Resource-aware Security Configuration for Constrained IoT Devices

Anonymous Author(s)

ABSTRACT

The Internet of Things (IoT) is the enabler for new innovations in several domains. It allows the connection of digital services with entities in the physical world. These entities are devices of different sizes ranging from large machinery to tiny sensors. In the latter case, devices are typically characterized by limited resources in terms of computational power, available memory and sometimes limited power supply. As a consequence, the use of security algorithms requires of them to work within the limited resources. This means to find a suitable implementation and configuration for a security algorithm, that performs properly on the device, which may become a challenging task. On the other side, there is the desire to protect valuable assets as strong as possible. Usually, security goals are recorded in security policies, but they do not consider resource availability on the involved device and its power consumption while executing security algorithms.

This paper presents an IoT security configuration tool that helps the designer of an IoT environment to experiment with the trade-offs between maximizing security and extending the lifetime of a resource constrained IoT device. The tool is controlled with high-level description of security goals in the form of policies. It allows the designer to validate various (security) configurations for a single IoT device up to a large sensor network.

ACM Reference Format:
Anonymous Author(s). 2023. Resource-aware Security Configuration for Constrained IoT Devices. In *Proceedings of Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The Internet of Things (IoT) allows to connect a variety of devices to each other and to the Internet. Most times small sensors provide information about the physical world. These devices are typically characterized by limited resources in terms of computational power, available memory and power supply. In this context, security aspects play an important role with the integration of IoT devices into new or existing environments. However, designing and developing a tailored solution, utilizing the available resources in an optimal way, is difficult and requires expert knowledge.

When designing an IoT environment, the assets and security goals are identified together with the stakeholder(s) and recorded within Security Policies (SPs). They describe who and under which circumstances an asset may be accessed or modified. Often, the description is in an abstract form and specific security mechanisms are not chosen until the implementation of the policy. In general, selecting a security implementation poses no problem, since modern computer systems have sufficient resources.However, the situation is different in the IoT domain, where constrained devices are deployed in remote places and are supposed to operate unattended for a long time.

For the protection of digital assets, a wide variety of security mechanisms and implementations is available. However, the use of complex security mechanisms can have a negative effect on the lifetime of an IoT device. For a system designer it is hard assess which mechanism can be used within the use case resources and how much it will influence the operational lifetime. This work describes a system that assists a developer in the design of an IoT ecosystem with respect to the selection and parametrization of security implementations for resource-constrained IoT devices to a) fulfill the security requirements stated in a SP; b) remain within available resources of the IoT device and the use case and c) identify the maximum security measures that can be applied for a given use case.

Here, a setup that allows the security mechanism to be executed on the IoT device itself is preferred. This way, true End-to-End (E2E) security can be realized, eliminating the requirement to trust the edge nodes, which may be operated by third parties. However, the option to outsource the execution of a security mechanisms to a "less" constrained edge/fog node should not be excluded, e.g. for cases where the protection of an asset on the IoT device is impossible within the available resources. Following the recommendations, the system designer can be sure to comply with the minimum technical security requirements stated in SP. The SPs shall be used to identify individual information flows, where an asset (in this case the data collected by the IoT device) is shared and hence must be protected.

The main contributions of this paper are:

- We describe desirable features for an IoT environment designing tool that combines resource availability with data security.
- We present available design and planning tools that can be used to plan an IoT ecosystem and compare

MSWIM, Oct. - Nov. 23, Montreal, Canada

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM. . . $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

them with regards to the aforementioned features. This includes research activities that try to mitigate resource limitations on IoT devices by distributing workload among different nodes within a network.
- We extend a Security Policy Language to capture available resource and define a number of transformation rules to convert the policies into a resource-aware Information Exchange Model.
- We extended a "cycle-true" emulator for MCUs and show how it can be used to estimate to resource consumption of an IoT application without the need of an actual deployment.

The rest of this paper is organized as follows: Section 2 lists a series of desired features we believe can help a designer of an IoT ecosystem to implement security measures in a more efficient way. Section 3 presents different types of solutions that have been developed in the past to address the issue of limited resources in the IoT or sensor networks in general. Section 4 describes a resource-aware model to capture individual information flows within a network followed by a description of the system design in Section 5. In Section 6, we evaluate the general concept in a small scale example, before concluding the paper in Section 7.

2 FEATURES & DESIGN GOALS

The design tool, as proposed in this paper, shall be used to plan a secure IoT environment. Data that is shared within this environment is supposed to be protected throughout the use case, i.e. not only during transport, but at application layer. The user shall be able to compare the impacts of different configurations in terms of resource consumption and IoT device lifetime. Following a list of features and design goals for the envisioned tool.

F-1 Usable by non-security Experts: The envisioned solution is meant to allow non-security experts to state high-level security requirements for the information that is collected by IoT sensors.

