A security mechanism for the Internet of Things in a smart home context

Dimitri Jonckers

Thesis submitted to obtain
the degree of Master of Science
in computer science engineering,
with an option in distributed systems

Supervisor:
Prof. dr. ir. Bart De Decker

Assessors:
Prof. dr. ir. Y. Berbers
Dr. A. Kimmig

Mentor:
Ir. A. Put

Academic year 2015-2016

Table of contents

Abstract

The Internet of Things (IoT) broadens the scope of the internet to tens of billions of devices. Because of the heterogeneity of the connected objects and their specifications, it becomes difficult to craft a general framework for the IoT and its security. This thesis aims to provide security and privacy for Internet of Things devices in a smart home setting.

The first and core contribution is the development of a gateway which stands at the border of the smart home, between the home's devices and outside users such as service providers. It is capable of providing confidentiality, authentication, authorisation and privacy and can take care of this on behalf of constrained devices which are incapable of securing themselves. The modular architecture includes several providers for each security domain, and can easily be extended in order to support more mechanisms. Another capability of the gateway is service discovery by looking up devices offering requested services.

The gateway enforces security based on policies which the user configures. A second contribution of this thesis is a policy description language designed for this purpose. It allows users to specify requirements for their devices and communication channels with other objects and parties, possibly located outside of the smart home.

Performance test results show a limited impact on performance, allowing tens of session set-ups per second and several hundreds of messages per second to be exchanged within a session. Hence, the gateway provides security in the IoT in a performant manner. The flexibility in supporting several security providers and the possibility to address services uniformly imply that the gateway will aid developers to securely create applications for the heterogeneous Internet of Things.

Korte Nederlandse samenvatting

Het Internet der Dingen (Internet of Things/IoT) paradigma zal zorgen dat tientallen miljarden toestellen met het internet verbonden zullen zijn. Omwille van de heterogeniteit van deze voorwerpen en de bijbehorende specificaties, wordt het moeilijk om een algemeen raamwerk voor het Internet der Dingen en de beveiliging ervan uit te werken. Deze thesis poogt beveiliging en privacy aan IoT toestellen te bieden binnen de context van een smart home.

In de eerste plaats heeft een literatuurstudie plaatsgevonden die een overzicht geeft van problemen binnen het Internet der Dingen, met een specifieke focus op beveiliging. Vervolgens is in deze studie gekeken naar bestaande oplossingen voor beveiliging van toestellen, data en privacy van gebruikers in een IoT context.

De belangrijkste bijdrage van deze thesis is het ontwerp van een gateway die zich aan de rand van het smart home bevindt, tussen de toestellen binnen het huis en gebruikers daarbuiten zoals dienstverleners (service providers). Deze gateway is in staat om confidentialiteit, authenticatie, autorisatie en privacy te verzorgen. Het is ook mogelijk deze aan te bieden voor computationeel beperkte toestellen die zichzelf niet kunnen beveiligen. De gateway heeft een modulaire architectuur, waarbij verschillende aanbieders van een bepaald beveiligingsaspect ondersteund worden en uitbreidingen mogelijk zijn. Zo is er compatibiliteit met diverse beveiligingsmechanismen, bijvoorbeeld verschillende vormen van authenticatie.

De gateway werkt met sessies die worden opgezet tussen Consumenten (Consumers) en Bronnen (Resources). Zo is het mogelijk om een bepaalde dienst aan te vragen aan de gateway, waarna deze een sessie zal opzetten met een toestel dat deze dienst aanbiedt. Binnen een sessie kunnen dan berichten worden uitgewisseld, waarbij de gateway kan zorgen voor verschillende beveiligingsvereisten voor aparte subkanalen. Zo is het mogelijk om toch gepast krachtige beveiliging te voorzien voor communicatie via het internet, terwijl dit niet mogelijk is voor de link met een zwakkere sensor binnen het huis.

De beveiliging die wordt afgedwongen wordt bepaald door een beleidstaal (policy language) die de gebruiker configureert. Een tweede bijdrage van deze thesis is zo'n beschrijvingstaal. Hierin kunnen de gebruikers vereisten uitdrukken voor hun toestellen en communicatiekanalen met andere objecten en partijen.

De resultaten van performantietests geven een zeer beperkte impact van de gateway aan. Zo is het mogelijk om tientallen sessies per seconde op te zetten of honderden berichten per seconde uit te wisselen binnen een sessie. Daarnaast biedt de gateway, zoals gezegd, een oplossing aan die verschillende beveiligingsmechanismen ondersteund en het mogelijk maakt om diensten op een uniforme manier aan te spreken. De gateway is dus een gepast mechanisme om op een veilige manier applicaties te ontwikkelen voor het heterogene Internet der Dingen.

List of figures and tables

List of figures

List of tables

1 Introduction

In recent years, the scope of the internet has increased considerably. Ten years ago, home users would use their desktop computers and laptops to access the web. In the last five years, smartphones became common, allowing access to the internet from anywhere. These mobile devices have now surpassed pcs and laptops to become the most frequently used method of accessing the internet in the United States [1].

The next broadening of this scope is enabled by the advent of smart devices [2] connecting to the internet. At present, a number of these have been introduced to the market. Smart TVs are already present in many households, enabling families to watch TV, but also offering a broad range of services, such as allowing them to browse the web, watch YouTube videos, and make Skype calls [3]. Other examples are smart fridges, smart thermostats and smart lightbulbs [4].

1.1 The Internet of Things

The emerging phenomenon in which all kinds of objects ("things"), such as the current-generation smart devices, become part of the internet, is called the *Internet of Things* (*IoT*). In this vision, a global information infrastructure is built, based on the internet [5], where many physical real-world objects will have a virtual counterpart [6] and will connect and interact to benefit humankind. This infrastructure is made possible by an improvement and integration of a number of technologies: identification, tracking, communication, sensors and distributed intelligence [7].

Hence, the Internet of Things will comprise a multitude of devices with varying capabilities. This also means some of the participating objects will have very limited resources.

At the end of 2014, 14 billion devices were already part of the IoT. There are various estimates for the expansion of this quantity in the next five years, which the U.K. government summarises as approximations between 20 and 100 billion devices by the year 2020 [8]. In 2015, the Internet of Things topped the Hype Cycle for emerging technologies, further demonstrating its importance and relevance [9].

The interaction between humans and a wealth of devices is bound to have a large impact on daily and professional life, by opening up opportunities for new and improved services. Many applications [2] [10] for the IoT are envisioned, being developed and already in use: smart homes, smart production environments, smart shopping, smart offices, smart cities, improved logistics and transportation, … .

In the next section, we will look at some of the challenges and open issues for the Internet of Things, including security. Subsequently, we will present the smart home

context as an application of the IoT, state the goals and contributions of this thesis, and give an overview of the further content.

1.2 Challenges and issues

The introduction of the IoT technology raises some important security concerns. Because of the tight connection between the real world and the IoT, its adoption could lead to security and safety breaches. As an example, if insufficient protection is in place, a burglar could replace a crude tool such as a crowbar with a smartphone to allow him access to the victim's house. Furthermore, as sensors all around humans are collecting and transmitting data, the need for privacy becomes apparent.

In a study in 2014, 57% of about 2000 interviewed consumers expressed worries about hacks and data breaches in the Internet of Things [11]. Another study [12], conducted in 2015, also interviewed about 2000 consumers and showed that almost 80% of them is worried about their privacy, specifically the collection and sharing of their data. It also showed these users want to know how companies preserve their privacy, identifying the need for openness about data gathering and protection.

These security risks are not simply perceived by users, but they exist in the real world and are present for many of the currently existing smart devices. To appreciate the seriousness of this issue, consider a 2014 study of 10 smart devices such as TVs, thermostats and sprinklers. Out of these, 8 were found to have insufficient authentication and authorisation, and 7 of them were lacking transport layer encryption [13]. Many breaches of smart devices have taken place in practice [14], a recent example being hacking a smart kettle to obtain access to the victim's network [15].

Protocol and network security is a key requirement to secure the Internet of Things [6]. Particularly important to achieve this, is the need for better (i.e. lightweight yet strong) cryptography. Many of the devices have limited resources available [16], but nonetheless encryption, authentication and access control are necessary to govern their use.

The need for privacy is recognised by lawmakers, resulting for instance in several revised principles and recommendations by the European Union [5]. However, a global framework and general legal principles are needed which are adopted by the whole IoT industry. Furthermore, technologies need to be developed to enforce these regulations [17].

In addition to security and privacy challenges, another major issue is harmonisation in the wide set of devices, protocols and services [7]. Various standardisation efforts exist, but many (home) devices still have their own specific application-layer protocol for controlling it. For true integration, standardisation needs to continue across all layers of the protocol stack.

Another set of open issues pertains to networking, specifically addressing and identification of IoT entities, and supporting their mobility [10]. The selection of protocols befitting the needs of the Internet of Things, such as a novel transport layer protocol, is particularly important to allow traffic modelling and quality of service.

Additionally, some more research is necessary in the field of energy consumption and energy harvesting. This is a key enabler to allow deployment of sensors in flexible environments, where a large battery which needs frequent replacement is impractical.

1.3 A smart home

The above discussion makes it clear that many devices have already become "smart". Nevertheless, it is also apparent that some problems related to security remain unsolved.

While all of the applications discussed in Section 1.1, such as smart cities and smart offices, offer interesting opportunities, this thesis will focus on the smart home context. Many of the devices that find their way to the market are aimed at consumers, and estimates show that over two third of U.S. households expect to own some smart devices by 2019 [11]. Therefore, many IoT devices will be found in the houses of families, which often have little technical expertise. Part of these objects will be relatively immobile devices such as sensors, smart fridges and smart TVs, and some others may enter and leave the house from time to time, examples being smartphones and wearables. Hence, a smart home will feature a diverse set of objects with heterogeneous characteristics and capabilities.

More formally, we can attempt to identify some categories to classify our smart home devices. Some of the objects are *sensors*, measuring physical phenomena such as temperature and humidity, or detecting motion or presence. Other devices are *actuators*, creating a mechanical effect like opening and closing doors or windows. Additionally, *input devices* are part of the smart home, an example being the input panel for controlling heating. Complementary, *output devices* are responsible for presenting information to users by means of a display, speaker, … . More powerful *computation* may be available at some devices. On the other hand, some are aimed specifically at *(permanent) storage*, for instance network-connected hard disk drives. Many devices will be *compound devices*, which combine properties from several categories. An interesting class are the *user interaction devices* that integrate input and output functionality by presenting notifications to users and allowing them to issue commands to the system.

Using the above description, it is clear that many subsystems will become part of the smart home. Lighting control and climate control (air conditioning and heating) will be integrated. In addition, devices such as smart TVs, HiFi sound systems and smart

fridges will become connected. Furthermore, common household appliances like an oven and a toaster will offer their services to the smart home. Another example are small chips such as RFID tags in everyday items such as clothing. At present, objects like smart cars still seem futuristic, but they will also become part of the smart home and of the Internet of Things once the technology matures. User interaction devices are frequently used in present day society already. Examples are laptops, personal computers and smartphones.

At any point in time, a set of some of the aforementioned devices will be connected to the house's network. We will henceforth refer to the network with the connected objects at one moment as the *home sphere*.

At the edge of the home sphere, it is possible to place a *gateway node/hub*. Conceptually, this forms the boundary between the home sphere and the rest of the internet. Therefore, the gateway is the connection point between the internet and a certain perimeter, in this case the smart home. This is useful, as it places the gateway in a perfect position to guard the home sphere. It could be implemented on a router, or it could be a separate device which secures all traffic passing through the boundary router.

The introduction of a gateway to monitor devices obviously creates a degree of centralisation. However, as some devices are unable to provide adequate security themselves, and as a fully distributed Internet of Things creates different security requirements, some centralisation is preferable in certain circumstances [18]. The idea of a hub to provide this protection has been hinted at [19], and we believe it is an appropriate solution as many real-world settings exist where IoT objects are inside a perimeter for a considerable amount of time. Therefore, the security gateway will be the main subject of this thesis.

1.4 Goals and contributions

In this work, we want to contribute to the field of IoT security by providing a means to protect the devices within a certain perimeter, more specifically the aforementioned smart home. In order to achieve this, we will use a gateway node at the border of the home sphere. This hub is responsible for monitoring IoT traffic and interactions between the protection domain and the outside world (i.e. the internet). The **gateway** is assumed to be a device with sufficient computing power and memory, such that it is more powerful than the average smart home object. These ample resources enable the hub to perform complex security operations.

The devices deployed in the perimeter may be of **heterogeneous** nature, in order to be suitable for the IoT with its wide variety of things. This also implies that we want to support the most resource-constrained devices. The hub should offer an **interface to these IoT objects** for applications and developers. It is given the responsibility to **protect these objects**, including those that **may lack the proper abilities** to secure

themselves, for example missing support for long (secure) keys. We assume that the devices are able to move around freely within the trusted network, however, when they leave the perimeter, they are no longer being secured by the hub.

In order to achieve this protection, a first requirement is the ability to **express security and privacy policies**. This configuration should be practical to use for non-expert users.

Additionally, the **design of the gateway** itself, which **enforces the policies**, is an obvious and central requirement. It must provide adequate authentication and access control to secure the devices and their communications from attacks from the internet. Furthermore, it must ensure privacy by limiting data access, according to the expressed policy. The gateway must also **offer access to the protected devices' services**, by providing interfaces to control them. Hence, the gateway should allow users to set up secure communications between devices based on given rules. The aspect of coming up with rules which satisfy the desires of several users, for example finding a compromise between resident and service provider policies, is not considered in this thesis.

The contributions of this thesis are:

- Literature review, focusing on security and privacy issues in the Internet of Things and on proposed solutions (some specifically for WSNs)
- Description of use cases for a smart home setting and identification of security and other requirements for a protection mechanism
- A policy specification language, allowing expression of policies regarding security (access control, message security, privacy)
- The design of a gateway, which monitors all data streams and enforces policies expressed in the aforementioned language.
  - Interface to access resources' services
  - Control over data entering and leaving the smart home

1.5 Overview of thesis content

First, a literature review is given in the next chapter, providing more details on IoT issues and looking at some related work. Subsequently, Chapter 3 presents the context more formally, introduces use cases for the system, and provides a requirement analysis. The design of the gateway is explained thoroughly in Chapter 4, followed by a description of the policy language in Chapter 5. A prototype of the design has been implemented, which is discussed in Chapter 6. The evaluation and discussion of the system is found in Chapter 7, followed by the conclusion of this thesis in Chapter 8.

2 Literature review

2.1 Issues in the Internet of Things

Even though the Internet of Things paradigm is promising in the sense of the opportunities it offers, a number of challenges remains. We first list a number of general issues, as presented in [7] and [10], and focus specifically on security and privacy afterwards.

Because of the wide variety of available IoT devices, many different protocols exist. In general, standardisation still has not been tackled completely, although a number of efforts (e.g. 6LowPAN [20] and EPCGlobal [21]) has been made. The largest problems remain at higher levels of the networking stack, where there is a need for clear standards to integrate the services offered by objects from different manufacturers.

