented by specialized edge computing devices that can detect, process, and interpret various features of video streams. Such features include, e.g., motion detection, object detection, or face recognition. Besides potentially improving security in specific domains, this information extraction process can also act as an enabler for application developers to provide valuable services to potential customers. Applications leveraging such information include e.g., biometric authentication (smartphone unlocking), locomotive systems (autonomous driving), fitness related applications (detecting and correcting movements during an exercise), traffic monitoring, law enforcement (tracking of fugitives e.g., drivers escape), and many more.

The process of AI-assisted video data analysis is typically composed of three major tasks. First, the video data has to be prepared for an AI-application to be able to work with such data. Second, one or more features have to be detected in a video frame (basically an image). Third, post-processing actions take place. A real-world application for a data protection-centered VAP is face-anonymization, i.e,. detected faces on each frame of the input video have to be made unrecognizable in each frame of the output video. Fig. 5.1 illustrates these three steps (P1-P3) based on the face-anonymization example.

The increasing number of cameras in public spaces cause growing concerns about the abuse of mass surveillance systems and the implications on personal privacy and freedom [SMM+09]. Therefore, adequate protection of private data is an increasing concern in all kind of domains making use of public video streams, such as health, financial, and social security. The most common straight forward generalized approach to protect sensitive data is the installation of access control mechanisms alongside various encryption techniques, in order to protect data at rest and in transit. An exemplary video analytics implementation at the edge might incorporate a computing unit, connected to a camera, encrypting and transmitting a video feed via Transport Layer Security (AI) to a cloud server, where some form of e.g., Role Based Access Control ensures that the decrypted video feed may only be processed by an entity with adequate permission or role.

In this work, we follow an orthogonal approach to the provided example. Instead of applying encryption or privacy preserving image transformation techniques to a recorded video feed, we focus on the extraction of relevant information from the feed with the help of sophisticated machine learning techniques. This information only is then transmitted and made available, and, of course, also protected by similar mechanisms as described

in the previous example. The (raw) video is never transmitted or persisted/distributed permanently.

We propose a secure system design, that leverages state-of-the-art access control mechanisms featuring a Key-Policy Attribute Based Encryption (KP-ABE) scheme. Furthermore, we present in more detail a use case for analyzing the use of protective face masks in public areas, which, given the global pandemic situation due to the Coronavirus Disease 2019 (COVID-19), is highly plausible and relevant. The performance of a sample use case implementation is evaluated, aiming to demonstrate the feasibility of such a system running on typical edge computing hardware.

5.2 Related Work

Privacy and security are critical non-functional aspects of video analytics systems, and remain active areas of research. Recent research in particular has identified edge computing as a key enabler for privacy-sensitive systems that deal with real-time video processing [SSX+15, GHB18]. We discuss both frameworks for privacy for video analytics and surveillance in general, as well as specific methods of edge computing that enable our approach.

In the broad context of privacy in video-based media spaces, Boyle et al. [BNG09] proposed a framework – a descriptive theory – that defines how one can think of privacy while analyzing media spaces and their expected or actual use. The framework explains three normative controls: solitude, confidentiality and autonomy, yielding a vocabulary related to the subtle meaning of *privacy*. A more technical introduction to video surveillance in general is given by Senior in [Sen09]. The paper briefly summarizes the elements in an automatic video surveillance system, including architectures, followed by the steps in video analysis, from pre-processing to object detection, tracking, classification and behavior analysis. Our proposed system builds on the high-level architecture described in this paper. We improve this architecture by considering AI-based video processing capabilities, and incorporate advanced security mechanisms. Furthermore, we suggest concrete hardware and software, proven to run with adequate performance in edge computing scenarios. Previous research on privacy mechanisms of video analytics systems often focuses on protecting the source video streams. For example, Upmanyu et al. proposed a privacy preserving video surveillance framework [UNSJ09]. Their approach follows a more 'traditional' strategy, i.e., image (frame) transformation. They split each frame into a set of random images, where each image by itself does not convey any meaningful information about the original frame. They implemented a secret sharing scheme based on the Chinese Remainder Theorem, thereby enabling distributed secure processing and storage of image data. A blockchain-based approach was introduced by [LP20]. They exploit the blockchain technology to ensure the integrity and security of a cloud-based intelligent surveillance system. Reducing bandwidth requirements, a novel Merkle-Tree method is used for efficient transmission of video data. Chattopadhyay et al. demonstrate how the practical problem of privacy invasion can be successfully

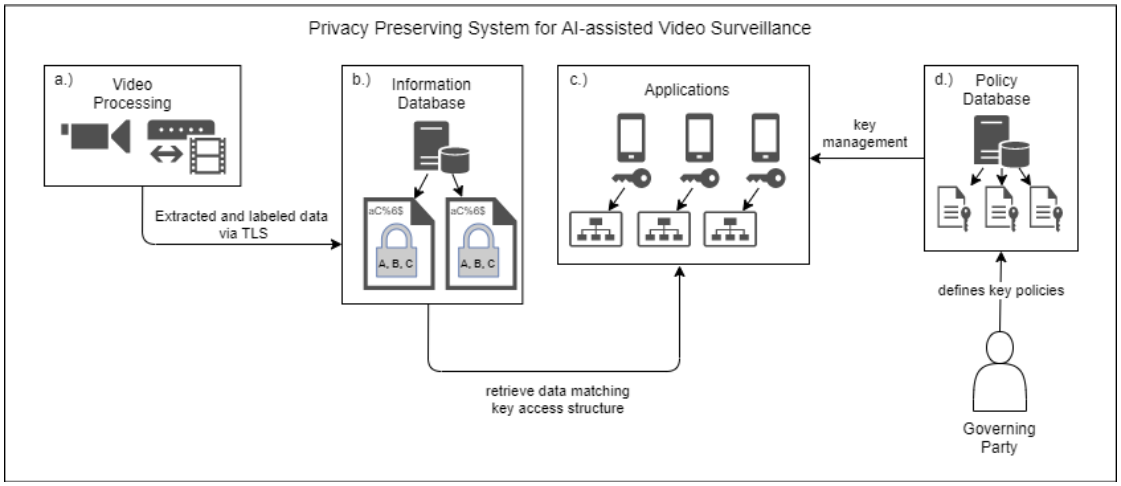

Figure 5.2: Schematic overview of our proposed privacy preserving system for AI-assisted video analytics

addressed through Digital Signal Processor (DSP) hardware in terms of smallness in size and cost optimization [CB07]. This is particularly useful for edge computing scenarios, where computational resources may be scarce. Their access control scheme is based on a asymmetric key exchange mechanism, while regions of interest in the image are encrypted via AES. The work of [EFG+09] also focuses on encryption of an individual images. They use two semantically secure additively homomorphic public-key encryption schemes, namely the Paillier and the DGK cryptosystem.

There is a lot more research focusing on encryption and anonymization strategies of image and video data. Of course, we have to notice that there are applications, for which our approach may not be feasible, and the transmission of (raw) video data is necessary. Other, more application-specific approaches, often involve the pre-processing of video streams to anonymize or obscure specific parts of a frame, i.e. *denaturing*. An example is the work of Schiff et al. [SMM+09] that proposes *Respectful Cameras*, i.e., cameras that respect the privacy preferences of individuals. Their practical real-time approach preserves the ability to monitor activity while obscuring individual identities. This is achieved by identifying colored markers such as hats or vests, which are automatically tracked by their system. The identities of people wearing, e.g., a colored vest, are obscured by adding a solid overlay over the face on every image frame. Satyanarayanan et al. [SSX+15] proposed GigaSight, an Internet-scale repository of crowd-sourced video that also enforces privacy preferences and access control, and leverages edge computing technology. GigaSight runs on a federated system of VM-based cloudlets that run the video analytics, which apply individual denaturing rules on all collected video feeds. Closer related to our approach is the work done by [GHB18]. They focus on camera-based digital manhunts of law enforcement agencies. Their approach leverages the inherent geo-distribution of fog computing systems to preserve privacy of citizens. Their architecture is divided into a cloud backend and edge devices. In the cloud, a specific facial recognition model

is trained on images of an individual (e.g., a fugitive) and then distributed across the edge devices. If a camera system, mounted on the edge device, detects a face it sends a notification to the cloud. Though the authors state that an authorization mechanism is implemented, in order to access manhunt related data, they do not provide any specific details.

To summarize, these closer related approaches all focus on protecting or denaturing the source video stream. Our system is different in that it ensures that no frames are ever transmitted, therefore requiring new system design considerations. This novel design is motivated by the fact that, many applications do not require analyzing or recording the raw video feed, but instead only require filtered frames or extracted metadata processed by other video analytics components.

5.3 Motivating Scenario

In different countries across the globe, cities have partnered with academic and industrial parties, deploying and testing edge infrastructure. Prominent examples are the Array of Things in Chicago (USA), the Smart City project in Aveiro, or the Padova Smart City project in Padova (ITA), to name just a few [CBSG17, Nun05, CZVZ14]. The most common infrastructure deployment is based on multiple sensor and compute units, typically mounted on e.g., lamp posts that are distributed across (a part of) the city. Adding video analytics capabilities to such an infrastructure can aid in various applications. The smart city project in Aveiro leverages video information to implement a modern traffic monitoring system, aiding governmental entities like law enforcement, fire department, ambulance service, or traffic control management. In combination with artificial intelligence capabilities, the extraction of relevant information from video regarding various domains can be processed in a more accurate and accelerated fashion, than being done by a human only [DA04, FBVC01, BZR+05, HBB+07]. The global COVID-19 pandemic brings new opportunities for AI-assisted video analytics in urban areas, potentially facilitating medical or social science research, as well as law enforcement.

5.3.1 Mask Detection Use Case

Due to the recent pandemic situation caused by COVID-19, many countries imposed an obligation for people to wear facial masks in certain (mostly public) areas. Detecting if people adhere to such obligations may not only be of interest to law enforcement but also for virus transmission research and medical analysis. Public surveillance distributed at the edge, supported by adequate machine learning techniques (models and prediction accuracy), is capable of aiding in the detection and provision of relevant information, i.e., identifying clusters or numbers of people not wearing facial masks at a given location. However, despite its potential beneficial use, privacy aspects still have to be considered and the protection of sensitive data ensured. Applications, whether law enforcement related or for academic or societal purposes, do not necessarily need to store (raw or compressed) video feed, neither must they have access at any time to (live) video data, in such given

use case described above. By implementing our system, relevant extracted information could in the simplest case be the number of people without masks per area, e.g., three people per 10 square meters. Combined with geodata of the surveyed area, interested parties would be able to get valuable insights and knowledge of peoples' behavior and adherence to possible obligations, and may take appropriate countermeasures. However, this is just a simplified example to demonstrate that there are real world use cases, where (governing) parties do not need to access a video feed directly in order to extract valuable information.

5.4 System Design

In this section we explain our proposed system design in detail, focusing on hardware and architecture. Section 5.4.1 will provide a view on the software side of the system. The foundation of our system is the architecture described in Section 5.1. Fig. 5.2 gives an overview of the involved components and mechanism incorporated. The proposed system assumes a processing and sensing unit (a), mounted at the edge of the network, e.g., on a smart lamppost. This unit comprises a high-resolution digital video camera, a computing device optimized for AI operations (e.g., NVIDIA Jetson TX2 [NVI20]), and a computing device (from now on referred as caching device) that caches the extracted video information. Additionally, the latter can also persist video data if needed. This is a typical hardware setup and environment for urban sensing applications featuring a VAP. Concretely, the smart city project in Aveiro mentioned previously, for example, uses a nearly identical hardware setup and environment for their traffic monitoring system, as used in the evaluation of this work. If prolonged caching or persistent storing of video data is needed, we strongly suggest writing the data to a secure drive (e.g., on a security module like [Zym20]), where it is additionally protected from physical access. As soon as the camera starts recording, video data is directly passed to the AI accelerated device. In order to enable extraction of information from the video feed in real time, we recommend that the video processing is done by a highly specialized AI-accelerated device. For simplicity reasons, we assume that one or more adequate pre-trained machine learning models are already deployed on this device. Complex machine learning training scenarios like federated learning as well as specifications for model architectures are out of scope of this work.

After the information is extracted, subsets of this data are labeled according to a specification provided by, e.g., a service provider or governing entity. A simple example may be the number of people in a certain area not wearing a mask, according to the motivating use case described in Section 5.3. The specific extracted information is an integer value, and the corresponding label could be `peopleWithoutMasks`. Notice that the extracted information could be the number of people per $1m^2$ or $10m^2$. Regardless of the area, the label for both types of information could be `peopleWithoutMasks`. An adequate and sophisticated labeling procedure may play a more important role when dealing with more complex scenarios.

The extracted and labeled information is then passed to the caching device, where it gets transmitted further to the information database (b). The information database can run anywhere in the compute continuum, and facilitates both protected real-time access as well as access to historic data for batch analytics. The transmission of all extracted information is protected by the well established AI standard, ensuring the integrity, authenticity, and confidentiality of data. Additionally, we suggest that a pinning mechanism [WSM+20] via modern HTTP extensions, e.g., HPKP [BHW17], is implemented. These mechanisms, though in case of HPKP controversial, provide additional authentication features for secure and trusted client-server communications.

The information database, once extracted information is received, is then being encrypted using the KP-ABE technique [GPSW06]. In this cryptosystem, ciphertexts are labeled with sets of attributes, i.e., our previously assigned labels. Furthermore, private keys are associated with access structures that control which ciphertexts a user is able to decrypt. Specific, fine grained access policies, define which user is allowed to access a certain labeled ciphertext for decryption. A user is able to decrypt a ciphertext if the attributes associated with a ciphertext satisfy the key's access structure. For instance, if Alice has the key associated with the access structure "X AND Y", and Bob has the key associated with the access structure "Y AND Z", they are not able to decrypt a ciphertext whose only attribute is Y by colluding. The KP-ABE system further allows deriving keys from other keys, based on their restriction hierarchy and access structure, i.e., each user's key is associated with a tree-access structure where the leaves are associated with attributes, allowing any user that has a key for access structure X to derive a key for access structure Y, if and only if Y is more restrictive than X.

In a KP-ABE, the encryptor exerts no control over who has access to the data they encrypt, except by their choice of descriptive attributes for the data. Rather, they must trust that the key-issuer issues the appropriate keys to grant or deny access to the appropriate users. In our case, users would be applications that may only need a specific subset of the extracted video information. A simplified example of access to encrypted data via policy defined access structure in a KP-ABE system is shown in Fig.5.3. Accessing only this subset of data is reflected in the application's private key (c), determined by a policy. An application, firstly when deployed, is issued with such a key and is notified by the policy database if its key is modified. This allows for a seamless fine-grained management of access control for any application without the need of a re-deployment. The policies are stored and managed at a dedicated independent location at the edge or in the cloud (d). Managing those policies could be done by governing parties for example; but in this work we do not further address this issue. Furthermore, we notice that if features not specified at design time are needed by an application, those unsupported feature extraction has to be re-implemented and deployed by e.g., a service provider or governing party. Once the extracted information is correctly encrypted, it is possible for applications to access this information via a well defined separate Application Programming Interface (API), via a corresponding Attribute-Based Access Control (ABAC) mechanism. The API never allows by design for an application

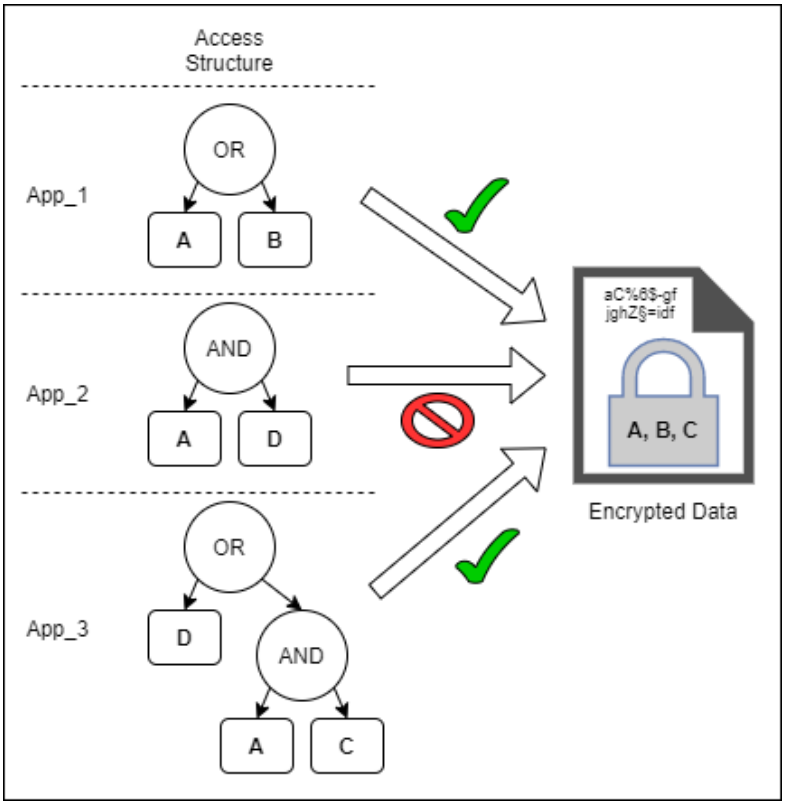

Figure 5.3: Access to encrypted data via policy defined access structure in a KP-ABE system

to directly communicate with a video processing unit at the edge, thereby prohibiting theoretical access to the video feed. An application is now able to further process the extracted information depending on their specific needs.

5.4.1 Sample Implementation

In this section we showcase a concrete exemplary implementation of our proposed system based on the scenario described in Section 5.3. Systems for face or object detection are well understood, as they have been broadly studied, implemented and evaluated over the recent years. Since the outbreak of COVID-19, machine learning models and their applications for mask detection have gained increased attention of researchers and developers world-wide.

The detection accuracy for real-time systems mostly depends on external factors like, lightning conditions, view angle and even skin color of observed persons. Video resolution only plays a minor role in detection accuracy but more in image processing performance [MCW+15]. In our example, we adapted the code from [Deb20] to run on the different hardware nodes of our evaluation testbed. We found that real-time performance

is achieved by averaging an accuracy of about 98%. The extracted information, i.e., probability of a mask being detected alongside with the count of people in a given frame, is labeled and transferred over the internet, encrypted via standard TSL, to the information database. On the information database runs a KP-ABE scheme [Agr20], which is responsible of the re-encryption of initial ciphertexts, i.e., incorporating the attribute groups into the ciphertext. In previous research, we have shown the feasibility of running data protection mechanisms on resource constrained IoT devices [LD19a]. The information database hosts a simple REST-API providing access to the extracted information, given proper authentication. A sample Android application is then able to query only information for labels which are reflected in the private key deployed on the smartphone. A potential limiting factor in this application chain is the network latency, which depends on multiple environmental factors. Therefore, we did not include measurements regarding network related performance into our evaluation. This sample application aims to showcase the scenario executed on dedicated edge computing hardware and evaluate the core systems tasks, i.e., AI-assisted object detection and encryption.

5.4.2 Evaluation

In our experiments, we focused on the AI-assisted information extraction process and the corresponding encryption tasks. Therefore, we deliberately executed the video processing tasks and the information database on the same device.
Our testbed comprises a heterogeneous set of typical edge computing hardware. First, a laptop with an Intel i7-7700 CPU@4.2GHz and 16GB RAM. Second, a Nvidia Jetson TX2 Developer Board with a ARM Cortex-A57@2GHz CPU and Pascal GPU and 8GB RAM. Third, a Raspberry Pi4 with a ARM Cortex-A72@1.5GHz CPU and 4GB RAM. The AI-assisted information extraction process, i.e., detecting the number of people wearing a mask, is computational expensive, but also (depending on the AI-model used; in our case Tensorflow) highly parallelizable. Therefore, we deliberately configured our system to use the GPU of the Nvidia Jetson board in the respective test run for the extraction part. Simply speaking, the many computing cores of a GPU, or even further specialized hardware like a Tensor Processing Unit (TPU), highly facilitate the inference tasks of AI-based processes, like e.g., object recognition. The experiments executed on the other devices are purely CPU bound, i.e., the GPU (even if available) is not used at all. For the evaluation, we chose three short publicly available video sequences, showing varying numbers of people wearing a mask. The first video shows a single person putting a mask on an off. The second video footage (labeled Multiple Persons in Fig 5.4) constantly shows five people taking on and off their masks, while the third video (labeled Crowd in Fig 5.4) shows a large amount of people (>10; some wearing a mask, some do not) constantly varying in number. These videos are the input for our system, where the number of people wearing a mask is extracted and encrypted on a Frames per Second (FPS) basis. If the FPS processed by our system matches the FPS the input video is recorded with (e.g., 25 FPS), real-time performance is achieved, i.e., a user could potentially read the extracted information in real-time, but obviously this still also depends on the network conditions. We have to notice that our mask detection implementation is a chaining process of a

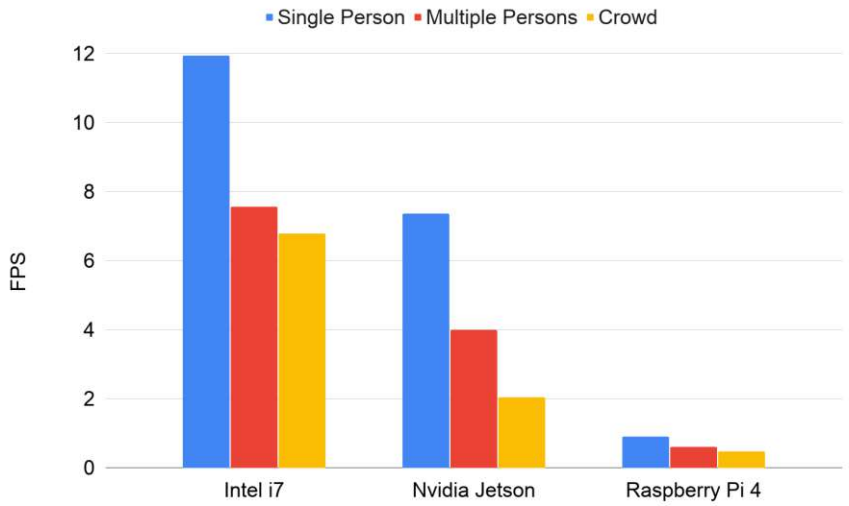

Figure 5.4: Performance of information extraction and encryption of different video input

face detection algorithm and a mask detection algorithm, each working with its own dedicated model, implemented in Python 2.8. The AI inference part is implemented using TensorflowLite for devices with ARM based architecture (Raspberry Pi and Jetson) and Tensorflow for x64 based devices (Intel i7). While the point of this work is not to implement a performance optimized mask-detection framework, this chaining procedure obviously greatly affects the overall performance of the system. Fig. 5.4 shows the results of our experiments. The overall performance is mainly affected by the AI-specific tasks and furthermore on the conditions and specifics of the information that needs to be extracted, e.g., an increase in number of people leads to a massive decrease in performance. The encryption tasks are commonly CPU-bound, scaling linearly with CPU-speed and/or are also dependent on the available specific encryption algorithm based hardware instruction of a given CPU, like e.g., the AES instruction set which is integrated into many modern processors [LD19b]. The extracted plain text information, concerning our scenario, is rather small (compared to e.g., encrypting the whole raw video data), hence the computational effort to produce ciphertext minimal. Therefore, the overall contribution to the performance capabilities of our system is neglectable compared to the information extraction process. Research has shown that modern edge computing devices are capable of executing AI-assisted video processing, without significant loss in performance [RF17, HRS+17, RCZ+18, LRCLP17]. Hence, in order to achieve real-time performance, the suggested KP-ABE-based encryption scheme is not a bottleneck in the system performance-wise, but rather the combination of used hardware and the nature of the AI-assisted feature extraction task. To address this problem, strategies like lowering the sampling rate of the video for feature extraction, reducing the input size, etc. could be incorporated. These kind of adaptation strategies and its impact on performance are presented in the following chapter 6.

5.5 Summary

Video cameras deployed in urban areas provide enormous value for novel smart cities applications, but at the same time, cause legitimate privacy concerns. These concerns are mostly related to the unrestricted access and recording of the raw video feed, and potential abuse for mass surveillance. We have found, however, that many applications do not require this access in the first place. Instead, we argue that video analytics should be pushed to the extreme edge, and direct access to the video feed should be avoided. To that end, we have presented a privacy preserving system for AI-assisted video analytics. It features a decoupling architecture that effectively hinders applications from directly accessing the underlying video feed, and instead allows them to advertise what type of information they require. Our system then extracts the information using existing AI-based video processing techniques, ensures that privacy preferences are met, and facilitates the secure access to the extracted information for both real-time and batch applications. A ciphertext (i.e., the encrypted information extracted from video data) is labeled with certain attributes, which only allows applications with a matching private key (i.e., the attributes corresponding to the labels of the ciphertext are encoded in the key) to decrypt and access the data. A KP-ABE security scheme ensures that only authorized parties have access to this extracted information. To allow for a more fine grained access control, security policies determine which application is able to decrypt specific subsets of the encrypted extracted data. The policies are stored and managed at a dedicated policy database, located at the edge or in the cloud. Furthermore, it is responsible for issuing keys to an application, as well as notifying applications if a key's attributes change. Hence, a seamless fine-grained management of access control for any application without the need of a re-deployment is achieved. In contrast to typical RBAC-based systems, the KP-ABE security scheme allows for a much more finer definition of who has access to what specific sets of data. However, such RBAC-based system are well established and dedicated software readily available, while KP-ABE is not as prominent which could potentially increase development effort when implementing the proposed system.
We showed that our system is able to run on typical edge computing hardware, by implementing and evaluating a simple, yet due to the recent pandemic situation highly relevant scenario. By pushing the video analysis to the edge and encrypting the data locally, a significant increase in data protection quality can be achieved. However, depending on the needs of the application and many other factors (which will be detailed in the upcoming chapter 6), local execution of such tasks (mainly AI-based tasks) potentially negatively affects the overall performance of a system, if compared to a traditional cloud-offloading scenario. However, we have to state that the KP-ABE security scheme does not significantly affect the overall performance of a VAP on its own. This is mainly due to the fact that the majority of computation time is taken by AI-based tasks, such as object detection. We deliberately chose to not measure overall round trip time or individual data transfer timings, because those values are typically heavily prone to network latency affected by various unstable environmental factors like signal strength

or interferences. Furthermore, such values would also heavily differ depending on the connection type used, for example wired versus bluetooth versus wireless.