F-2 Bridging Semantic Gap between high-level description to low-level configuration: There is a semantic gap between the security requirements by stakeholders from a high-level to specific implementations and configurations at a low-level. The high-level SP should be defined formally and processable in a (semi-) automated way.

F-3 Individual Information Flows to entire Sensor Networks: Handling the constraints of a specific IoT device requires to identify all information flows it is involved in. (Mostly, where is acts as source or destination.) This means it must be possible to extract individual information flows out of the SPs.

F-4 Specify Use Case Resources: The problem with constrained resources in IoT environments is rooted in constrained resources in a use case in general. If a sensor had unlimited time to provide its readings, even the slowest processor would have sufficient computational power. However, in real-world scenarios resources of a use case are limited and must be respected in the system design phase. As these resources depend on the use case, they must be determined in accordance with the stakeholder, i.e. within the high-level SP.

F-5 Issue-Specific Level: When concerned with limited use case resources that are available to individual IoT devices providing specific assets, the SP must describe address a specific use case. Alternatively, following the categorization defined by the National Institute of Standards and Technology (NIST) in [15], the policy must be `issue-specific` specification level. Having SPs specifying concrete use cases is necessary to specify resource availability for individual information flows and involved devices.

F-6 Emulation-based Resource Consumption Estimation: The number of candidates for a suitable configuration on the IoT device grows exponentially with the number of algorithms/implementations (including different libraries for the same algorithm), the number of available parameters and their possible values and the number of available Micro-Controller Units (MCUs). Executing possible candidates in an emulation environment allows to consider new combinations for which no consumption data was gathered yet.

F-7 Maximizing Security within available Resources: The system shall find a set of security configurations that are operating within the limited resources of the use case. These configurations will describe different algorithms and their parametrization. As a consequence, the achievable security level will also differ and hence a ranking mechanism must allow the user to judge about the tradeoff between security level and resource consumption.

F-8 Support for Algorithms with Task-Offloading: Although E2E encryption is the preferred way to protect an asset, we acknowledge that there might be cases, where the execution of security algorithms must be shifted to an external entity with less constrained resources. The system shall support the offloading of (parts of) a security algorithm for the benefit of saving resources or extending the lifetime of an IoT device.

3 RELATED WORK

In this Section, we present related works that address the issue of resource-constrained IoT devices for a secure data provisioning. We focus on tools that allow to design of sensor networks with regards to the challenges using IoT devices, e.g. limited energy source and computational power.

In [17], Ramadan et al. present `SensDep`, a design tool to optimize cost and coverage in sensor networks. The tool divides an area into zones and tries to place active sensors into each zone, where each sensor has different attributes such as reliability and lifetime (battery depletion). However, in their performance evaluation, characteristics such as the battery lifespan are generated randomly. Furthermore, the influence of different algorithms executed on the sensor is not considered. Vibha Prasad et al. developed `ANDES`, an "ANalysis-based DEsign tool for wireless Sensor networks" [16]. The goal of the tool is to improve the predictability of the performance of a Wireless Sensor Network (WSN) design

and reduce the costs for testing. The focus in ANDES is to experiment with different communication scheduling algorithms. The tool can be seen as an extension to the Architecture Analysis and Design Language (AADL), meaning the designer of the system must be able the write AADL documents, which is not trivial. Wireless Sensor Network Deployment Design (`WSN-DD`) by David Santiago [9] is a tool to capture high-level requirements of a WSN. The wizard-like user interface was developed as browser-based map application, where sensors and obstacles can be placed. The tool allows to specify sensor data streams. Requirements are expressed as Quality of Service (QoS) expectations, whose feasibility is evaluated. The expected lifetime can be estimated analytically, based on a cost model. However, there is no relation to the protection of the data provided by the WSN. `Tinker`, by Jeremy Elson et al. [12], is a collection of software modules for the processing of data in data streams. Tinker provides features to analyze the effects of data loss compared to a complete data set. With this, unreliable IoT sensor nodes and network connections, typical in WSNs, are acknowledged. The user-groups targeted by Tinker are developers of IoT applications. However, Tinker itself is not an in-depended application, but must be integrated into one. `Que` (by Chu et al. [10]) implements Tinker and is a framework to simulate/emulate sensor networks. Que addresses' the topics scalability, realistic data, interoperability with other components, and longevity. The last one is dealt by minimizing the sampling rate and duty-cycle of a sensor node. The paper also touches the topic of energy reduction, but only by performing data reduction on application level, resulting in fewer/shorter radio transmission of the IoT sensors. Aazam et al. [3, 4, 2] developed a model to allocate resources of a fog node for customers, based on their previous behavior. They named their solution MEdia FOg Resource Estimation (`MeFoRE`). Their assumption is, that an end device that interrupted the usage of a fog node in the past is likely to do so in the future. As these users seem not interested in QoS, less fog resources are allocated. Salah ud din et al. [11] propose a protocol named Power-Efficient Wireless Multimedia of Things (`PE-WMoT`). The main idea behind PE-WMoT is to reduce unnecessary transmissions of sensor nodes in a (sub-)cluster that observe the identical area (field of view).