Furthermore, a range of networking issues remains. As a first sub problem, addressing and identification of devices needs to be handled. Therefore, it is necessary to map the identity of devices to (IP) addresses, for example to go from a RFID tag identity (64-96 bit) to 128 bit IPv6. Also, the ability to look up the devices based on characteristics or groups is necessary. To achieve this, a further evolution of the Object Name Server concept may be needed, which needs to provide a bidirectional mapping between an object description and an identity.
Additionally, the mobility of devices within the network should be supported, while respecting scalability and adaptability conditions.
Lastly, a new transport layer protocol may need to be developed. The TCP protocol is not properly fit for the Internet of Things, because of the traffic patterns which mostly consist of short packages which are not connection-oriented, and because of the congestion control and buffering mechanisms which are inappropriate for IoT objects.

A final problem is energy consumption. Some objects are able to use passive technology, meaning that they operate without a battery. More research is needed, however, as in a relative sense the energy consumption for transmitting information is very large for small devices [22].

2.1.1 Security and privacy

As exemplified in the introduction, there is a strong need for elaborate security and privacy mechanisms. In fact, according to [23], these issues form a major slowdown on the large-scale adoption of the IoT.

The first difficulty originates in the heterogeneity of the devices. Many of them are resource-constrained, making implementation of complex security operations impractical [16]. This heterogeneity also denotes that a varied set of security mechanisms is supported by the objects. Furthermore, many of these devices lack memory

protection [7]. In addition to performance, the usability should be taken into account when incorporating security and privacy measures [19]. Users must be able to enjoy protection without the need to become technical experts.

The increased scope of the internet also means that new threat models are likely to surface [6]. As an example, an attacker could be both internal and external at the same time, as the "perimeter" becomes fuzzy in an IoT context [18]. Furthermore, physical attacks may occur more often, as many objects are open and publicly accessible.

Cryptography is believed to be a crucial building block for security mechanisms [6], but the involvement of low-end devices creates the need for it to be lightweight. An architecture for key management is pivotal for cryptography, but it continues to be a problem in the IoT.

Furthermore, authentication and authorisation are needed for managing data and resource access. However, a comprehensive access control framework still needs to be established [24].

Other issues are in the field of trust and governance, notably the need for a mechanism to negotiate trust between parties [10]. Additionally, there is work left in the area of fault tolerance and accountability [5].

In addition to the above security requisites, privacy is a key requirement for the IoT. With many sensors collecting data and cheap storage available to persist it, it becomes more difficult for users to control their own data [5]. [7] adds that traditional solutions such as the Platform for Privacy Preferences (P3P) are inappropriate, because the user does not control all surrounding sensors which gather information about them. Hence, there clearly is a need for means to enable users to control their data. Furthermore, regulators may have to revise their privacy principles and create new legislation to protect civilians' data. This will require new technologies to support the enactment of laws and regulations [17].

In general, [16] recognises a standard security infrastructure as a remaining open issue, while [10] identifies the need for a general framework and enforcement mechanism for privacy in the Internet of Things.

2.2 Related work

This section will examine available security solutions and techniques for some of the aforementioned security and privacy challenges.

2.2.1 Cryptography & key management

The importance of encryption was already brought up as an important enabler for security in the previous section. A basic choice is between a symmetric and an asymmetric version of cryptography. Using the former is more convenient and resource-friendly, but has some problems with scalability and key management. The latter is more flexible and facilitates key distribution, but has a disadvantage when it comes to performance [16].

As resource constraints make implementation of suitable encryption in software very difficult, the possibility of performing it in hardware has been explored [25]. A comparison was made between a hardware implementation of the symmetric AES-128 to software versions of AES-128 (both a pure and an optimised variant) and some other symmetric ciphers on a CC2420 chip. The hardware encryption has a clear advantage in terms of execution speed, and a slight advantage in terms of memory consumption. The software versions' execution time was not excessive, but would be much greater if asymmetric algorithms had been used.

A significant requirement is the availability of a flexible and scalable key management system. An overview of techniques is given in [26]. It claims that public key cryptography is not usable for most IoT applications because of the resource consumption. It discusses various pre-shared key approaches, depending on the amount of keys at server and client side. Furthermore, it recognises mathematical key management systems as suitable techniques for the IoT, with better security properties, but more overhead than pre-shared keys.
Another classification is given by [27], which identifies the shared master key, pre-shared, and public key approaches. Again, the public key approach is disregarded because of performance issues. A shared master key should not be considered secure, because one compromised node has access to all communication.

On the other hand, the Sizzle security architecture design paper [28] argues that public key technologies are feasible if using Elliptic Curve Cryptography (ECC) variants. On low-end motes, this resulted in an execution time of about one second for the key negotiation phase. As this phase is not needed frequently, this is an acceptable overhead.

Pre-shared keys have been used considerably more often than public key approaches. One possibility is to use a gateway which has a pre-distributed key for each node in the network [29].
Another implementation was made for the PAKA system in the context of WSNs [30]. A gateway is used, but direct communication between nodes is possible. It uses a pre-shared approach, and builds up a path through other nodes if two nodes do not have a pre-shared key to communicate with each other. The PAKA system is secure against node and gateway capture, but lacks extensive testing on realistic systems and a clear test set-up description.

Other pre-shared key systems use a probabilistic approach, providing protection as long as a certain number of nodes has not been captured [27].
In general, pre-sharing offers better performance, but has limited scalability. For larger networks, either more pre-distributed key material is needed, which consumes resources, or a very powerful gateway is needed, which may not be realistic.

2.2.2 Confidentiality, authentication & access control

Confidentiality is achieved by encrypting message contents, disallowing anyone who does not have the key from reading it. Encryption techniques were already discussed in the previous section, hence we start immediately with some designs.

The Leap+ system [31] uses symmetric keys, resulting in an efficient way of ensuring message confidentiality. The disadvantage is that it is only useful for static environments. Furthermore, it is insecure if there are captured nodes during pairwise link establishment.
TinySec [32], and more recently MiniSec, included in TinyOS [33], provide confidentiality, but only at the link layer.
The Datagram Transport Layer Security (DTLS) [34] protocol provides TLS functionality for UDP connections, including encryption. It offers internet-level security for datagrams. It is possible to use DTLS to build an end-to-end security architecture [35] using the public key infrastructure, but this system has not been tested on the most low-end devices.
Sizzle similarly provides end-to-end encryption [28], with ECC for key establishment. A gateway is used, but the nodes themselves perform the encryption and key negotiation.
Overall, the encryption of the message contents is done using symmetric keys, which decreases the load on the device significantly. None of these systems, however, make optimal use of the gateway for decreasing the load on the constrained nodes.

Another open issue is authentication in the Internet of Things. Physically unclonable functions have been proposed as a way of generating unique secret keys for proving the identity on highly constrained devices. This uses hardware characteristics resulting from small manufacturing imprecisions to yield an unclonable way of calculating secrets [36]. This technique is, however, very low-level and difficult to use unless hardware constructors provide support for it.

At a higher level, a number of authentication mechanisms has already been designed. The aforementioned TinySEC provides link-layer authentication, and end-to-end solutions based on the public key architecture exist, too [28] [35].
The Leap+ system [31] is based on symmetric keys, and provides support for clusters (groups) to create a more flexible system.
Another authentication system using asymmetric keys (ECC) is presented in [37]. When a user attempts to establish a connection with a device, he is redirected to an authorative agent, which checks the user's identity and authenticates the IoT device,

and sets up a key between the user and the device. This paper did not present any performance characteristics for an implementation, though.
Another interesting scheme uses four-step authentication using a gateway [29], even though the user interacts directly with the device. It uses pre-distributed keys between the node and the gateway, and an extensive security analysis is given for the system. The performance, however, is only analysed in a theoretical way, and as the system was designed for a WSN context, it may not be appropriate for the IoT.

It is clear that a relatively large number of authentication mechanisms has been designed. Several of these make use of a more powerful gateway to free the devices themselves from this task. In the smart home context, and in many other IoT contexts in general, it is important to take the heterogeneity of the devices into account. Devices are likely to incorporate support for a certain number of specific mechanisms, potentially including some of those discussed in this section. Therefore, there is a need to provide interoperability between these devices and services, which has not been addressed for the IoT at present.

For authorisation in the Internet of Things, attribute-based access control is an often-explored alternative to classical approaches, which are difficult to maintain in a distributed setting. A recent performance analysis [38], however, shows that even for smartphone-class devices, the overhead becomes excessive when internet-level security is required. Hence, attribute-based authorisation is not feasible for the devices themselves.
Nonetheless, several designs have been made. An example design is fdac, which provides fine-grained access control by expressing rules as a tree access structure with AND, OR and NOT commands [39]. However, it has only been tested on relatively powerful objects, and has been designed for distributed settings, so it is not the most suitable solution for a smart home context.
Another design [40] was made specifically for the Internet of Things, but remains theoretical and lacks an implementation with performance data from a realistic setting.
As the Internet of Things involves a large number of users and applications interacting with a large number of devices, there is a need for rich access control expressions. Enforcing these is, however, infeasible for many of the objects involved.

2.2.3 Privacy

Some sources argue that privacy goals may interfere with security goals and that hence a balance should be established [19]. The *Privacy by design* principles [41], on the other hand, state that such false trade-offs should be avoided and that integration between the two ambitions is possible.

Several security mechanisms take privacy into account. The aforementioned attribute-based technologies allow anonymous authentication and authorisation, by proving that a user satisfies certain criteria without revealing the actual values [42]. This

has been integrated in existing systems, for example in the ABC4Trust project [43]. Some designs are aimed specifically at the Internet of Things, an example being ePass [44], but as discussed, the devices themselves are often not capable of performing these resource-intensive operations.

Whenever data has to leave the protected perimeter, we may want to ensure its privacy is respected at the external parties. Data tagging has been proposed as a way of enriching data streams [45], and it has been shown this allows the modelling of information flow control [46]. Adding 8 bits of metadata already yields significant semantic liberty, and allows the system to express the distinction between data that contains identifiable information and information without personal identifiers. The system is shown to be feasible on a low-end PIC device. Hence, this tagging approach is efficient, but it should be used properly at the service provider side. As it is based on a trusted computing base, the information itself has not been anonymised. Furthermore, there will be more overhead if checksums for the tags are added.

It is also possible to let users specify access control lists for their data. This is used in [47] to enable a user to specify that data is readable for specific friends or only for herself, via a website. For a smart home, such explicit access lists may be a feasible approach. However, it would be desirable to make such a system compatible with various existing access control techniques.

A final privacy technique is to anonymise the data itself. When providing *k-anonymity*, the data of one user is indistinguishable from the data of (k-1) others. An example is CASTLE [48], which anonymises dynamic stream data by clustering. While these techniques are useful, it would be most practical to execute them at the service provider end. The user side, for instance the smart house, does not have data from several users available to create clusters. We would therefore expect this technology to be used by service providers, which use it to anonymise data tagged to be personal as soon as possible.

2.2.4 Expressing policies

Several expressions for security preferences have been developed. Some of these escribe authentication and authorisation specifically, while others aim to give a description of systems and incorporate the ability to express (security) constraints for their components.

The security policy framework for WSNs, described in [49], focusses on expression authorisation of requests exchanged by nodes. It uses mathematical expressions for trust and criticality, allowing transactions to take place if their criticality is not higher than the amount of trust in the other node. The trust is based on proximity between nodes, based on the assumption that nodes which are physically nearby are more likely to be trustworthy. While this allows for rather convenient and easy-to-

compute authorisation expressions, no research into generalising them to allow different ways to characterise trust. Hence, the system lacks flexibility to make it viable in more universal network setups.

The aforementioned fdac allows extensive access control expressions using tree-structured arrangements of logical combinations [39], and attribute based systems such as ePass [44] similarly enable users a rich expressiveness.
ABC4Trust includes attribute-based policies [43] and provides policy expression for authentication and authorisation, described in XML format.
The eXtensible Access Control Mark-up Language [50] (XACML) also offers description of access control policies as an extension of XML.
While all of these offer abstractions for a subset of security preferences, integration into a complete set of abstractions for all protection needs is required. Furthermore, most of these have not been developed for the IoT context.

The Security Assertion Mark-up Language [51] (SAML) another extension of XML, aims to express security policies in general. While it was not designed for the IoT, it does attempt to allow a broader range of expressions.

Remora is a framework which promotes component-based development of sensor systems [52]. While it separates the description from the implementation and allows abstractions for dependencies, it does not consider security.

The Service Component Architecture [53] (SCA) was also designed for describing components, and includes a policy framework allowing the expression of some security constraints. It aims to be a universally-usable framework, so it is not IoT-specific.
The Architecture Analysis and Design Language [54] (AADL), on the other hand, was designed specifically for component description in embedded systems. It contains language extensions for data constraints, some linked to security.

3 Context

As described in the introduction, the setting this work focuses on is a smart home context. The home sphere contains the house's network and all objects connected to it at a specific point in time. Note that this network can actually consist of several networks in parallel, supporting different technologies such as WiFi and 802.15.4. Hence, there could be multiple access points. Nonetheless, the gateway node (hub) is located at the boundary between the home sphere and the internet. Figure 1 shows a graphical representation of the context.

Several entities interact with the home sphere. These actors of interest can be divided in three distinct categories. A first category is formed by *residents*: people who live in the smart home and who may access it from either inside or outside. Some of these residents may have more rights than others. As an example, consider people who have more rights in their own room or adults having greater privileges than children.
Other users are the *visitors*, which are physically present on the site of the house but are not permanent habitants. They too will want to make use of some of the smart house's services and they may bring their own devices into the house, which need to interface with the system.
The final category contains the *service providers*. These third parties use the IoT as part of the service they offer. A non–exhaustive list contains the police, energy company and weather forecast provider.

3.1 Use cases in the smart home

In order to illustrate typical behaviour of the smart home and interaction between users and the system, a number of scenarios involving the smart home is presented.

Interfacing with devices. When a resident wakes up and starts moving around the house, the system keeps track of her position as she crosses room or zone boundaries. Identifying the resident could be achieved by facial recognition or by tracking the location of a user identifier such as a smartphone or a RFID tag embedded in the resident's clothes. The sensors send this information to a domotics hub. This system uses this information to accommodate the room this person is in. In order to achieve this, the smart house loads the person's user preferences, which may be retrieved from the user's smartphone or which may have been previously stored inside the home sphere. Obviously, the system needs to be sure that the input is reliable, meaning that the identity has indeed been recognised and that the preferences are genuine. Based on these user preferences and taking into account other parameters such as the time of day, appropriate commands are sent to the climate control and the lighting system through the gateway. These systems need to be sure the commands are authentic and authorised, so as to avoid misuse. As an example, in case of a quarrel with the neighbours, they should not be able to control the heating to make the house uncomfortably cold.