CHAPTER 6

Towards Understanding the Adaptation Space of AI-Assisted Data Protection for Video Analytics at the Edge

Edge computing facilitates the deployment of distributed AI applications, capable of processing video data in real time. AI-assisted video analytics can provide valuable information and benefits in various domains. Face recognition, object detection, or movement tracing are prominent examples enabled by this technology. However, such mechanisms also entail threats regarding privacy and security, for example if the video contains identifiable persons. Therefore, adequate data protection is an increasing concern in video analytics. AI-assisted data protection mechanisms, such as face blurring, can help, but are often computationally expensive. Additionally, the heterogeneous hardware of end devices and the time-varying load on edge services need to be considered. Therefore, such systems need to adapt to react to changes during their operation, ensuring that conflicting requirements on data protection, performance, and accuracy are addressed in the best possible way. Sound adaptation decisions require an understanding of the adaptation options and their impact on different quality attributes. In this chapter, we identify factors that can be adapted in AI-assisted data protection for video analytics using the example of a face blurring pipeline. We measure the impact of these factors using a heterogeneous edge computing hardware testbed. The results show a large and complex adaptation space, with varied impacts on data protection, performance, and accuracy.

After a brief introduction, the rest of the chapter is structured as follows: Section 6.2 describes the exemplary scenario motivating our approach. The adaptation space, divided

into different phases of an application, as well as for infrastructural considerations, is presented in Section 6.3 An Evaluation based on a face-anonymization pipeline is presented in Section 6.4 In Section 6.5 we investigate and compare our approach to related work in the field. Finally, Section 6.6 concludes the chapter.

6.1 Introduction

Video feeds generated by distributed devices enable a variety of applications. For example, in smart cities, videos from cameras can be used for traffic monitoring, accident reporting, and law enforcement applications, among others [CBSG17, Nun05, CZVZ14]. In some cases, also user-generated content may be available. E.g., in the case of an accident, videos taken by passers-by with their smartphones may also aid the work of first responders.

Videos from public spaces may contain sensitive data associated with special security or privacy requirements [MAM15, SWZL12]. For example, people's faces or cars' license plates are personally identifiable information, and processing such information is subject to data protection regulations, such as the GDPR in the European Union [AJL+21].

Recent advances in artificial intelligence, particularly in machine learning, enable effective automatic video processing [DA04, FBVC01, BZR+05, HBB+07]. AI can also help in satisfying data protection requirements; e.g., faces and license plates can be automatically detected by ML-based object detection algorithms, and then anonymized by further video manipulation methods.

Usually, processing videos is very resource-intensive, but the end devices producing the videos are resource-constrained. Video processing may be offloaded to the cloud to benefit from the virtually unlimited computational capacity of the cloud. However, offloading to the cloud is associated with high latency and high network load. A better solution is to deploy devices – called edge nodes – with sufficient computational capability near the network edge. End devices can offload some video processing functionality to nearby edge nodes, thereby benefiting from low-latency access to computational capacity without overloading the core network. With the appropriate distribution of functionalities between end devices, edge nodes and the cloud, optimal performance can be achieved [Man19].

A challenge for such systems is the dynamic variability of their run-time environment [BFI19, BFGL20, VF13a]. For example, the number of smartphones offloading video processing to an edge node may change at run time. Also the capabilities of the connected smartphones and the properties of the videos to process can change over time. Thus, video processing systems at the edge must be self-adaptive to be able to react to changes at run time. Self-adaptation involves monitoring the system and its environment, analyzing whether changes threaten the satisfaction of the requirements, and the planning and execution of adaptations if needed to ensure the continued satisfaction of requirements [KC03]. For example, a smartphone may be able to perform face anonymization locally as long as the video contains only one face, but when the number of faces in the video

increases, the processing may have to be offloaded to ensure proper operation. This requires an automatic run-time adaptation of the face-anonymization pipeline.

To enable adaptations at run time, appropriate adaptation rules have to be defined at design time, specifying what adaptation to perform in which situation. Existing approaches to creating self-adaptive systems assume that the designer is able to define the appropriate adaptation rules [ST09]. For defining adaptation rules, it is crucial to understand i) what changes in the environment may happen, ii) what self-adaptations the system may perform, and iii) how those changes and self-adaptations impact the relevant system properties [SMM18]. For video analytics at the edge, these questions are complicated, since there are many different types of possible environment changes and self-adaptations, with intricate implications on a variety of system properties. Environment changes may happen both at the infrastructure level and the application level, and also self-adaptations are possible on both levels.

This work presents the first detailed study on the adaptation space of AI-assisted data protection for video analytics at the edge. We use the example of a ML-based face-anonymization pipeline which can be distributed between end devices and edge nodes. Our contributions are as follows:

- We identify relevant parameters of the environment that can change, possible self-adaptations that the system can perform, and key system metrics.
- We perform extensive measurements in a heterogeneous testbed to determine the effect of different environment changes and self-adaptations on the identified metrics.

Such results are needed to have a solid basis for designing the adaptation logic for AI-assisted data protection for video analytics at the edge. Our work is a first step in this direction and the results are potentially applicable to a wide field of other possible scenarios beyond face anonymization.

6.2 Motivating Scenario

We consider a use case of traffic monitoring in a smart city. Cameras are installed across the city to monitor the traffic on the streets. The video feed of each camera is streamed to a nearby edge node, as shown in Fig. 6.1. Edge nodes are computational resources deployed throughout the city, for example in a Road-Side Unit (RSU) or a smart lamppost. The video feeds from the cameras are anonymized in the edge nodes before they are forwarded to the cloud. Traffic control experts use a cloud-based application to identify and analyze potential traffic incidents, such as traffic jams.

In addition to the statically deployed cameras, also citizens' smartphones may connect to a nearby edge node and stream their video feeds to the edge node. In this case, the anonymization of the video feed may happen either in the smartphone or in the edge node. The anonymization of video feeds is performed by a Video Analysis Pipeline, as

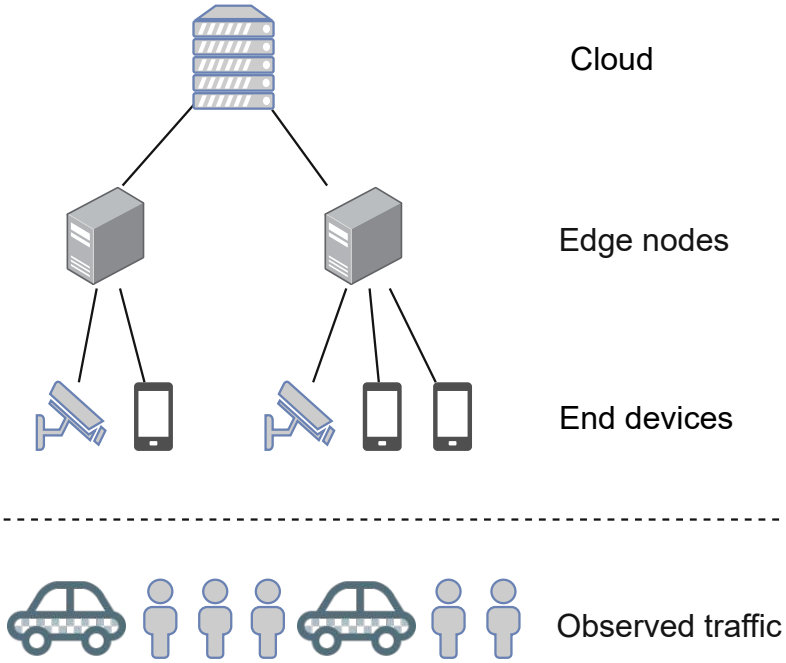

Figure 6.1: Edge infrastructure in the considered use case

schematically depicted in chapter 5, Fig. 5.1. This face-anonymization pipeline consists of three phases, where i) the video is pre-processed, ii) faces are detected in the video frames, and iii) faces are made unrecognizable.

During the operation of the system, many changes are possible. For example, assume that a smartphone is taking a video feed and anonymizing it using a face-blurring technique in real time. As more people join the scene, more faces need to be detected and blurred in the video, which increases the computational needs of the face-anonymization pipeline. If the computational capacity of the smartphone is not sufficient anymore, an adaptation is needed. One possibility is to offload one or more phases of the face-anonymization pipeline to a nearby edge node. Another possibility is to resort to a less resource-consuming anonymization technique (e.g., drawing a black rectangle over the faces instead of Gaussian blurring). Which of these potential adaptations is most appropriate has to be decided on the fly, depending on the given situation and on the implications of potential adaptations.

6.3 Adaptation Space for Data Protection

Feature Extraction, Object Detection or Motion Tracking are prominent examples of AI enhanced video analytics, usually designed, implemented and integrated into a VAP. Optimizing the performance of a VAP operating at the edge is a major research topic. The concept of edge computing in combination with the increasing potential of AI, becomes

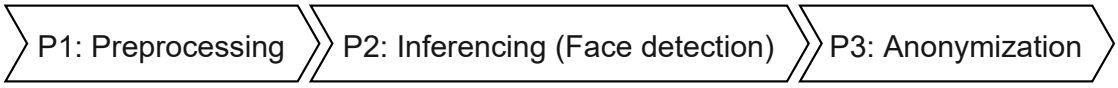

Figure 6.2: Parts of an exemplary face-anonymization application

Adaptation	Phase	Data protection	Performance
Compiled Programming Language	P0	/	+
Greyscaling	P1	-	+++
Adequate Video Resolution	P1	+	-
Frame Skipping	P1	-	+
Adequate AI-Model	P2	+++	+
Batching	P2	/	++
AI-Inference Chaining	P2	++	- - -
Sophisticated Anonymization	P3	+	-
Enable Dedicated Hardware	*	/	+++
Overclocking	*	/	++
Migration	*	+/-	++
Offloading	*	-	++

Table 6.1: Impact of adaptations on data protection and performance

increasingly relevant for this area of research. Various optimization objectives can be considered, including execution time, latency, and inference accuracy. Optimization is carried out during either design time or run time of an application, resulting in adaptations on the infrastructure or application level. Accuracy plays a crucial role for applications incorporating data protection mechanisms, e.g., anonymization, that have to adhere to specific privacy regulations like the GDPR. An application may have to sacrifice potential performance benefits from optimization strategies to achieve the necessary level of accuracy. The heterogeneous and dynamic environment in edge computing greatly increases the complexity of such optimization strategies. Therefore, we need a better understanding of the main levers for performance and data protection in each of the three phases of the VAP. To ease reading and correlating each phase with the respective adaptation levers, the Fig. 5.1 in chapter 5 is shown here again as Fig. 6.2.

Table 6.1 displays the most relevant adaptations we identified and their impact on data protection and performance. For application adaptations, the relevant phase of the VAP (P1-P3) is indicated in the table. P0 is a special case of a pre-runtime phase. Infrastructure adaptations can relate to any phase, indicated by a *. The impact of an adaptation on data protection and performance is shown by + (positive impact), - (negative impact), or / (no significant impact).

The trigger for such adaptations can be a change on the application level or on the infrastructure level. Application-level triggers include changes in video content (e.g., the number of people in the video, the size of their faces in the video, the angle of faces

to the camera, lightning conditions), changes in the number of video feeds to process, or changes in application requirements (e.g., different frame rate required depending on the purpose of the video feed). Infrastructure-level triggers include changes in the available hardware (e.g., mobile end devices connecting to or disconnecting from an edge node) and in the trustworthiness of hardware (e.g., a camera becoming compromised by a physical attack).

In the following, we describe the identified possible adaptations of each phase in more detail and reason about their impact based on the example of Section 6.2.

6.3.1 Adaptations of the pre-processing phase (P1)

Video data needs to be prepared for processing by a face detection framework. This may involve encoding, decoding, or transcoding the video, greyscaling the frames, or resizing each frame. Greyscaling is needed since most object recognition frameworks (e.g., TensorFlow) take greyscaled images as input. Transforming the video can be a heavy computational task, depending on how the video is recorded (frame rate, resolution, codec, etc.). Manufacturers like Nvidia (NVEnc) integrate dedicated chips into their hardware to facilitate this task. Encoding, decoding or transcoding may have a significant impact on overall performance of a VAP, but do not affect data protection quality directly. While resizing and greyscaling are computationally not very expensive (in the sub-millisecond range per frame even on a Raspberry Pi4), adaptations concerning resizing an image could negatively affect data protection. This is because face detection may produce less accurate results on the resized video, leading to undetected and thus non-anonymized faces in the output video.

Adaptations changing the frame rate of an input video should take the context and nature of the input video into account. If (near-)real-time performance (and experience of a user) is the goal, the frame rate should not go below 24 FPS. However, if this is not possible due to computational limitations, skipping a given number of frames, i.e., providing only each k-th frame to the face detection step, provides a viable option for videos without abrupt changes. Frame skipping assumes that a face detected in frame n is in the same location in frames $n+1, \ldots, n+k-1$ as well. Thus, anonymization operates for each of these frames on the location where a face was detected in frame n. For videos with abrupt changes, this may degrade anonymization quality.

6.3.2 Adaptations of the inference phase (P2)

The performance of the second phase of the pipeline is heavily dependent on the face-recognition framework itself as well as on the inference model used by the framework. In the context of AI-based inference tasks, performance is not only related to processing speed but also to the accuracy with which an object like a face is detected in an image. To detect faces in a video, a framework looks at every frame of the video, trying to infer if a face is present in the given frame. The accuracy is mostly dependent on factors like face-angle or lightning conditions in the picture [MCW+15]. A well-trained model

embedded in modern frameworks like e.g., TensorFlow will generally allow for high inference speed with high accuracy. In some cases, inference may take significantly longer if more than one face is present in the frame. Such frameworks are not necessarily available and/or optimized for each system architecture like e.g., ARM, x86, x64. Differences in performance and accuracy can also occur due to the usage of legacy versions of such AI frameworks.

There are several possibilities for adaptations concerning inference that aim to improve accuracy and/or performance. The most common practice in AI-based VAPs is batching. Modern AI frameworks provide an API, allowing an application to send multiple frames in one request and process them in parallel. Adapting the batch size and frequency can be considered viable adaptations because they do not negatively affect data-protection quality, but may improve performance. Changing the pre-trained model for inference is another possibility. This adaptation becomes relevant if the analyzed video is prone to dynamic changes as it may increase accuracy and/or performance, but may also decrease data protection quality if accuracy drops due to the model change.

In a scenario as described in Section 6.2, accuracy may also be improved if multiple inference steps are chained, e.g., first, persons are detected in a video frame, then on the resulting regions of the frame, face detection is performed. Such strategies heavily affect overall performance, but may be necessary to adhere to data protection regulations and policies. Another adaptation is switching between implementations in different programming languages. For example, Python is heavily used for AI applications in the research community, but being an interpreted language, applications written in Python generally perform worse than using a compiled language like C or C++.

6.3.3 Adaptations of the anonymization phase (P3)

The third phase involves the actual anonymization of detected faces in a video frame, i.e., a graphic overlay is drawn over the face in the image. A face-detection framework returns a bounding box (an area inside an image) that corresponds to, or covers the face detected inside a frame. This bounding box is then used to draw an overlay onto the image. Although it is well researched that making a face unrecognizable in visual data does not fully guarantee the anonymity of a person, it still facilitates such anonymization task to a high degree, mostly by making the de-anonymization process significantly harder for an attacker.
Both performance and data protection quality depend on the desired graphic nature of the overlay. One possibility is blurring the face in an image by applying a Gaussian Blur Transformation function to the image part cropped using the previously described bounding box. Pixelating the face area can be considered an improvement in anonymization: the area is divided into an $n \times m$ image-tile matrix and each tile is transformed with e.g., Gaussian Blur. A simple alternative, with minimal performance impact, maybe a more blunt approach, where the cropped image part is just painted with a single color, resulting in e.g., a white circle drawn 'over' the face. This approach may from a GUI perspective of an application not be considered as beautiful as the Gaussian blur, but is

significantly less constraining on computing performance. However, similar to resizing and greyscaling described in Section 6.3.1, the differences in execution times between these anonymization strategies are small compared to the inference time. Adaptations to improve data protection quality, specifically anonymization, most likely have a greater impact on user experience than on performance.

6.3.4 Adaptations of the infrastructure

The execution time of a task often heavily depends on the capabilities, e.g., CPU speed, architecture (ARM, x64, etc.), or type of hardware (CPU, GPU, etc.) available, of the device the task is running on. Executing all three phases on the same device may limit the overall performance of the VAP. A common approach is to decouple the phases of a VAP and execute them separately on different devices. Offloading tasks to the cloud or powerful edge nodes bears great potential to minimize execution time, but comes with the downside of increasing latency [RCLC17, LQB18]. Data locality requirements may hinder offloading, potentially forcing a task that operates on sensitive data, to run on specific premises [MMPS19].

Infrastructure adaptations may be useful in any of the three phases. Phase 2 can benefit the most from infrastructure adaptations in terms of performance, but may also suffer from downsides like additional energy costs and/or laborious human intervention. Infrastructure adaptations typically do not directly influence accuracy, hence they can increase performance without compromising data protection quality. Activating (pre-installed and available on demand) dedicated hardware like a GPU or a Tensor Processing Unit (TPU) are viable infrastructural adaptations. In virtualized environments, migrating the inference component to a more powerful device can be a beneficial adaptation. However, data locality aspects and privacy policies need to be considered. Overclocking hardware may also be a possible adaptation that should be carried out carefully. Dedicated AI hardware like e.g., an Nvidia Jetson device, provides interfaces and scripts to change performance profiles (e.g., CPU voltage, fan speed, clock speed) on the fly. However, these adaptations may also increase energy consumption and potentially decrease hardware life expectancy. A further adaptation possibility is to execute the inference task on CPUs with different instruction sets. Modern CPUs provide a lot of low-level instructions, besides the usual arithmetic and logic, known as CPU extensions. Examples of such extensions are Streaming SIMD Extensions (SSE) or Advanced Vector Extensions (AVX), see [Int16, AMD12] for details. In particular, AVX introduces Fused Multiply-Accumulate (FMA) operations, which speed up linear algebra computation, namely dot-product, matrix multiply, convolution, and more. Almost every AI-based application involves many of those operations, hence will be faster on a CPU that supports AVX and FMA [AS20]. AI frameworks, like TensorFlow, support special instruction sets (Single Instruction Multiple Data) and other features that increase performance. However, in the case of Tensorflow for example, the default distribution is built without including such CPU extensions. Hence, if a system developer relies on a CPU (e.g., in the absence of a GPU or TPU), they have to manually compile it for their given architecture.

Table 6.2: Devices used for the experiments

Name	CPU	GPU	RAM
Desktop PC	AMD Ryzen 5600X@3.7GHz	not used	32GB
Laptop	Intel i7-7820HQ@2.9GHz	not used	16GB
Intel NUC	Intel i5-7260U@1.5GHz	not used	16GB
Raspberry Pi4	ARM Cortex-A72@1.5GHz	not used	4GB
Nvidia Jetson TX2	ARM Cortex-A57@2GHz	Nvidia Pascal, 256 cores	8GB

6.4 Evaluation

We implemented the face-anonymization pipeline described in Section 6.2 using Python 3.6, OpenCV 3.4 and TensorFlow 1.14. Parameters can be used to choose between different versions of each phase of the pipeline. We performed a series of experiments to evaluate the impact of different factors on the performance of the face-anonymization pipeline. We performed the experiments on a heterogeneous testbed consisting of several different types of devices, shown in Table 6.2. The source code and the results are available online[1].

Overall, the empirical findings reinforced the results of the theoretical analysis of Section 6.3. In the following, we present some of the quantitative findings from the experiments.

Fig. 6.3 shows the impact of the used hardware on the VAP's performance. The fastest device (desktop PC) offers roughly 30 times higher performance than the slowest one (Raspberry Pi). Thus, offloading computations to a more powerful device offers huge potential for improving performance. Additionally, dedicated hardware like a GPU has huge potential for performance gains. The Raspberry Pi used in our evaluation features a similar CPU like our used Nvidia Jetson board. However, due to the availability of outsourcing the inference task to the GPU of the Jetson board (instead of running it on the CPU like the Raspberry Pi did), overall it performed nearly 10 times better than the Raspberry Pi, and even slightly outperformed the Intel NUC, which has a far more powerful CPU that the Jetson board.

Fig. 6.4 shows the impact of the number of faces in the video on the VAP's performance. The results for the NUC are shown; the results for the other devices are similar. Processing a video with many faces leads to a performance loss of about 5% compared to a video containing a single face. This is an example of the impact of an environmental factor. The system has no influence on the number of faces in the video, but the system may have to adapt to react to changes in the number of faces in the video, to counteract the performance loss. This becomes especially important for mainly performance oriented systems that have to deal with a dynamic and mostly heterogeneous (video) environment. Obviously, an increased number of faces in a video stream also increases the chances

[1] https://github.com/clemenslachner/EdgeAIAdaptations

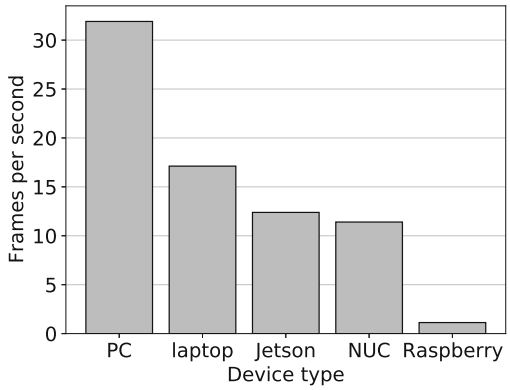

Figure 6.3: Throughput of the face-anonymization VAP run on different devices

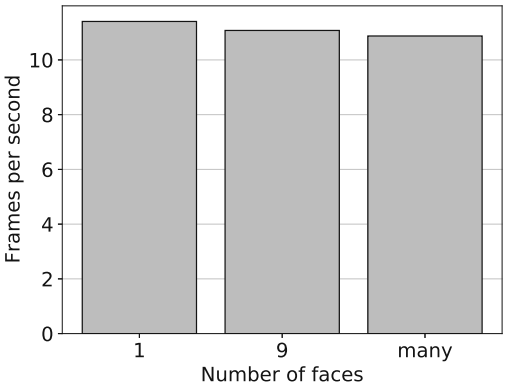

Figure 6.4: Throughput of the VAP on the NUC, depending on the number of faces in the video

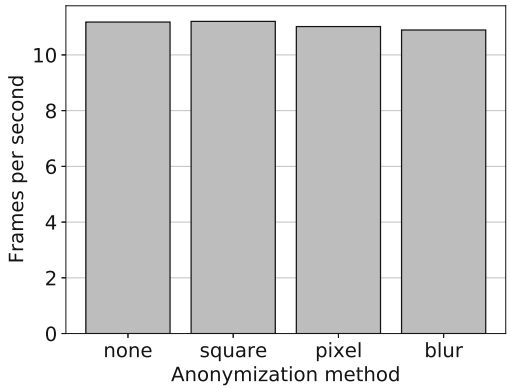

Figure 6.5: Throughput of the VAP on the NUC, with different anonymization methods

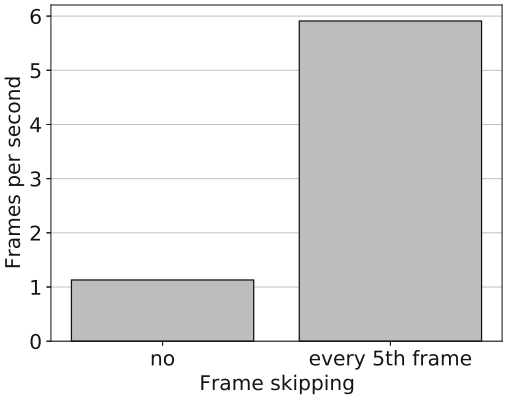

Figure 6.6: Effect of frame skipping on the throughput of the VAP on the Raspberry Pi

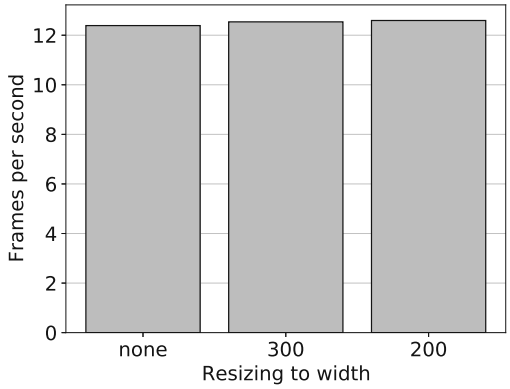

Figure 6.7: Effect of resizing on the throughput of the VAP on the Jetson board

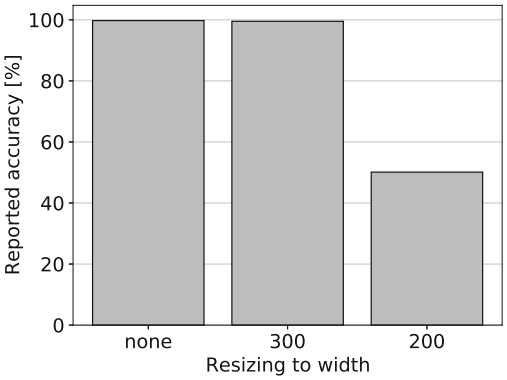

Figure 6.8: Effect of resizing on the accuracy of the VAP on the Jetson board

of faces to remain undetected, thus not anonymized. In such situations swapping the inference model may yield better results, however there is no empirical method available to determine that. Hence, such an adaptation would have to be verified, e.g., at design time where different models could be tested, by a (human) system operator.