The topic of designing a WSN or an IoT ecosystem is often addressed by Network Simulators (`NetSims`). NetSims can be used for implementing and investigating the performance of a system under varying parameters and configurations by considering multiple "what-if" cases. The focus of network simulators is analyzing communication protocols in different scenarios. Prominent network simulators include Network Simulator 2 and 3 (NS-2, NS-3), TOSSIM, OMNeT++, or MatLab/SimuLink. The interested reader is referred to [18]. The support to simulate the resource consumption is integrated in only few simulators out-of-box, but as simulators are generally extendable, own energy models could be added in principle. However, security is usually handled at network layer via Transport Layer Security (TLS), limiting the range of available security algorithms and configurations.

The EU projects SECURED [21] and ANASTACIA [22] investigated ways to define security requirements for Software Defined Networks (SDN). For this, the project developed the policy language `High-level Security Policy Language (HSPL)` to control the use of security functions on network fog/edge devices. With it, general protection requirements are expressed by typically non-technical end-users and is based on a set of predefined vocabulary. The syntax follows a subject - predicate/action - object approach, followed by the possibilities to add further conditions in the form of key-value pairs. The subject specifies the user accessing/performing an operation on the object. The action is the operation to perform. The three components (subject, action, object) need to be predefined, depending on the use case as well as their possible combinations. HSPL allows to gather the security policies with the stakeholder on a high level and in a formal way. These policies are later transformed into configuration files for security software running on the fog device, for example generating nftables[1] firewall rules.

In Table 1, a comparison of the tools for the design of resource-constrained networks and IoT environments, and their support of the features listed in Section 2 is given. The semantic gap between security requirements and low-level configuration (**F-2**) is mainly addressed by HSPL and to some degree by WSN-DD. Some approaches focus on the allocation of resources in fog nodes only **F-8**, but not IoT end device (**F-3**). The offloading of complex tasks also serves the purpose of dealing with limited resources and sounds reasonable. However, it requires to trust these (third party) fog nodes. In case when the data provided by the IoT devices is confidential, using offloading techniques becomes difficult or impractical. SensDep and ANDES address technical experts **F-1** and focus on the positioning of nodes within an network. Tinker and Que as well are not meant for the use by stakeholders but rather developers **F-5**. Same is true for prominent network simulators like NS-3 or OMNeT++. Given a suitable simulation model, they are able to predict resource consumption **F-6**, but are difficult to use. The policy language HSPL, provided by the SECURED project, was the most promising solution, but like MoFeRE and PE-WMoT concentrated on resources within a fog/edge node. None of the tools relates security algorithms with their resource consumption (**F-4**). The last column represents the proposed system, which is here named eHSPL as it extends the HSPL language.

4 INFORMATION EXCHANGE MODEL

This section specifies the Information Exchange Model (IEM) developed to describe the secure exchange of information between two or more parties, where at least one of them is considered to be constraint in terms of resources. The model includes at least two parties that are involved in the transfer of the information, mainly those responsible for the execution of security mechanisms. There is at least one security goal that needs to be applied in the protection of the information.

[1] https://netfilter.org/

Table 1: IoT Design Tools Comparison Overview

Solution → Property ↓	SensDep	ANDES	WSN-DD	Tinker	Que	MeFoRE	PE-WMoT	NetSim	HSPL	eHSPL
F-2	×	×	✓	✓	×	×	×	×	✓	✓
F-3	✓	×	✓	✓	✓	✓	✓	✓	✓	✓
F-1	×	×	✓	×	×	×	×	×	✓	✓
F-4	✓	⊗	✓	×	×	✓	✓	✓	×	✓
F-5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
F-6	×	×	×	×	✓	×	✓	✓	×	✓
F-7	?	×	⋆	×	×	×	×	✓	×	✓
F-8	✓	✓	✓	×	✓	✓	✓	✓	×	✓

✓ - supported; × - not supported; ⊗ - not supported, but planned; ⋆- only reliability; ? - unknown

The model is able to specify the security requirements on a high- and a low-level. The model is divided into four concepts, which are described in the following subsections.

4.1 Information Flow $\mathcal{F}$

An Information Flow $\mathcal{F}$ describes the exchange of an information in form of a *Message* M between at least two *Participants*, a data *Provider* P_P and a *Receiver* P_R. In between other Participants may assist the transport by forwarding M. Participants are connected to one another by *CommunicationLinks*. M contains at least one (digital) *Asset* A, which must be protected and which is of interest by an attacker. The distinction between an Asset and a Message is done, since the size of a M depends on the communication protocol, whereas the size of an A is determined by the size of the digital information, e.g. a sensor measurement. Within the IEM, A is a information that is gathered by/provided to a resource-constraint IoT device.