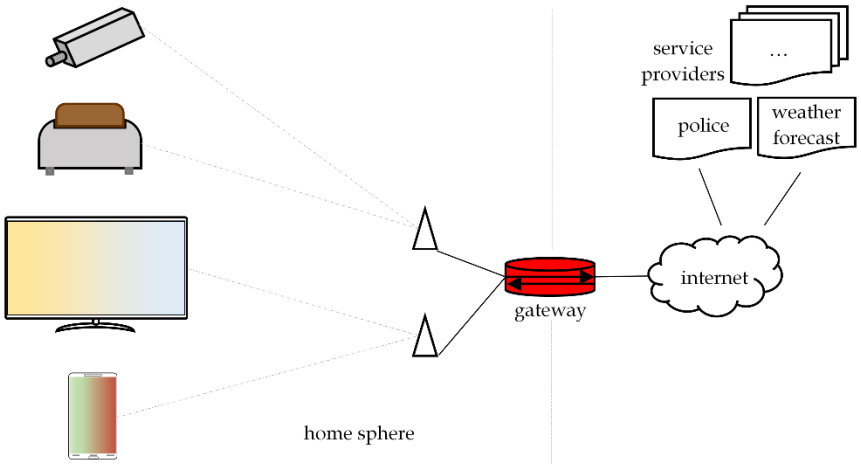


Figure 1: The smart home architecture

This discussion implies that the domotics system keeps track of the position of people inside the house. Complementary to accommodating the rooms when a resident or visitor enters, the system is able to undo the adjustments to a room when all people have left. The system sets the heating to a low-power mode and switches off the lighting. The fact that specific people are inside the house, let alone the exact locations of these people within the house, clearly implies that privacy-sensitive information is involved. Even though knowledge of the exact identity enables the system to offer personal services by using user preferences, the users will not want this information to leave the home sphere or may demand guarantees about confidentiality or anonymity if it does. They may, for instance, let the gateway define that service providers are disallowed to store the information longer than necessary or let it send them (pseudo-)anonymised data.

Service providers. At 10 AM every morning, a service provider, more specifically the energy company, interfaces with the smart home. Their system is scheduled to retrieve an overview of recent energy consumption in the investigated house. Upon receiving the request, the gateway checks its authenticity, after which the request is delivered to the smart meter. This is the conceptual subsystem monitoring energy flow and energy use inside the house. The smart meter composes an overview, which it sends back to the energy company. The latter expects this overview to be authentic: the readings must be genuine to avoid misuse by the user. Additionally, the residents will most likely prefer to keep this information confidential, meaning that no third party should be able to read the contents. Hence, the gateway ensures that the message travels the internet securely, even though it is not be able to modify the contents of the message itself.

Later that afternoon, the resident wants to go to the supermarket for some groceries. She requests a shopping list using her smartphone. The smart fridge composes the shopping list, and the system presents this shopping basket on the user's smartphone, along with pricing information retrieved from the store's catalogue via the gateway. When the user approves, the system sends this order to the store's online shopping system. In this order, a token is included allowing the supermarket to get the payment directly from the user's bank. Hence, this scenario involves two service providers, the supermarket and the bank, but direct communication only takes place with the former. Nonetheless, some measures need to be taken to ensure a correct (indirect) transaction with the latter provider. Therefore, it is important that the payment token is sufficiently secure. Measures are taken to safeguard the confidentiality of the message and to guarantee the integrity of the payment information is enforced. This should ensure that the correct amount of money is subtracted from the resident's balance and that it is transferred to the intended recipient. Furthermore, this transaction is allowed to be executed only once.

When the resident leaves the house to go to the supermarket, she turns on the alarm mode using the control panel. Note that as part of the smart house paradigm, the alarm system is unlikely to be a separate subsystem, but is instead an IoT application using the home sphere. A bit later, the sensors detect a breach: a window is opened, followed by motion detection in the hallway. The system will then compose a notification describing the observation, and the gateway will relay it to the alarm company's monitoring application. This notification needs to be validated by the alarm company, because they need to be certain the observation is coming from the rightful source so as to honour the contract with the house's residents and to avoid misuse. Therefore, the notification is authenticated before leaving the home sphere, even though the sensors may not be able to provide strong authentication themselves. When the observations are received, the alarm company will take appropriate action, such as notifying a local branch of the company. Furthermore, the alarm company's response may involve sending information back to the smart house, for instance sounding an alarm in an attempt to scare away the intruders.

Mobile access. Later on, the resident returns home after picking up the groceries and wants to turn on her Dutch oven such that it is preheated by the time she arrives. She uses her smartphone to access the home sphere remotely, and issues the command to the system. The gateway verifies the user's identity, after which it forwards the command to the oven. A new element in this interaction is the communication by a resident from outside the house's network. Again, an identity integrity check is a critical step, as it prevents safety issues with devices like ovens.

Introducing new devices. In addition to the groceries, the resident has brought home a compact smart TV for her kitchen. After finishing supper, she installs the new device. As part of the configuration, she uses her smartphone to instruct the gateway to allow the TV to become part of the home sphere. After doing so, the gateway is set to handle the device's network traffic. The TV registers its services

and sets up a connection with external service providers, such as the TV manufacturer's updating system. The resident wants this configuration process to be as straightforward and as short as possible.

Visitors. In the afternoon, the resident invites a friend over to the house. When the visitor enters the house's perimeter, her devices will request access to the system by identifying themselves. Afterwards the gateway may decide to grant them access to the home sphere or parts thereof, or it may deny the request. To make this decision, it validates the visitor's objects, and identifies the smart house such that these things also have assurances about the authenticity of the home. After the successful registration of the visitor, she will be able to use some of the house's smart devices, depending on the residents' preferences. Similar to the scenarios described above, the system will track the visitor's location and may use the visitor's user preferences to accommodate some characteristics of the rooms.

When the visitor leaves the house, she may explicitly unregister or the hub may simply detect that the visitor's devices are no longer part of the network. Upon this logout, some of the cached information about this user is deleted. At this point, the visitor is no longer authorised to use any of the smart home functionalities.

The above scenarios exemplified some key behaviour of the smart house system. They showed interactions between a resident and the system, both from inside and outside the house's network. Others featured exchanges which were not instantiated directly by the user, but instead only involved communication between systems. Furthermore, exchanges with several service providers were illustrated, some of them including sensitive information. The next section will use this setting to extract the gateway's requirements. Finally, the visitor role played a part in showing dealings with foreign devices.

3.2 Requirements analysis

The use cases in the previous section reveal a set of important requirements. The ensuing paragraphs will look at the functional requirements, including the security and privacy needs, and a set of non-functional demands. Afterwards, the threat model is described.

3.2.1 Functional requirements

A first functional requirement that is present in all of the use cases, it that the gateway should be an *interface for the IoT objects* within the perimeter. Since it stands in front of the things from the point-of-view of the internet, it should offer an API that gives access to the home sphere for applications and services.

The remaining and most prevalent portion of the functional requirements is associated with security and privacy. It should be possible to express these demands, and a system is needed which enforces them.

As a first security observation, it is clear that *authentication* is of paramount importance. This applies to communication within the home sphere, such as sensor readings and commands for actuators and other subsystems. It is also necessary in dealings with devices and systems outside of the home sphere, for example when a service provider or mobile access is involved. Furthermore, this need arises when identifying foreign and new devices.
In some cases, for instance when using the supermarket's shopping system, two-way authentication is required.
Because of the heterogeneity of the objects in the home sphere, the authentication mechanisms they support will also be heterogeneous. The gateway must be able to operate with all of these diverse methods.

In addition to authentication, this credential information should be used to authorise the requests. Resources in the home sphere need *access control* policies to ensure correct operation upon receiving a message with authentication.

Another requirement is *confidentiality*, for example for payment information or energy consumption details. Note that some messages may not demand confidentiality, for instance, the value of a temperature reading may not be very secretive. Nonetheless, for many messages confidentiality is desirable, meaning that encryption and decryption must be supported. However, the resource constraints of some of the devices should be taken into account.

The former requisites indicate that the gateway should support authentication, authorisation and confidentiality. Sometimes, it may need to provide these functionalities on behalf of low-end devices, or it may have to ensure more powerful security for messages travelling the internet. These requisites also imply that the gateway should provide key management.

To ensure *privacy*, some tools are needed in addition to confidentiality: the *scope* of some of the data needs to be limited to the house, and information leaving the house should be protected. The latter could be achieved by including *privacy meta-data*, describing how the data may be processed and stored, or by *anonymising* or pseudo-anonymising it.

As mentioned before, all of these security demands need to be specified if they are to be enforced as expected. There will be certain preferences with regards to security, which the user must be able to specify in a *policy language*. The gateway then uses these specifications as directives for protecting the data and devices in the home sphere.

3.2.2 Non-functional requirements

Apart from the functional requirements, some other demands are identified, which are side conditions that should not be overlooked. First off, chief requisites are generality, communication and transparency. This implies that the hub needs to be compatible with a variety of devices, networks, services and protocols. This was already discussed briefly in the context of offering an interface to heterogeneous devices. Furthermore, it needs to be flexible by supporting the dynamicity of a network where objects may enter and leave. There should also be support for mobility of users and remote access by their user interaction devices.

Another requirement is performance: the impact of the hub on the services should be minimised. Furthermore, some scalability is required, although an upper limit on the amount of objects may be assumed, as infinite scalability is not necessary in the context of a smart home.

A final necessity is user friendliness: configuration and use of the IoT should be straightforward for users in a domestic context, often without technical background. This means addition and removal of devices and defining policies must not be complex.

3.2.3 Threat model

As the main design goal of the gateway is to offer security, it is important to know against which attacks to protect and what the adversaries are capable of. Therefore, this subsection outlines the threat model, showing the types and capabilities of assumed attackers.

The first threat considered is an internet-based, or *external attacker*: an outside attacker that attacks the home sphere from the internet. This attacker is either *passive* or *active*. The former is only able to capture messages which leave the home sphere and to read their contents if they are insufficiently protected. This is the case if the messages are transmitted in plaintext form or by breaking insecure encryption, for example by executing a brute-force attack for messages with a short-length key. The goal of the attacker is to gain unauthorised access to information.
On the other hand, in an active attack, the adversary modifies the contents of the messages and sends these altered versions to the destination. Alternatively, she may create new or duplicate messages or prevent the messages from reaching its destination. This kind of attacker's goals are to install false information at one of the parties involved, or to gain control over or to harm their systems or operations. An alternative method of disrupting the normal functioning of the home sphere is to carry out a denial-of-service (DoS) attack. This type of attack is not considered in this thesis, however, as other solutions (i.e. firewalls) are available to protect the whole network domain.

A second kind of threat originates from the service providers. Some of them may be classified as *curious but honest*, trying to gain more information about the smart home and the users than intended, while staying within the limits set for them. This especially impacts the privacy of the users. Other service providers may perform requests and commands to smart home devices which were not specified in their service contract and in the policies they declared to the gateway. In this case, the service provider is considered to be malicious or compromised.
Conversely, the identity providers (certificate authorities) are considered to be completely trustworthy.

A final set of attacks are *inside* attacks, executed from within the home sphere perimeter. This attack could be performed by a malicious *visitor* entity. The person herself should not necessarily be the source of the attack: one of her devices could be compromised and responsible. In general, the goals of this attacker will be similar to those of the outside attacker: obtain, control, alter or harm data and resources. The difference is that this assailant may have access to some more information or resources, which are not available to an outside attacker, as they are inside the network. Examples are key material on the attacker's node, or access to the private home network, allowing it to easily read messages that are open to all nodes. Specifically, an inside attacker may try to execute *elevation of privilege*, where she attempts to bypass restrictions and policy enforcement to obtain more rights.
Another attack source for executing inside attacks is formed by compromised objects in the home sphere itself. The goals and attack vectors are similar to those of the dishonest visitor.

3.2.4 Summary

Based on sample use cases, we have identified functional and non-functional requirements for the gateway. Furthermore the threat model has been drafted, showing the attacker's side of the context. The following list briefly summarises the identified requisites:

- Offer an interface giving access to objects inside the home sphere
- Authentication: Managing identity information, providing authenticity on behalf of devices, checking authenticity of foreign devices and services, two-way authentication
- Authorisation: Enforcing access control policies to control device access
- Confidentiality: Encryption and decryption using keys of sufficient key length, key management
- Privacy: Limiting data scope, protect data leaving the home sphere by adding privacy meta-data or anonymising it
- The ability to express these security and privacy preferences in policies, which are directives for the gateway
- Communication, generality, transparency and flexibility: Support for a wide range of devices

- Performance and (limited) scalability: Avoid excessive overhead while providing protection
- User friendliness: Keep configuration simple, avoid unnecessary complexity for users

4 Gateway design

This chapter discusses the design of the gateway developed to secure the home sphere by enforcing policies for authentication, authorisation, confidentiality and privacy. The policies themselves are explained in more detail in the next chapter.

First, some design concepts are clarified, including an explanation of the entities interacting with the gateway and the resulting interaction model. Then, an overview of the architecture is given, followed by a more detailed discussion of the component responsible for policy enforcement. Section 4.4 discusses mapping of abstract addresses to specific devices, followed by a description of the key management and authentication in Sections 4.5 and 4.6, respectively. Sections 4.7 and 4.8 describe session set-up and regular message handling, and Section 4.9 describes miscellaneous gateway behaviour. Finally, Section 4.10 describes the interface offered by the gateway, by means of describing the types of messages it handles.

4.1 Design concepts

As Figure 1 showed previously, the gateway stands between the outside network (internet) and the inside home sphere network. Therefore, messages entering and leaving the home sphere pass through the gateway. Furthermore, communication between home sphere devices also travels through the gateway, such that it becomes the central point for checking security.

4.1.1 Resources and Consumers

Conceptually, a distinction is made between two entity types: Resources and Consumers. A **Resource** is something which offers functionality, while a **Consumer** makes use of this functionality. Hence, messages containing commands and requests will be sent by Consumers to Resources, and Resources may send replies and information to Consumers. The next paragraphs will explain these entity types in more detail.

A Resource provides functionality. Hence, it could be either a device along with all of its services, a specific service of a specific device, a service regardless of the device offering it, or a group of Resources:

```
Resource = device OR device.service OR service
     OR resourceGroup
```

Examples of these are, respectively:

EnergyMeter, EnergyMeter.usageinfo, getTemperature, HeatingUnits

Resources all revolve around functionality offered by devices. Formally, a **device as a resource** is described with these attributes:

```
{name,address,internal,listOfServices}
```

Here, the *address* refers to the network address, and *internal* is a flag indicating whether or not this resource is found inside the home sphere.
When a service is called as a resource, it will need to be mapped to a specific device providing it. This is explained in Section 4.4 of this chapter.

A **Consumer**, on the other hand, is something or someone that makes use of a Resource. Hence, this could be a device, a user or a group of Consumers:

`Consumer = device OR user OR consumerGroup`

Formally, a **device as a consumer** contains the following attributes:

`{name,address,internal,authenticationDetails}`

Again, a network *address* and *internal* flag are included, complemented with *authenticationDetails*. These details describe how the device should authenticate towards the system. The various authentication mechanisms are described in Section 4.6.
A **user** symbolises a real-world actor, and is actually a group of devices which the actor owns.

As was shown above, the basic building blocks for both Consumers and Resources are **devices**. A device is a physical entity located inside or outside of the home sphere. For the gateway and for this thesis, a device has an associated Resource and Consumer identity, or one of them if it only takes one role. This gives a clear conceptual model of messages exchanged between a Consumer and a Resource. In the next chapter, policies will also be defined on basis of these concepts.

The result of this distinction is that each device will be a Resource, a Consumer, or both. It is a Resource when it is accessed and when its services are used, and a Consumer when it uses the services of other devices. Both roles are combined when a device both offers and uses services.

One basic observation that deserves specific attention is that Resources are found both inside and outside of the home sphere, and so are Consumers. The gateway will deal with inside Resources used by outside Consumers, outside Resources used by inside Consumers, and inside Resources used by inside Consumers. If both Resource and Consumers are located outside, there obviously will not be any enforcement by the gateway involved.