Fig. 6.5 shows the impact of different anonymization methods on the performance of the face-anonymization pipeline. The performance difference between the fastest and slowest anonymization method is about 2.8%. The difference is small because the anonymization method only affects the performance of the last phase of the face-anonymization pipeline. The execution time of the face-anonymization pipeline is dominated by the second phase (face detection), hence accelerating the third phase has only limited effect. Across all devices, we could observe that the execution time of tasks located in P3, such as blurring is similar to P1 tasks (resizing and greyscaling) as described in Section 6.3.1, ranging in the sub 10ms area. This highlights the importance of P2 levers, as main driver for performance gains. However, if maximum performance is desired/required in a system, and e.g., UI-aspects are secondary, P3 levers also offer marginal performance gains.

Fig. 6.6 shows the impact of frame skipping on the performance of the face-anonymization pipeline. If face detection is performed only for every 5th frame, this also leads to roughly a 5 times performance increase for the whole face-anonymization pipeline. Hence, activating frame skipping or changing the number of skipped frames is a very effective adaptation.

Fig. 6.7-6.8 show the impact of resizing the frames on the measured performance and the accuracy reported by the AI framework. Scaling the video to smaller sizes leads to a slight increase in performance, but may lead to a dramatic loss in accuracy. The figure clearly shows that the effect of resizing on the throughput of the VAP is minimal. This also holds true for all other devices in our testbed, which underlines the importance of understanding the effect of different adaptations on key metrics. Hence, we argue that the observed minor increase in performance of resizing an image is neglectable, especially considering the possible negative effects on accuracy.

6.5 Related Work

Privacy and security are critical aspects of video analytics systems. Recent research identified edge computing as a key enabler for privacy-sensitive systems dealing with real-time video processing [SSX+15, GHB18]. Today's hardware capabilities potentially enable real-time video processing at the edge where, typically, data originates. In the context of privacy in video-based media spaces, Boyle et al. [BNG09] proposed a framework – a descriptive theory – that defines how one can think of privacy while analyzing media spaces and their expected or actual use. The framework explains three normative controls: solitude, confidentiality and autonomy, yielding a vocabulary related to the subtle meaning of *privacy*. A more technical introduction to video surveillance is given by Senior in [Sen09]. The paper briefly summarizes the elements in an automatic video surveillance system, including architectures, followed by the steps in video analysis,

from pre-processing to object detection, tracking, classification and behavior analysis. Our implementation builds on the high-level architecture described in that paper, and adds AI-based video processing capabilities. Chattopadhyay et al. demonstrate how the practical problem of privacy invasion can be successfully addressed through DSP hardware in terms of smallness in size and cost optimization [CB07]. This is particularly useful for edge computing, where computational resources may be scarce. Much research focuses on encryption and anonymization of image and video data. Other, more application-specific approaches, often involve pre-processing of video streams to anonymize or obscure specific parts of a frame. An example is the work of Schiff et al. [SMM+09] that proposes *Respectful Cameras*, i.e., cameras that respect the privacy preferences of individuals. Their real-time approach preserves the ability to monitor activity while obscuring individual identities. This is achieved by identifying colored markers such as hats or vests, which are automatically tracked by their system. The identities of people wearing, e.g., a colored vest, are obscured by adding a solid overlay over the face in every frame.

Several authors proposed using adaptations to cope with dynamic changes of edge computing systems. Breitbach et al. combine different data placement and task scheduling policies to adaptively react to changes in the system context [BSEB19]. Gand et al. introduce a fuzzy controller for self-adaptive container orchestration for edge devices [GFEI+20]. Samir and Pahl propose using adaptations for self-healing of edge cluster systems [SP19]. Wang and Xie develop an algorithm for the adaptive choice of parameters in mobile augmented reality systems [WX20]. All those approaches rely on only carefully selected parameters, with focus on a particular problem in the video analytics domain. However, the big picture was missed, and some of the potential solutions to a problem presented in their work may introduce another problem related to either data protection or performance. As we have seen, identifying the parameters and their impact for AI-assisted data protection for video analytics at the edge is a non-trivial task, which has not been solved yet. Our generalized work and more holistic approach thus paves the way towards effective adaptation algorithms for AI-assisted privacy-preserving video analytics at the edge.

6.6 Summary

This chapter presents preliminary results of the first systematic study of the adaptation space of AI-assisted data protection for video analytics at the edge. Using a face-anonymization pipeline as running example, we identified several possible adaptations and analyzed their impact on performance and data protection. We also implemented and empirically evaluated several of the considered adaptations. The results show a wealth of different adaptation options on both the infrastructure and application level. The results also show that the impact of these adaptations varies significantly. In general, adaptations that deal with the inference part of a VAP have the most significant impact on performance, most notably the incorporation of dedicated AI hardware such as a GPU or TPU. Choosing an appropriate machine learning model for a given task, on the other hand, yields the best results in terms of data protection, due to a significantly

increased accuracy with which objects are identified in video frame. There are several adaptations that potentially increase performance, but negatively impact data protection. For example, frame skipping provides a significant boost for performance. However, this adaptation could potentially lead to losses in anonymity. For example, in a dynamic video situation (parts of) a face could not be entirely covered by an e.g., blurring effect, because the location of the effect (the inferred location of a face) was determined based on a (four times) earlier frame. Swapping an AI model on the other hand, may lead to better accuracy, hence increased data protection, but comes at the price of performance. A limitation every AI-assisted VAP faces is determining the level of accuracy, as there is (yet) no empirical method available to check if e.g., a face was correctly detected in a video frame or not. Hence, we argue that adaptations, such as model swapping are best extensively tested (with appropriate test video material) at design time. Overall, we could show the variety of different adaptation possibilities and their impact on performance and data protection, which potentially aids developers and system designers of an AI-assisted VAP in optimizing the software.

CHAPTER 7

A Data Protection Focused Adaptation Engine for Distributed Video Analytics Pipelines

The design, development, deployment and operation of a distributed Video Analysis Pipeline (VAP) at the edge of the network is highly complex. Concrete specifications of various infrastructural parts may not be fully known at design-time of a VAP and an application's part of the VAP may be executed on different hardware that may also change during run-time. Additionally, a VAP may use many different software components to fulfill its concrete task. Those software components may also be distributed across different nodes that may be maintained by different parties. Hence, software characteristics, such as efficiency, reliability or functionality, could also VAP greatly. Those circumstances heavily influence system requirements such as data protection, performance and energy consumption. Hence, determining an optimal system configuration during design- and runtime of a VAP becomes a tremendous challenge. The concept of adaptation aims to mitigate this problem. In the domain of adaptive systems, several solutions are proposed in literature to optimize either one particular performance aspect of a VAP, e.g., execution time or latency, or focus on minimal energy consumption, or calculate a trade-off including some of those aspects. However, nowadays, most systems utilizing a VAP that records personally identifiable data have to adhere to some form of data protection regulation, such as the GDPR. Still, adaptations to increase aspects that are not related to performance, such as data protection requirements, are often second to previously mentioned performance or energy consumption characteristics of a VAP. While there is state of the art literature dealing with data protection related adaptations, most of them solely focus on increasing certain security or privacy aspects of a system, but leaving previously mentioned performance or energy consumption characteristics out of scope. To the best of our knowledge, there is no solution that covers all of these

aspects. In this chapter, we present a data protection focused adaptation engine that leverages the application- and infrastructure based adaptation space of a distributed VAP. It features an optimization algorithm to improve data protection, performance and energy consumption characteristics of a distributed VAP. The engine employs an extended system model and adaptation rules that are based on previous research.The approach builds upon two well established paradigms in adaptive systems, namely the Monitoring-Analysis-Planning-Execution (MAPE) loop principle and Event-Condition-Action (ECA) rules.

After a brief introduction, the chapter is structured as follows: First, we state the problems and challenges a distributed VAP faces with respect to data protection, performance and energy consumption in Section 7.2. Furthermore, we describe a real world use case, that is derived from a use case from FogProtect, motivating our approach and illustrating the associated challenges to data protection. Section 7.3 gives the essential background information that is needed to understand our approach. Details on the implementation of the engine are described in Section 7.4. In Section 7.5 we evaluate the feasibility and applicability of our approach. Research related to this work is discussed in Section 7.6. Section 7.7 discusses our findings and concludes the chapter.

7.1 Introduction

Edge Computing comprises a variety of different connected devices with minimal to average computing power. These devices continue to permeate deeper into our personal environment as well as in commercial and industrial areas, by sensing, processing, and storing all kind of data [VF13a]. The design, development, and deployment of distributed edge applications, e.g., AI-assisted Video Analysis Pipeline, is highly complex [Dea07]. Due to the heterogeneous hardware environment of such systems, concrete specifications of various infrastructural parts may not be fully known at design-time of a distributed application. Furthermore, various application parts (e.g., microservices) will be executed on different infrastructures and could be migrated during run-time. While the cloud is to be expected, mainly due to virtualization and containerization techniques, to have virtually unlimited computing power and storage, the capabilities of edge nodes may VAP widely. This heterogeneity of capabilities increases even more in the area of IoT, where energy consumption becomes an additional important factor. Hence, not knowing the exact infrastructural details on where a distributed application will be deployed and executed on becomes a tremendous challenge [BFI19, BFGL20, VF13a]. Additionally, a VAP may use many different software components to fulfill its concrete task. Those software components may also be distributed across different nodes that may be maintained by different parties. Hence, software characteristics, such as efficiency, reliability or functionality, could also vary greatly. Another problem a VAP may likely face, is the handling of sensitive data, such as PII, in order to allow its workflow to comply to privacy policies [DJP11, LRD19a] or security provisions such as the GDPR of the EU. For such applications, these circumstances call for data protection [DWK15a] to be the topmost priority, while performance and energy consumption come second. Integrity,

confidentiality, availability, undetectability, and unobserveability are the key elements of such protection mechanisms. The overall risk that such a system is confronted with could be assessed by looking on its various attack surfaces that are exposed to an adversary. For example, in the context of AI-assisted video analytics, the potential leakage of PII (e.g., recorded faces) from video data, e.g., due to unencrypted transmission of images (frames), is a characteristic threat to data protection of such a system [FWKM15]. However, implementing adequate mechanisms to fulfill data protection requirements is often second to improvements for performance characteristics of a system, especially in the area of Edge Computing and IoT [MAM15, SWZL12]. The concept of adaptation aims to support tackling this challenge. Based on the classical definition of control theory an adaptive system monitors its own performance and adjusts its parameters in the direction of better performance [NA12]. Computing time, data storage or latency are concrete exemplary manifestations of this classical understanding of performance. For defining adaptation rules, it is crucial to understand (i) what changes in the environment may happen, (ii) what self-adaptations the system may perform, and (iii) how those changes and self-adaptations impact the relevant system properties. Hence, adaptations in the system have to be designed in a way so that this heterogeneous infrastructural and software environment is also taken into account. Regarding performance related adaptations, several solutions are proposed in literature to optimize either one particular performance aspect of a VAP, e.g., execution time or latency, or focus on minimal energy consumption, or calculate a trade-off including some of those aspects [ZSL+18, ZSYM17, HM20, Kim20]. While there is state of the art literature dealing with data protection related adaptations, most of them solely focus on increasing certain security or privacy aspects of a system, leaving previously mentioned performance characteristics out of scope [BSG+18, BGR+15, TPGN16]. To the best of our knowledge, there is no solution that covers all of these aspects. In this work, we present a data protection focused adaptation engine that leverages the application- and infrastructure based adaptation space of a distributed VAP. The engine employs an extended system model and adaptation rules that are based on previous research from a collaborative H2020 project[AJL+21], namely FogProtect[1]. The model was specifically extended and enhanced to meet the requirements of AI-assisted VAPs at the Edge. The adaptation rules cover the application layer, as well as the infrastructure layer of a VAP system. Furthermore, the engine now features an optimization algorithm to improve performance, energy consumption and data protection of a distributed VAP and its functionalities.

Based on a real-world use case, we can show that our approach efficiently and effectively mitigates data protection risks in a running system. Additionally, it opts to find the most suitable system configuration based on an operators preferences regarding performance, energy consumption, and available functionalities (Quality of Service (QoS)).

[1] https://fogprotect.eu/

7.2 Problem Statement

In this section we describe the challenges for data protection and performance for each major task a VAP comprises. Performance in context of a VAP is not only related to computational capabilities of the system. It also covers qualitative aspects of the analysis part, i.e., the accuracy with which desired features can be inferred from video data. Accuracy is typically determined by the used AI-model and respective framework. The computational performance of each task is tightly coupled to the hardware capabilities of the device executing those tasks. Running all of the three major tasks on a single computing device can have a significant impact on the overall performance of a VAP. If performance on a single device becomes an issue, a common approach is to decouple the three main tasks of the face-blurring pipeline and execute them separately on different devices. However, this pragmatic approach may often pose a non-trivial challenge to edge computing use cases, considering distributed applications running in a heterogeneous hardware and software environment. Therefore, in [LMD21], we got a better understanding of the main drivers decreasing performance in each of the three major tasks of a VAP, in order to develop adaptation strategies to enhance performance and/or data protection quality. Hence, the approach presented in this chapter builds on top of that, enabling us to incorporate the important aspects into the proposed solution.

7.2.1 Motivating Example

To better illustrate the problems and challenges of a VAP we describe a real world use case from a FogProtect partner, motivating our approach. The use case takes place in the smart cities domain and featuring a modern urban monitoring system[2]. In general, the use case scenario can be considered as a multi-tenant system with a highly dynamic device setup. Smart lampposts, equipped with computing nodes and various sensors, are installed and distributed across the city. The heterogeneous capabilities between nodes stem from the fact that (in our specific real world use case) in an urban environment smart lamppost can be manufactured, equipped and mounted by different vendors. These smart lampposts sense and process data and share this data across the network. Data may be transferred from one lamppost to another node or directly to the cloud for further processing and/or storage. Traffic monitoring is one concrete exemplary scenario in our use case. A smart lamppost records or streams video data from a mounted camera to a computing node, where the data is further processed, streamed or stored. Various parties may access this data for further analysis. For example, this enables traffic controllers to identify and analyze potential incidents and problems regarding traffic, such as a traffic jam or a car accident, which then inform potential first responders to act accordingly. Imagine a smart lamppost records a car incident and the data gets transmitted to a traffic controller. Such data flow, as well as its content, is prone to many security and privacy threats, such as wiretapping and personal data leakage. It may not be beneficial

[2]See: `https://urbanplatform.city/`

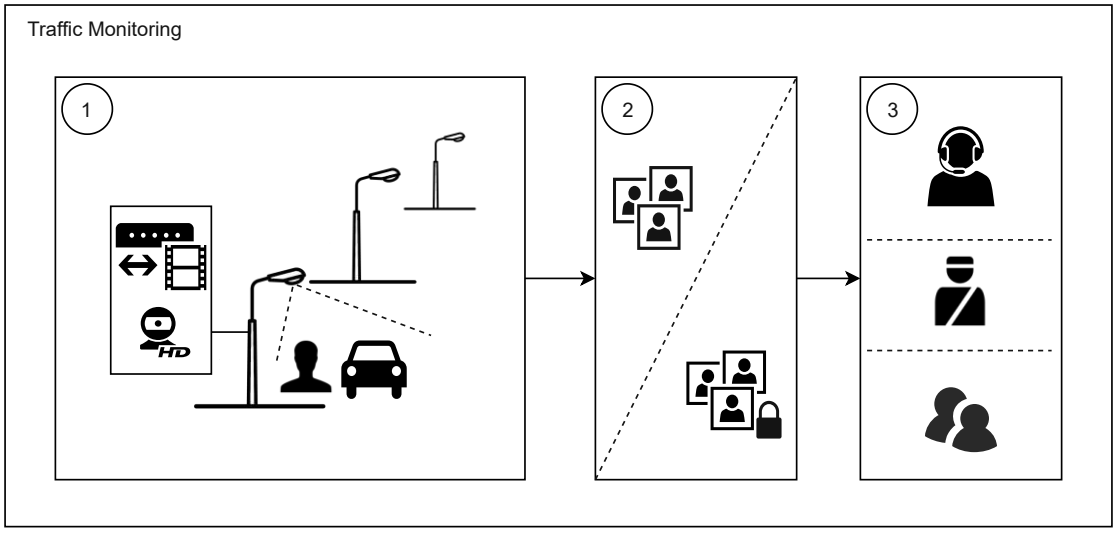

Figure 7.1: Smart lampposts recording video data that potentially contains PII. Depending on a users access rights, either anonymized or raw video is transmitted.

or desired for a traffic controller to see potentially recorded faces of people nearby the accident or even license plates of cars involved in the accident or also nearby. However, a first responder, like law enforcement, may be interested to actually identify people (as potential witnesses) or license plates of cars (in cases of e.g., hit and run scenario). Hence, depending on e.g., access rights, specific adaptations on nodes have to be triggered. A concrete example of such an adaptation is to activate an object blurring algorithm manipulating a recorded video stream. Fig. 7.1 shows an overview of the scenario.

The box with label 1 displays smart lampposts, distributed across the city, each equipped with a camera and a computing node that acts as a video gateway. A lamppost records it's environment, e.g., people and cars on the street, and transmits the video to another computing node, where it can be further processed, or viewed by a user. However, as depicted in the box with label 2, depending on the user's access rights, the video is either anonymized or not upon request. The box with label 3 displays exemplary roles of users, a traffic operator, a law enforcement officer and a normal end user, each with different access rights. While the law enforcement officer would have access to the raw video, the other two users would only be allowed to watch the anonymized version of the video.

7.2.2 Challenges to Data Protection and Performance

From a high level perspective, an AI-assisted VAP faces similar data protection challenges as any other distributed application that stores, processes and transmits sensitive data. Sticking to the information security principle of the Parkerian Hexad [Par12], such systems have to be protected against security breaches affecting one or more of the fundamental attributes of information, namely: Confidentiality, Possession or Control,

Integrity, Authenticity, Availability and Utility. Identifying the risks associated with those security breaches and how to mitigate them is typically a design-time activity carried out by information security experts following standards like the ISO27000 [Int18]. A concrete implementation of the scenario described in Section 7.2.1 makes heavy usage of AI-based tasks in order to provide additional services like automatic license plate detection, but also enable or enhance data protection mechanisms. A simple, yet not easy task to ensure and respect the privacy of people recorded by a system like we described, is the anonymization of personally identifiable features like faces of people or license plates of cars. Furthermore, the actual physical location where (personal) data is processed plays an important role in regulations like e.g., the GDPR in Europe. Additionally, a distributed VAP needs to take care of secure data transmission as well. A secure data transmission may not only cover encrypted communication, but could also leverage cutting edge AI-techniques like model splitting[ZCL+19] to further enhance privacy. Basically, each phase of the VAP faces specific challenges to either performance and/or data protection. In the first phase (P1), the system is typically concerned with video pre-processing tasks, such as encoding or up/downsampling. Such tasks are commonly done if the camera records with different parameters (e.g., framerate or resolution) than the desired output video, i.e., the video data that will be transmitted to, and analyzed by, a component of P2. However, transforming video data to a specific format is a heavy computational task, hence significantly affecting the performance of P1. Therefore, manufactures like Nvidia integrate dedicated chips into their hardware and offer dedicated SDKs to facilitate this task[3]. The performance of the second task of the pipeline is heavily dependent on the used AI-framework as well as on the underlying infrastructure it operates on. In the context of AI-based inference tasks, like stated previously, performance is not only related to processing speed but also to the accuracy with that e.g., an object was detected in an image. A well pre-trained model embedded in modern sophisticated frameworks like e.g., *TensorFlow* or *YOLO* will generally allow for higher inference speed with high accuracy. However, such frameworks are not necessarily available and/or optimized for each system architecture like e.g., ARM, x86, x64, but developers constantly aim to port a framework to another architecture or provide lightweight alternatives such as *TensorFlow Lite* or *YOLOv3-tiny*. Furthermore, differences regarding performance can also stem from the usage of legacy versions of such machine learning frameworks. Executing this second major task of the pipeline on devices featuring hardware like a GPU or even dedicated AI-hardware like TPUs (Google) or Neural Processing Unit (NPU) (Microsoft) do also heavily contribute to performance gains. This stems from the fact that inference tasks are parallelizable to a high degree and TPUs or NPUs are custom ASIC chip-designed from the ground up for machine learning workloads. The aforementioned heterogeneity of the underlying hardware infrastructure makes deployment, migration or offloading inference based tasks or components increasingly complex. The third task involves potentially multiple post-processing activities. Regarding the example of face-anonymization, this would be the anonymization process of recognized face in a video frame, i.e., a graphic overlay is drawn over the face in the image. Although it is well researched that making a

[3] https://developer.nvidia.com/nvidia-video-codec-sdk

face unrecognizable in visual data does not fully guarantee the anonymity of a person, it still facilitates such anonymization task to a high degree, mostly by making the de-anonymization process significantly harder for an attacker. However, other typical post-processing steps could include again transcoding or encoding tasks, sampling rate conversions, resolution alterations, etc. to e.g., facilitate and optimize for streaming the output video.
Another important aspect of a VAP is QoS. Due to hardware or software constraints of the node executing (parts of) a VAP, the processing of data protection mechanisms of the application may not keep up with a reasonable output video quality, i.e., the frames per second (FPS) of the output video are far too low compared to the input video. Hence, a person viewing the output video would experience a significant loss in QoS using the application.

7.3 Background Information

In this section we provide background information that is needed to understand the approach and additionally builds the foundation of our work. First, we will give a short description on the term data protection from a legal perspective according to the GDPR. Our approach is not limited to that kind of data only (i.e., personal data, as explained later), but is rather capable of operating with any kind of sensitive data. Second, we give a brief overview on previous work, i.e., the main components from RADAR [MKL+21], which we extended, updated and enhanced for our approach. Third, we explain what kind of adaptations are possible within our approach.

7.3.1 Data Protection

The term data protection is not very well defined from a global perspective. It is related to information security but not the same. The ISO 27000:2018 standard describes information security as the "preservation of confidentiality, integrity and availability of information" [Int18]. Furthermore, it states that other information security aspects, such as authenticity, accountability, non-repudiation, and reliability, should also be considered. According to the GDPR, data protection mechanisms should prevent personal data breaches [Gen16]. Personal data means "any information relating to an identified or identifiable natural person", e.g., name, address or location data. A personal data breach means "a breach of security leading to the accidental or unlawful destruction, loss, alteration, unauthorised disclosure of, or access to, personal data transmitted, stored or otherwise processed". Hence, data protection includes information security aspects such as confidentiality or integrity concerning personal data. However, data protection goes beyond information security, for example by defining specific roles as well as their rights and obligations related to personal data. Our approach is able to handle the following roles according to the GDPR:

- Data Subject: A data subject is "an identified or identifiable natural person . . .

who can be identified . . . by reference to an identifier such as a name . . . or to one or more factors specific to the . . . identity of that natural person".

- Data Controller: A data controller "determines the purpose and means of the processing of personal data".
- Data Processor: A data processor "processes personal data on behalf of the data controller".
- Third Party: A third party is authorized to process personal data under the direct authority of the controller or processor.