In a flow $\mathcal{F}$, M may pass additional entities, which can be classified as active and passive. Passive participants will be called *Gateway* and active ones *Proxy*. Gateways forward a message to the next participant, while Proxies execute security mechanisms. For example, a Proxy can perform algorithms to (re-)encrypt M. The name Proxy was chosen to relate to the Proxy Re-Encryption (PRE) [19] schemes. A CommunicationLink is used by two participants to transfer M. They may support different communication technologies, which consume different amount of resources. The Information Flow concept is depicted in Figure 1 at the bottom left.

4.2 Devices and Resources concept

Each Participant involved in a flow $\mathcal{F}$ uses a form of *Device* to receive, transmit or process M. Each Device has a set of *Resources* and *Capabilities*, that can be used in $\mathcal{F}$. In the context of message exchange in an Information Flow using resource-constrained IoT devices the most relevant Resources are: Computational Power (cycles per second, number of cores), Memory (Flash, RAM), Communication Technology (throughput, duty cycles), Energy Consumption (amount of current drawn in different power modes of the MCU, including periphery such as the sensor itself and communication device) and financial cost. In addition to the resources provided by the Device, the use-case itself provides Resources. Typically these are described in the form of use case requirements and include: Time (available time to transfer M from P_P to P_R), Energy Source (capacity of a battery if applicable) and Budget. The concept is depicted in the top left corner in Figure 1.

4.3 Performance concept

A flow $\mathcal{F}$ possesses a series of *Performance Metric* (P_M), indicating the ability of a *Device* to perform a *Security Implementation* (S_I) (see next concept) and the resulting resource consumption. P_Ms need to be determined by either analyzing the algorithm of the S_I or experimentally, e.g. through emulation. There are two types of P_M: a P_M either describes the amount of resources that are going to be consumed when executing the S_I, if the resource is consumable in nature. As an example, the load level of a battery is reduced a little over time as a consequence of executing a S_I. Secondly, a P_M can describe the amount of assets that can be processed in a certain amount of time, e.g. how many bytes of data can be encrypted/decrypted per second.

In addition, we define *Performance Indicator* (P_I) as a combination of all relevant P_M that influence a S_I. The P_Ms can be weighted to reflect use case resources that describe a hard threshold, e.g. real-time requirements. The intention is to be able to compare different implementations, where one might require more time than an other, but allocating less memory at the same time. The concept is depicted in the top right corner in Figure 1.

4.4 Security Specification and Configuration concept

The goal of the IEM is to describe and process and transform high-level SPs, which were developed together with a stakeholder, into concrete security configuration recommendations. For this, the level of abstraction must be refined. This concept defines the following abstraction levels: *Security Service* (S_S) (e.g. confidential, integrity), *Security Mechanism* (S_M) (e.g.

encryption/steganography or digital signatures/Message Authentication Code (MAC)), *Security Implementation (S_I)* (e.g. AES, Twofish, Blowfish, RSA), and Security Configuration (S_C) (S_Is and their parametrization that usable in the scenario).

5 SYSTEM DESIGN

This section describes the design of the envisioned IoT environment security design tool. First, an extension to the HSPL policy language is proposed to allow for the annotation of resources and the identification of information flows. Following are transformation rules to interpret these HSPL policies. Thirdly, modifications on an emulator for ATMEL MCUs have been made to allow for a scenario specific estimation of resource consumption. This emulator has been chosen because its ability to emulate the well known, cheap and easily available Arduino platform. Furthermore, the emulator has an energy monitoring subsystem implemented already, which forms the basis of the resource consumption predictions.

5.1 eHSPL- Extended High-level Security Policy Language

In order to guide the resource estimation from a high-level Security Policy starting point, ANASTACIA's HSPL was extended. HSPL was chosen as it is an easy to understand SP that is defined formally and thus can be processed in an automatic way. We refer to the extended version of HSPL as eHSPL.

Although not explicitly stated by the ANASTACIA project [5, 1], the documentation for HSPL-fields indicates that they only allow to specify characteristics and limitations for HSPL-objects. To be able to annotate the HSPL policies with additional information required in the IEM, we introduce a set of specific HSPL-fields that affect the HSPL-subject as well. In that respect, we extend the original concept with the following HSPL-fields to annotate HSPL-subjects with further properties, namely:

- **within**: With this field real-time requirements in the provision of data can be specified. This field describes the maximum transfer time of the message M from provider P_P to receiver P_R.
- **has energy source**: This field is important for IoT devices with a limited energy source, i.e. that are battery powered. Its value describes the amount of energy stored within the battery in mAh (milli-amp-hour).
- **runs autonomous for**: With this field, the maintenance interval for an IoT device can be specified. It is closely related to the field **has energy source** and is used to define an upper threshold for the consumption of the energy resource.
- **with budget**: In cases where the device type is not known upfront, this field can be used to specify a maximum budget that can be spent. The system then can iterate through different candidates to search for the most suitable one.