4.1.2 The message passing paradigm and sessions

The gateway is designed in a *message passing* model fashion. This means that messages are exchanged between the entities and that they are being intermediately processed by the gateway. This model stands in contrast to a (remote) method invocation view. The message passing model was chosen because designing the protocols based on message exchanges stands closer to reality, and will hence result in a thinner middleware layer at the devices. Furthermore, interaction via messages

results in less dependency on the programming language used for the implementation.

One way to use the gateway would be to make it completely transparent from a Consumer and Resource point-of-view, such that they need not be aware that a gateway is present. The advantage of this strategy is that no middleware is required at the devices. Unfortunately, this would make it difficult for the gateway to achieve its security goals. The gateway would not be involved in session set-up, which means that the raw bytes the gateway sees in messages are virtually meaningless. Furthermore, without sharing some secrets with the devices involved, it would not be possible to interpret most of the messages. A non-transparent design makes it possible to specify different security requirements for communication between each entity and the gateway. Hence, it allows the gateway to enforce stronger security demands for messages travelling the internet, and more lightweight security for constrained devices inside the home sphere. Another advantage of a non-transparent design is that Consumers may specify a request for a service, which the gateway maps to a specific device supplying it.

Therefore, the design assumes that the devices are aware of the presence of the gateway. First off, a protocol has been designed which involves the Consumer requesting a session with a Resource. This message is sent to the gateway, which will deny or grant the request. In case of a granted request, the gateway will supply session information, such as security parameters and session keys, and an *abstract identity*. From there on, the Consumer will send messages including the abstract identifier to the gateway, which will forward them to the Resource after enforcing security. The session may be ended by a specific request, or it may be automatically removed after some time or after some amount of idle time. The session set-up protocol and regular message handling are explained in Sections 4.7 and 4.8, respectively.

This message handling interaction model was created based on separate sequences made for a message from some user (resident, visitor or service provider) to a home sphere device, from a home sphere device to some user, and from one device to another. It became clear that these were all very similar, and could be abstracted as done in the current design, including the Consumer and Resource concepts. Hence, this message handling model is generic and applicable to use cases such as the alarm company requesting a presence reading from a motion sensor, a resident requesting a list of items in the smart fridge, a resident remotely switching on the oven from outside of the home, a presence detection device sending a reading to the climate control system, and potential responses in all of these cases.

4.2 Architecture overview

The high-level design of the gateway (Figure 2) consists of three front-ends and seven internal modules.

The `Internet FrontEnd` and `Home Sphere FrontEnd` are the front-ends connecting the gateway to the outside and inside network, respectively. They offer an interface for accessing devices by providing a proxy for their services and device interfaces. They allow setting up a session with entities inside and outside the smart home, and sending them messages, such as commands from service providers or other devices. Hence, they are the main entry point for regular gateway traffic.

The `Config FrontEnd`, on the other hand, allows users to make changes to the gateway's configuration, such as adding devices and users and changing access policies.

The `Network Manager` manages the front-ends for the internal and external networks. It maps between IP addresses and more abstract entity representations used internally by the gateway.

The most important internal component for providing the functional security requirements is the `Enforcement` module. It fulfils these requirements by *enforcing the policies* for a given connection between the entity and the gateway or between the entity and the other entity. It is also responsible for selecting an appropriate message security mechanism for a session, based on the policies, and creating matching session key material. As this `Enforcement` provides many different functionalities and is composed of several submodules, it is discussed in more detail in section 4.3.

The `Message Handler` is the gateway's central unit for processing messages, either requests or replies, from devices. This decision is based on policies for the associated devices and session information for that specific connection. This information will be gathered by using the `Message Handling Information Retriever`. It will use the `Enforcement` module for *imposing security demands* whenever applicable. It is also responsible for *mapping* abstract service requests to explicit devices, which is discussed in more detail below (Section 4.4). Furthermore, it transforms serialised sequences of bytes by extracting the metadata intended for the gateway from the actual payload, and allows the reverse transformation where internal gateway messages are serialised.

As the name implies, the `Identity Manager` contains a register of information regarding identities of all entities. This includes *key material*, such as public keys of service providers and compatible devices, and shared symmetric keys of other devices and device clusters. It also includes the gateway's private keys for use in some interactions. Furthermore, all devices and their Consumer and/or Resource identities are registered here. This means that authentication details for Consumers

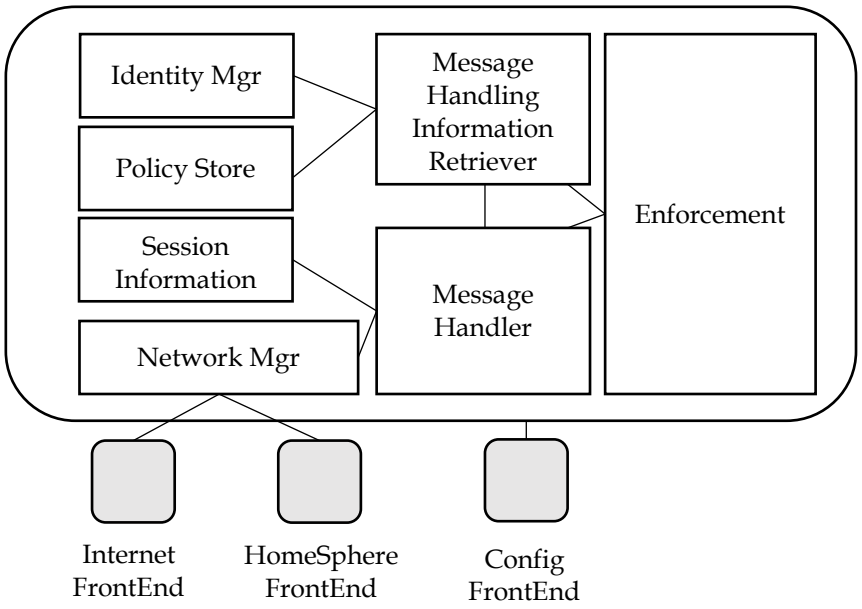


Figure 2: The gateway architecture

are available here. The composition of groups is also found at the `Identity Manager`, as is information about the functionality and services the devices offer.

Management of policies is handled by the `Policy Store`. It contains a *register of policies* for home sphere devices and users. The `Policy Store` offers an interface to enable the `Message Handling Information Retriever` to *retrieve the matching policies* when given the entities involved. Furthermore, residents configuring the policies are able to inspect, add, modify and remove policies.

The `Session Information` module provides temporary storage of information pertaining to a specific active connection between two devices. Hence, it allows the gateway to keep *state* information, which may be used to correctly and efficiently handle messages. For example, when mapping an abstract service to a device, it may store the fact that service provider A is using device B_1, such that subsequent messages are not relayed to other devices B_x offering the same service. Another example is storage of the result of an approved stateless access control check upon receiving a service provider request, such that this check is not performed again when a response is returned. It also contains the temporary key material for that session.

4.3 The Enforcement Module

This section details the further decomposition of the `Enforcement` module introduced in the overview of the architecture. The decomposition is depicted in Figure 3.

The `Enforcement Module Manager` will receive message payloads and the associated policies. It has a register of available enforcement modules and their capabilities, allowing it to call the appropriate modules based on the policies.

A first subsystem of enforcement modules is `Authentication`. As the gateway aims to support several mechanisms for authentication of entities, it will, in turn, contain submodules implementing different techniques. Hence, the gateway design allows inclusion of username/password credentials, anonymous attribute-based authentication and others. In Section 4.6, authentication is explained in more detail.

Another subsystem is `Authorisation`. As the name implies, it verifies whether Consumers have access to the Resource they are trying to reach. As this module may do fine-grained access checks, it is in part responsible for fulfilling the privacy requirements. Based on the type of service requested, for example a request for a presence reading, a request may be identified as a violation of privacy policies, which should be blocked by the gateway.

The `Message Security` submodule provides security of messages (payloads). Hence, based on the message security type, it may provide confidentiality, authenticity and integrity, or a subset thereof.

The `Cryptography` module offers symmetric and asymmetric encryption schemes to the higher-level modules described above. Hence, it does not provide policy enforcement by itself, but instead offers its services to other enforcement components, such as encryption, decryption and message authentication codes.

The `Privacy` module is a component where further privacy-related functionality is located. While privacy is also a part of other modules, such as using anonymous authentication and fine-grained authorisation, this module is responsible for anonymising or pseudo-anonymising data. This is achieved by removing certain elements of the data based on their sensitivity. Adding tags to the data tuples as metadata is one way of indicating that data or parts thereof are sensitive or personal. Alternatively, this can be deduced from the type, service or device. Based on this information, part of the data can be filtered out or generalised. It is also responsible for applying *sticky policies*, where the metadata concept is extended beyond the home sphere. In this case, the gateway enriches the data which leaves the smart home with tags. These are then expected to be used by the service provider's trusted computing base to appropriately handle the data.

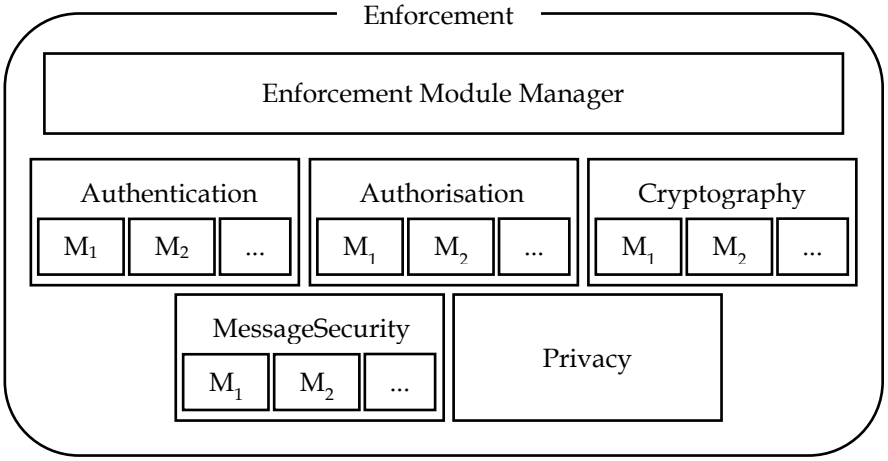


Figure 3: The Enforcement Component

4.4 Entity name and service mapping

As part of offering an interface towards the smart home devices, the gateway has to relay messages to the appropriate device after enforcing security policies.

In the simplest case, a request is addressed to a device as a resource DEV_A, or to a specific service thereof $DEV_A.SRV_W$. An example would be a request to `EnergyMeter`. As we are using a non-transparent model, the Consumer sends the name of the requested entity to the gateway, which identifies the appropriate device. This also allows us to use several aliases for the same device, for use by different Consumers. The mapping between the entity name(s) and the associated Resource is available from the `Identity Manager`.

Another possibility is a request addressed to a Resource group. In this case, the `Identity Manager` is aware of the group composition. Authorisation is based on the policies of all group members. When messages are exchanged in a session, the gateway will aggregate responses of all members before relaying the response to the Consumers.

Finally, requests may be addressed to a service. A realistic example is an internal request from a resource-constrained device to the `Compute` service for performing more resource-intensive computational tasks. To satisfy this type of request, the gateway should find some device offering this requested functionality. This `Identity Manager` is able to supply this information. The Consumer need not know the exact device, as it sends messages to an abstract identifier anyway. The gateway, however, needs to store the mapping with the selected specific device in the `Session Information` module.

Similarly, the Resource will send its messages to an abstract identity which the gateway associates with the Consumer.

As can be seen from the above discussion, the service mapping may play a role in providing privacy. This is possible because it allows the gateway to hide the actual device itself behind a proxy name or service.

4.5 Key management

As the gateway aims to support various security mechanisms, it will need to handle several types of keys. This section overviews the supported keys and how they are managed. Key management is tightly linked to the `Identity Manager`, which manages device and user identities and stores their key material, to the `Session Information`, which stores session keys, and to the `Enforcement` module which will require the material, for example to check authentication.

A first important category is the *session set-up key material* which is present for a longer period of time and which, as the name implies, is used for setting up sessions. Within a session, different *session keys* are used.

The first supported method for long-term keys is the **public key infrastructure**, where the gateway holds the public key of the devices using this method. Adding a new device supporting this method to the gateway may be done by providing a certificate (for example, a X.509 certificate). The gateway validates this using a known and trusted Certificate Authority (CA), possibly traversing a chain of issuers. Afterwards, trust is established, and the gateway has a validated public key for this device, which is stored in the `Identity Manager`. Similarly, this will be the main key technique used for identification in interactions with service providers.

The gateway itself has a series of private keys and associated public keys. These may be created and the public key distributed upon registration of a service provider or device. This way, the gateway could have an asymmetric key pair for each service provider.

The gateway also supports **shared-secret** methods. This could be a secret key which is shared with a device. Both unique keys, associated with one device, and group keys are possible. Another secret that is shared with the other party is a password. Username/password schemes are supported for authentication, and are explained in the next section.

Whenever a session is established, a temporary shared secret is generated for use during that session. This session key is distributed by encrypting it with the long-term key material. In the case of a shared symmetric key, this results in straightforward symmetric encryption. When an asymmetric key pair is involved, it is

encrypted using the other party's public key such that this party is able to recover the key by decrypting with its private key.

The set-up of a shared session secret will always be used to minimise the amount of asymmetric key operations. This session secret could be a key for encryption or for a message authentication algorithm, if authentication without confidentiality is required.

4.5.1 Applicability

Connections between the gateway and outside parties can be interactions with either service providers or residents in a remote session. For service providers, the public key infrastructure is used as much as possible. This creates a flexible and scalable system for external interaction. Remote sessions can likewise use public keys to identify the remote device. Alternatively, a shared secret could be used, if the user registered it to the gateway beforehand. Visitors are very similar to remotely-logged residents. They will also use public keys as much as possible, but a shared secret is also possible.

Devices inside of the home sphere will use either public key infrastructure or a unique shared secret. The former is possible when the device supports it, while the latter is more suited for low-end devices and will therefore be used more frequently. This does mean that upon adding the device to the gateway, it will be necessary to specify the key.

4.5.2 Device groups

Groups allow direct communication between sets of devices and hence do not require all their messages to pass through the gateway. Obviously, there is a need for restrictions (policies) for the creation of such a group. Upon group creation, the gateway will need to distribute key material to all parties.

A first method is to use the gateway for the establishment of the key and the consequent distribution.

The following notation is used for the pseudo-code:

```
secret_A_B: a secret shared between party A and party B
puA:        a public key of party A
prA:        a private key of party A
H(m,k):     hash of message m, using key k
E(m,k):     encrypt message m, using key k
D(m',k):    decrypt encrypted message m', using key k
```

In this fragment of pseudo-code, device A requests group creation. The gateway generates a secret which will be shared between the group members. It sends this secret and the `groupinfo` (such as the list of members) to the newly formed group.

```
DevA -> GW: request group([list of group members DevX])

GW:         generate secret_G1
            for all DevX with symmetric key:
                  send newgroup(E(secret_G1,secret_GW-X),
                        groupinfo)
            for all DevX with asymmetric key:
                  send newgroup(E(secret_G1,puX),
                        groupinfo)
```

In case any of the group members chooses not to participate in the group, it can simply discard the group information it receives. An alternative would be to use a confirmation step where the device indicates this explicitly.

Another method for generating the secret is to allow the device A to generate the group secret itself. In this case, the device sends it to the gateway in encrypted form, which simply decrypts it instead of generating a secret itself. The rest of the code remains the same.
In both scenarios, the gateway has access to the group key and is ultimately responsible for approving the group. This leaves the gateway in control to apply restrictions.