From a legal point of view, data protection must be ensured in an edge computing system processing personal data. Therefore, several information security aspects need to be taken into account when operating a system that processes sensitive data, hence adhere to the GDPR. In order to model and operate such systems with RADAR, we decided to enrich the basic approach with information security aspects from the Parkerian Hexad [Par12]. The Parkerian Hexad adds three additional attributes to the traditional security attributes of the CIA triad (confidentiality, integrity, availability). This enables our updated approach to cover the following aspects of information security:

- Confidentiality
- Possession or Control
- Integrity
- Authenticity
- Availability
- Utility

7.3.2 Foundation of our approach

This work is based on the so called RADAR (Run-time Adaptations for DAta pRotection) approach presented in [MKL+21]. RADAR aims at ensuring data protection in dynamically changing cloud-based systems and other related system concepts such as edge or fog computing. RADAR is deployed as a central control unit that manages the self-adaptive system. It is assumed that both monitoring of the system and execution of adaptations in the system are carried out by the system itself or components between the system and RADAR. The Eclipse Modelling Framework (EMF)[4] combined with Henshin[5] is used to enable model-based runtime adaptations, specifically optimized towards data protection. Details on the adaptation rules, their format and underlying

[4] https://www.eclipse.org/modeling/emf/
[5] https://www.eclipse.org/henshin/

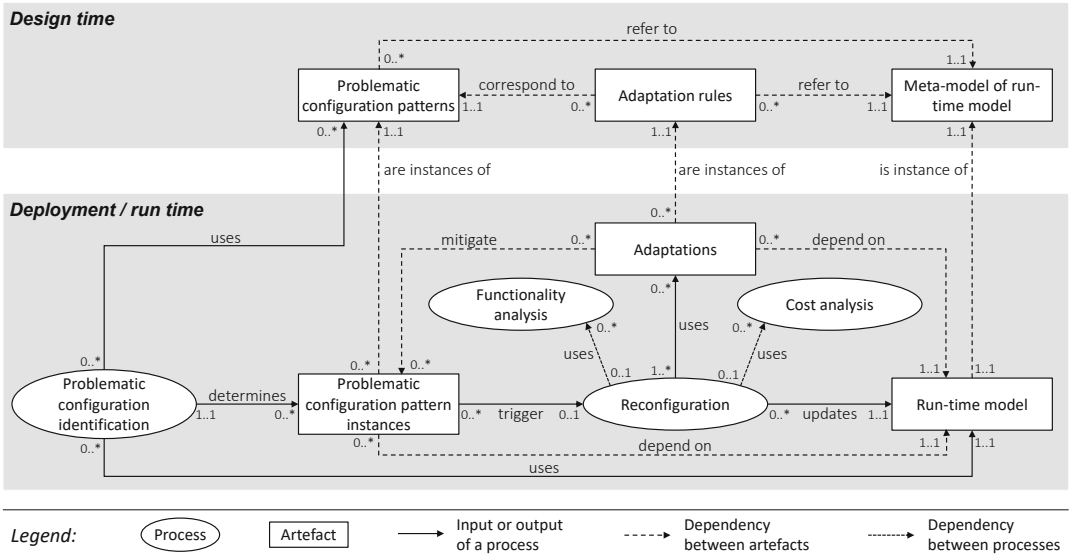

Figure 7.2: Overview of the RADAR approach

RARAD Adaptation language can be found in [MKL+21]. As shown in Fig. 7.2, RADAR consists of multiple components that are described now.

Meta-Model

RADAR includes a *meta-model* that defines the modelling constructs that can appear in the *run-time model*, a language to specify *problematic configuration patterns* (Problematic Configuration Pattern (PCP)s), and *adaptation rules* to mitigate problematic configurations. Thus, the meta-model ensures compatibility between the problematic configuration patterns, adaptation rules, and the run-time model. Previous research has shown that using established modelling languages from the area of security, such as UMLsec or SysML-Sec, is not sufficient to model neither edge computing systems nor data protection aspects related to edge computing systems [Lau21]. Therefore, an independent framework is needed.

The meta-model is created by using the Eclipse Modelling Framework (EMF) and its graphical editor. In general, the meta-model is similar to UML class diagrams. Nodes are represented as classes, edges are represented as relations between classes. Properties of nodes are represented as attributes and object-oriented concepts like inheritance are supported. The meta-model extends the well established TOSCA standard, a modeling language defined by the Organization for the Advancement of Structured Information Standards (OASIS) [6]. In the RADAR meta-model nodes can be any kind of entity of a system, like a compute node or a GDPR role as described in section 7.3.1. From a

[6] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca

high level perspective, a compute node typically comprises various software components. These components are also modeled as nodes. Which hardware and software components belong to a certain type of compute node is modeled via relations.

Problematic Configuration Patterns

A *PCP* describes a pattern that exists in the *run-time model* if the system is in a problematic state. A run-time model represents relevant aspects of the system and its environment in the style of an Unified Modeling Language (UML) object model. It is fully based on the meta-model and gets continuously updated by a system monitoring component at run-time. PCPs are defined at design-time and specifically focus on configurations that threaten data protection or QoS aspects of the system or lead to high overall costs. By cost, we mean, for example, the cost of cloud rental that Infrastructure as a Service (IaaS) providers incur. At run-time, instances of PCPs can be detected by using RADARs pattern-based algorithms. The goal is to avoid as many PCP instances inside of the run-time model as possible. To design PCPs a graphical modeling language, called PCP Language, was created. Similar to the run-time model it is in the style of UML object diagrams. To create PCPs the PCP Language is mapped to Henshin transformation rules. Henshin also enables graphical modelling by providing a GUI tool.

Adaptations

Whenever the run-time model changes, an algorithm is checking whether *instances of PCPs* can be found in that model (see *problematic configuration identification* in Fig. 7.2). If this is the case, RADAR uses *adaptation rules* to identify possible solutions to mitigate the PCP instance (see *Reconfiguration* in Fig. 7.2). Adaptations rules are created at design time. They are associated with the PCP that they should mitigate. For each PCP multiple adaptations can exist. Each adaptation rule captures a potential type of adaptation in a well-formed manner. An adaptation involves adding or removing objects and relations as well as changing attribute values according to the specific rule. Each adaptation rule is split into three parts, namely (i) a PCP and its instance that have to exist in the run-time model, (ii) a precondition that adds further constraints on the run-time model, and (iii) the adaptation action that describes changes to the run-time model. A graphical Adaptation Language was defined to design adaptation rules and was mapped to Henshin transformation rules as well.

If RADAR cannot find any PCP instances, it tries to detect improvement / optimization adaptations that increase the amount of working functions (see *Functionality analysis* in Fig. 7.2) and lower the overall costs (see *Cost analysis* in Fig. 7.2) without creating new threats to data protection. After RADAR has found the optimal adaptation (in the best case all PCPs are mitigated, the amount of available functions is at its highest and the overall costs at their lowest) the adaptation will be executed on the system. However, a predefined prioritisation defines whether the amount of available functions or the overall costs are more important.

It has to be noted that there may be multiple PCP instances in the run-time model, there may be multiple adaptations to mitigate a given PCP instance, and an adaptation to mitigate a PCP instance may also mitigate or create other PCP instances. Hence, a sequence of adaptations may be needed to mitigate all PCP instances. In this sequence, the order of adaptations may be important because an adaptation may become applicable only after another adaptation was carried out.

In [LMD21] we investigated the adaptation space of AI-assisted data protection for resource constrained VAPs. These findings build the foundation of the modeled adaptions designed, implemented and evaluated in this work and described in the following sections.

7.4 Adaptation Engine Extensions

In this section, we describe the basic functionality of our proposed data protection focused adaptation engine. Furthermore, we explain how the underlying concepts of previous research were improved in order to tackle the challenges faced by an edge computing based distributed VAP, as described in section 7.2.

The meta-model, as described in section 7.3, was neither able to tackle detailed infrastructural aspects of an edge computing based system, nor was it capable of modeling fine-granular security and privacy controls that are vital for many systems, e.g., a Video Analysis Pipeline, that need to adhere to some sort of data protection regulation such as the GDPR. Moreover, it was not possible to consider additional metrics like energy consumption or performance of a system while finding the optimal adaptation. In addition, it was also not possible to change the order of the prioritization of such metrics while solving data protection concerns in the first place. Therefore, we made several extensions and enhancements to our previous work.

7.4.1 Conceptual extension

We started by developing conceptual extensions and enhancements of the RADAR meta-model to address the gaps mentioned above. The extensions made to the meta-model can be seen in Fig. 7.3. It has to be noted that this figure only shows newly added classes and relations. A full representation of the meta-model in the style of an UML class diagram can be found online[7]. In the following, the changes and newly added model constructs are explained in detail.

Infrastructure and Application Aspects

To enable the modeling of infrastructural details we enhanced the RADAR meta-model by adding new nodes, relations and attributes. Especially the concepts of the so called “compute” and “software component” node were thoroughly revised. First, we now employ inheritance to distinguish between different types of compute nodes based on

[7] See `https://git.uni-due.de/fogprotect/vap-adaptation-engine`

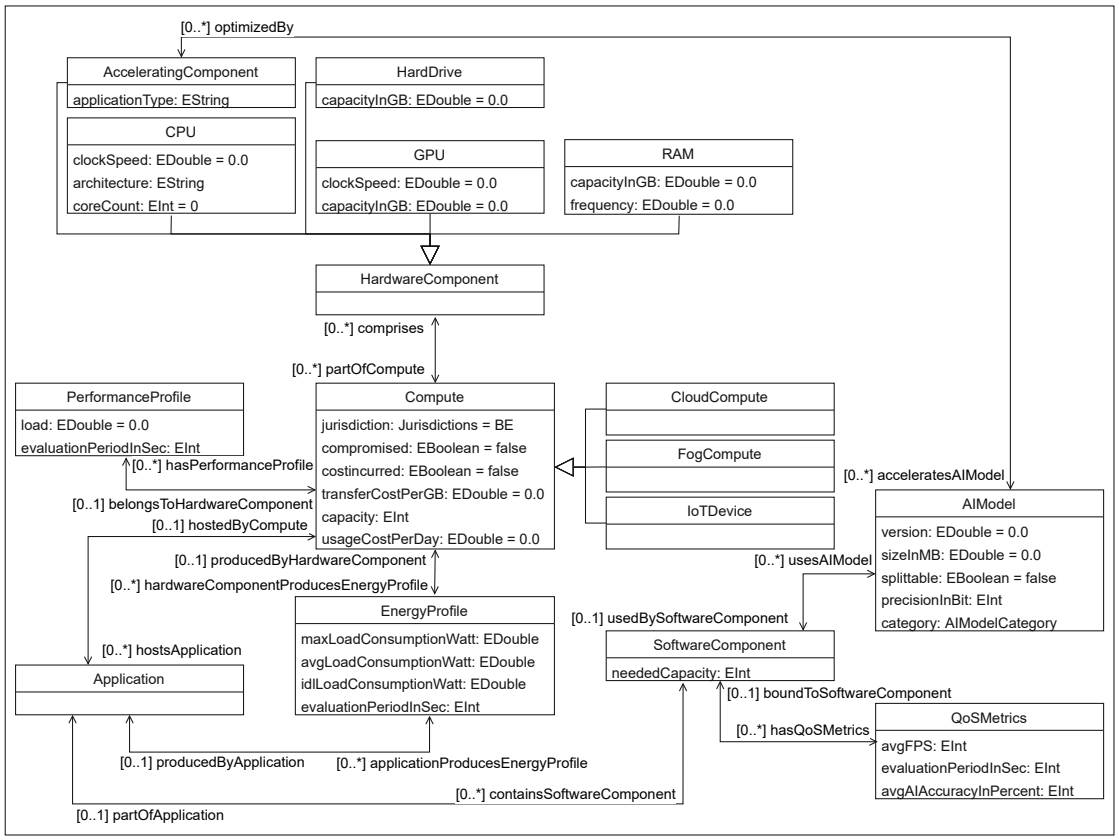

Figure 7.3: The reworked and extended part of the RADAR meta-model

the three layers of edge computing architecture. The new types are called “IoTDevice”, “FogCompute” and “CloudCompute”. This enables modeling more precise PCPs and adaptation rules while also keeping the possibility to model generic compute nodes.

Second, each compute node can now comprise hardware components. This concept enables modeling of detailed hardware information that may be relevant when the engine is trying to identify PCP instances or trying to find the optimal adaptation, especially with regards to performance and energy consumption variables as described in section 7.4.2. Again, inheritance is used to represent different types of hardware components, namely “CPU”, “GPU”, “AcceeleratingComponent”, “HardDrive” and “RAM”. They differ on the basis of certain attributes like e.g., “clockspeed” or “capacityInGB”.

Third, each hardware component may have a performance and energy profile assigned. We introduced those profiles to the meta-model to allow energy consumption and performance to be considered when running the adaptation algorithm. We assume that at runtime a monitoring component provides the required information as average values for a selected evaluation period.

Lastly, we also rethought the concept of software components. Prior to our changes,

each compute node was able to host multiple software components. Now, we added a node called “application” that is hosted by a compute node, and a software component describes a dedicated part of an application, for example an API or a processing algorithm. Therefore, software components take over specific tasks like communication. To represent the communication between software components we are using the following chain of nodes: software component - data flow (consisting of one or multiple data records) - software component. Thus, an application does not replace a software component, it rather comprises and clusters multiple software components. Additionally, an application can also be linked to an energy profile, similar to hardware components, to e.g., identify and handle the main drivers of total energy consumption of a node.

To store additional details that can be used to evaluate the overall performance of the managed system when searching for adaptations, each software component is related to the nodes called “QoSMetrics” and “AIModel”. Currently, QoS metrics only consider metrics that are relevant for VAPs (average frames per second, average AI accuracy in percentage). However, further metrics can easily be added when using the engine in other environments. The AI model that may be used by a software component is represented by a new node storing information like the precision, the AI-model category (e.g. face detection or dedicated object detection) and whether the model is splittable or not. Furthermore, an AI task, leveraging a specific AI-model, can be accelerated by an accelerating hardware component, such as a GPU or TPU. An example run-time model consisting of all newly added or reworked concepts can be found online in our git repository. The run-time model is based on the use case described in section 7.2.1 and will be discussed in detail in section 7.5.

Data Protection Aspects

We also extended the meta-model to enable modeling of security features. A security feature represents the bridge between a data record that needs to be stored or transferred in a secure way, and hardware or software components that are securing the data record. Our model covers confidentiality, integrity, availability, possession control, authenticity and utility as types of a security feature based on the Parkerian Hexad described in section 7.3.

In addition, we enhanced the PCP language. Prior to our extensions, it was only possible to check whether an attribute value is equal to a reference value that has to be defined at design time. Now, PCPs can also check whether an attribute value is lesser or greater than a reference value. This enables the coverage of specific VAP data protection aspects and QoS metrics, such as meeting the minimum requirements of FPS or AI model accuracy. The changes made to the PCP language and the mapping to the Henshin language are shown in Fig. 7.4 and are highlighted in green. It can be seen that a Henshin condition has to be used to compare an attribute to a given value.

Furthermore, reference values can now be changed at runtime which allows reacting to changing expectations from the environment. For example, additional software could

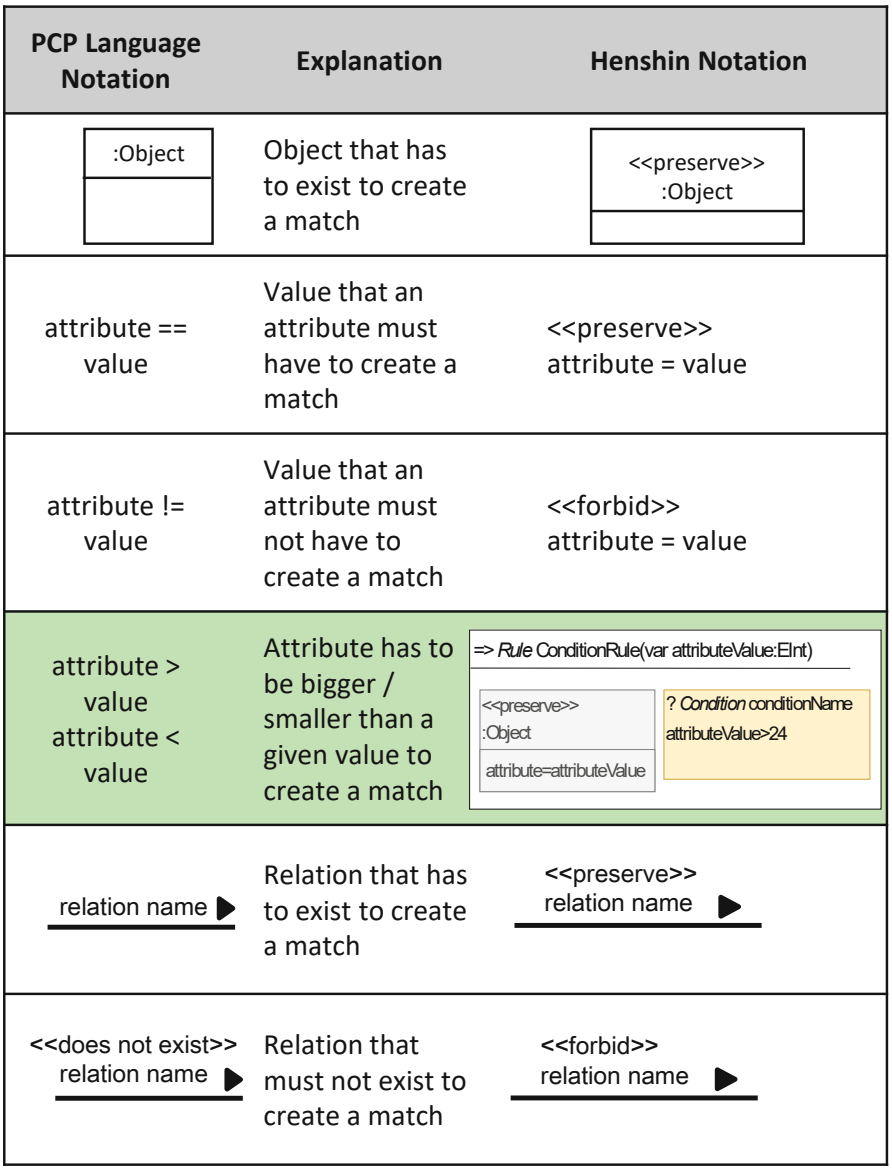

Figure 7.4: Extensions to the PCP language and mapping to Henshin

be used to calculate a minimum FPS value that needs to be met by an object blurring software component to ensure that data protection and QoS requirements are fulfilled.

7.4.2 Algorithmic Improvements

The main goal of each adaptation is to ensure adequate data protection. However, as stated in section 7.3 multiple solutions to mitigate data protection risks and therefore ensuring data protection may exist. These adaptations may have a different impact on

metrics such as energy consumption, performance, costs, and available functionality. In order to adapt the system in the best possible way, it is therefore necessary to include both data protection and above-mentioned metrics in the adaptation planning algorithm. However, because protecting personal data is a priority, it is not possible to balance data protection and system metrics. By improving the algorithms of RADAR, we enable the evaluation of further metrics, namely energy consumption and performance. Additionally, we reworked the way available functions and total costs are calculated for a run-time model or proposed adaptation model. Lastly, the order of the metrics is no longer predefined but dynamically changeable at runtime by exposing a dedicated API.

First, functionalities can now be modeled similar to PCPs by using the same notation as the PCP language and the same modeling tool provided by Henshin. Therefore, a functionality can, for example, be modeled by a system architect who defines what a functionality in the given system represents. Therefore, a functionality is now represented as a sub-graph and will be identified by using graph-pattern-matching. The amount of available functions can be understood as a QoS metric. The overall goal is to opt for a high amount of available functions. It should be noted that the calculation of the amount of available functions, costs, energy consumption and performance follows a global approach. A run-time model may consist of context nodes that are not part of the managed system and thus should not be considered when calculating global values. Hence, we are only considering values to be part of the calculation if they are part of a functionality instance that was found by our graph-pattern-matching algorithm.

Second, by using the information from the newly added nodes “EnergyProfile”, “PerformanceProfile”, “AIModel” and “QoSMetrics” it is now possible to calculate the total energy consumption and the performance value of a model. To calculate total energy consumption, all *EnergyProfile* nodes that are part of a functionality instance are considered. The average load consumption in Watt of each energy profile is used to calculate the overall sum. To calculate a performance value, we are using an equation that results in a numeric value. It combines different attribute values that use different scales like FPS (natural numbers) or AI model accuracy (percentage). Therefore, those different attribute values need to be normalized first. However, considering FPS as an example, a linear transformation to normalize values between 0 FPS and 60 FPS would not be feasible and accurate in order to compare two nodes. For the human eye every video with a frame rate around 24 FPS is perceived as fluid. Differences in frame rates above 24 FPS are significantly harder to notice, than the differences below that value. In terms of QoS a human may not see the difference between 50 and 60 FPS. However, the perceived difference between 14 and 24 FPS is significant. Internally our engine expects such metrics as scalar values between 0 and 1 to combine them with other performance related metrics to calculate an overall performance value. Therefore, we used a logistic function to normalize the FPS value where the slope grows steep for values between 0 and 24, but flattens out for values above 24. Equation 7.1 represents the mathematical notation of the logistic function.

To calculate the global performance value, we first identify all *QoSMetric* objects and

AIModel objects that are in relation to a software component that is part of an identified function. Second, we normalize the FPS value stored in the QoSMetrics objects. Afterwards, the normalized FPS value and the percentage value of the AI model accuracy are summed up in a distribution of $\alpha = 50\%$ as shown in Equation 7.2. However, our approach can be extended to use any kind of function that is appropriate for a specific QoS metric, as long as the output values are between 0 and 1. The newly added performance metric has been added to Equation 7.2 and α needs to be adjusted to achieve the desired *weighting* for overall QoS.

$$f(x) = (\frac{(a-b)}{(1+e^{-c*(x-d)}))} + b) = A \tag{7.1}$$

and

$$P = A * \alpha + B * (1 - \alpha) \tag{7.2}$$

where:

a = the curve's top asymptote value
b = the curve's bottom asymptote value
c = the logistic growth rate of the curve
d = the x value of the sigmoid's midpoint
x = input FPS value
A = normalized FPS
B = AI model accuracy in percent
P = performance value
α = performance metric weight

Algorithm 7.1 shows the model analysis algorithm. The algorithm is given a model M as an input to perform the model analysis (line 1). At first, two empty sets to store the PCPs and available functionalities found in M are created (line 2-3). Afterwards, placeholders to store the global numeric cost, energy consumption and performance value are created (line 4-6). By using graph-pattern matching, the engine is searching for PCP instances in M using a set of all PCPs designed at design time. Whenever a PCP instance is found, it is stored in PCP_M (line 7-12). The same procedure is used to identify instances of predefined functionality patterns. Whenever an available functionality is found, it is stored in F_M. By using the functionality instance costs, energy consumption, and performance are calculated and the result is added to the global placeholders C_M, E_M, P_M. (line 13-21). Finally, the PCP instances (PCP_M), the functionality instances (F_M), and the calculated system metrics are stored in M (line 22-24).

To determine whether a possible adaptation is better than another, we are comparing models. As stated in Section 7.3, the engine is searching for the best possible system configuration by evaluating sequences of adaptations that are reachable from the current run-time model. To do so, a search tree is constructed. A child node represents the run-time model obtained from the run-time model of the parent node through an adaptation.

Algorithm 7.1: Model analysis algorithm

```
1  Input: M ;                                   // model to be analysed
2  PCP_M ← {} ;                                 // set of PCP instances in M
3  F_M ← {} ;                                   // set of available functions in M
4  C_M ← 0 ;                                    // global costs of M
5  E_M ← 0 ;                                    // global energy consumption of M
6  P_M ← 0 ;                                    // global performance value of M
7  PCP_A ← getPCPs() ;                          // set of all designed PCPs
8  foreach pcp ∈ PCP_A do
9      if pcp exists in M then
10         add pcp to PCP_M
11     end
12 end
13 F_A ← getFunctions() ;                       // set of all designed functions
14 foreach f ∈ F_A do
15     if f exists in M then
16         add f to F_M
17         C_M += calcCosts(M, f)
18         E_M += calcEnergyConsumption(M, f)
19         P_M += calcPerformance(M, f)
20     end
21 end
22 M.setPCPs(PCP_M)
23 M.setFunctions(F_M)
24 M.setMetrics(PCP_M.length, F_M.length, C_M, E_M, P_M)
```

The search tree is built up during the search, by iteratively adding unexplored child nodes to the nodes already visited. An evaluation of different strategies has shown that a best-first search approach leads to the best results [MKL+21]. The model comparison takes the prioritization of the comparison metrics into account. By using an API, a human operator can decide which metric should be used first to optimize the VAP after ensuring adequate data protection.

Algorithm 7.2 explains how the best model is determined. First, the algorithm keeps track of the best solution and the path from the root node to the respective solution (line 1-2). Moreover, the current prioritization order of the metrics list is queried (line 3). As long as unexplored nodes exist (set of nodes S is not empty, starting with the current run-time model M^{RT}, line 4-5), the algorithm is searching for the best node. While evaluating nodes from S, a selected node M out of S is first analyzed with regard to the amount of PCP instances, the amount of available functions, the overall energy consumption, the overall performance value and the total costs (line 6-7). Afterwards, M is compared to the actual best node by comparing both models. Whenever two models

Algorithm 7.2: Model comparison algorithm

```
1  best ← NULL ;                          // best solution found so far
2  bestPath ← NULL ;                  // path from root to best solution
3  ML ← getListOfMetrics() ;                  // sorted list of metrics
4  S ← {M^RT} ;                  // set of nodes that are being explored
5  while S ≠ ∅ do
6      M ← selectNode(S);
7      AnalyzeModel(M);
8      i ← 0;
9      CM ← ML(i) ;                          // comparison metric from ML
10     while CM ≠ ∅ do
11         CompareModels(M,best,CM);
12         if M is better than best in regard to CM then
13             best ← M;
14             bestPath ← path from M^RT to M;
15             break
16         end
17         if M is worse than best in regard to CM then
18             break
19         end
20         i++ ;                  // M and best are equal in regard to CM
21     end
22     T ← generateChildren(M);
23     S ← S \ {M} ∪ T;
24     if termination criterion then
25         break
26     end
27 end
28 return bestPath
```

are compared, the algorithm picks a metric in a defined order from the list of metrics (line 8-9). As long as there are further comparison metrics (line 10), the selected comparison metric is used to compare M to the best solution (line 11). If the calculated values for a metric are equal the next metric is used for comparison, otherwise M is either better or worse than the best solution (line 12-21). If M is better than the best solution in regard to a specific comparison metric, the best solution is replaced by M, the path to the best solution is stored, and the comparison ends (line 12-16). The comparison also ends if M is worse than the best solution (line 17-19). It is important to state that the amount of PCP instances will always be the first metric that will be used for comparison (line 9). In the end it should be possible to state which of the two models is optimal. After the current node has been evaluated, it is removed from S and the children of the node are added to S instead (line 22-23). To avoid an endless search in the potentially unbounded

space of possible solutions, the search is aborted when a predefined termination criterion is met (lines 24-26). Such a termination criterion could be for instance a maximum allowed time budget for the execution of the algorithm. Finally, the algorithm returns the adaptation sequence leading to the best found solution (line 28), which should then get executed in the real system.