Table 2: Participant identification based on HSPL Actions

Type → Action ↓	Provider	Gateway	Receiver	Proxy	Asset
provides	✓	-	-	-	-
publish	✓	-	-	-	-
protects*	✓	-	-	-	✓
(not)authorised to access	-	-	✓	-	(✓)
receives	-	-	✓	-	-
subscribes	-	-	✓	-	-
requests	-	-	✓	-	-
forwards	-	✓	-	-	-
converts	-	-	-	✓	(✓)

- **every**: This field is used to specify the update frequency of the IoT device in which new data is provided. The field has to be some sort of time value, such as 1 second(s). It is expected that the value of this field is smaller than the one in the field **within**.
- **has size**: With this field, the size of an asset (i.e. the object in the eHSPL policy) can be specified.
- **is a**: This field is used to specify a device for the corresponding HSPL-subject. This might be relevant if a certain microcontroller must be used due to other scenario requirements.

Next it is necessary to identify the roles of the participants as well as an asset from the eHSPL policies. In order to do so, new eHSPL-actions were defined together with a mapping of those actions to the possible roles in an Information Flow, as shown in Table 2. For most eHSPL-actions the mapping should be self-explanatory as it states what the role does. A special eHSPL-action is the `protects` action as it is used to a) identify an asset and b) to list the security service S_S for this asset. Using this action the HSPL syntax is exceptionally: *subject* `protects` <list of security services> *asset*.

5.2 eHSPL Policy to Information Flow Transformation

For the previously presented concept, we implemented a transformation tool, applying the transformation rules from the eHSPL into an instance of the IEM. In a first step the eHSPL policies are ordered in such a way that those with a `protect` action come first. Each policy with this action is considered to declare an new instance of an Information Flow $\mathcal{F}$. Iterating through the other policies new restrictions are added to a flow. Which flow is identified by the eHSPL object (i.e. the flows' Asset). For example, consider the following two policies:

- Sensor A protects Confidentiality of temperature reading.
- User B is authorized to access temperature reading.

Both policies belong to the same Information Flow as both specify restrictions for the same asset *temperature reading*. *Sensor A* has the role of provider with the action *protects Confidentiality* and *User B* the role of a receiver with the action *authorized to access*. Note how the Sensors action is also used to specify the S_S *Confidentiality*.

After all eHSPL policies have been processed, in a second stage the Information Flows are tried to be merged. That is the case when multiple assets have the identical provider, receiver and security service(s) specified. This is important in terms of resource consumption as the assets can be transferred in a single message. For example, in our evaluation scenario, the provider provides a temperature and humidity reading from a single sensor.

A manual step in the transformation process remains the selection of a security mechanism S_M for the S_S specified in the eHSPL policies (in the above example Confidentiality). Although only the security mechanism 'Encryption' may be realistic for the S_S Confidentiality, with other S_Ss, multiple S_M options might be applicable.

In a next step, another component named "Firmware Composer" gets as input information about a single flow and generates a firmware image for a specific MCU, where a specific security implementation S_I is used on a data blob of the same size as the asset. The user of the transformation tool can select which S_I to try. The firmware generation uses a template engine with S_I specific templates for each operation of a participant. The firmware code is compiled in the PlatformIO development environment, providing estimations about the memory consumption. The firmware is then passed to an emulation environment to analyze the energy consumption (i.e. processing time). The emulation environment is described in more detail in the next section.

5.3 AVRORA: Atmel MCU Emulator Extensions

To determine the resource consumption of an security implementation within an emulation requires a "cycle-true" emulator. That means, each instruction in the firmware is emulated in the same amount of cycles as it would require when the firmware is executed on the real MCU. This does not mean that the emulation must be performed in the same amount of time as the execution on real hardware would require, only that the execution is done in the same amount of cycles. Often, emulation environments such as Apple's Rosetta or QEMU translate compiled code for one processor architecture to another processor architecture, which makes it impossible to determine the execution time (and thus resource consumption) reliably [8]. We decided to use the AVRORA emulator [7] developed at the University of California, Los Angeles (UCLA), which supports AVR MCU used by the popular and widely available Arduino Leonardo platform (equipped with an AVR ATMega32u4 MCU[13]). This allowed a direct comparison between emulation and the execution on real hardware. Furthermore, support for the Arduino Uno platform (AVR ATMega328P MCU) was added.

A benefit of the AVRORA emulator is its ability to track the energy consumption during emulation. For this, the current drawn by the MCU in each mode (active and sleep modes) were measured and implemented into AVRORA's energy model. The energy model is implemented as a Finite State Machine (FSM), where each state represents a mode of the MCU. The FSM keeps track for how many cycles it remained in a specific state. By knowing the MCU frequency, the time spent in each mode and in combination with the operating voltage, the energy consumption can be calculated. The energy models for the Arduino Leonardo and Uno were also added.