4.6 Authentication

4.6.1 Challenge/Response (handshake) methods

Challenge/response methods are used by the gateway for establishing authenticity. Hence, they are executed when setting up a session, and are the responsibility of the `Authentication` enforcement module. This allows the gateway to support a username/password scheme (shared secret), an authentication mechanism based on public keys, and anonymous authentication.

When a **shared secret**, such as a username/password combination or an access token, is used, the following pseudo-code shows how `party 2` is authenticated:

```
Party 1:    generate random x
            send x to Party 2

Party 2:    xd = H(x,secret_1-2)
            send xd to Party 1

Party 1:    check xd by computing H(x,secret_1-2)
```

In this case, the first party (e.g. the gateway) wants the second party (e.g. a resident, visitor, …) to authenticate. It generates a random value (nonce), which the second party hashes using the shared secret. As the secret is shared, the first party simply repeats this operation to verify the identity.
When **mutual** authentication using a **shared secret** is required, this is algorithm is extended, resulting in the code below:

```
Party 1:    generate random x
            send x to Party 2

Party 2:    x' = H(x,secret_1-2)
            generate random y
            send x',y to Party 1

Party 1:    check x' by computing H(x,secret_1-2)     (A2)
            y' = H(y,secret_1-2)
            send y' to Party 2

Party 2:    check y' by computing H(y,secret_1-2)     (A1)
```

The code is very similar to the code for one-way authentication. The difference lies in the second party generating a random value itself, such that the first party also executes the step of hashing it with the shared secret. This means that `party 2` and `party 1` are authenticated at points `A2` and `A1`, respectively.

It is also possible to authenticate using a mechanism based on **public keys**. The protocol starts by a random value `x` being generated by `party 1` (typically the gateway) and encrypted using `party 2`'s public key. The second party is able to decrypt this nonce, and encrypts it again using the other party's public key. When `party 1` receives this message, it is able to verify the authenticity of `party 2` by decrypting `x''` and checking it contains the random `x` it originally sent. The code below shows this protocol used twice to provide **mutual** authentication:
Again, `A2` and `A1` mark the points where, respectively, `party 2` and `party 1` are authenticated.

```
Party 1:    generate random x
            x' = E(x,pu2)
            send x' to Party 2

Party 2:    x = D(x',pr2)
            x'' = E(x,pu1)
            generate random y
            y' = E(y,pu1)
            send x'',y' to Party 1

Party 1:    check x'' by computing D(x'',pr1)         (A2)
            y = D(y',pr1)
            y'' = E(y,pu2)
            send y'' to Party 2

Party 2:    check y'' by computing D(y'',pr2)         (A1)
```

If only one-way authentication is required, steps related to `y` are simply left out and the identification is finished at point `A2`.

Another alternative approach is **anonymous authentication**, which allows authentication based on some attributes one party possesses, without revealing their true values to the other party. This can be used for communication with service providers supporting it, and offers more privacy than traditional authentication methods.

As for the applicability of these authentication methods for service providers, the main methods used will be the public key infrastructure method and anonymous authentication. The latter is used to authenticate the smart home or parts thereof to the service providers, whenever possible and required. The opposite direction, where a service provider could anonymously authenticate to the gateway, is considered less important and useful and will hence not be supported.

Remote sessions and visitors use public key authentication when possible, or may use a shared secret scheme such as a username/password combination. In case of a visitor, it is natural to imagine him receiving an access token from a resident.

For devices inside the home sphere, as with the key management, public keys or unique shared secrets may be used. Alternatively, it may suffice to have **network authentication**. This basically means that the gateway will do no specific authentication checks, but authenticity is based upon the network identity of the device inside the home network.

4.6.2 Authentication within sessions

After initially establishing authentication with one of the protocols described above, it will still be necessary to provide authentication within a session.

If the message security policies dictate that confidentiality is a requirement, the system uses **authenticated encryption** for subsequent messages exchanged with an entity. This provides authentication, integrity and confidentiality. The messages are secured using an "**encrypt, then authenticate"** scheme.

If message security indicates a need for authentication and integrity without confidentiality, a **message authentication code** is computed and attached to each message.

In both cases, a shared session key will be required. This is generated and distributed during session set-up.

4.7 Session set-up

This section will explain the session set-up protocol more closely. First, the protocol is described as viewed externally, including interactions with the Consumer and Resource and a high-level view of the gateway's behaviour. Then, an internal perspective explains how the components from Section 4.2 are involved in this protocol.

The high-level view of the protocol is shown in Figure 4.The Consumer will start by sending a session set-up request to the gateway, in which the requested Resource is specified. This could be a specific device or could be more abstract, as explained in the 'Entity name and service mapping' section. Hence, the message `(src,dest,data)` will look like `(Consumer,GW,RequestedResource)`, as the message itself is addressed to the gateway. The source Consumer could be outside of the home sphere, belonging to a service provider or a resident in a remote session. It could also be located inside of the home sphere, including a user interaction device of a resident or visitor.

The first step will be to authenticate the Consumer, according to its authentication details. Various mechanisms are possible, as explained in Section 4.6.1, and the mechanism may influence the amount of messages exchanged in this step. For example, mutual authentication would require supplementary communication in this step. Afterwards, the abstract requested resource is mapped to specific devices by name resolution and service mapping (Section 4.4) and the applicable policies are retrieved.

Subsequently, some enforcement steps are executed. This includes an authorisation check, and some privacy checks may also need to be carried out. When all of these are satisfactory, session material is set up. The required message security mechanisms are selected, based on the policies. This security mechanism need not be the same for communication with all of the devices. Furthermore, it is possible to have different mechanisms for both directions of communication. According to these requirements, session keys are generated for all of the devices involved. Furthermore, the gateway creates abstract identifiers for the Consumer and Resource to address their communication partner.

In the next step, a session is set up with all of the specific Resources. The gateway will distribute the session parameters, such as the abstract identifier, message security parameters and key material. As a result, the Resource could request authentication, and the gateway may want to carry out mutual authentication.

Finally, the gateway stores the session material and distributes the Consumer's share to her in a *session granted* message. If a failure occurs at any point during this protocol, for example because of a failed authorisation check, the gateway will send a *session failed* message, potentially including a problem description.

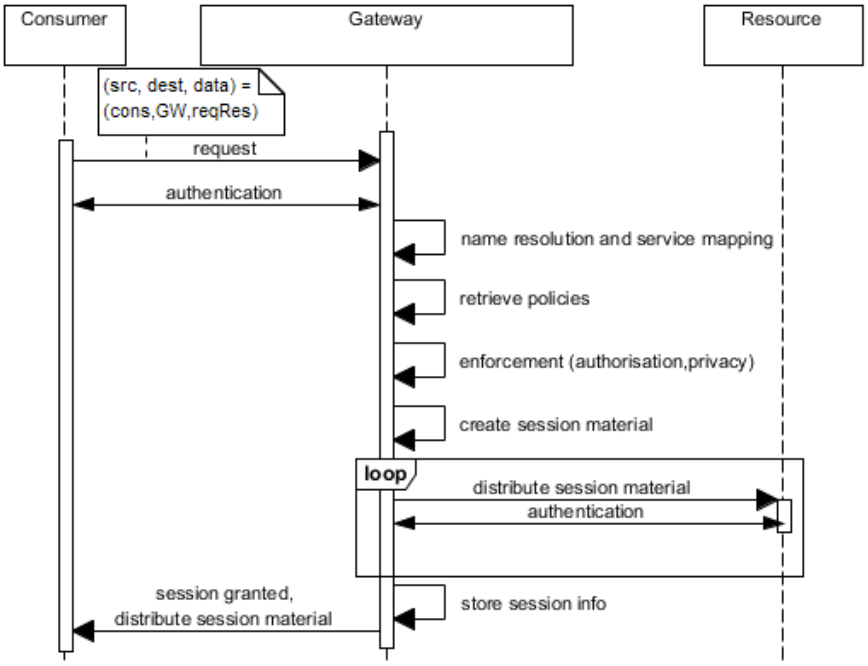


Figure 4: Session set-up protocol, showing a successful execution

The internal handling of session set-up messages is shown in Figure 5 and Figure 6. For readability, some result communication is not depicted separately. Messages arrive at the `Network Manager`, which identifies the Consumer identity of the sender and passes the message to the `Message Handler`. The authentication details are retrieved from the `Identity Manager` (step 2). Authentication itself is done in step 3. The `Enforcement Module Manager` will select the appropriate enforcement module based on the authentication details. In steps 3b through 3e, the Consumer receives a challenge and sends a reply. The proof is verified by the appropriate enforcement module. Steps 3g and 3h are optional, and allow the gateway to authenticate itself.

In the next steps, name resolution and service mapping is done using the `Identity Manager`, and policies are retrieved from the `Policy Store`. The `Identity Manager` needs to be used to check what groups a device belongs to, such that policies specified for a group containing the device are also selected. After step 7, the gateway has a list of all specific Resources in this session and a list of policies that may need to be enforced.

Enforcement checks such as authorisation and privacy are done in step 8. In step 9, the `Enforcement` component is called again, this time for selecting a message

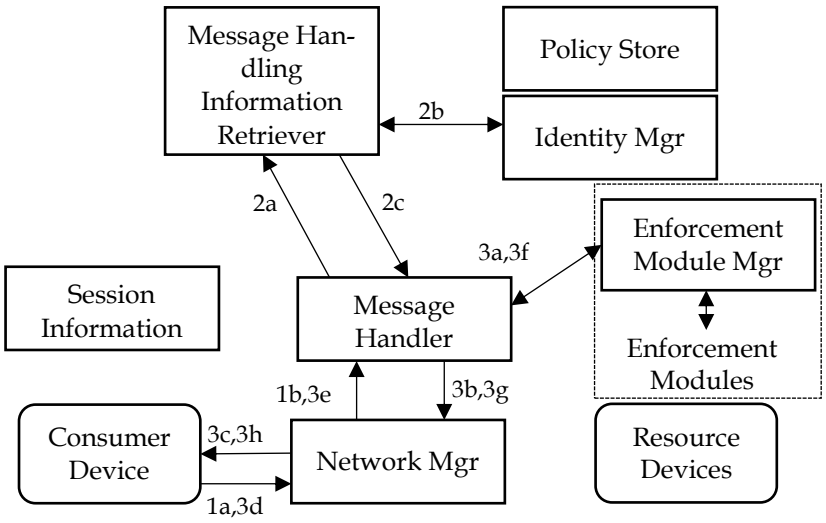


Figure 5: Internal handling of session set-up, part 1

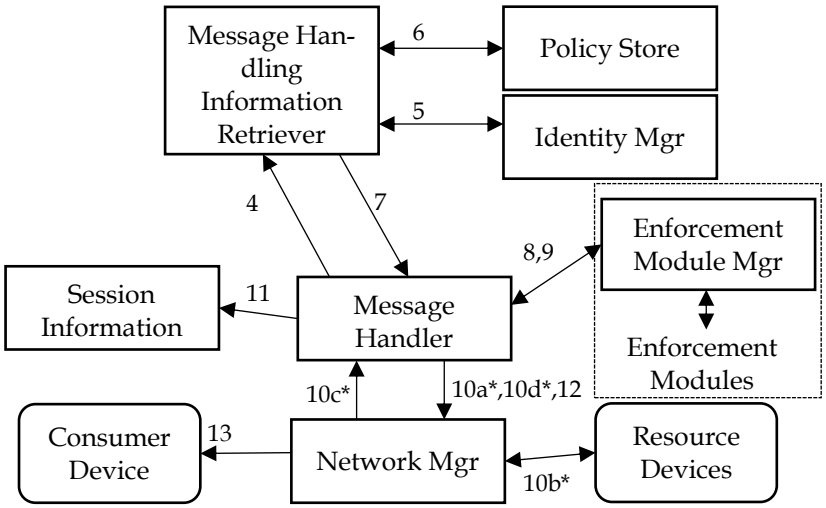


Figure 6: Internal handling of session set-up, part 2

security mechanism and creating key material. Sessions with the Resource, potentially consisting of several specific Resources, are set up in step 10. If the gateway does not need to authenticate, step 10d is not necessary. Finally, session material is stored in `Session Information` (step 11) and the Consumer's session parameters are sent in steps 12 and 13.

4.8 Regular message handling

This section describes the handling of regular messages exchanged within a session. These are messages sent by the Consumer to a Resource via the gateway using an abstract address identifier, or vice versa. Figure 7 shows Consumer to Resource message handling, Figure 8 shows the opposite direction. While the handling is very similar, separate figures were created to highlight the differences.

Here, the `(src,dest,abstractDest,data)` message will look like `(Consumer,GW,abstractIDOfResource,rawdata)` or `(Resource,GW, abstractIDOfConsumer,[metadata,rawdata])`. In some cases, the messages sent by a Resource will contain metadata. It is assumed this additional information may be added by devices in the home sphere to specify tags or other directions meant for the enforcement of policies. An example is the use of tags to specify that data or parts thereof are sensitive and to specify sticky policies.

The gateway will start by fetching all session information. This includes key material, potential specific information of some enforcement modules and all policies.

Afterwards, some security enforcement steps are executed. The gateway starts by checking the message security, during which the message is decrypted and authenticity and integrity are checked.

Subsequently, an authorisation step completes the access control. Note that this step may not be necessary, if during session set-up a stateless access rule was selected and this was stored as specific information for the authorisation module. As mentioned, this step also plays an important role in guaranteeing privacy, as fine-grained authorisation is used to check what kind of data and device is accessed.

In the next steps, some additional actions are done to enforce privacy. This may involve checking the sensitivity of data, indicated by tags or deduced from the device or service class, and potentially filtering it out. An example is transforming exact GPS coordinates to broader area description such as a city name.

Now this message has been checked and it is ready for forwarding. The subsequent steps depend on whether the destination is a Consumer or Resource. The latter case is the simplest, as Figure 7 shows, as the message simply needs to be sent to the Resource. This destination could contain multiple specific Resources, though. For each of these, the message is prepared for forwarding by applying the appropriate message security. Afterwards, the secured message is sent to the destination device. Figure 8 shows a slightly different handling for messages originating at a Resource.