As already stated, the algorithm is capable of using different prioritization orders for the list of comparison metrics. The order of the metrics can be changed at runtime by using an API while the engine keeps the amount of PCP instances (i.e., threats to data protection) always as the metric with the highest priority to optimize towards.

7.4.3 Summary

To summarize, we highlight the most important additions and upgrades to RADAR in order to fulfill the needs of a data protection focused edge computing system, particularly for a distributed AI-assisted VAP:

- Added security and privacy features to the meta-model according to the Parkerian Hexad model.
- Added machine learning entities to the meta-model to enable the modeling of AI-based systems.
- Enhanced the meta-model to allow for fine grained modeling of infrastructural aspects of a system.
- Added energy consumption and performance as part of optimization goals.
- PCPs can now handle set operators, thereby enabling the algorithm to dynamically compare either monitoring variables or static variables. This is for example needed to enable software compatibility or version checks.
- The adaptation algorithm is now able to change the prioritization of different metrics namely costs, QoS (available functions), energy consumption, performance, while always aiming for the lowest amount of PCP instances.

7.5 Evaluation

In this section we demonstrate how the engine behaves with different configurations and correctly solves PCPs, based on a simplified example implementation of the traffic monitoring use case as described in Section 7.2. The runtime behavior of RADAR has been evaluated in previous work [MKL+21]. Additional information on the evaluation of the pattern-matching algorithm can be found on the Henshin website[8].

[8] https://www.eclipse.org/henshin/publications.php

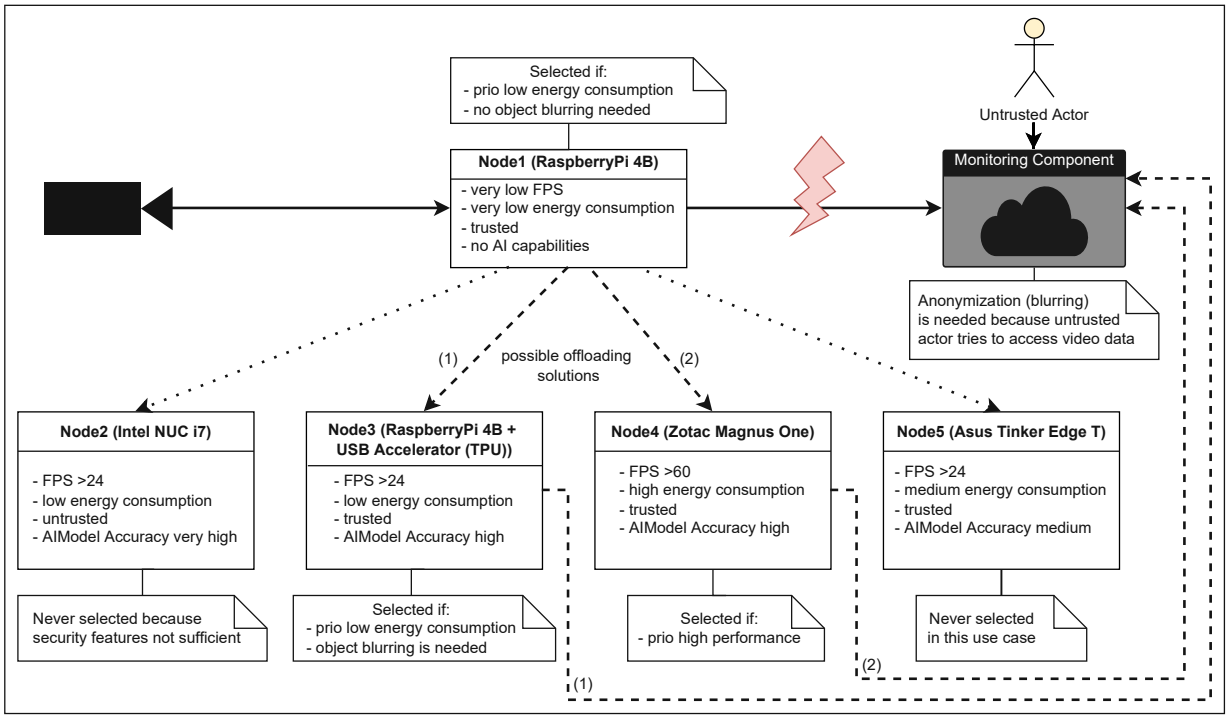

Figure 7.5: Descriptive problem representation of the traffic monitoring use case model

7.5.1 Evaluation of the considered Use Case

In order to evaluate our approach we modeled a system configuration comprising five heterogeneous compute nodes and defined a scenario typical for the considered traffic monitoring use case. The system configuration can be seen as a part of the traffic monitoring system, e.g., a large crossing with five smart lampposts. Initially, the system is configured so that low energy consumption is prioritized. As long as only trusted actors access the monitoring component, an unanonymized video is transferred.

Fig. 7.5 displays a descriptive simplified model representation that reflects our considered system configuration and use case scenario. It can be seen that the different edge nodes differ in their video processing performance (frames per second), their AI model accuracy (object detection accuracy in per cent), their average energy consumption (in watt), and their security features (trusted / untrusted). The nodes model real world edge computing nodes, highlighted by their factory naming, e.g., a RaspberryPi 4B, and their respective capabilities. Solid lines represent the initial situation. Dotted lines represent possible offloading solutions that are not taken into account due to data protection and optimization reasons. Dashed lines represent possible offloading solutions depending on the prioritization of performance vs. energy consumption. In the following, the different scenarios are described in detail.

In the use case scenario, we handle a request from an untrusted user wanting to access

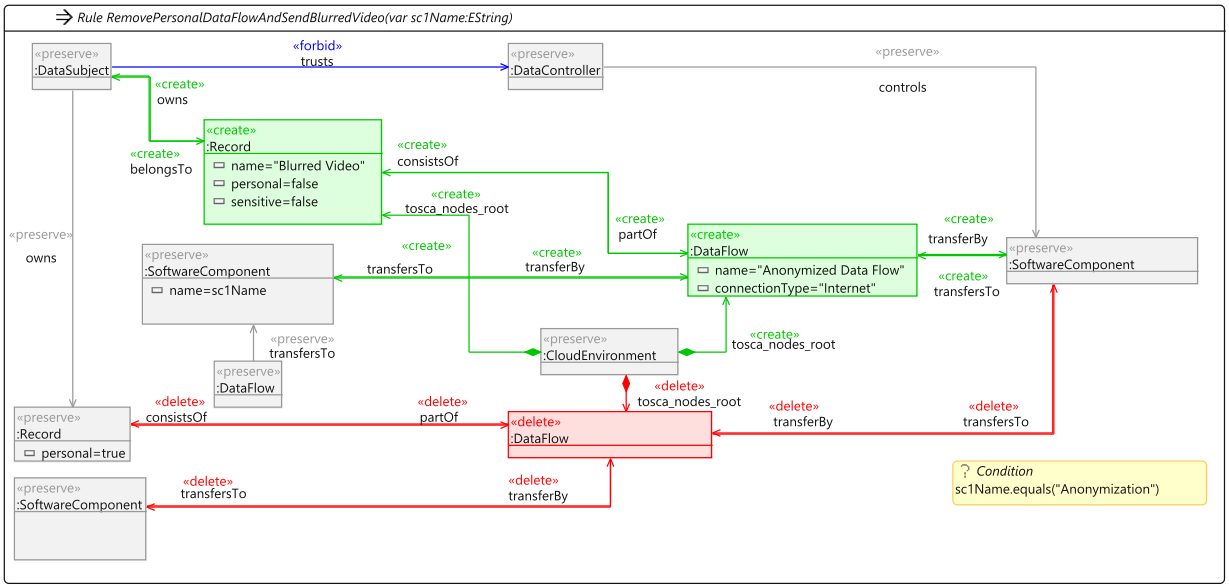

Figure 7.6: Adaptation rule that activates video anonymization

video data that is recorded by a smart lamppost. This actor is only allowed to access anonymized video data. The access of the untrusted actor is detected by our engine as a PCP instance, highlighted in Fig. 7.5 with a thunderbolt. Therefore, the goal is to mitigate this PCP instance while maintaining low energy consumption. As described in section 7.4, the algorithm combines several possible adaptations and compares different system configurations.

The best solutions consist of two adaptations that need to be both carried out subsequently. First, an anonymization software component needs to be activated, and a data flow transferring video data with blurred faces and number plates needs to replace the data flow transferring the unanonymized video. The associated adaptation rule modeled with Henshin is shown in Fig. 7.6. Concretely, it shows that whenever a *personal record* is transferred via a *data flow* to a *software component* that is controlled by an *untrusted data controller* (precondition), the particular data flow is removed and a new *data flow* transferring a new *non-personal video* from a *software component* called "Anonymization" to the *software component* used to access the video is created (adaptation action). The corresponding PCP is not represented separately because it is already part of the adaptation rule as its precondition.

It has to be noted that another solution would be to adapt the system in a way in which the video stream would be stopped. However, this adaptation implies a drastic decrease in available functionality and should therefore only be used as a last resort.

In this particular use case, the adaptation that activates video anonymization is the best choice. However, another PCP instance is created when enabling video anonymization at *Node1* because *Node1* is not capable of processing the video fast enough (output

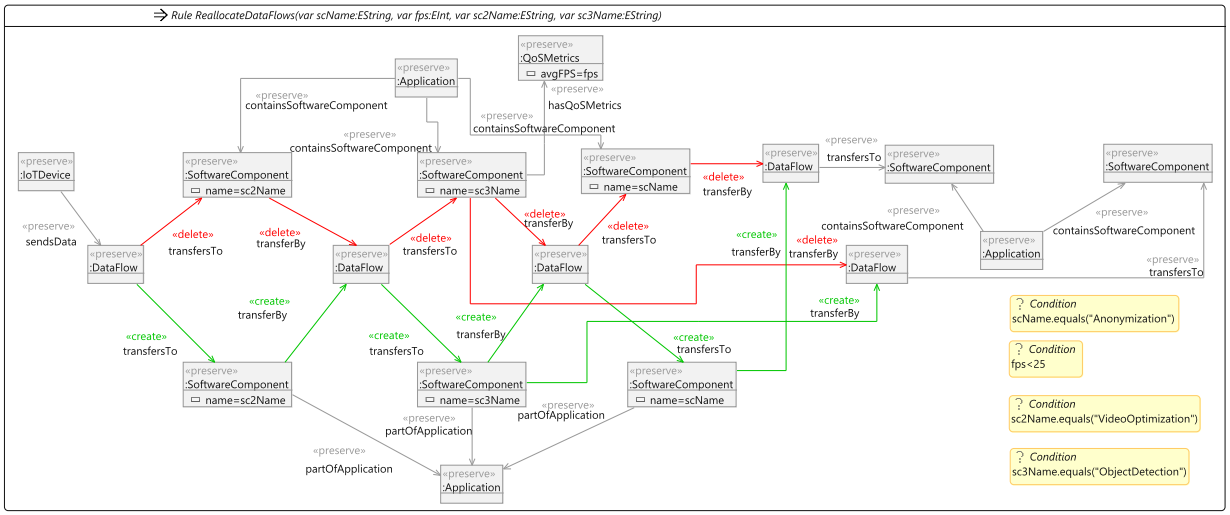

Figure 7.7: Adaptation rule that reallocates data flows

video FPS < 25) to ensure QoS at an acceptable level. Hence, a second adaptation that offloads the video processing from *Node1* to another node has to be carried out. Fig. 7.7 shows the corresponding adaptation rule. Again, the PCP instance is already part of the adaptation rule as its precondition: Whenever the *avgFPS* of an *QoSMetrics* object is below 25 and a *software component* called "Anonymization" is transferring video data to another *software component* that is not part of the same *application* a PCP instance exists. In this case, the adaptation rule stipulates the following adaptation action: reallocate all existing *data flows*, starting at an *IoT device* and ending at *software components* that are not part of the edge nodes *application*. When reallocating, switch from each *software components* used before to *software components* that have the same name and are hosted on another edge nodes *application*.

Four different system configurations have to be compared by the engines' algorithm, each related to one of the four existing nodes shown in Fig. 7.5. All four nodes are capable of processing the video adequately, allowing for an continuous video anonymization. However, in this particular use case only *Node3* and *Node4* are reasonable possibilities because an offloading to these nodes mitigates all PCP instances and optimizes the system configuration based on the desired metric prioritization.

When prioritizing optimal energy consumption, *Node2* should be selected due to its low energy consumption. However, the algorithm always considers data protection the top most priority before optimizing the system configuration towards a QoS goal. Reallocating the video processing from *Node1* to *Node2* leads to a new PCP instance because *Node2* is untrusted. Therefore, *Node2* is not taken into consideration when comparing possible system configurations in terms of improving the system configuration.

A reallocation to *Node3*, *Node4*, or *Node5* does not lead to a new PCP instance. When comparing the remaining possible system configuration options in terms of energy con-

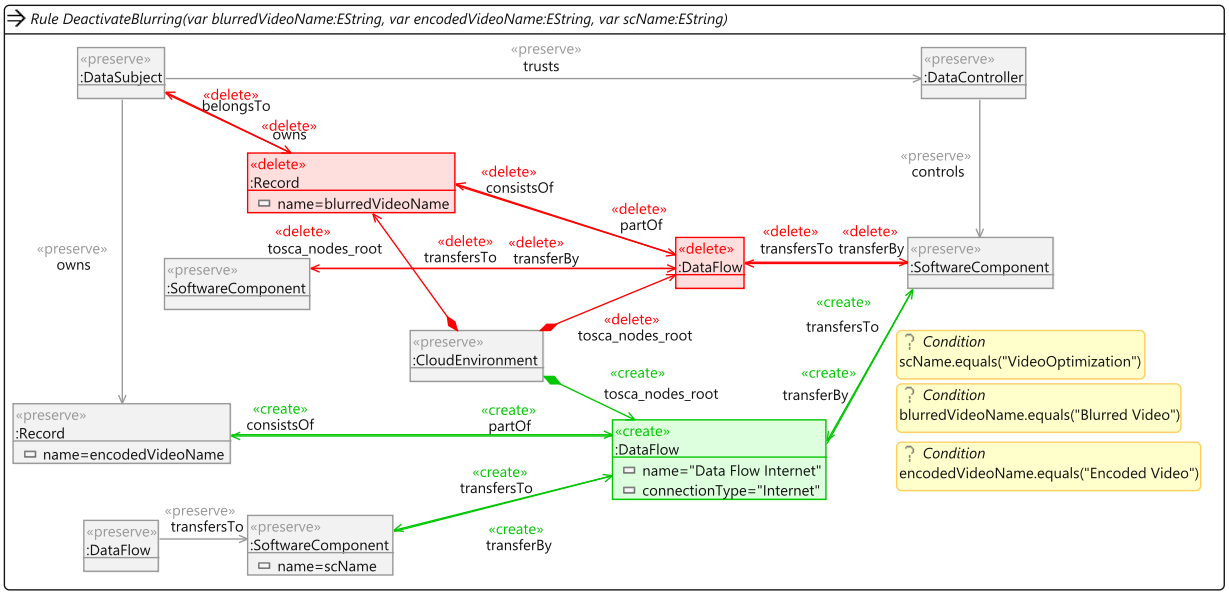

Figure 7.8: Adaptation rule that deactivates video anonymization

sumption, *Node3* represents the best choice. Therefore, both adaptations –activating object blurring and reallocating the video processing– are carried out at once to enable video access to an untrusted actor. This is represented by the dashed line from *Node1* to *Node3* and from *Node3* to the *Monitoring Component* in Fig. 7.5.

When changing the prioritization from low energy consumption first to high performance first, our engine detects a possible improvement adaptation by comparing different system configurations that are created by using the adaptation rule described in Fig. 7.7. Three out of four possible system configuration changes lead to newly arising PCP instances or worse performance. Only the reallocation of the video processing to *Node4* does improve the overall system performance because *Node4* can process the video with the highest amount of FPS combined with a high AI model accuracy. Therefore, this adaptation is carried out. The dashed line from *Node4* to the *Monitoring Component* in Fig. 7.5 shows this scenario.

Should the untrusted actor quit accessing to the monitoring component, object blurring is not longer needed. The engine detects a possible improvement and uses the adaptation rule shown in Fig. 7.8 to change the system configuration accordingly. The respective rule does the opposite of the rule shown in Fig. 7.6, i.e., removing the *data flow* transferring an *anonymized video record* and reactivating a *data flow* transferring an *unanonymized video record*. Depending on the selected prioritization order, the video processing remains at *node4* (highest performance) or an adaptation based on the adaptation rule shown in Fig. 7.7 (lowest energy consumption when no object blurring is needed) is carried out, reallocating the video processing to *Node1*.

In Fig. 7.9 the impact of the selected adaptations is shown. $S1$ represents the initial

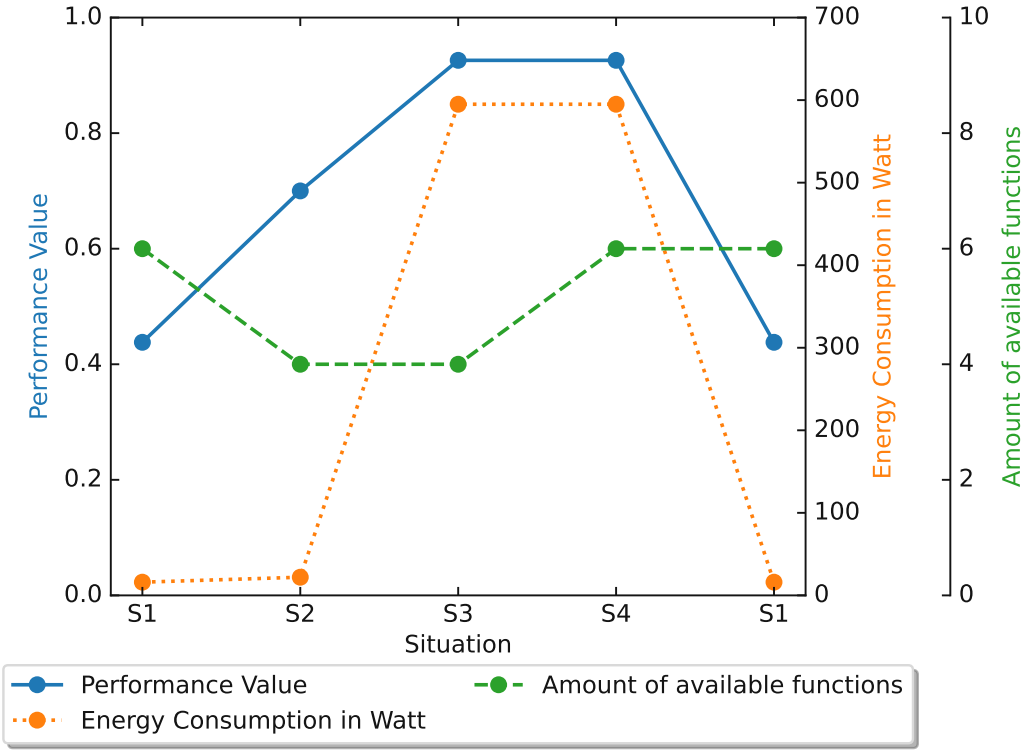

Figure 7.9: Changes in the metric values after the described adaptations

situation where video blurring is deactivated and energy consumption is prioritized. The VAP system consumes a low amount of energy while the performance is sufficient. $S2$ represents the system metric values after video blurring was activated. Due to offloading video processing from *Node1* to *Node3*, an increase in performance and energy consumption can be seen. However, due to the activation of video blurring, the amount of available functionality has decreased, because seeing an unanonymized video was designed as part of a functionality pattern. After changing the prioritization from *low energy consumption first* to *high performance first*, both performance and energy consumption increase substantially. This is due to the migration from *Node3* to *Node4*. *Node4* does support video processing and AI-Task offloading to a dedicated video card, which affects both performance and energy consumption. The system metrics in this situation are represented by $S3$. When video blurring is no longer needed and therefore deactivated, the amount of available functions increases (see $S4$) again. If, in addition, the prioritization is changed back to low energy consumption first, then the original situation $S1$ is adopted due to the migration from *Node4* to *Node1*.

A detailed version of all described scenario steps in the style of a graphical run-time model (similar to an UML object diagram) as well as graphical representations of PCPs

and adaptation rules can be found online[9]. Moreover, the code base and an instruction how to run this example use case are included there.

7.6 Related Work

Broadly speaking, the mitigation of threats to data protection of a system is a two-dimensional problem. First, a mitigation concept comprises either design-time activities or run-time activities or a combination of both. Second, it is typically specific to the system domain and underlying infrastructure. Research dealing with risk analysis, e.g., [PMM18, AHF+15, PHMN16, Lau21], as a design time activity is related to our work, i.e., it would be needed to identify and model PCPs (and how to resolve them) as suggested by our approach. Our approach is agnostic to a specific risk analysis approach, therefore it could be seen as complementary research. However, research that aims to mitigate data protection threats via run-time adaptations is closer related to our approach.

A generic approach was proposed by Bürger et al., which handles *Essential Security Requirements*. This is achieved by triggering *Security Maintenance Rules* if changes in the *Security Context Knowledge* are detected [BSG+18, BGR+15]. Another generic mechanism by Tsigkanos et al. describes a model checking approach to analyze threats and plan adaptations to mitigate those [TPGN16]. They use bigraphs to model the topology and security requirements of the system. Their approach takes a lot of execution time, while our approach builds upon RADAR, which we could show is well performing even with larger system models. Additionally, both of the above mentioned proposed adaptation mechanisms focus solely on the mitigation of data protection threats, but do not take other parameters into account, such as QoS, performance or energy consumption. We identified several other papers related to our work that take such parameters into account. [GGAM+16, GGDM+18, KPIM19] take into account various types of costs, response time and/or performance. However, they mostly focus on specific mitigation strategies and/or approaches dedicated to specific attack vectors. For example, Nostro et al. solely deal with attacks performed by insiders and how to prevent those[NCBB14]. Nguyen et al. proposed a mitigation strategy that only involves migration of virtual machines[NSD18]. Our proposed solution is able to process arbitrarily complex adaptations that are optimized to also provide the desired QoS, performance and energy consumption.

In the context of privacy in video-based media spaces, Boyle et al. [BNG09] proposed a framework – a descriptive theory – that defines how one can think of privacy while analyzing media spaces and their expected or actual use. The framework explains three normative controls: solitude, confidentiality and autonomy, yielding a vocabulary related to the subtle meaning of *privacy*. A more technical introduction to video surveillance is given by Senior in [Sen09]. The paper briefly summarizes the elements in an automatic video surveillance system, including architectures, followed by the steps in video analysis, from pre-processing to object detection, tracking, classification and behavior analysis. In [UAT+19], the authors introduce a video analytics framework to process real-time

[9] See `https://git.uni-due.de/fogprotect/vap-adaptation-engine`

video streams and also do batch video analytics. However, they focus on the performance aspect of a distributed VAP, in terms of scalability, effectiveness and fault-tolerance, leaving data protection out of scope. Sada et al. proposed an edge computing based video analytics architecture [SBM+19]. Their solution leverages federated learning strategies in order to reduce the computational load of cloud infrastructure. Additionally, their training data remains on the edge servers, thereby increasing privacy. In contrast to our proposed solution, their approach incorporates federated learning, while we focus on local execution of AI-based tasks and protecting data agnostic to specific security and privacy mechanisms. Furthermore, we focus on the heterogeneity aspect in edge computing based VAPs, and the adaptation aspect of the system, hence increasing the flexibility and applicability of our approach.

Our proposed solution is specifically tailored to meet the requirements of a self-adaptive data protection aware VAP. Hence, we also identified research in the domain of self-adaptive (distributed) AI-assisted VAPs. As stated in section 7.2, the inference tasks of an AI-assisted VAP is computationally expensive. Hence, as well as in other domains than VAPs, many authors propose offloading those computationally expensive tasks to the cloud. However, executing inference in the cloud, especially for real-time video analysis, often incurs high bandwidth consumption, high latency, reliability issues, and privacy concerns. Therefore, many researchers follow the edge computing paradigm, i.e., processing data closer to the data source. For example Liu et al. proposed EdgeEye, which enables developers to transform models trained with popular deep learning frameworks to deployable components with minimal effort [LQB18]. In [JAB+18], they introduced a controller that dynamically picks the best configurations for existing Neural-Network-based Video Analytics Pipeline. Their aim is to achieve higher accuracy with the same amount of resources, or achieve the same accuracy but utilizing less of the available resources. Zhang et al. propose a flexible serverless-based approach to facilitate fine-grained and adaptive partitioning of cloud-edge workloads for multiple concurrent video query pipelines. Their goal is to achieve real-time responses given a highly dynamic input workload. In contrast to our approach, those papers focus solely on improving one specific performance aspect (either accuracy or latency) of an AI-assisted VAP.