In the evaluation scenario (see Section 6) the MCU has to perform a task in a specified interval and in between is allowed to change into a sleep mode. To realize this the Watchdog Timer (WDT) is utilized, however, switching back into the active state triggered by the WDT was not implemented in AVRORA. Thus, the emulator was extended by reading the watchdog control register Watchdog Timer Control Register (WDTCSR) ([6] p. 60), which holds the information about how long the MCU is supposed to sleep. With this time and the frequency of the MCU, the number of cycles in sleep mode is calculated and the internal clock is fast forwarded by this number of cycles before switching back into the active mode. Before switching back, the FSM stores the number of cycles that should have been spent in this state for the purpose of energy calculation. This way, the sleep intervals can be emulated without actually spending a lot of emulation time in one of the sleep states.

6 EVALUATION

In this section we provide an evaluation about the ability of the system to predict the runtime of an hypothetical scenario. In the scenario, a temperature and humidity sensor (a DHT22 [2]) is used to provide new sensor reading every twenty seconds for a period of 24 hours. In between, the MCU goes into sleep mode ("PWR_DOWN"). As security goal, the confidentiality of the sensor readings must be protected by encrypting the data before transmission. The readings are embedded into a JavaScript Object Notation (JSON) message (40 bytes in size), encrypted and transferred via Universal Asynchronous Receiver Transmitter (UART) interface. The sensor is connected to an Arduino Leonardo hardware platform with an ATMega32u4 MCU.

To protect the confidentiality the encryption algorithm Advanced Encryption Standard (AES) is used in Cipher Block Chaining (CBC), Electronic Code Book (ECB) and Counter Mode (CTR) mode. As implementations serve the AESLib library [14] as well as the TinyAES [20] implementation, implementing the 3 selected encryption modes. The former is executed with a key length of 128 bit and only CBC (the only supported key length), while the later is executed with the key lengths 128 bit and 256 bit.

The use case was described in two very simple eHSPL security policies describing a sensor as data provider using

[2] https://www.adafruit.com/product/385

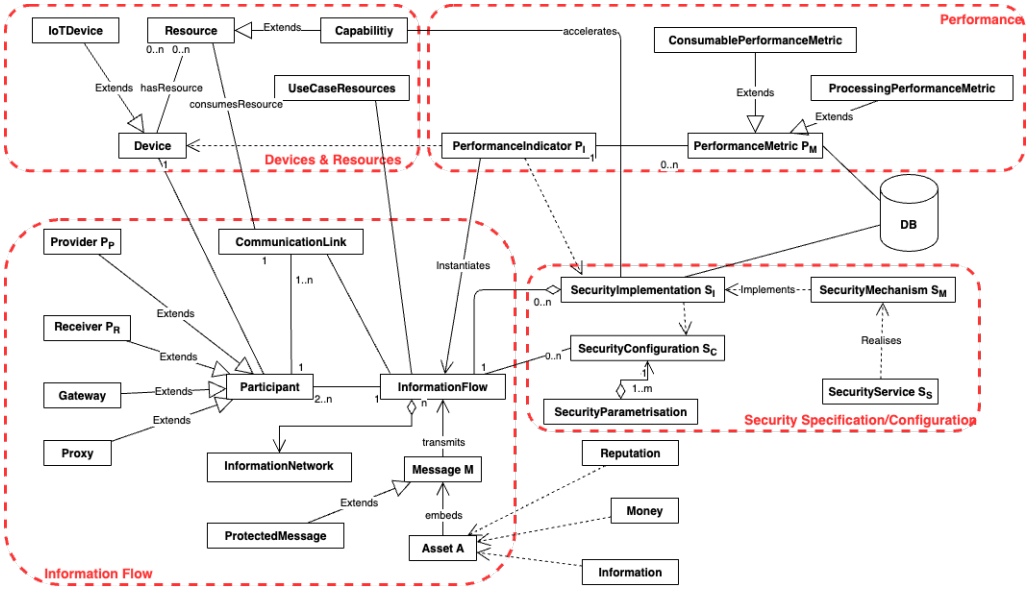

Figure 1: Overview of the Information Exchange Model

the eHSPL action 'protects' with 'Confidentiality' as security service. The timing constraint of 20 seconds between each measurement and the size of the asset were described using the according eHSPL fields 'has size' and 'every'. Furthermore, a PC as receiver with no further constraints regarding its resources was described in a seconds eHSPL policy. During the processing of the eHSPL policies both policies have been merged, since both address the same asset (sensor readings). Next, a firmware was generated, utilizing the various security implementations on a dummy asset of the same size. As the runtime of a security implementation is independent of the cleartext message except the size, the MCU will spend the same amount of time for the encryption. Then, the firmware was compiled and executed with the modified AVRORA emulator (section 5.3) for a emulated time of 24 hours.