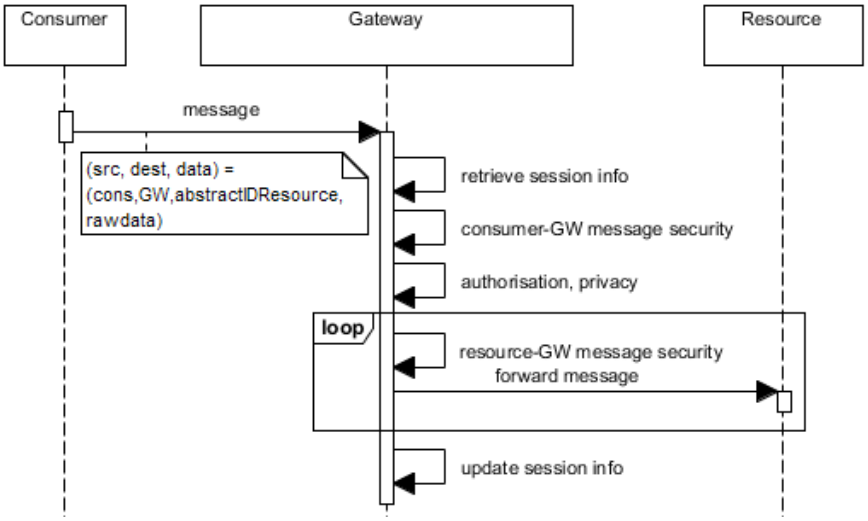


Figure 7: Regular message handling: Consumer to Resource

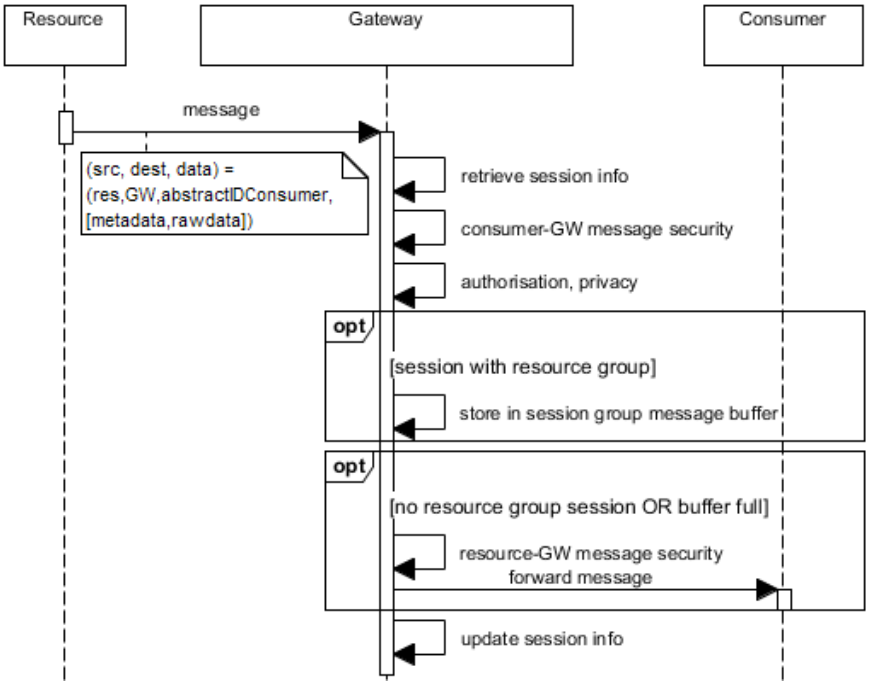


Figure 8: Regular message handling: Resource to Consumer

If the session involves a group of Resources, the message is stored in the session group message buffer. This temporarily stores messages until all members have sent. Here, added messages are combined into a single message for the Consumer. Subsequently, if the session contains only a single Resource, or if all session Resources have sent a message to the buffer, the message is secured and sent.

Note that this is a generic sequence, and that some steps may be skipped, depending on the policies and stored enforcement-specific information. As an example, message encryption may be skipped if the policies do not dictate a need for it. As another note, this behaviour also applies to communication between smart home devices, as explained before. An alternative approach for these devices is to set up a cluster, after which messages are transferred directly between the devices.

4.9 Other gateway behaviour and interactions

This section describes some miscellaneous gateway behaviour. First, private session are explained, followed by an overview of adding devices and policies.

4.9.1 Private communication with a device

Some devices are capable of authenticating and setting up a secure channel themselves, and will hence authenticate and encrypt their messages without need for a gateway. A user may want to request a *private session* with such a device, which basically means the gateway's security functionality is bypassed. This means that a service provider could engage in private communication with a home sphere device. An example is the Energy Company interacting with the smart meter, which is a device owned by that service provider but located in the smart home.

The user will send a request for a private session to the gateway. In response, the gateway will authenticate the sender and check the policies to see if the session is allowed. If this is the case, the user will then be able to use the requested Resource without gateway interference. This could happen by means of messages passing through the gateway and which are simply forwarded, or by direct communication using a different channel.

Obviously, this mode of operation presents a trade-off between the security provided by the gateway when using its default interaction model, and the flexibility of not having this model and communicating with a device in private. The impact and considerations will be further discussed in Section 7.2.

4.9.2 Configuration

Adding devices and services. As exemplified by the use case of adding a smart TV to the home sphere, the gateway must provide support for adding new devices. The device may be identified manually by the user, or the gateway could receive a notification from the network that a new device is available, after which further configuration steps by the user are required. Then, some device details need to be defined. First off, it will be necessary to specify the Resource and/or Consumer identities of the device and supply a name. Furthermore, security information is required. This means that the session set-up mechanism and key material, and the entity authentication mechanism and material are specified. For Resources, the services need to be described.

Most of this information is stored in the `Identity Manager`. It will contain the name, session set-up key material, authentication details and a list of services. From then on, the `Identity Manager` will be able to select this device if its service is being requested. The IP address of the device and its mapping to internal entity identities is stored in the `Network Manager`.

Specifying restrictions and requirements. Afterwards, it is possible to specify restrictions and requirements for the device. This includes specification of access control and privacy limitations and requirements for message security. These are specified by means of policies, detailed further in Chapter 5, and they will be stored in the `Policy Store`. Before storing them, they will be parsed in order to create an internal presentation of the policies.

When visitors enter and are given access to the home sphere, they may also have their own devices. Adding these is very similar to the interaction described above. As the gateway aims to enforce the policies of the smart home (the residents), the visitor is not allowed to specify limitations herself.

4.10 Consumer and Resource interfaces

The gateway offers it services, including access to home sphere, in a message-oriented fashion. Consumers and Resources send messages, which arrive at the `Network Manager` and which are then handled by other components, as described in this chapter. This section gives a brief and abstract overview of the types of messages exchanged with the gateway.

- `Session request {requested Resource}`
  A session request is sent to the gateway by a Consumer. It specifies the Resource that the sender wants to access. As explained in Section 4.4, this could be an abstract description or a service identifier. This request will eventually result in a `Session granted` or `Session denied` message.

- `Session granted {requested Resource, abstract Resource address ID, session security information}`
  A message sent to a Consumer by the gateway, after successful session set-up, as described in Section 4.7. The message contains the description of the requested Resource, such that the Consumer knows which request this grant belongs to. Furthermore, an abstract identifier is included, which the Consumer should include in the `Regular messages` sent to the gateway for that session. Finally, session security information is included, such as the message security type for both directions of communication and the associated session keys.

- `Session denied {requested Resource, description}`
  This message is sent by the gateway to notify a Consumer that his request, identified by including the requested Resource, was denied. The description is used to provide further information about the failure.

- `Session set-up {abstract Consumer address ID, session security information}`
  A message sent to a Resource by the gateway, during session set-up. As descried in Section 4.7, this will occur for each of the specific Resources involved in the session, before granting the session with the Consumer. As with the distribution of session information in a `Session granted` message, the abstract identity of the communication partner and session security details are attached.

- `Session end {abstract ID communication, description}`
  A message sent by Consumers, Resources or the gateway to inform that a session is no longer active. An abstract identifier is used to identify the session. If a Consumer or Resource sends this message, this will be the abstract identifier it used to address its communication partner. If the gateway sends it, it is the abstract identifier that the receiving Consumer or Resource for addressing its partner.

- `Authentication request {authentication type, challenge}`
  An authentication request may be sent by Consumers, Resources or the gateway. It requests entity authentication of the destination by means of the specified type for the specified challenge.

- `Authentication response {proof}`
  This message is sent by a Consumer, Resource or by the gateway in response to an `Authentication` request. It uses the included proof to authenticate.

- `Authentication response & request {proof, authentication type, challenge}`
  This message simply combines an authentication response with a request. It is a shorthand message in case the party proving its identity also needs the other party to authenticate.

- `Regular message to gateway {abstract ID destination, metadata, payload}`
  A regular message exchanged within a session, sent by a Consumer or Resource. The abstract identifier of the destination is included, as established during session set-up, such that the gateway is able to handle and relay the message appropriately. Metadata may be included if the message originated at a Resource, in which case this could, for example, contain tags to aid in enforcing privacy. The payload is a simple sequence of raw bytes.

- `Regular message from gateway {abstract ID sender, metadata, payload}`
  This message is the next step after a `Regular message to gateway`, and is sent by the gateway to a Resource or Consumer. It includes the abstract identifier of the sender, such that the receiver is able to identify the session this message belongs to. Metadata is optional and could, for example, be used to specify sticky policies when sending a message to a service provider.

- `Group request {list of group members}`
  A request to create a group with the specified list of members, as explained in Section 4.5.2. If the group creation is allowed, the gateway will generate key material and distribute it to all group members.

- `Group request {list of group members, temporary key}`
  Similar to the `Group request` described above, except that the requesting party has already generated the group key material.

- `Group set-up {group information, temporary key}`
  Message sent by the gateway to group members, after creation of a group. The group information contains the group members and potentially other meta-information.

- `Private session request {requested Resource}`
  Used to request a private session with a Resource, as described in Section 4.9.1.

5 Policy Definition Language

In this section, the supported policies are explained. Subsequently, these are exemplified by describing policies for some scenarios, and some extension points of the language are explained.

5.1 Expressing policies

As explained in Section 4.1.1, the gateway protects Resources, and checks messages sent between Consumers and Resources. First off, support for these entities in the policy language will be clarified. Thereafter, policy language elements are introduced for authorisation to control access to the resources, and for message security and privacy requirements.

The emphasis is on the supported concepts, but a syntax is set to allow a tangible description. In this syntax, these language elements have the following meaning:

`[ ]` – optional parts
`xlist = xmember1, xmember2, ...` – a list
`*` – a wildcard indicating all members of some entity

Optionally, a priority may be added for a policy. This may be necessary as several conflicting policies for one resource or message are possible, for example in case a default rule is overwritten by a more specific one. It is assumed a higher number indicates higher priority. As unification of policies is not the topic of this thesis, no scheme has been devised for resolving conflicts.

5.1.1 Identities

As mentioned, the system contains Resources and Consumers. A simple text name is used to indicate these entities, regardless of the exact type. By default, a specific service of a specific device is written as:

device.service

A Resource group or device group is simply described as a list of all members:

resourceGroup = *resource1*, *resource2*, ...
consumerGroup = *consumer1*, *consumer2*, ...

As members are easily added and removed, they are also used to model roles. Some default groups supported by the policy language are:

`Visitors, Residents, Devices, RemoteLogins`

For devices, it is possible to express ownership as follows:

deviceOwner *device consumer*

This indicates that the *consumer* is owner of the *device*. As a result, the *consumer*'s policies also apply to the *device*.

As a physical entity may have both a Resource and Consumer identity in the system, it may be convenient to get the Resource(s) identity when given the Consumer identity, and vice versa. This is possible using the following constructs, respectively:

```
resourceIdentityOf (consumer)
consumerIdentityOf (resource)
```

Furthermore, it should be possible to express the ability to configure consumer group composition (e.g. adding members). For example, a subgroup of residents may be allowed to change the Visitors group. Therefore, the expression:

```
allowGroupConfig consumerGroup consumer
```

indicates that the *consumer* is allowed to configure the *consumerGroup*.

On top of Consumers and Resources, the language allows description of a **communication channel**. This is a connection between a Resource and a Consumer, or part thereof, over which messages are sent in which the Consumer utilises the Resource:

```
communication resource <- consumer
communication resource -> consumer
communication resource – consumer
subcomm_GW_Consumer resource <- consumer
subcomm_GW_Consumer resource -> consumer
subcomm_GW_Consumer resource – consumer
subcomm_GW_Resource resource <- consumer
subcomm_GW_Resource resource -> consumer
subcomm_GW_Resource resource – consumer
```

As shown, the channel may be one-way, with an arrow indicating the direction, or bidirectional. The first three channels apply to the entire end-to-end communication, while the next groups of three apply only to the part of the communication between gateway and Consumer (inner three) or gateway and Resource (bottom three). These *subcomm* channels allow to set different demands for the message up to reaching the gateway and after going through it. For example, this makes it possible to express need for strong confidentiality on the internet part of the message's route, while setting lower demands while the message is in the home sphere.

Finally, **aliases** for Consumers, Resources and communication channels are supported:

```
alias consumer name
alias resource name
alias commChannel name
```

This sets the *name* as an alias. Using an alias may be convenient when writing down policies for longer entity names, for example for communication channels. There is also a functional necessity for aliases, for example when several service providers use different identifiers to connect to the same device.

5.1.2 Authorisation

Allowing or denying access to a Resource is done using the following constructs:

```
grant access to resource (resourcelist) to
     (consumerlist) [condition (conditionlist)]
     [priority (number)]
deny access to resource (resourcelist) to
     (consumerlist) [condition (conditionlist)]
     [priority (number)]
```

This controls authorisation to the Resources in the given list. Optionally, conditions may be added. A first set of conditions is time-related, for example to only apply the role in a specified interval. Others are count-related, for example to limit the amount of accesses within a given time interval. They may also depend on the current home sphere context in other ways, for example by only applying when a specific device is in a specified state. These three examples are described formally below:

```
Time startTime-endTime
MaxAmount maxCount per time duration
Status device statusDescription
```

Brackets, 'AND' and 'OR' may be used to specify more complex combined conditions.

An example of authorisation, which grants access to a device and a service of another device to a group of users, is given below:

```
grant access to resource (WashingMachine, EnergyMeter.
     usageinfo) to (RemoteLogins) priority (1)
```

5.1.3 Message security & privacy requirements

Other language elements allow specification of requirements for connections:

```
specify req for (listOfCommunicationChannels)
     [priority (number)]
```

This sets the requirement *req*, which is detailed below, with optional priority. The rule is always specified for a list of communication channels. This could be specifically for communication between a Resource and a Consumer, but the '*' wildcard is possible for more general statements.

The *req* can describe various policies. A first subset specifies message security requirements, which influence the mechanism used within sessions:

```
req =
     confidentiality OR
     confidentiality (specificdemands) OR
     messageAuthentication OR
     messageAuthentication (specificdemands)
```

These specify a need for confidentiality and for message authentication (authentication and integrity). As shown, it is possible to simply specify that confidentiality or message authentication is required. As an alternative, specific demands may be added, for example to specify a specific algorithm such as AES.

Other requirements are privacy-related. These may be used to specify the need for filtering or for sticky policies:

```
req =
     filter (specificdemands) OR
     stickyPolicy (specificdemands)
```

In case of sticky policies, the gateway will attach these to messages, and the service provider is expected to enforce them. Table 1 shows some pre-defined sticky policy expressions.

Also, it is possible to specify that no policies should be enforced, and hence messages should simply be relayed:

```
req =
     none
```

Here is an example which uses the requirements specification to set a confidentiality requisite for a service's outgoing traffic after travelling through the gateway:

```
specify requirement confidentiality for
     (subcomm_GW_Consumer EnergyMeter.usageInfo -> *)
```

The '`*`' wildcard makes the rule apply to any Consumer using the *EnergyMeter*.

5.2 Examples of use

To show the language in use for realistic situations, this subsection gives some examples of policies for scenarios from the use cases (Section 3.1).

One of the use cases involved presence of people in a room triggering a reaction of the heating and lighting systems. Here are some policies for the connection between the *HeatingController* and the heating devices in the living room:

```
LivingRoomHeating = LivingRoomHeatingUnit1,
     LivingRoomHeatingUnit2
grant access to resource (LivingRoomHeating) to
     (HeatingController)
```

Specific sticky policy demand	Meaning
`limitUse (noThirdParty)`	Information only for use by the receiving service provider.
`noStorage`	Information is only for direct usage, no storage allowed.
`limitStorage (anonymised)`	Information should be anonymised before storage.
`limitStorage (time: 5 days)`	Information should be deleted after 5 days.