To summarize, to the best of our knowledge there is no approach in literature that allows for data protection focused run-time adaptations in the domain of AI-assisted Video Analytics Pipelines, that also takes performance, QoS and energy consumption into account.

7.7 Summary

Operating a distributed VAP comes with many associated challenges. Adhering to data protection regulations, dealing with changes in computational load, targeting low energy consumption or facing a heterogeneous hardware and software environment are prominent examples of those challenges. In order to provide an adequate QoS and comply to data protection policies, a VAP has to react and adapt to face those challenges. While there

is state of the art literature dealing with either performance or data protection related adaptations, most of them solely focus on increasing certain security or privacy aspects of a system, leaving previously mentioned performance characteristics out of scope or the other way round. To the best of our knowledge, there is no solution that covers data protection, computational performance and energy consumption aspects.

In this chapter, we presented a data protection focused adaptation engine that leverages the application- and infrastructure based adaptation space of a distributed VAP. The engine employs a system model and adaptation rules that are based on previous research. The model was specifically extended and enhanced to meet the requirements of AI-assisted VAPs at the Edge. Furthermore, the engine features an optimization algorithm to improve performance, energy consumption and data protection of a distributed VAP and its functionalities. Using a traffic monitoring use case as running example, we demonstrated how the engine behaves with different configurations and correctly solves problematic configurations during design- and runtime.

In previous work, scalability experiments have shown that the engine is capable of handling models with up to 200 nodes in a given time frame of 10 seconds [MKL+21]. In our considered use case scenario the engine is not limited by the size of the run-time model. Thus, the engine is capable of always finding the best solution in minimal time. To test how long it takes for the engine to determine the best solution in this kind of scenario we measured the time between the change of the run-time model using the monitoring API until the engine found the best solution. The tests were executed on a typical edge node equipped with an Intel Core i5-4690K processor with 3.5GHz clock frequency and with 16 GB DDR3 memory. The node was running the Windows 10 OS and JDK14.0.1 as the Java environment. After hundred rounds of testing, an average evaluation time of round about 1.4 seconds was measured. It should be mentioned that this measured time does not equal the interval between a real world system change causing a PCP instance and the final system change caused by the call to execute the optimal adaptation by our engine. We tested our engine independent from the managed self-adaptive system, as this system could be a limiting factor that we cannot influence.

Thus, our solution is mainly limited by monitoring and adaptation execution capabilities of the self-adaptive system. On the one hand, the possible frequency of monitoring reports as well as the quality and level of detail of the monitoring reports will influence our proposed approach. For example, if details are missing, our engine may not detect a PCP instance or possible adaptation. If the report frequency is too low, it takes longer for a PCP instance to be detected and thus until the self-adaptive systems system configuration is adapted. Furthermore, the adaptation execution capabilities of the self-adaptive system may limit our approach because only supported system adaptations can be taken into account when then engine is searching for adaptations. Moreover, an adaptation selected for execution may not be carried out successfully. In this case, our engine can only try to carry out the proposed adaptation again or try to carry out the next best adaptation.

Our evaluation has shown that the adaptation engine is capable of solving data protection risks in a Video Analysis Pipeline while also finding the optimal system configuration

regarding global energy consumption, system performance and available functionalities. However, our approach is not limited to VAPs. Systems that face similar challenges, such as surveillance applications, can also implement our meta-model, PCP language, and adaptation language. If extensions to the meta-model are needed, this changes do not affect existing run-time models, PCPs, or adaptations. Therefore, our approach is transferable to many edge systems dealing with data protection and optimization problems at runtime. Additionally, our previous research [MKL+21] has shown that data protection risks related to location and jurisdiction restrictions are also covered by our approach. Therefore, the range of different data protection problems this approach can handle is not limited to the content of the data but also handles geospatial or environmental constraints, such as data locality requirements.

CHAPTER 8

Conclusion and Future Work

This chapter concludes the thesis and provides an outlook on future work. Concretely, Section 8.1 summarizes the main scientific contributions made throughout this thesis to the urban sensing domain. Section 8.2 revisits the research questions formulated in the beginning of the thesis and explains how those questions were addressed. Lastly, Section 8.3 discusses the limitations of our findings and provides a detailed outlook how each of the concepts and mechanisms presented in this theses will be further improved or enhanced.

8.1 Summary of Contributions

Urban sensing comprises multiple domains, such as public surveillance, environment monitoring, smart health, crowd sourcing/sensing, smart buildings and many more. Stemming from those domains, challenges to data protection are a major concern, as the various services offered by an urban sensing application often process sensitive data such as personally identifiable information or environmental parameters needed to guarantee the smooth operation of the service. Tackling those challenges becomes inherently important as the majority of urban sensing applications incorporate resource constrained devices operating at the edge of the network. These constraints often hinder system designers and developers to adequately comply to security and privacy regulations, such as the GDPR. The concept of self-adaptation can be seen as a potent enabler to tackle those challenges. In this thesis, we exploit the adaptation space to improve and enhance data protection in resource constrained urban sensing environments.
The scope and constraints of adaptations performed in a system typically stem from policies. Hence, we first present a system model that builds the foundation of an adaptive urban sensing system that has to adhere to some form of data protection regulation. The model focuses on data protection aspects, such as fine grained access control, and supports the definition of privacy policies and how to enact them inside the system.

As part of our privacy model, we define privacy levels to incorporate a finer formal description granularity. Furthermore, it leverages the context of data to make decisions about how these data need to be processed and managed to achieve an adequate level of data protection. We thereby enable flexibility in implementation, and aid system architects or developers in designing and building decentralized urban sensing systems that can make use of privacy-sensitive data, while complying to complex privacy policies of a given domain. Compared to existing approaches, which are mostly tailored to a specific use case or domain [XSSC06, KAB09, CRJS13, Klo17], our model focuses on context-awareness and corresponding actions edge devices have to take. We demonstrate our approach based on a scenario from the Smart Health domain, where privacy is considered a critical requirement.

Second, we need to get a better understanding on the main drivers and limitations of the adaptation space for data protection in urban sensing systems. With respect to the previously mentioned infrastructural characteristics of such systems, those drivers and limitations typically correspond to performance and energy consumption and how those two aspects impact data protection. Therefore, we evaluated several data protection mechanisms by measuring their performance and energy consumption on representative low tier edge devices. The evaluated data protection mechanism include cryptographic block and stream ciphers, secure hashing algorithms, digital signature algorithms and algorithms needed for key exchange protocols. Our measurements provide results that can aid the design and development of secure urban sensing systems, incorporating resource constrained devices, thus facilitating an appropriate design of privacy policies and respective adaptation space. In particular this becomes relevant for adaptive systems, e.g., edge nodes running an risk-assessment engine which needs the end devices (i.e., our resource constrained IoT devices) to be flexible concerning data protection methods. The recently emerged *tinyML* paradigm would be a concrete example that could benefit from such adaptation mechanisms, with its applications in the domains of agriculture, healthcare or smart manufacturing.

Third, based on those measurements, we implemented ORIOT, a Source Location Privacy (SLP) System for resource constrained devices, specifically micro controllers. SLP is a problem, commonly found in the environment monitoring domain, where the origin of sensed data needs to be protected or undisclosed to certain parties. Well established SLP mechanisms, such as TOR, perform poorly or not at all on such MCUs, hence a system in need of SLP has to be adapted to overcome those constraints. Our performance measurements, show the feasibility of integrating ORIOT into existing or planned urban sensing systems. Similar to Tor, it leverages techniques of the onion routing principle. Compared to other approaches, our system does not rely on heavy cryptographic algorithms to provide anonymity. On the other hand, ORIOT avoids network broadcasting strategies as used by different proposed SLP systems. By setting a specific path length for our message transfer, a well balanced trade-off between network load and SLP level is achieved.

ORIOT is applicable to applications that need to ensure source location privacy. A well-known example in literature is the Panda-Hunter Game, where a WSN is deployed

in a forest to monitor pandas. Hunters take the role of an adversary, trying to capture the panda. The goal is to prevent the hunter from locating the source, i.e., the sensor attached to a specific panda. Fourth, we investigated data protection concerns, and how to address them, in the domain of public surveillance. Specifically, we are interested in Video Analysis Pipelines and the challenges to data protection such systems are confronted with, i.e., leakage of PII from recorded and transmitted video data. Arguably, not transferring video data at all provides a major benefit in terms of protecting PII. However, many systems in this domain need to aggregate or process the information contained in the video data at some central location, e.g., in the cloud, in order to provide some added value in the form of services or to enhance an application. Hence, we proposed a privacy preserving system for AI-assisted video analytics that extracts relevant information from video data and governs the secure access to that information. It features a decoupling architecture that effectively hinders applications from directly accessing the underlying video feed, and instead allows them to advertise what type of information they require. Our system then extracts the information using existing AI-based video processing techniques, ensures that privacy preferences are met, and facilitates the secure access to the extracted information for both real-time and batch applications. A ciphertext (i.e., the encrypted information extracted from video data) is labeled with certain attributes, which only allows applications with a matching private key (i.e., the attributes corresponding to the labels of the ciphertext are encoded in the key) to decrypt and access the data. A KP-ABE security scheme ensures that only authorized parties have access to this extracted information. To allow for a more fine grained access control, security policies determine which application is able to decrypt specific subsets of the encrypted extracted data. The policies are stored and managed at a dedicated policy database, located at the edge or in the cloud. Furthermore, it is responsible for issuing keys to an application, as well as notifying applications if a key's attributes change. Hence, a seamless fine-grained management of access control for any application without the need of a re-deployment can be achieved by adapting those policies at run-time. We demonstrate the feasibility of our approach by evaluating a face detection application, deployed on typical edge computing infrastructure. Such an application may be part of any public surveillance system in a e.g., smart city. A concrete example would be traffic monitoring where several parties may not necessarily need to store (raw or compressed) video feed, neither must they have access at any time to (live) video data containing PII. However, there are applications that still rely on transmitting video data where our system may not be feasible. Hence, we further explored the adaptation space of AI-assisted data protection for a VAP operating at the edge. We identified factors that can be adapted in AI-assisted data protection for video analytics using the example of a face blurring pipeline. We measured the impact of these factors using a heterogeneous edge computing hardware testbed. The results showed a large and complex adaptation space, with varied impacts on data protection, performance, and accuracy. Our work thus paves the way towards effective adaptation algorithms for AI-assisted privacy-preserving video analytics at the edge.
Lastly, to actually exploit the application- and infrastructure based adaptation space of

AI-assisted data protection in AI-assisted video analytics, we developed a data protection focused adaptation engine for distributed VAPs. It employs an extended system model and adaptation rules to meet the requirements of AI-assisted VAPs at the Edge. Furthermore, it features an optimization algorithm to improve performance, energy consumption and data protection of a distributed VAP and its functionalities.
Additionally, it opts to find the most suitable system configuration based on an operators preferences regarding performance, energy consumption, and available functionalities (QoS). Using a traffic monitoring use case as running example, we demonstrated how the engine behaves with different configurations and correctly solves problematic configurations during design- and runtime.

8.2 Research Questions Revisited

In the beginning of this thesis, the following research questions were formulated:

(**RQ**$_1$) *How can edge computing based data protection policies be formulated and processed in order to build the foundation for an adaptive urban sensing application?*

In chapter 2 we addressed this question by proposing a novel model that supports the definition and enactment of privacy policies based on context-aware edge computing. Defining a consistent model enables a coherent definition of policies that ensure privacy of data, also handling data locality. Our privacy model combines and correlates certain levels of privacy (e.g., visibility constraints on specific data sets) with a given context. The determination of a specific context is best handled closest to the according environment. Therefore, edge computing is well suited for this task, where every edge device is exposed to a certain and specific context. To achieve a higher degree of flexibility in implementing privacy policies we propose a context-aware decision making process to dynamically adapt to changing environment situations at the edge. Context in computer science can be interpreted in many different ways. In the focus of this chapter, we use the term *context* as environmental information recognizable by edge devices. This could be information about the network and its topology, connected devices, spatial information, proximity, location or time. We argue that a context-aware system is then able to adequately interpret changes in the environment and react to them in a predefined manner.

(**RQ**$_2$) *What parameters need to be evaluated and considered for data protection focused adaptation strategies and how can those be applied in low tier resource constrained urban sensing environments?*

We addressed the first part of this question in chapter 3. Regarding the second part of the question, we developed an adapted SLP system, based on the previously gained insights, in chapter 4. Urban sensing systems continue to permeate deeper into our personal lives. IoT devices sense, process, and store all kinds of data which poses various challenges to security and privacy aspects, especially to applications running on resource constrained devices. Hence, we evaluated selected, well established data protection mechanisms that enable confidentiality, integrity and authenticity of data. Specifically, we looked into the performance and energy consumption of different cryptographic block and stream ciphers, secure hashing algorithms, digital signature mechanisms, and key exchange protocols executed on state-of-the-art resource constrained devices. Our results ease the calculation of thresholds incorporating performance aspects, data protection criteria and respective energy consumption values, thus facilitating the design and development of secure self-adaptive systems in low tier urban sensing environments.
We applied those evaluation results by developing an adapted source location privacy (SLP) system, namely ORIOT. ORIOT is a SLP preserving system that leverages techniques from the well established Onion Routing paradigm. It is specifically

designed for low tier devices, i.e., devices lacking computing power. Symmetric and asymmetric cryptography are combined with a path assembling strategy to realize anonymity of a node transmitting messages in a network to a specific destination.

(**RQ**$_3$) *What are the implications on the adaptation space regarding AI-assisted data protection mechanisms in mid tier urban sensing applications?*

In this thesis we focused on use applications running in the public surveillance domain. Specifically, we looked into Video Analysis Pipelines and investigated data protection challenges such systems are confronted with. However, our results are applicable to other domains incorporating AI-assisted data protection mechanisms as well. In chapter 6 we presented preliminary results of the first systematic study of the adaptation space of AI-assisted data protection for video analytics at the edge. Using a face-anonymization pipeline as running example, we identified several possible adaptations and analyzed their impact on performance and data protection. We also implemented and empirically evaluated several of the considered adaptations. The results show a wealth of different adaptation options on both the infrastructure and application level. The results also show that the impact of these adaptations varies significantly. We categorized these results into either application based adaptations or infrastructure based adaptations. Such results are needed to have a solid basis for designing the adaptation logic for AI-assisted data protection for video analytics at the edge. Our work is a first step in this direction and the results are potentially applicable to a wide field of other potential use cases.

(**RQ**$_4$) *How can we exploit the adaptation space in an edge based urban sensing application?*

In chapter 5, we proposed a privacy preserving system for AI-assisted video analytics that extracts relevant information from video data and governs the secure access to that information. The system ensures that applications, leveraging extracted data, have no access to the video stream. An attribute-based authorization scheme allows applications to only query a predefined subset of extracted data. Security policies determine which application is able to decrypt specific subsets of the encrypted extracted data. Hence, run-time adaptations made to those policies allow for a dynamic and fine grained access control to sensitive data.
In chapter 7, we present a data protection focused adaptation engine for AI-assisted urban sensing applications. The engine features an optimization algorithm to improve data protection, performance and energy consumption characteristics of the system. Specifically, it leverages the application- and infrastructure based adaptation space of a distributed VAP, but it is easily extendable, hence it can be also implemented for many other urban sensing applications. Based on a real-world traffic monitoring use case, we showed that our approach efficiently and effectively mitigates data protection risks in a running system.

8.3 Future Work

In this theses, we investigated how adaptation strategies could enhance urban sensing applications to mitigate risks to data protection while still maintaining adequate performance. In the context of urban sensing, this is necessary because many edge devices executing those applications are limited in their capabilities, such as computing performance, energy consumption or storage. Obviously, within the scope of this thesis, we cannot address all of the problems associated with data protection and constrained resources at the edge in an urban sensing environment. Hence, there are still several open challenges that we intend to address in future work. In this section, we outline the identified shortcomings and discuss possibilities for future research.

8.3.1 Privacy Policy Model

Part of our future work will be further investigation in the field of context-aware privacy enforcement and prototypical implementations of policy enforcing techniques for different kinds of edge devices. Currently, our proposed model focuses mostly on data access controls. Hence, we plan to enhance our model by incorporating other data protection related controls as well. Specifically, our intent is to integrate the findings presented in chapter 3 into the model. Furthermore, we plan to incorporate the updated model into a adaptation strategy for resource constrained devices, we have already developed in [Kai21]. Incorporating AI-based mechanisms into the decision process is another future action point that will be investigated.

8.3.2 Adaptation Strategies

There are still data protection mechanisms that we have not evaluated for their usability in resource constrained urban sensing environments in terms of performance and energy consumption. Future work will include further measurements on performance and energy consumption of asymmetric encryption algorithms, like RSA or El-Gamal. This results will also be incorporated into the development of ORIOT. Additionally, regarding ORIOT, we will include packet padding and noise generation to mitigate attacks like timing/traffic analysis. We will also revisit and improve the strategy to balance performance, energy consumption and data protection presented in [Kai21]. Currently, this balancing strategy relies on adaptations based on pre-calculated threshold values for energy consumption and performance and pre-defined so called security stages. Those stages represent different configurations for data protection algorithms, which in turn are based on the STRIDE risk model from Microsoft. Currently these configurations reside on a central controlling device (triggering the adaptations on a device) in the form of simple text files. First, we want to investigate the potential of distributing (parts of) the adaptation logic to the end-devices. Second, as described in section 8.3.1 we want to get rid of the text file representation of adaptation policies, and instead employ our developed privacy model. Third, simulation experiments are planned in order to make concrete statements on

scaling aspects and how much potential for e.g., energy savings there may be in large scale system such as smart grids or similar.

8.3.3 AI-Assisted Video Analytics Pipelines

In the future, we intend to extend our research to further types of edge AI analytics applications and devices in general, as well as with further relevant metrics. In addition, we plan to transform the results to performance models that can be used to automatically reason about possible adaptations at run time. Those models will feed into the adaptation engine, presented in chapter 7. Additionally, regarding our proposed adaptation engine, future work will focus on applying the adaptation engine to other urban sensing domains. This makes perfect sense, since our approach is extendable by adjusting the meta-model and by adding further PCPs and corresponding adaptations. However, based on the respective domain, additional metrics, such as business metrics, will be taken into account. Furthermore, due to the nature of edge systems, decentralizing our approach and deploying it multiple times in the edge network of a self-adaptive system may enable better performance and faster responses to PCPs. Therefore, further research is needed to find a suitable way of coordinating multiple adaptation engines. Regarding our proposed privacy preserving system for AI-assisted video analytics, we plan to encapsulate the system into a framework, where a user is able to simply plug in a desired machine learning model and specify the information that will be extracted alongside with the corresponding labels via an easy-to-use markup language. Additionally, we plan to look into *coral.ai*, which offers a complete toolkit to build products with local AI on-device inference capabilities [Res20]. Furthermore, we plan to evaluate this framework on additional edge computing hardware varying in performance related capabilities like CPU, RAM, HDD, etc., also incorporating and implementing more complex scenarios. We plan to do simulation experiments in order to evaluate the feasibility and practicability of our approach in large scale systems. Furthermore, the approach is also planned to be integrated into our adaptation engine, e.g., by supporting run-time adaptations to the KP-ABE related key policies.

APPENDIX A

Performance and Energy Consumption Graphs of Data Protection Mechanisms for Resource Constrained IoT Devices

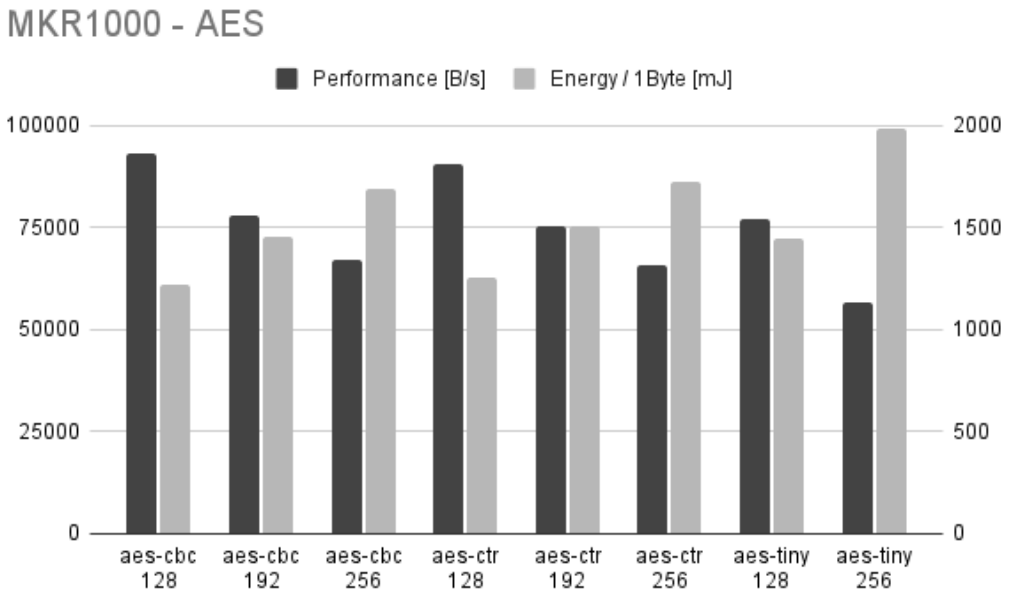

Figure A.1: MKR1000: Performance and Energy Consumption of different AES configurations

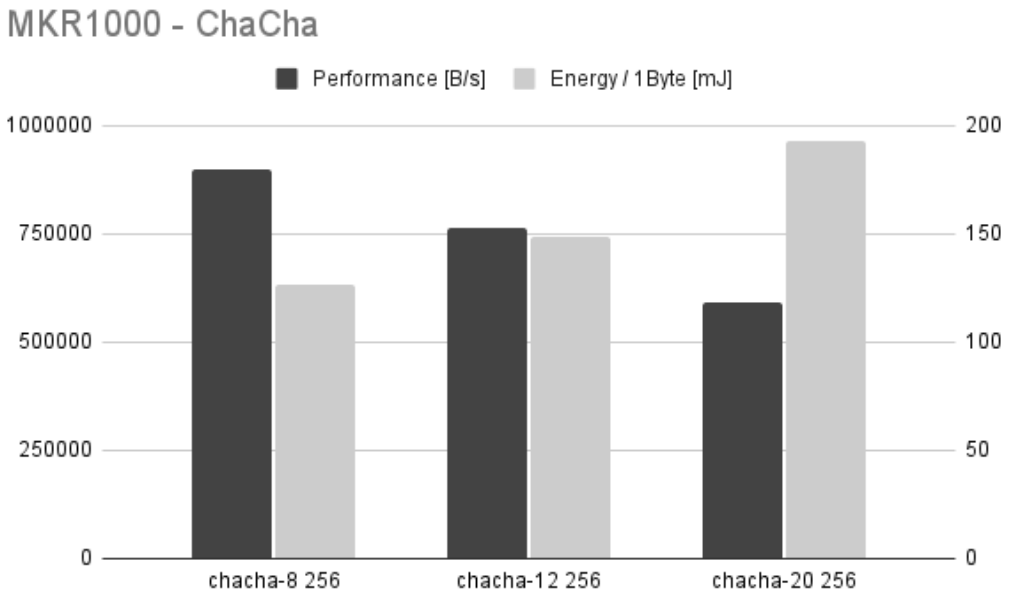

Figure A.2: MKR1000: Performance and Energy Consumption of different ChaCha configurations

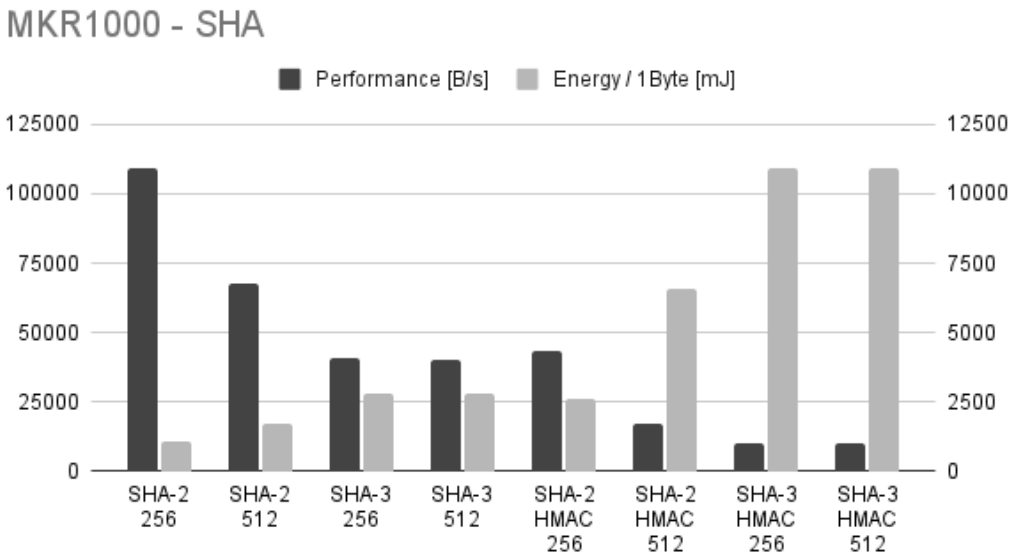

Figure A.3: MKR1000: Performance and Energy Consumption of different elliptic curve operations