In this fictive scenario, the measurement frequency was deliberately high to be able to see the influence on energy consumption in such a small time frame. As power supply served two lithium-ion batteries connected in series with a capacity of 700 mAh. When fully charged, the batteries have a voltage of about 3.7 V each. Table 3 shows the drop of the voltage of the two batteries after 24 hours as well as the average voltage. Furthermore, the required charge C to re-fill both batteries (Batt1 and Batt2) is shown. The charge C was measured using the `SKYRC iMAX B6 mini` battery charger. With that, the energy E were calculated as follows:

$$E = (C_{Batt1} + C_{Batt2})/t_e * U_{avg} * t_s \quad (1)$$

with emulation time t_e equal to 24 hours, i.e. the runtime of the experiment, and the simulation time t_s (for convenience as well 24 hours). The results are compared with the predicted energy consumption by the emulator in the last column. The predictions are close to the measurements, but marginally differ between the different key lengths. As shown in the last column, the deviation is about 1% between measurement and emulation for TinyAES and 2% for AESLib. A reason for the deviation between emulation and measurement could be, that the emulator assumes a constant voltage by the power supply. The voltage of lithium-ion batteries however decreases as the battery is discharged.

Table 4: Predicted Energy consumption in Joule for different configurations and roles.

Role	Conf.	Leo	Uno
Provider	TAES 128 ECB	3251	8496
Provider	TAES 128 CBC	3667	8669
Provider	TAES 128 CTR	3745	8701
Provider	AESL 128 CBC	4149	8868
Receiver	TAES 128 ECB	4802	9140
Receiver	TAES 128 CBC	5659	9495
Receiver	TAES 128 CTR	4899	9180
Receiver	AESL 128 CBC	5233	9318

Table 4 shows the influence of different security configurations and on different hardware platforms. In this imaginary scenario, a sensor provides a new reading every 10 seconds for a day. Again the two implementations TinyAES (TAES) and AESLib (AESL) are used with fixed key size of 128 bit and modes. As hardware platform the Arduino Uno R3 (ATMega329P MCU) and Leonardo (ATMega32u4 MCU) were emulated. It shows that encrypting with AESLib consumes most energy. Decrypting requires more energy in all configuration. Furthermore, CBC with TinyAES requires most.

Table 3: Experiment

Configuration			Voltage U [V]			Charge C [mAh]			Calculation			Emulation	
Algo	Key Len	Mode	max	min	avg	Batt1	Batt2	Σ	I [mA]	P [mW]	E [mJ]	E [mJ]	δ
AESLib	128	CBC	7.97	7.48	7.7278	161	162	323	13.458	104.00331	8,985,885.84	9,127,694.651	2%
TinyAES	128	CBC	8.1	7.49	7.73388	159	162	321	13.375	103.44065	8,937,271.7	8,993,735.439	1%
TinyAES	192	CBC	8.01	7.48	7.7306	162	166	328	13.666	105.65153	9,128,292.48	8,993,735.44	-1%
TinyAES	256	CBC	8.02	7.49	7.7392	161	159	320	13.333	103.18933	8,915,558.4	8,993,735.442	1%

7 CONCLUSION

To extend the lifetime of IoT devices requires the efficient usage of the limited resources on all levels. In this paper, we addressed the issue of resource-constrained IoT devices by selecting and configuring suitable security algorithms in an IoT environment. Our proposed solution starts by capturing the security goals within an extended version of the HSPL policy language. The policies are transformed into an instance of the presented IEM afterwards. With this, different security implementations and configurations are tested by generating a firmware image, which is later executed within a cycle-true emulator to estimate the later resource consumption. For an IoT sensor gathering data in a cyclic way, this allows to predict the runtime of an iteration and with that the energy consumption over a longer period of time.

In the future, we plan to scale up the prediction of the resource consumption from single Information Flows to entire sensor networks. Here, the predictions about the resource-consumption from single Information Flows shall be used as input for simulation models of the network. This will allow to investigate the workload of entities in a network that must handle multiple data streams, i.e. the participants Gateway and Proxy, and in which cases they become a bottleneck.

REFERENCES

[1] *AANASTACIA Project - GitLab repsitory*. GitLab. URL: https://gitlab.com/anastacia-project (visited on 10/08/2021).

[2] Mohammad Aazam and Khaled A. Harras. "Mapping QoE with Resource Estimation in IoT". In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. 2019 IEEE 5th World Forum on Internet of Things (WF-IoT). Apr. 2019, pp. 464–467. DOI: 10.1109/WF-IoT.2019.8767254.

[3] Mohammad Aazam, Chung-Horng Lung, and Ioannis Lambadaris. "MeFoRE: QoE based resource estimation at Fog to enhance QoS in IoT". In: *2016 23rd International Conference on Telecommunications (ICT)*. 2016 23rd International Conference on Telecommunications (ICT). May 2016, pp. 1–5. DOI: 10.1109/ICT.2016.7500362.