Table 1: Some predefined sticky policies

The group *LivingRoomHeating* is defined as a group of Resources, and the *HeatingController* is given access to it.

Another element of this scenario is the collection of presence information by the *HeatingController*. This will result in rules such as:

```
grant access to resource (HeatingController) to
     (PresenceSensor1)
```

Here the *HeatingController* is the Resource, which is used (for example, a *sendReading* service) by a sensor.

If the *HeatingController* were located externally instead of in the home sphere, for example if it were offered by a service provider, then sticky policies could be included when sending sensor information. Here is an example limiting the further distribution of data:

```
specify requirement stickyPolicy
     (limitUse (noThirdParty)) for
     (communication HeatingController <-
     PresenceSensor1)
```

Another use case described the Energy Company requesting consumption details from an energy meter in the smart home. In this case, the energy meter itself provides authenticity for the message contents. In the service provider's authentication profile, the need for public key authentication will be specified. The gateway still has to check the Energy Company's identity and provide confidentiality for the outgoing energy consumption information. Furthermore, access is only allowed once each day at a specific interval:

```
grant access to resource (EnergyMeter.usageInfo) to
     (EnergyCompany) condition (Time 21:00-23:00 AND
     MaxAmount 1 per time 24 hours)
specify requirement confidentiality for
     (subcomm_GW_Consumer EnergyMeter.usageInfo ->
     EnergyCompany)
```

A final set of examples applies to the alarm mode. This involves sensors for doors and windows relaying their readings to the external alarm company:

```
windows = kitchenwindow, livingroomwindow1, ...
doors = frontdoor, kitchendoor, ...
grant access to resource (AlarmCompany.sendReading) to
     (windows, doors)
specify requirement confidentiality for
     (subcomm_GW_Consumer * <- windows,
     subcomm_GW_Consumer * <- doors)
specify requirement stickyPolicy (limitStorage (time:
     1 days)) for
     (subcomm_GW_Consumer * <- windows,
     subcomm_GW_Consumer * <- doors)
```

First, the two groups of Consumers are defined. In this case the Consumers are devices in the smart home, using an information reporting service of a service provider Resource. These two groups are given access to this service in the third rule. Furthermore, confidentiality is required for the outgoing readings, for the part of the communication where they travel from the gateway to the alarm company. The final requirement adds a sticky policy to the outgoing readings.

5.3 Extension points

A number of policy language elements were explained in this chapter. However, there are also several parts where the language may be extended. Obviously, this means that the gateway must have enforcement modules to support them.

A first extension is related to the conditions. First off, more conditions may be specified, as they were intentionally kept simple in the base language. Furthermore, users may want to set conditions for requirements, instead of only for authorisation expressions.

Another extension is the inclusion of requirements other than those mentioned here. For example, if the gateway is capable of anonymising information, it needs to be described in new privacy requirements.

The *specificdemands* for confidentiality, filtering and sticky policy requirements are also highly flexible. Their definition was kept abstract, as the exact parameters depend on the gateway's enforcement capabilities.

6 Implementation

Based on the gateway design described in Chapter 4, a prototype implementation has been created in Java. This prototype includes the normal interaction model of session requests and regular messages exchanged within sessions, with support for various security mechanisms.

An overview of the prototype's components is given in Figure 9. Looking back at Figure 2 on page 27, it corresponds to the architecture described in the chapter on the gateway's design. Interfaces in the component diagram correspond very well to the associations between components in the architecture. Hence, the functionalities and responsibilities are exactly those explained in Section 4.2, and will not be repeated here. The `Logging` component that has been added here is the only major difference. As the name implies, it receives updates on the activities and state of other components and logs them. It uses the `Identity Manager` to retrieve names of entities, allowing it to create an externally readable representation of these objects.

6.1 Entities and the Identity Manager

Consumers, Resources, communication channels and devices are all *entities* in the system. Figure 10 shows how they are related. The figure shows how devices may have a Consumer and Resource identity, and shows the different types of Consumers and Resources, as described in Section 4.1.1. Here, a `DeviceResource` signifies a device as a Resource, and a `DeviceServiceResource` is a specific service of such a specific device. Furthermore, the figure shows that a communication channel is a link between a Resource and a Consumer, along with a direction and a sub communication specification. This sub communication is necessary to express security policies that apply between the gateway and a Consumer or Resource.

The underlying structure of the `Identity Manager` is shown in Figure 11. It has components responsible for managing devices, Consumers and Resources. Each of them holds a collection of registered entities of the respective type, and allows adding and removing them. They are also responsible for mapping names and aliases of those entities. They contain a mapping between the entity type and names in both directions. This allows them to efficiently go from an internal representation to a name, as required when this information needs to be shown, and from a name to an internal form. The latter case is used when parsing policies and, in case of Resources being looked up, for finding the requested Resource.

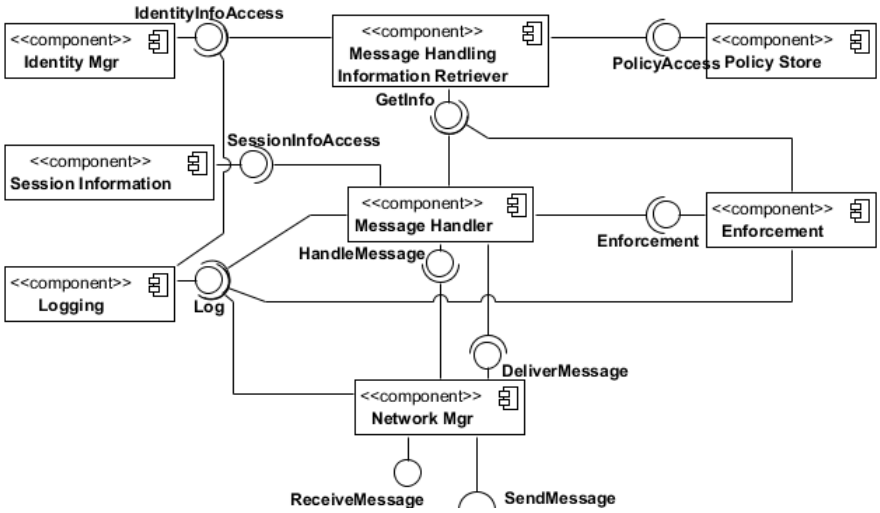


Figure 9: Component diagram of the gateway

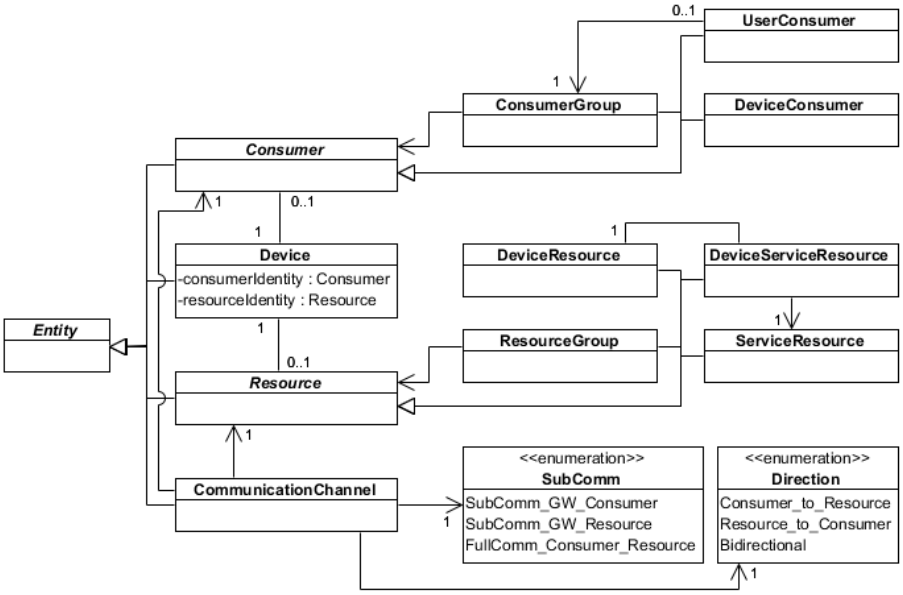


Figure 10: Entities in the system

Furthermore, one component manages communication channels. In this case, not all communication links are registered. Instead, the component only registers alias names for use in policy description.

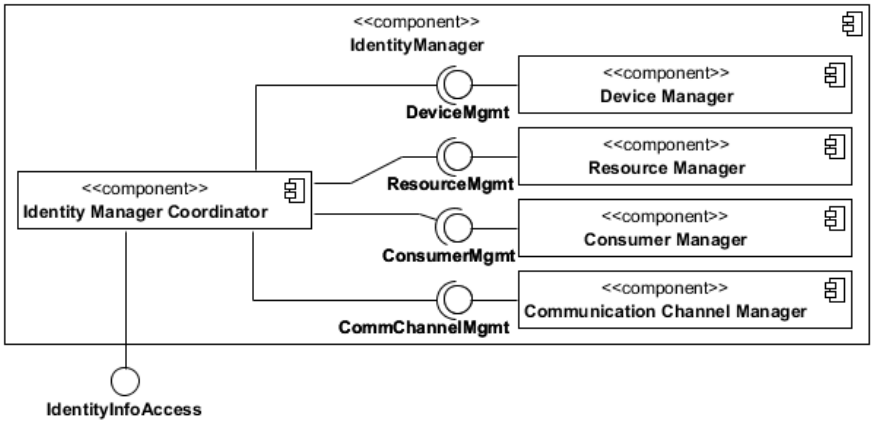


Figure 11: Subcomponents of the Identity Manager

The `Resource Manager` component is the main module carrying out name resolution and service mapping. As it holds a mapping between Resources and names, the name resolution is a straightforward step. Furthermore, the `Resource Manager` holds a mapping between a service description and a list of devices offering it. This allows efficient look-up of a device offering a service that is requested, as it is not required to iterate over all devices. This means that, when adding a device, it is included in the list for each of the services it offers.

Two other functionalities offered by the identity management components are the computation of **subsets** and **supersets** of Resources and Components. Concretely, the subset of a `DeviceResource` consists of all the `DeviceServiceResources` it offers, the subset of `ServiceResource` consists of all the `DeviceResources` offering it, and a group consists of all its members. Supersets are then computed by using these subsets. If an entity belongs to the subset of a second entity, that second entity is a superset of the first one. These sets are used when searching for applicable policies. A policy defined for some entity also applies to a second entity if that second entity is in the first one's subset. Alternatively, the policy applies if the first entity is the second entity's superset. This applicability check is necessary, as policies may be defined for groups and services, in which case they apply to members and devices offering the service.

6.2 The Message Handler

The structure of the `Message Handler` component is shown in Figure 12. The main handling of messages is done by the `Message Handler Worker`. It holds the logic for handling each type of message the gateway receives, and will create messages to send to other entities. It will use the `Enforcement` module for policy

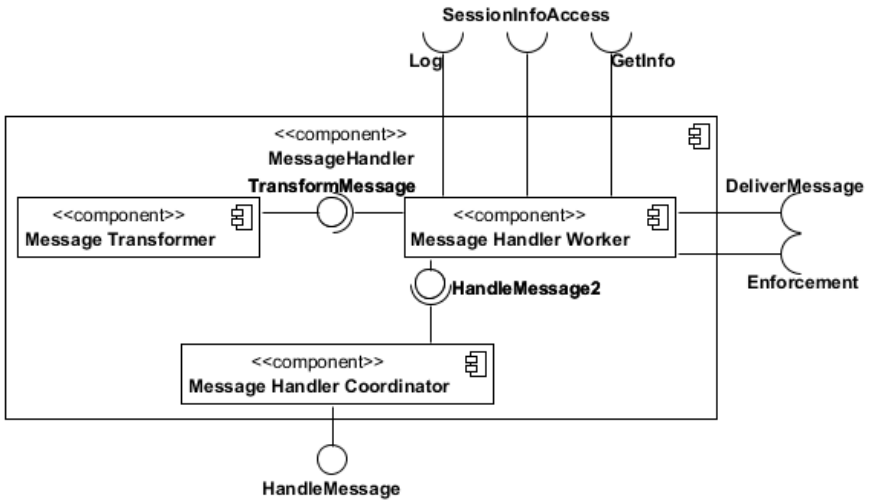


Figure 12: Subcomponents of the Message Handler

enforcement and will control the flow for session set-up and regular message handling (Sections 4.7 and 4.8). It has a queue of messages which are left to handle.

The `Message Handler Coordinator` receives messages from the `Network Manager` and ensures they are handled by a `Message Handler Worker`. This allows for a design with multiple workers handling messages, running in different threads or, potentially, on different machines. In the current prototype, there is only a single worker. The coordinator simply adds all messages to this worker's queue. However, this could be extended with several workers easily, as the `Message Handler Worker` is stateless.

Finally, there is a `Message Transformer` subcomponent. This transforms between sequences of bytes exchanged with other devices and an internal message format where gateway metadata is separated from the raw byte payload for the communication partner.

6.3 Policy enforcement

Storage of policies is the responsibility of the `Policy Store`. When a session is requested, all policies between the Resource and Consumer are retrieved. To achieve this, the `Message Handling Information Retriever` gives the `Policy Store` the supersets for the Resource and Consumer. Applicable policies are those for which both entities are in the respective supersets. For efficiency reasons, the policy store is indexed both by Resource and Consumer. This is a time/space trade-off, where some more space is used by saving policies twice in order to achieve faster retrieval.

The enforcement of policies by the gateway is rather straightforward. The enforcement submodule `Message Security` will decide on the message security mechanism based on confidentiality and message authentication requirements. The `Authentication`, `Authorisation` and `Privacy` submodules are each responsible for enforcing the policies of the categories their names imply.

The `Authentication` submodule is capable of creating challenges and checking proofs in case of authentication of the other party, and of proving the identity of the gateway itself. In order to make the authentication of the other party work for various mechanisms, there is a method which will return a result code, complemented with a byte array for the other party and a byte array for local storage. The former contains all information the other party needs to prove its identity. This could be a simple nonce, or could be more complex, such as an authentication policy complemented with a challenge. The part that is stored locally is required when the gateway verifies the other party's proof. To give some examples of the content of these arrays, consider simple password authentication. In this case, both arrays will contain the nonce. In case of the public key protocol from Section 4.6.1, the original random is stored locally, while the other party gets this random that has already been encrypted using its public key. The specific modules for each mechanism are responsible for interpreting all of this information correctly.

Note that, for the enforcement, it is possible the modules will need to make use of the `Message Handling Information Retriever`. The `Enforcement` module also offers the ability to evaluate conditions, which may be attached to policies. Furthermore, each module may store *enforcement specific details* for messages in that session, as explained in Chapter 4.

The `Enforcement Module Manager` is responsible for selecting the appropriate submodule based on the requirement description. This submodule will further relay the enforcement to its submodule for the applicable mechanism.

As a final note, the direction of the message needs to be taken into account when enforcing requirements. If the sender is the Consumer, then rules for *subcomm_GW_Consumer* need to be considered, and similarly for a Resource and *subcomm_GW_Resource* rules.

6.4 Implementation details

The gateway prototype has been implemented in Java. Several entity authentication and session security mechanisms have been included, demonstrating the gateway's flexibility. Java 8 has been used for the implementation, and the Priman framework [55] has been included for providing some authentication functionality. The framework requires some providers that are not included in the standard Java platform, but are available through the BouncyCastle cryptography toolkit [56].