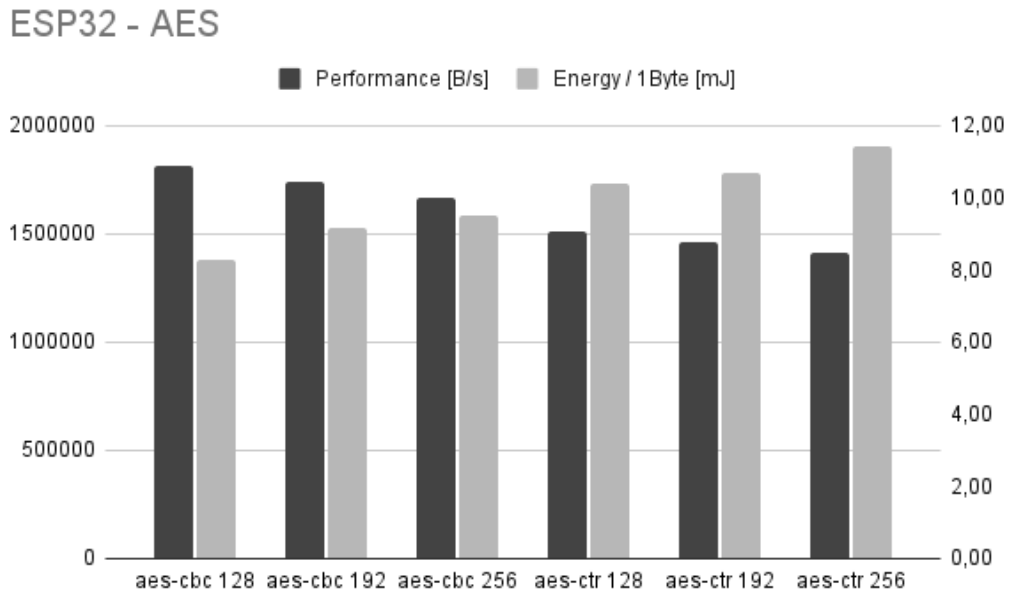

Figure A.4: ESP32: Performance and Energy Consumption of different AES configurations

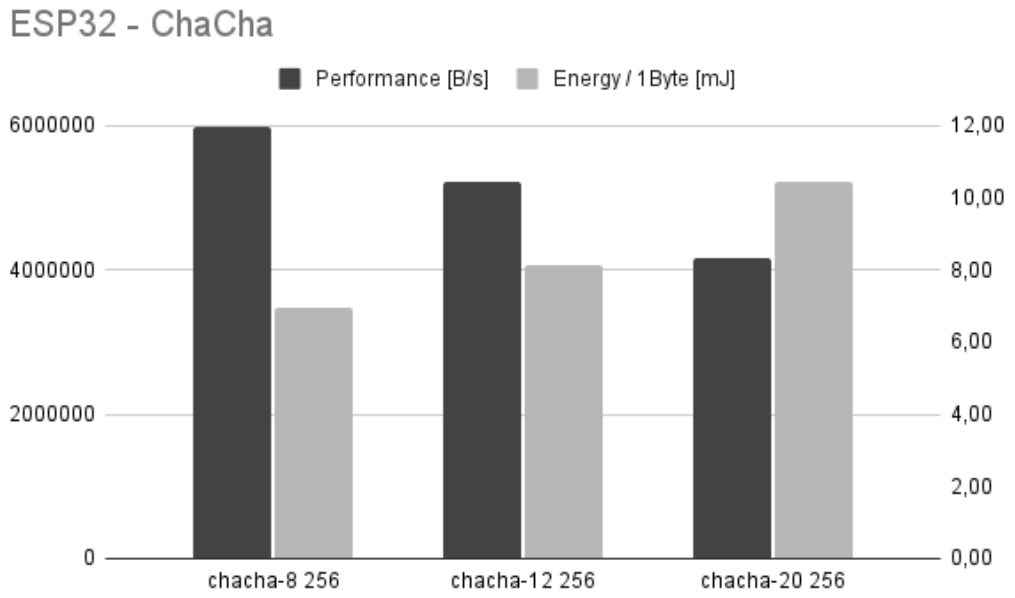

Figure A.5: ESP32: Performance and Energy Consumption of different ChaCha configurations

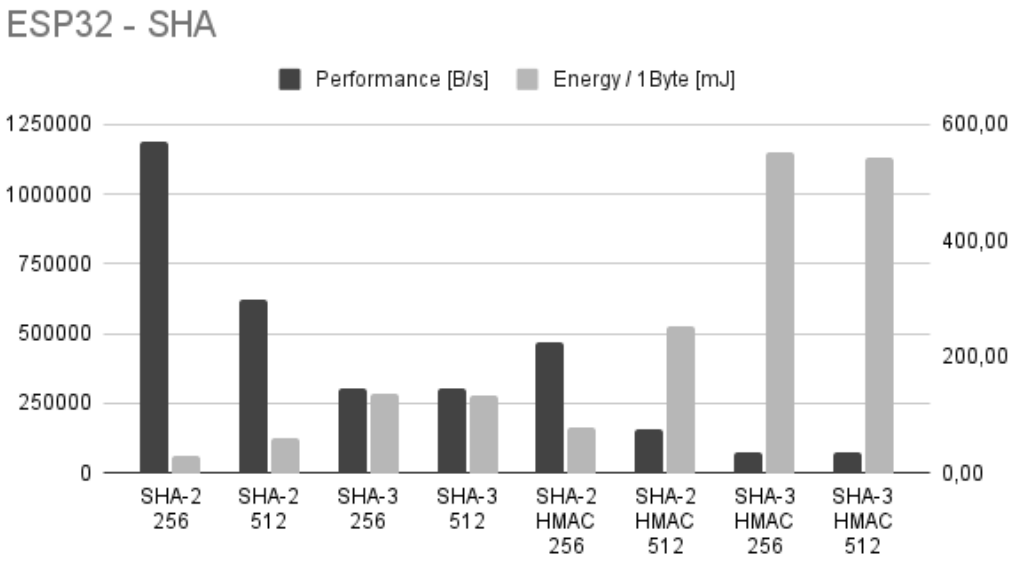

Figure A.6: ESP32: Performance and Energy Consumption of different elliptic curve operations

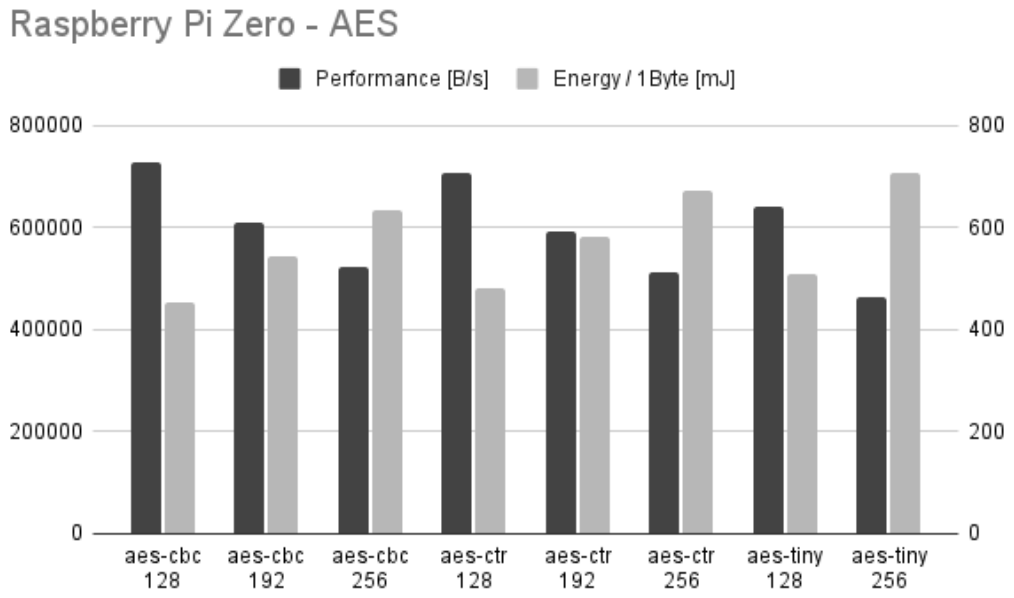

Figure A.7: Raspberry Pi Zero: Performance and Energy Consumption of different AES configurations

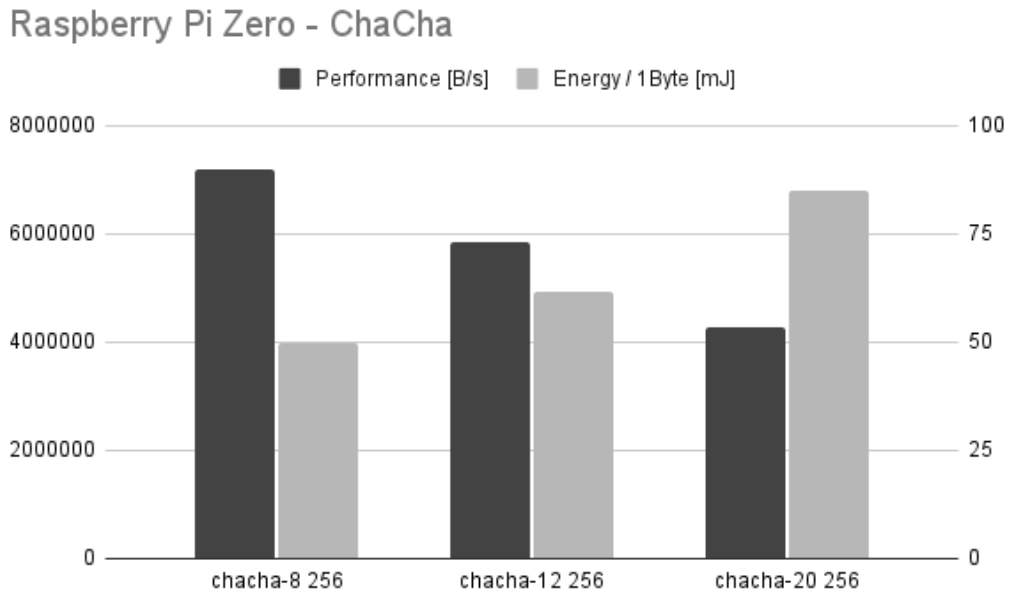

Figure A.8: Raspberry Pi Zero: Performance and Energy Consumption of different ChaCha configurations

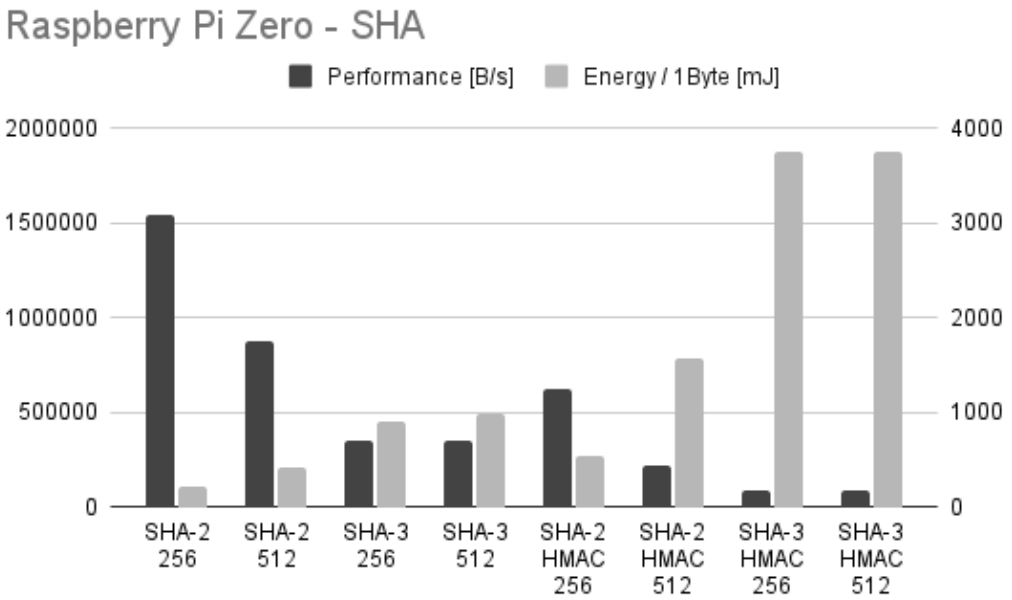

Figure A.9: Raspberry Pi Zero: Performance and Energy Consumption of different elliptic curve operations

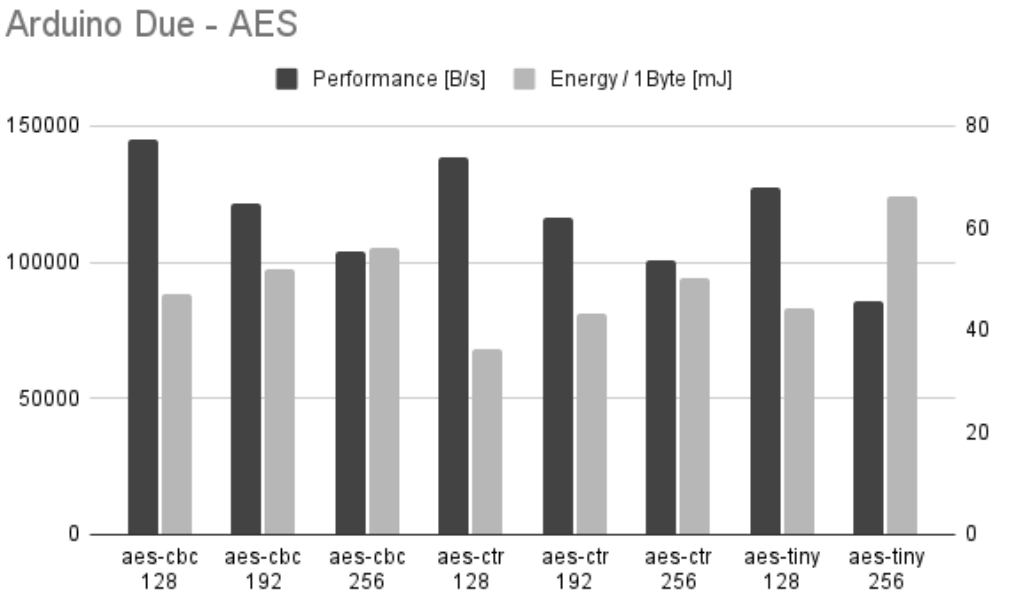

Figure A.10: Arduino Due: Performance and Energy Consumption of different AES configurations

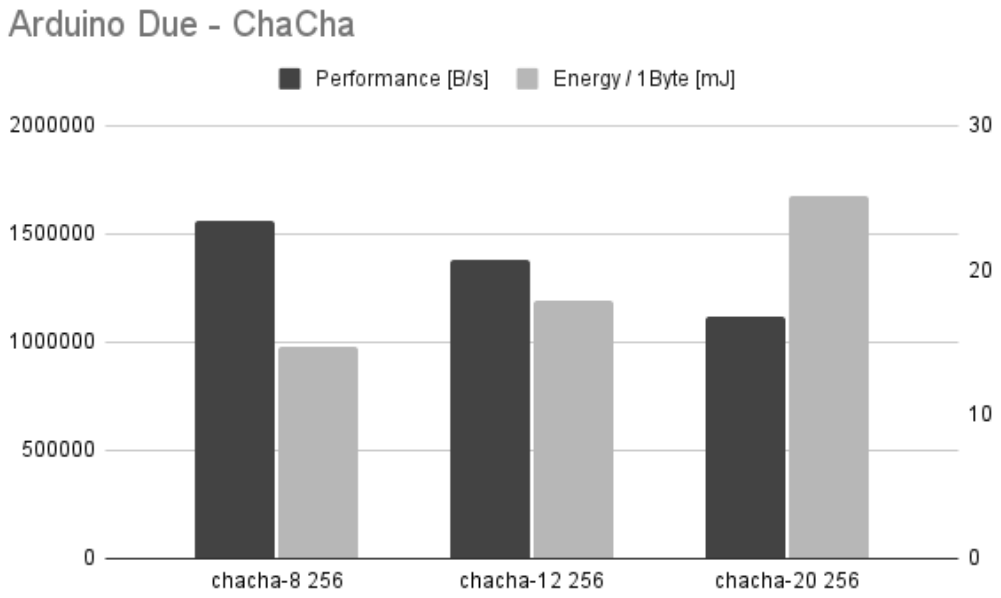

Figure A.11: Arduino Due: Performance and Energy Consumption of different ChaCha configurations

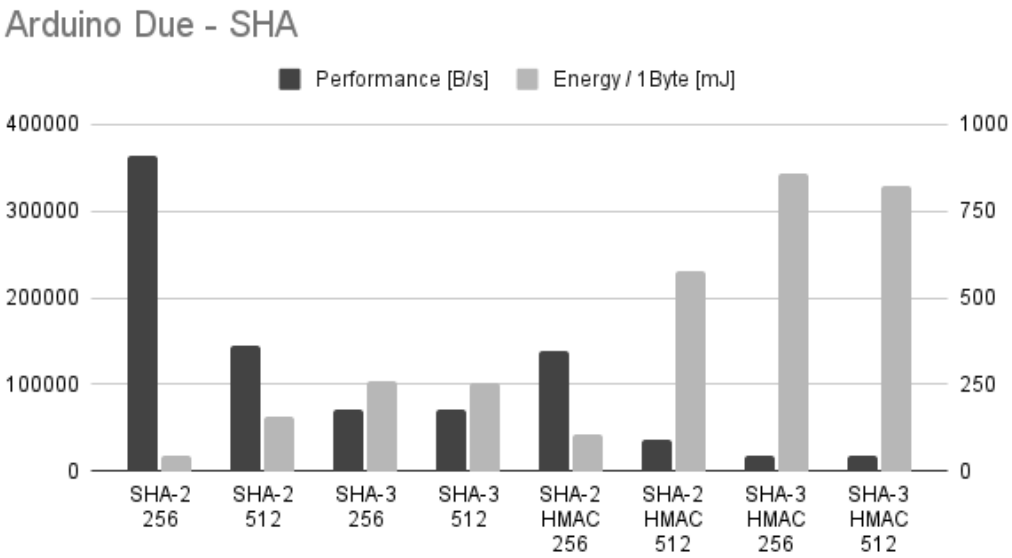

Figure A.12: Arduino Due: Performance and Energy Consumption of different elliptic curve operations

List of Figures

List of Tables

List of Algorithms

Acronyms

ABAC Attribute-Based Access Control. 63

AES Advanced Encryption Standard. 29, 30, 32–35, 37–39, 54, 60, 66

AI Artificial Intelligence. xi, xii, 7, 8, 10–12, 57–63, 65–67, 69–77, 79–81, 84–88, 93, 95, 97, 98, 101, 102, 105, 106, 108, 109, 113, 114, 116–118, 127

API Application Programming Interface. 63, 65, 75, 97, 99, 101, 109

AR Anonymous Routing. 47

ASIC Application-Specific Integrated Circuit. 88

AVX Advanced Vector Extensions. 76

CBC Cipher Block Chaining. 32–35, 37

CO-OP Context Operations. 18–21

COVID-19 Coronavirus Disease 2019. 59, 61, 64

CRUD Create, Read, Update and Delete. 19, 20

CTR Counter. 32–35, 54

DHKE Diffie-Hellman Key Exchange. 31

DSP Digital Signal Processor. 60, 80

DTSL Datagram Transport Layer Security. 50, 54

ECA Event-Condition-Action. 84

ECC Elliptic Curve Cryptography. 30–32, 41, 54

ECDH Elliptic-Curve Diffie-Hellman. 31, 43, 51, 53, 54

FMA Fused Multiply-Accumulate. 76

FPGA Field Programmable Gate Array. 28

FPS Frames per Second. 65, 74, 89, 95–98, 104, 105

GDPR General Data Protection Regulation. 3, 4, 8, 14, 70, 73, 83, 84, 88–91, 93, 111

GPS Global Positioning System. 19, 22

GUI Graphical user Interface. 22, 75, 92

HIPPA Health Insurance Portability and Accountability Act. 3, 6, 14

HMAC Keyed-Hash Message Authentication Code. 29, 31, 35

IaaS Infrastructure as a Service. 92

IBE Identity Based Encryption. 46

IoT Internet of Things. 3, 11, 14, 23, 25–30, 32, 41, 43–47, 53–55, 65, 84, 85, 94, 104, 112, 115, 129

JSON JavaScript Object Notation. 32

KI Künstliche Intelligenz. x

KP-ABE Key-Policy Attribute Based Encryption. 59, 63, 65–67, 113, 118

M2M Machine to Machine. 40

MAC Message Authentication Code. 31

MAPE Monitoring-Analysis-Planning-Execution. 84

MCU Microcontroller Unit. 26, 29, 30, 33, 35, 38–40, 112

mEHR Mobile Electronic Health Record. 14, 15, 20–22

ML Machine Learning. 7, 8, 70, 71

NPU Neural Processing Unit. 88

PAM Policy Application Manager. 17, 18, 20, 21

PAP Policy Administration Point. 17

PBC Pseudonym Based Cryptography. 46, 47

PCP Problematic Configuration Pattern. 91–101, 103–107, 109, 110, 118, 127

PDP Policy Decision Point. 17, 18

PEP Policy Enforcement Point. 17, 18

PII Personally Identifiable Information. xii, 7, 84, 85, 87, 113, 127

PIP Policy Information Point. 17

PKG Private Key Generator. 46

QoS Quality of Service. 85, 89, 92, 95–98, 101, 104, 107, 108, 114

RBAC Role Based Access Control. 7, 14, 15, 21, 23, 58, 67

REST Representational State Transfer. 65

RFID Radio-Frequency Identification. 15, 22

RSU Road-Side Unit. 71

SDK Software Development Kit. 88

SHA Secure Hash Algorithm. 29–32, 35–37, 39

SLP Source Location Privacy. x, xi, 7, 43–48, 112, 115

SN Sink Node. 47

SPR Single Path Routing. 44

SSE Streaming SIMD Extensions. 76

SSL Secure Sockets Layer. 6, 26

TCP Transmission Control Protocol. 50, 54

TPU Tensor Processing Unit. 76, 80, 88, 95

TSL Transport Layer Security. 6, 7, 26, 49, 50, 54, 65

UDP User Datagram Protocol. 50, 54

UML Unified Modeling Language. 92, 93, 106

VAP Video Analysis Pipeline. x–xii, 8, 11, 12, 58, 62, 67, 71–73, 75, 77, 79–81, 83–89, 93, 95, 101, 106, 108–110, 113, 114, 116

WSN Wireless Sensor Network. 3, 26, 27, 46, 47, 49, 112

XACML eXtensible Access Control Markup Language. 15

Bibliography

[ABC+07] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 598–609. Acm, 2007.

[Agr20] Shashank Agrawal. Attribute-based encryption, 2020. `https://github.com/sagrawal87/ABE`.

[AHF+15] Azadeh Alebrahim, Denis Hatebur, Stephan Fassbender, Ludger Goeke, and Isabelle Côté. A pattern-based and tool-supported risk analysis method compliant to iso 27001 for cloud systems. *International Journal of Secure Software Engineering (IJSSE)*, 6(1):24–46, 2015.

[AHWR18] Sultan Alharby, Nick Harris, Alex Weddell, and Jeff Reeve. The security trade-offs in resource constrained nodes for iot application. *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, 12(1):52–59, 2018.

[AJL+21] Dhouha Ayed, Eva Jaho, Clemens Lachner, Zoltán Ádám Mann, Robert Seidl, and Mike Surridge. FogProtect: Protecting sensitive data in the computing continuum. In *Advances in Service-Oriented and Cloud Computing: International Workshops of ESOCC 2020, Revised Selected Papers*, pages 179–184, 2021.

[AMD12] AMD. New "bulldozer" and "piledriver" instructions, 2012. `http://developer.amd.com/wordpress/media/2012/10/New-Bulldozer-and-Piledriver-Instructions.pdf`.

[AS11] Moncef Amara and Amar Siad. Elliptic curve cryptography and its applications. In *International Workshop on Systems, Signal Processing and their Applications, WOSSPA*, pages 247–250. IEEE, 2011.

[AS20] Hossein Amiri and Asadollah Shahbahrami. Simd programming using intel vector extensions. *Journal of Parallel and Distributed Computing*, 135:83–100, 2020.

[AV16] U Arjun and S Vinay. A short review on data security and privacy issues in cloud computing. In *Current Trends in Advanced Computing (ICCTAC), IEEE International Conference on*, pages 1–5. IEEE, 2016.

[B+12] Carsten Bormann et al. Guidance for light-weight implementations of the internet protocol suite. *Lightweight Implementation Guidelines (LWIG) Working Group, Internet Engineering Task Force (IETF), work in progress, draft-ietf-lwig-guidance-02*, 2012.