[4] Mohammad Aazam et al. "IoT Resource Estimation Challenges and Modeling in Fog". In: *Fog Computing in the Internet of Things: Intelligence at the Edge*. Ed. by Amir M. Rahmani et al. Cham: Springer International Publishing, 2018, pp. 17–31. ISBN: 978-3-319-57639-8. DOI: 10.1007/978-3-319-57639-8_2. URL: https://doi.org/10.1007/978-3-319-57639-8_2 (visited on 01/25/2023).

[5] *ANASTACIA Project - Advanced Networked Agents for Security and Trust Assessment in CPS / IOT Architectures*. URL: http://www.anastacia-h2020.eu/ (visited on 10/08/2021).

[6] "ATmega16U4/32U4 Datasheet". In: (), p. 438.

[7] *Avrora - The AVR Simulation and Analysis Framework*. URL: http://compilers.cs.ucla.edu/avrora/ (visited on 03/17/2022).

[8] Sorav Bansal and Alex Aiken. "Binary Translation Using Peephole Superoptimizers." In: *OSDI*. Vol. 8. 2008, pp. 177–192.

[9] David Santiago Bonilla Bonilla and Ixent Galpin. "WSN-DD: A Wireless Sensor Network Deployment Design Tool". In: *Data Analytics*. Ed. by Andrea Calì et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 146–152. ISBN: 978-3-319-60795-5. DOI: 10.1007/978-3-319-60795-5_15.

[10] David Chu et al. "Que: A Sensor Network Rapid Prototyping Tool with Application Experiences from a Data Center Deployment". In: European Conference on Wireless Sensor Networks (EWSN '08). Jan. 1, 2008. URL: https://www.microsoft.com/en-us/research/publication/que-a-sensor-network-rapid-prototyping-tool-with-application-experiences-from-a-data-center-deployment/ (visited on 03/06/2023).

[11] Muhammad Salah ud din et al. "Improving resource-constrained IoT device lifetimes by mitigating redundant transmissions across heterogeneous wireless multimedia of things". In: *Digital Communications and Networks* 8.5 (Oct. 1, 2022), pp. 778–790. ISSN: 2352-8648. DOI: 10.1016/j.dcan.2021.09.004. URL: https://www.sciencedirect.com/science/article/pii/S235286482100064X (visited on 01/26/2023).

[12] J. Elson and A. Parker. "Tinker: a tool for designing data-centric sensor networks". In: *2006 5th International Conference on Information Processing in Sensor Networks*. 2006 5th International Conference on Information Processing in Sensor Networks. Apr. 2006, pp. 350–357. DOI: 10.1145/1127777.1127830.

[13] Pavlo Ilin. *PIlin/avrora-arduino*. URL: https://github.com/PIlin/avrora-arduino (visited on 01/18/2023).

[14] Davy Landman. *Arduino AESLib*. URL: https://github.com/DavyLandman/AESLib (visited on 01/17/2023).

[15] Michael Nieles, Kelley Dempsey, and Victoria Pillitteri. *An Introduction to Information Security*. NIST Special Publication (SP) 800-12 Rev. 1. National Institute of Standards and Technology, June 22, 2017. DOI: 10.6028/NIST.SP.800-12r1. URL: https://csrc.nist.gov/publications/detail/sp/800-12/rev-1/final (visited on 11/28/2022).

[16] Vibha Prasad et al. "ANDES: An ANalysis-Based DEsign Tool for Wireless Sensor Networks". In: *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*. 28th IEEE International Real-Time Systems Symposium (RTSS 2007). ISSN: 1052-8725. Dec. 2007, pp. 203–213. DOI: 10.1109/RTSS.2007.28.

[17] R. Ramadan, K. Abdelghany, and H. El-Rewini. "SensDep: a design tool for the deployment of heterogeneous sensing devices". In: *Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems*. Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems. Apr. 2006, 10 pp.–53. DOI: 10.1109/DSSNS.2006.15.

[18] Richa Sharma, Vasudha Vashisht, and Umang Singh. "Modelling and simulation frameworks for wireless sensor networks: a comparative study". In: *IET Wireless Sensor Systems* 10.5 (2020), pp. 181–197. ISSN: 2043-6394. DOI: 10.1049/iet-wss.2020.0046. URL: https://onlinelibrary.wiley.com/doi/abs/10.1049/iet-wss.2020.0046 (visited on 02/28/2023).

[19] K. Suksomboon et al. "In-Device Proxy Re-encryption Service for Information-Centric Networking Access Control". In: *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. 2018, pp. 303–306. DOI: 10.1109/LCN.2018.8638129.

[20] *tiny-AES-c/aes.c at master · kokke/tiny-AES-c*. GitHub. URL: https://github.com/kokke/tiny-AES-c (visited on 08/26/2022).

[21] Marco Vallini. *FP7 project SECURED deliverable "D4.1 Policy specification"*. 2015.

[22] Alejandro Molina Zarca et al. *H2020 project ANASTACIA deliverable "D2.5 Policy-based Definition and Policy for Orchestration Final Report"*. 2018.