[Ber09] Daniel J Bernstein. Introduction to post-quantum cryptography. In *Post-quantum cryptography*, pages 1–14. Springer, 2009.

[BFGL20] Antonio Brogi, Stefano Forti, Carlos Guerrero, and Isaac Lera. How to place your apps in the fog: State of the art and open challenges. *Software: Practice and Experience*, 50(5):719–740, 2020.

[BFI19] Antonio Brogi, Stefano Forti, and Ahmad Ibrahim. Predictive analysis to support fog application deployment. *Fog and edge computing: principles and paradigms*, pages 191–222, 2019.

[BGR+15] Jens Bürger, Stefan Gärtner, Thomas Ruhroth, Johannes Zweihoff, Jan Jürjens, and Kurt Schneider. Restoring security of long-living systems by co-evolution. In *2015 IEEE 39th Annual Computer Software and Applications Conference*, volume 2, pages 153–158. IEEE, 2015.

[BHW17] William J Buchanan, Scott Helme, and Alan Woodward. Analysis of the adoption of security headers in http. *IET Information Security*, 12(2):118–126, 2017.

[BNG09] Michael Boyle, Carman Neustaedter, and Saul Greenberg. Privacy factors in video-based media spaces. In *Media Space 20+ Years of Mediated Life*, pages 97–122. Springer, 2009.

[BSEB19] Martin Breitbach, Dominik Schäfer, Janick Edinger, and Christian Becker. Context-aware data and task placement in edge computing environments. In *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. IEEE, 2019.

[BSG+18] Jens Bürger, Daniel Strüber, Stefan Gärtner, Thomas Ruhroth, Jan Jürjens, and Kurt Schneider. A framework for semi-automated co-evolution of security knowledge and system models. *Journal of Systems and Software*, 139:142–160, 2018.

[BWJ+19] Utsav Banerjee, Andrew Wright, Chiraag Juvekar, Madeleine Waller, Anantha P Chandrakasan, et al. An energy-efficient reconfigurable dtls cryptographic engine for securing internet-of-things applications. *IEEE Journal of Solid-State Circuits*, 54(8):2339–2352, 2019.

[BZR+05] Claus Bahlmann, Ying Zhu, Visvanathan Ramesh, Martin Pellkofer, and Thorsten Koehler. A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pages 255–260. IEEE, 2005.

[CB07] Ankur Chattopadhyay and Terrance E Boult. PrivacyCam: A privacy preserving camera using uCLinux on the Blackfin DSP. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[CBSG17] Charles E Catlett, Peter H Beckman, Rajesh Sankaran, and Kate Kusiak Galvin. Array of things: A scientific research instrument in the public way: Platform design and early lessons learned. In *Proceedings of the 2nd International Workshop on Science of Smart City Operations and Platforms Engineering*, pages 26–33. ACM, 2017.

[Cea11] Jianneng Cao and et al. Castle: Continuously anonymizing data streams. *IEEE Transactions on Dependable and Secure Computing*, 8.3:337–352, 2011.

[CFV13] Simone Cirani, Gianluigi Ferrari, and Luca Veltri. Enforcing security mechanisms in the ip-based internet of things: An algorithmic overview. *Algorithms*, 6(2):197–226, 2013.

[Coh06] Simon Cohn. Privacy and confidentiality in the nationwide health information network. National Committee on Vital and Health Statistic `http://www.ncvhs.hhs.gov/060622lt.htm`, 2006. (Accessed 2018-03-26).

[CRJS13] Supriyo Chakraborty, Kasturi Rangan Raghavan, Matthew P Johnson, and Mani B Srivastava. A framework for context-aware privacy of sensor data on mobile systems. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, page 11. ACM, 2013.

[CZVZ14] Angelo Cenedese, Andrea Zanella, Lorenzo Vangelista, and Michele Zorzi. Padova smart city: An urban Internet of Things experimentation. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6. IEEE, 2014.

[DA04] Fadi Dornaika and Jörgen Ahlberg. Fast and reliable active appearance model search for 3-D face tracking. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(4):1838–1853, 2004.

[Dea07] A Dearie. Software davelopment past present and future. In *Proc. of IEEE Conference on Future of Software Engineering*, pages 269–284, 2007.

[Deb20] Chandrika Deb. Face mask detection, 2020. https://github.com/chandrikadeb7/Face-Mask-Detection.

[Den14] Murat Dener. Security analysis in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 10(10):303501, 2014.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22.6:644–654, 1976.

[DJP11] Naipeng Dong, Hugo Jonker, and Jun Pang. Challenges in ehealth: From enabling to enforcing privacy. In *International Symposium on Foundations of Health Informatics Engineering and Systems*, pages 195–206. Springer, 2011.

[DWK15a] Joerg Daubert, Alexander Wiesmaier, and Panayotis Kikiras. A view on privacy & trust in iot. In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 2665–2670. IEEE, 2015.

[DWK15b] Joerg Daubert, Alexander Wiesmaier, and Panayotis Kikiras. *A view on privacy & trust in IoT*. Communication Workshop (ICCW)IEEE International Conference on. IEEE, 2015.

[EFG+09] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. Privacy-preserving face recognition. In *International symposium on privacy enhancing technologies symposium*, pages 235–253. Springer, 2009.

[EW17] Levent Ertaul and Arnold Woodall. Iot security: Performance evaluation of grain, mickey, and trivium-lightweight stream ciphers. In *Proceedings of the International Conference on Security and Management (SAM)*, pages 32–38. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2017.

[FBVC01] R Feraund, Olivier J Bernier, J-E Viallet, and Michel Collobert. A fast and accurate face detector based on neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):42–53, 2001.

[FCK95] David Ferraiolo, Janet Cugini, and D Richard Kuhn. Role-based access control (rbac): Features and motivations. In *Proceedings of 11th annual computer security application conference*, pages 241–48, 1995.

[Fea15a] Muhammad Umar Farooq and et al. *A critical analysis on the security concerns of internet of things (IoT)*. International Journal of Computer Applications 111.7, 2015.

[Fea15b] Sebastian Funke and et al. *End-2-End privacy architecture for IoT*. Communications and Network Security (CNS)IEEE Conference on. IEEE, 2015.

[FWKM15] Muhammad Umar Farooq, Muhammad Waseem, Anjum Khairi, and Sadia Mazhar. A critical analysis on the security concerns of internet of things (iot). *International Journal of Computer Applications*, 111(7), 2015.

[Gen16] General Data Protection Regulation. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC. *Official Journal of the European Union*, page L119, 2016.

[GFEI+20] Fabian Gand, Ilenia Fronza, Nabil El Ioini, Hamid R Barzegar, Shelernaz Azimi, and Claus Pahl. A fuzzy controller for self-adaptive lightweight edge container orchestration. In *10th International Conference on Cloud Computing and Services Science (CLOSER)*, pages 79–90, 2020.

[GGAM+16] Gustavo Gonzalez-Granadillo, Ender Alvarez, Alexander Motzek, Matteo Merialdo, Joaquin Garcia-Alfaro, and Hervé Debar. Towards an automated and dynamic risk management response system. In *Nordic Conference on Secure IT Systems*, pages 37–53. Springer, 2016.

[GGDM+18] Gustavo Gonzalez-Granadillo, Samuel Dubus, Alexander Motzek, Joaquin Garcia-Alfaro, Ender Alvarez, Matteo Merialdo, Serge Papillon, and Hervé Debar. Dynamic risk management response system to handle cyber threats. *Future Generation Computer Systems*, 83:535–552, 2018.

[GHB18] Martin Grambow, Jonathan Hasenburg, and David Bermbach. Public video surveillance: Using the fog to increase privacy. In *Proceedings of the 5th Workshop on Middleware and Applications for the Internet of Things*, pages 11–14, 2018.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98, 2006.

[HASW17] Asma Haroon, Sana Akram, Munam Ali Shah, and Abdul Wahid. E-lithe: A lightweight secure dtls for iot. In *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, pages 1–5. IEEE, 2017.

[HBB+07] Arun Hampapur, Sergio Borger, Lisa Brown, C Carlson, Jonathan Connell, Max Lu, Andrew Senior, V Reddy, C Shu, and Ying-Li Tian. S3: The IBM smart surveillance system: From transactional systems to observational systems. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, pages IV–1385. IEEE, 2007.

[HM11] Darrel Hankerson and Alfred Menezes. *Elliptic curve cryptography*. Springer, 2011.

[HM20] Mohamed K Hussein and Mohamed H Mousa. Efficient task offloading for iot-based applications in fog computing using ant colony optimization. *IEEE Access*, 8:37191–37201, 2020.

[HP15] Nguyen Phong Hoang and Davar Pishva. *A TOR-based anonymous communication approach to secure smart home appliances.* Advanced Communication Technology (ICACT)17th International Conference on. IEEE, 2015.

[HRS+17] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.

[Int16] Intel. Intel 64 and ia-32 architectures software developer's manual, 2016. `https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-1-manual.pdf`.

[Int18] International Organization for Standardization. Information technology – Security techniques – Information security management systems – Overview and vocabulary(2700:2018). Standard, International Organization for Standardization, February 2018.

[JAB15] Sarra Jebri, Mohamed Abid, and Ammar Bouallegue. *An efficient scheme for anonymous communication in IoT.* Information Assurance and Security (IAS)11th International Conference on. IEEE, 2015.

[JAB+18] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 253–266, 2018.

[JKJ07] Ari Juels and Burton S Kaliski Jr. Pors: Proofs of retrievability for large files. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 584–597. Acm, 2007.

[KAB09] David Kotz, Sasikanth Avancha, and Amit Baxi. A privacy framework for mobile health and home-care systems. In *Proceedings of the first ACM workshop on Security and privacy in medical and home-care systems*, pages 1–12. ACM, 2009.

[Kai21] Sebastian Kaindl. Balancing performance, energy consumption and security in resource constrained environments. Master's thesis, TU-Wien, 2021.

[KC03] Jeffrey O Kephart and David M Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[Kea05] Pandurang Kamat and et al. *Enhancing Source-Location Privacy in Sensor Network Routing*. The 25th IEEE International Conference on Distributed Computing System599-608, 2005.

[Kim20] Sungwook Kim. New application task offloading algorithms for edge, fog, and cloud computing paradigms. *Wireless Communications and Mobile Computing*, 2020, 2020.

[Klo17] David C Klonoff. Fog computing and edge computing architectures for processing data from diabetes devices connected to the medical internet of things, 2017.

[KMVOV96] Jonathan Katz, Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[KPIM19] Kyriakos Kritikos, Manos Papoutsakis, Sotiris Ioannidis, and Kostas Magoutis. Towards configurable cloud application security. In *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 684–689. IEEE, 2019.

[KST11] Misha Kay, Jonathan Santos, and Marina Takane. mhealth: New horizons for health through mobile technologies. *World Health Organization*, 64(7):66–71, 2011.

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical report, Technical Report CSL-98, SRI International Palo Alto, 1979.

[Lau21] Laufer, Jan and Mann, Zoltán Ádám and Metzger, Andreas. Modelling data protection in fog computing systems using umlsec and sysml-sec. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 777–786, 2021.

[LD19a] C. Lachner and S. Dustdar. A performance evaluation of data protection mechanisms for resource constrained iot devices. In *2019 IEEE International Conference on Fog Computing*, ICFC'19, pages 47–52, 2019.

[LD19b] C. Lachner and S. Dustdar. A performance evaluation of data protection mechanisms for resource constrained iot devices. In *2019 IEEE International Conference on Fog Computing (ICFC)*, pages 47–52, 2019.

[LKS10] Jongdeog Lee, Krasimira Kapitanova, and Sang H Son. The price of security in wireless sensor networks. *Computer Networks*, 54(17):2967–2978, 2010.

[LL18] Fang Liu and Tong Li. *A clustering-anonymity privacy-preserving method for wearable iot devices*. Security and Communication Networks(2018, 2018.

[LMD21] Clemens Lachner, Zoltán Ádám Mann, and Schahram Dustdar. Towards understanding the adaptation space of ai-assisted data protection for video analytics at the edge. In *2021 IEEE 41st International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 7–12. IEEE, 2021.

[LP20] Donghyeok Lee and Namje Park. Blockchain based privacy preserving multimedia intelligent video surveillance using secure merkle tree. *Multimedia Tools and Applications*, pages 1–18, 2020.

[LQB18] Peng Liu, Bozhao Qi, and Suman Banerjee. EdgeEye: An edge service framework for real-time intelligent video analytics. In *Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking*, pages 1–6, 2018.

[LRCLP17] Zongqing Lu, Swati Rallapalli, Kevin Chan, and Thomas La Porta. Modeling the resource requirements of convolutional neural networks on mobile devices. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1663–1671, 2017.

[LRD19a] Clemens Lachner, Thomas Rausch, and Schahram Dustdar. Context-aware enforcement of privacy policies in edge computing. In *2019 IEEE International Congress on Big Data (BigDataCongress)*, pages 1–6. IEEE, 2019.

[LRD19b] Clemens Lachner, Thomas Rausch, and Schahram Dustdar. Oriot: A source location privacy system for resource constrained iot devices. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019.

[LRD21] Clemens Lachner, Thomas Rausch, and Schahram Dustdar. A privacy preserving system for ai-assisted video analytics. In *2021 IEEE International Conference on Fog and Edge Computing (ICFEC)*. IEEE, 2021.

[MAM15] Farooq Muhammad, Waseem Anjum, and Khairi Sadia Mazhar. A critical analysis on the security concerns of Internet of Things (IoT). *International Journal of Computer Applications (0975 8887)*, 111(7), 2015.

[Man19] Zoltán Ádám Mann. Optimization problems in fog and edge computing. In Rajkumar Buyya and Satish Narayana Srirama, editors, *Fog and edge computing: Principles and paradigms*, pages 103–121. Wiley, 2019.

[MCW+15] Tomasz Marciniak, Agata Chmielewska, Radoslaw Weychan, Marianna Parzych, and Adam Dabrowski. Influence of low resolution of images on reliability of face detection and recognition. *Multimedia Tools and Applications*, 74(12):4329–4349, 2015.

[MEB16] Yassine Maleh, Abdellah Ezzati, and Mustapha Belaissaoui. An enhanced dtls protocol for internet of things applications. In *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pages 168–173. IEEE, 2016.

[Mic95] Peter Michel. Medication dispensing device, January 24 1995. US Patent 5,383,865.

[MIG13] George Moldovan, Anda Ignat, and Martin Gergeleit. Group-based anonymous on-demand routing protocol for resource-restricted mobile ad hoc networks. In *International Conference on Ad-Hoc Networks and Wireless*. Springer, Berlin, HeidelbergSpringer, Berlin, Heidelberg, 2013.

[MKL+21] Zoltán Ádám Mann, Florian Kunz, Jan Laufer, Julian Bellendorf, Andreas Metzger, and Klaus Pohl. Radar: Data protection in cloud-based computer systems at run time. *IEEE Access*, 9:70816–70842, 2021.

[MMPS19] Zoltán Ádám Mann, Andreas Metzger, Johannes Prade, and Robert Seidl. Optimized application deployment in the fog. In *International Conference on Service-Oriented Computing*, pages 283–298. Springer, 2019.

[MS18] Lilian Mutalemwa and Seokjoo Shin. *Strategic Location-Based Random Routing for Source Location Privacy in Wireless Sensor Networks*. 2291, Sensors 18.7, 2018.

[NA12] Kumpati S Narendra and Anuradha M Annaswamy. *Stable adaptive systems*. Courier Corporation, 2012.

[NCBB14] Nicola Nostro, Andrea Ceccarelli, Andrea Bondavalli, and Francesco Brancati. Insider threat assessment: A model-based methodology. *ACM SIGOPS Operating Systems Review*, 48(2):3–12, 2014.

[NSD18] Minh Nguyen, Priyanka Samanta, and Saptarshi Debroy. Analyzing moving target defense for resilient campus private cloud. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 114–121. IEEE, 2018.

[NSS13] Dmitry Namiot and Manfred Sneps-Sneppe. Geofence and network proximity. In *Internet of Things, Smart Spaces, and Next Generation Networking*, pages 117–127. Springer, 2013.

[Nun05] Flávio Nunes. Aveiro, Portugal: Making a digital city. *Journal of Urban Technology*, 12(1):49–70, 2005.

[NVI20] NVIDIA. Nvidia jetson tx2 module, 2020. `https://www.nvidia.com/de-de/autonomous-machines/embedded-systems/jetson-tx2/`.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 33–43. ACM, 1989.

[OftAoSIS] XACML Technical Committee Organization for the Advancement of Structured Information Standards. Xacml. `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml`.

[oHS96] US Department of Health and Human Services. Health insurance portability and accountability act, 1996. `https://www.cdc.gov/phlp/publications/topic/hipaa.html`.

[OPD16] Ankhbayar Otgonbayar, Zeeshan Pervez, and Keshav Dahal. *Toward anonymizing iot data streams via partitioning.* Mobile Ad Hoc and Sensor Systems (MASS)IEEE 13th International Conference on. IEEE, 2016.

[oS19] Protection Commission of Singapore. Guide to basic data anonymization techniques, 2019. `www.pdpc.gov.sg/-/media/Files/PDPC/PDF-Files/Other-Guides/Guide-to-Anonymisation_v1-(250118).pdf`.

[Par12] Donn B Parker. Toward a new framework for information security? *Computer security handbook*, pages 3–1, 2012.

[PCM17] Paolo Palmieri, Luca Calderon, and Dario Maio. An anonymous internetwork routing protocol for the internet of things. *Journal of Cyber Security and Mobility*, 6.2:127–146, 2017.

[PHMN16] Liliana Pasquale, Sorren Hanvey, Mark Mcgloin, and Bashar Nuseibeh. Adaptive evidence collection in the cloud using attack scenarios. *Computers & Security*, 59:236–254, 2016.

[PMM18] Alexander Palm, Zoltán Ádám Mann, and Andreas Metzger. Modeling data protection vulnerabilities of cloud systems using risk patterns. In *International Conference on System Analysis and Modeling*, pages 1–19. Springer, 2018.

[Pro19] Tor Project. You should let people choose their path length, 2019. `www.torproject.org/docs/faq.html.en#ChoosePathLength`.

[RCLC17] Xukan Ran, Haoliang Chen, Zhenming Liu, and Jiasi Chen. Delivering deep learning to mobile devices via offloading. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, pages 42–47, 2017.

[RCZ+18] Xukan Ran, Haolianz Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. Deepdecision: A mobile deep learning framework for edge video analytics. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 1421–1429. IEEE, 2018.

[RD19] Thomas Rausch and Schahram Dustdar. Edge intelligence: The convergence of humans, things, and ai. In *2019 IEEE International Conference on Cloud Engineering*, IC2E'19, pages 86–96. IEEE, 2019.

[Res20] Google Research. A local ai platform to strengthen society, improve the environment, and enrich lives, 2020. `https://coral.ai/`.

[RF17] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[RGP15] Kaushik Ravichandran, Ada Gavrilovska, and Santosh Pande. Pimico: Privacy preservation via migration in collaborative mobile clouds. In *System Sciences (HICSS), 2015 48th Hawaii International Conference on*, pages 5341–5351. IEEE, 2015.

[RR16] D Jamuna Rani and S Emalda Roslin. Light weight cryptographic algorithms for medical internet of things (iot)-a review. In *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*, pages 1–6. IEEE, 2016.

[RRG+91] Deborah Russell, Debby Russell, GT Gangemi, Sr Gangemi, and GT Gangemi Sr. *Computer security basics*. " O'Reilly Media, Inc.", 1991.

[SAKR12] M Sethi, J Arkko, A Keranen, and H Rissanen. Practical considerations and implementation experiences in securing smart object networks. *draft-aks-crypto-sensors-02 (WiP), IETF*, 2012.

[SBM+19] Abdelkarim Ben Sada, Mohammed Amine Bouras, Jianhua Ma, Huang Runhe, and Huansheng Ning. A distributed video analytics architecture based on edge-computing and federated learning. In *2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pages 215–220. IEEE, 2019.

[Sch03] Markus Schumacher. *Security engineering with patterns: origins, theoretical models, and new applications*, volume 2754. Springer Science & Business Media, 2003.

[SCZ+16] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[SD16] Weisong Shi and Schahram Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

[Sea12] Hui Suo and et al. *Security in the internet of things: a review.* Computer Science and Electronics Engineering (ICCSEE)international conference on. Vol. 3. IEEE, 2012.

[Sen09] Andrew Senior. *Protecting privacy in video surveillance*, volume 1. Springer, 2009.

[SKW15] Florian Schaub, Bastian Könings, and Michael Weber. Context-adaptive privacy: Leveraging context awareness to support privacy decision making. *IEEE Pervasive Computing*, 14(1):34–43, 2015.

[SM10] Suhaila Omer Sharif and SP Mansoor. Performance analysis of stream and block cipher algorithms. In *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, volume 1, pages V1–522. IEEE, 2010.

[SMM+09] Jeremy Schiff, Marci Meingast, Deirdre K Mulligan, Shankar Sastry, and Ken Goldberg. Respectful cameras: Detecting visual markers in real-time to address privacy concerns. In *Protecting Privacy in Video Surveillance*, pages 65–89. Springer, 2009.

[SMM18] Stefan Schoenen, Zoltán Ádám Mann, and Andreas Metzger. Using risk patterns to identify violations of data protection policies in cloud systems. In *Service-Oriented Computing – ICSOC 2017 Workshops*, pages 296–307. Springer, 2018.

[SP19] Areeg Samir and Claus Pahl. Self-adaptive healing for containerized cluster architectures with Hidden Markov Models. In *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 68–73. IEEE, 2019.

[SSX+15] Mahadev Satyanarayanan, Pieter Simoens, Yu Xiao, Padmanabhan Pillai, Zhuo Chen, Kiryong Ha, Wenlu Hu, and Brandon Amos. Edge analytics in the Internet of Things. *IEEE Pervasive Computing*, 14(2):24–31, 2015.

[ST09] Mazeiar Salehie and Ladan Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 4(2):1–42, 2009.

[Sta17] William Stallings. *Cryptography and network security: principles and practice.* Pearson Upper Saddle River, 2017.

[SWZL12] Hui Suo, Jiafu Wan, Caifeng Zou, and Jianqi Liu. Security in the Internet of Things: a review. In *2012 International Conference on Computer Science and Electronics Engineering*, volume 3, pages 648–651. IEEE, 2012.

[TPGN16] Christos Tsigkanos, Liliana Pasquale, Carlo Ghezzi, and Bashar Nuseibeh. On the interplay between cyber and physical spaces for adaptive security. *IEEE Transactions on Dependable and Secure Computing*, 15(3):466–480, 2016.

[UAT+19] Md Azher Uddin, Aftab Alam, Nguyen Anh Tu, Md Siyamul Islam, and Young-Koo Lee. Siat: A distributed video analytics framework for intelligent video surveillance. *Symmetry*, 11(7):911, 2019.

[UNSJ09] Maneesh Upmanyu, Anoop M Namboodiri, Kannan Srinathan, and CV Jawahar. Efficient privacy preserving video surveillance. In *2009 IEEE 12th international conference on computer vision*, pages 1639–1646. IEEE, 2009.

[VF13a] Ovidiu Vermesan and Peter Friess. *Internet of Things: Converging technologies for smart environments and integrated ecosystems*. River Publishers, 2013.

[VF13b] Ovidiu Vermesan and Peter Friess, editors. *Internet of things: converging technologies for smart environments and integrated ecosystems*. River Publishers, 2013.

[WSM+20] Jeffery Walton, John Steven, Jim Manico, Kevin Wall, and Ricardo Iramar. Certificate and public key pinning, 2020. `https://owasp.org/www-community/controls/Certificate_and_Public_Key_Pinning`.

[WWRL10] Cong Wang, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-preserving public auditing for data storage security in cloud computing. In *Infocom, 2010 proceedings ieee*, pages 1–9. Ieee, 2010.

[WX20] Haoxin Wang and Jiang Xie. User preference based energy-aware mobile AR system with edge computing. In *IEEE INFOCOM – IEEE Conference on Computer Communications*, pages 1379–1388. IEEE, 2020.

[XSSC06] Yang Xiao, Xuemin Shen, BO Sun, and Lin Cai. Security and privacy in rfid and applications in telemedicine. *IEEE communications magazine*, 44(4):64–72, 2006.

[YYSL12] Jean Yang, Kuat Yessenov, and Armando Solar-Lezama. A language for automatically enforcing privacy policies. In *ACM SIGPLAN Notices*, volume 47, pages 85–96. ACM, 2012.

[ZCL+19] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.

[ZO11] Hessam Zakerzadeh and Sylvia L. Osborn. Faanst: fast anonymizing algorithm for numerical streaming data. *Data privacy management and autonomous spontaneous security*, pages 36–50, 2011.

[ZSL+18] Guowei Zhang, Fei Shen, Zening Liu, Yang Yang, Kunlun Wang, and Ming-Tuo Zhou. Femto: Fair and energy-minimized task offloading for fog-enabled iot networks. *IEEE Internet of Things Journal*, 6(3):4388–4400, 2018.

[ZSYM17] Qiliang Zhu, Baojiang Si, Feifan Yang, and You Ma. Task offloading decision in fog computing system. *China Communications*, 14(11):59–68, 2017.

[Zym20] Zymbit. A security module for raspberry pi, 2020. `https://www.zymbit.com/blog-security-module-raspberry-pi/`.