# lines of code (incl. tests & comments)	9611
# lines of code (excl. tests)	5834
# classes (excl. inner classes)	118
# packages	25
# top level packages	14

Table 2: Prototype statistics

For entity authentication, the gateway provides support for password authentication and a public key system, both of which were described in Section 4.6.1. Furthermore, it is possible to choose for *no authentication*, which could be used if simple network authentication suffices. All of the cryptographic primitives for these mechanisms are provided by the standard Java Cryptography Extension [57]. Finally, Priman is used to give access to Idemix anonymous authentication [58].

For message security, the implemented modules are *authenticated encryption, authentication only* and *no security*. If the gateway finds a confidentiality requirement in the policies, authenticated encryption is selected. If a message authentication requirement is found without the need for confidentiality, the gateway will only perform authentication. Authenticated encryption is implemented by AES in GCM mode. GCM provides confidentiality, authenticity and integrity [59], and will create a nonce for each message which is sent alongside the ciphertext. Authentication without confidentiality is provided by attaching a message authentication code. In both cases, a symmetric key is used which is established during the session set-up. Again, the cryptographic primitives required are found in the Java Cryptography Extension, though Java 8 is necessary for supporting all providers by default.

No specific privacy modules for filtering or sticky policies have been included in the prototype. Furthermore, support for private sessions has been left out, as this is not the main mode of operation and not the most interesting part of the gateway because it does not show its security functionality.

The gateway offers it interface by handling the types of messages described in Section 4.10. The exact syntax of these messages is included as an appendix (Appendix A: Syntax of exchanged messages). Some miscellaneous prototype statistics are found in Table 2.

6.5 Consumer and Resource middleware

A middleware framework provides the functionality allow Consumer and Resource devices to communicate with the gateway. This middleware ensures the communication protocols are followed. They allow applications to send messages via the gateway and handle authentication with the gateway. Consumer middleware also contains the logic for initiating a session set-up. The middleware integrates different specific enforcement modules for handling message security and authentication. It

is also possible to use a middleware without enforcement module, which simply takes messages and handles the headers. This is useful in case the application itself handles encryption.

This strategy could be generalised to make one middleware which integrates different enforcement modules and hence support multiple security mechanisms. In this case, the middleware would effectively become a slimmed down version of the gateway software. Specifically, there would be a module responsible for enforcement, integrating several mechanisms, much like the gateway's `Enforcement` module.

In this implementation, the simpler and more compact method of middleware with a fixed security mechanism was chosen. The focus of this thesis is on the gateway itself anyway, and an extensive and complex middleware is not suited for all Internet of Things devices. This model is hence sufficient in the context of this thesis, and the middleware implementing some specific mechanisms is also an appropriate abstraction for devices implementing these methods themselves.

7 Evaluation & discussion

This chapter will evaluate the presented gateway system and discuss its ability to satisfy the goals of the thesis. First, performance is described, followed by a discussion of the security and privacy aims, the policy language and some miscellaneous requirements.

7.1 Performance

The performance tests aim to measure the overhead caused by the gateway for typical and frequent scenarios executed by the gateway. These tests were run on a laptop with an Intel Core i3-3337U processor (2 physical cores of 1.80 GHz) with 8GB of RAM on Windows 8.1. Furthermore, they were executed using the default Oracle Java 8 virtual machine. First, the performance of session set-up is tested, where a Consumer such as a service provider requests access to some Resource. Thereafter, exchange of regular messages between a Consumer and Resource within a session is evaluated. Finally, there is a brief description of the message overhead caused by the system's headers.

7.1.1 Session set-up

A first set of tests executes a session set-up where the Consumer requests access to a service using several forms of key material used for the set-up, several authentication mechanisms and several session security requirements. Note that this will require service mapping. The tests measure the time taken from the moment the message enters the gateway up to when a *session granted* message is sent to the Consumer. The time taken by the Resource to process the *session setup* message and by the Consumer to respond to *authentication request* messages is not counted, such that only the activities within the gateway boundary are timed. Furthermore, a separate timer measures the time taken by cryptographic operations such as decrypting a message and checking authentication. Then, the overhead caused by the gateway itself is obtained by subtracting this value of the cryptography timer. Timings are done by calling methods to start and stop a stopwatch.

For the first four tests settings, the Consumer, the Resource and its service are the only entities registered in the system. Furthermore, the system contains an authorisation policy to allow access and requirements for the security of the subchannels. The security parameters are shown in Table 3, and the results are found in Table 4. The session set-up material can be symmetric (128-bit AES), 2048-bit asymmetric, or *none* (in which case plaintext is sent).

Test	1	2	3	4
Set-up key material Consumer	symmetric	symmetric	asymmetric	symmetric
Set-up key material Resource	symmetric	asymmetric	symmetric	none
Authentication Consumer	password	password	public key	public key mutual
Authentication Resource	password	anonymous	none	none
Session security Consumer → GW	encr&auth	auth	encr&auth	encr&auth
Session security GW → Consumer	encr&auth	encr&auth	encr&auth	encr&auth
Session security GW → Resource	auth	auth	encr&auth	none
Session security Resource → GW	auth	encr&auth	encr&auth	none

Table 3: Session set-up tests (1): security parameters (key material used for session set-up, authentication mechanisms, message security requirements) for the first test settings

Test	Avg. (stdev) total	Avg. (stdev) crypto only	Avg. (stdev) without crypto
1	6.84 (2.35)	4.02 (1.18)	2.82 (1.33)
2	472.22 (12.04)	454.19 (109.23)	18.03 (3.34)
3	37.94 (6.50)	35.71 (6.27)	2.23 (0.58)
4	63.92 (9.71)	32.34 (6.31)	31.58 (6.18)

Table 4: Session set-up tests (1): Averages in milliseconds (50 samples per test). Average overhead differs significantly for different security parameters, but most of the overhead is attributable to the cryptography.

For authentication, the mechanisms used were password authentication with a 128-bit nonce for hashing, anonymous authentication involving 1 credential, Section 4.6.1's public key mechanism using 2048-bit keys, or *none*. Session security is 128-bit authenticated encryption, authentication using 128-bit SHA-1, or *none*. For each test, security parameters are drawn from these sets of options and 50 sessions were set up.

The results of the first test show that on average, the gateway adds about 2.82 ms of overhead, disregarding the cryptography time. The total time shows an average of 6.82 ms, which adds the time for password encryption, setting up 2 session keys, and securing these using symmetric encryption.

The second test has a Resource for which asymmetric encryption is used for session set-up and which uses anonymous authentication. Hence, this would most likely occur if the Resource is offered by a service provider. The results clearly show a much higher overhead compared to the previous test. The gateway's identity proof takes about 450 ms, resulting in an average overhead of about half a second. If cryptography is disregarded, however, the delay is only 18.03 ms on average. The reason for the higher value compared to the first test, is that the gateway has slightly more work in this scenario. Firstly, the anonymous authentication module must deserialise the other party's message into a challenge and a policy and must serialise the response. Furthermore, this scenario has different requirements for subchannels based on the direction.

A scenario corresponding to a relatively constrained Resource without authentication is shown in test 3. The results are quite similar to those of the first test. The average without cryptography of 2.23 ms is close to what was found in the first test. The average of the total times is higher, which is expected as some asymmetric mechanisms are used.

The final test corresponds to a really constrained Resource, which does not support any security. The average without cryptography of 31.58 is significantly higher than the values in the first and third test. This is because mutual public key authentication is used, such that the gateway also needs to authenticate towards the Consumer and more messages need to be handled.

One more important note about all of these tests is that, as expected, some variance in the results is shown. The threaded message handling is definitely a contributing factor in that domain. Nonetheless, this makes the tests representative for actual functioning of the gateway.

In tests 5 through 7, described in Table 5 and Table 6, the security parameters from test 1 are used, but more resources and policies are added. In test 5, this means that 10 extra devices are added as Resource, all with some services and some overlapping services. Furthermore, 3 levels of groups are added. In test 6, there are 50 extra devices compared to test 1 with 4 levels of groups. Test 7 adds 30 applicable policies to the requested service.

We notice that test 5 results in a barely noticeable difference. With 50 extra devices (test 6) the average without cryptography increases to 3.82 ms. Most influence, however, is seen when applicable policies are also added, resulting in an average without cryptography that is about twice that of test 1. This is explainable by the extra work of checking the authorisation policies.

Test	Description
5	Test 1 with 10 extra devices as Resource
6	Test 1 with 50 extra devices as Resource
7	Test 6 with 30 extra applicable policies

Table 5: Session set-up tests (2): Test description

Test	Avg. (stdev) total	Avg. (stdev) crypto only	Avg. (stdev) without crypto
1	6.84 (2.35)	4.02 (1.18)	2.82 (1.33)
5	5.79 (2.68)	2.89 (1.00)	2.90 (1.80)
6	7.52 (2.60)	3.70 (1.56)	3.82 (1.63)
7	9.71 (6.05)	4.04 (4.65)	5.67 (2.70)

Table 6: Session set-up tests (2): Averages in milliseconds (50 samples per test). Overhead remains limited when more Resources and applicable policies are added, up to amounts that are unlikely to be surpassed in a smart home context.

From these session set-up tests, it turns out that the overhead caused by the gateway is relatively small. Even with 51 devices as a Resource and over 30 applicable policies, the set-up takes less than 6 ms without cryptography and less than 10 ms in total, on average. In a smart home context, it is not expected that many more Resources will be added. Different security parameters result in other average execution times, but it is important to note that this always stays very well below 100 ms. It is not expected that there will ever be 10 session set-ups per second, so there should not be any problem even on less powerful machines.

7.1.2 Regular message handling

This series of tests measures the overhead of handling regular messages within a session. The first three tests measure the time taken from the moment a message from a Consumer enters the gateway, up to its processed version being sent to the Resource. Test 4 to 6 measure the overhead for the opposite direction. Table 7 and Table 8 describe the test parameters and summarise the results. The delay for 50 exchanged messages of 64 bytes were measured for each test. Note that, for these tests, the key material used for session set-up and the authentication mechanism are irrelevant. The only difference between them is the session security mechanism. In test 1, 128-bit authenticated encryption and 128-bit SHA-1 authentication are used for the Consumer and Resource subchannel, respectively. In test 4, these requirements have been switched, meaning that in both tests the gateway will decrypt a message and create a message authentication code for forwarding. This results in tests 1, 2 and 3 corresponding to test 4, 5 and 6 in terms of session security.

Test	Session security subchannels
1	Consumer: encr&auth, Resource: auth
2,5	Consumer: encr&auth, Resource: encr&auth
3,6	Consumer: auth, Resource: auth
4	Consumer: auth, Resource: encr&auth

Table 7: Regular message handling tests: Description of security parameters of the tests.

Consumer → GW → Resource

Test	Avg. (stdev) total	Avg. (stdev) crypto only	Avg. (stdev) without crypto
1	1.30 (0.42)	0.56 (0.22)	0.74 (0.33)
2	2.18 (0.52)	1.44 (0.37)	0.74 (0.31)
3	0.92 (0.42)	0.20 (0.09)	0.72 (0.38)

Resource → GW → Consumer

Test	Avg. (stdev) Total	Avg. (stdev) crypto only	Avg. (stdev) without crypto
4	0.81 (1.53)	0.64 (1.51)	0.17 (0.05)
5	1.52 (0.55)	1.32 (0.51)	0.20 (0.09)
6	0.45 (0.70)	0.24 (0.65)	0.21 (0.25)

Table 8: Regular message handling tests: Averages in milliseconds (50 samples per test). Overhead is limited to at most a few ms, and is higher for the Consumer to Resource direction due to an authorisation check.

The results show that, regardless of the session security, the average overhead without encryption in tests 1 to 3 is almost identical. The average total delay varies more, which is expected as authenticated encryption creates more load than authentication. Similarly, the averages without encryption in tests 4 to 6 are very close together, and the averages of the total vary as expected. It is interesting to note that a message from Consumer to Resource takes significantly longer than one in the opposite direction. This is because the former also requires an authorisation check.

Of course, in a real deployment it is the total overhead that matters for the applications. This involves an extra step of decrypting the message and encrypting it again for the destination. Obviously, this presents extra delay compared to a direct communication without the gateway, but this is a price to pay for the security features of the gateway and the flexibility of allowing different mechanisms for different subchannels which it offers. Nonetheless, the overhead never exceeds a few milliseconds, meaning that several hundreds of messages per second are possible. Even if the gateway were to run on less powerful gateway, it will still be able to process all of the smart home's messages without problems. As mentioned, the authorisation check for messages from Consumer to Resource create extra overhead. This may be

eliminated if during session set-up it is assessed that the granted authorisation is stateless, such that subsequent checks are skipped. On the other hand, some more overhead is to be expected if privacy modules are added.

7.1.3 Message headers

Another area to consider is the message header overhead caused by the gateway, which causes some networking overhead. Because interaction is done with the gateway, devices need to adhere to the specified message format (see Section 4.10). The primary overhead that is of interest is that of regular messages. In the prototype's syntax (see Appendix A: Syntax of exchanged messages), this adds the strings "`message|dest:`*`receiverID`*`|`" or "`message|sender:`*`senderID`*`|`" to the payload. This means that at least 15 or 17 bytes are added (with an abstract identifier of size 1). This is a rather low amount, but it may still be desirable to reduce this further for devices in networks with a very low maximum transmission unit, to lower the chance of more frames being sent. Such a decrease could be achieved by simply using a number to identify the message type (e.g.: "1" for regular messages) and leaving out the *dest*/*sender*, as the first type is only sent to the gateway and the second type is only sent by the gateway. However, for academic purposes of keeping the messages easier to read, this optimisation was not performed.

7.2 Security & privacy

This section will discuss how the gateway system fulfils several requirements related to security and privacy. The first step, for users to express their preferences in policies, is accomplished through the policy language described in Chapter 5 and which will be discussed in the next section.

The design of the gateway in Chapter 4 showed an architecture of the gateway, which is capable of enforcing entity authentication, authorisation, message security (confidentiality, authenticity, integrity) and privacy. The prototype (Chapter 6) made this design tangible and includes support for most of the security features.

The main component responsible for providing this functionality is the `Enforcement` module. It has been designed with submodules for each function domain (authentication, message security, …), and in turn each of these contains several submodules for different techniques and providers.

The prototype demonstrates that the design is realistic. For authorisation, the specific provider's module receives policies and some context information, and the result will state whether or not access is allowed. For authentication challenge creation, the specific module receives an authentication profile and creates a challenge and a part for local storage (for later verification, see Section 6.3). Similarly, for prov-

ing its own authentication, it receives a challenge and returns proof, and for checking authentication proof, it receives proof and the aforementioned locally stored part and returns a result code. Hence, the provider's module takes care of the exact details and the formatting of the information, and the rest of the gateway system does not depend on this.

For message security, the provider's module converts between secured byte arrays and plaintext arrays. It does so for regular messages, in which case the content depends on the devices and application, and for session set-up, in which case the content is session material that needs to be secured. In each case, the rest of the system will simply deal with raw streams of bytes, which is independent of the chosen provider.

As a result of this, the security goals of confidentiality, authentication, authorisation and privacy are reached. Several modules ensure the security of the message, and the system provides flexibility in terms of supported mechanisms and algorithms.

While support for privacy modules is included in the design, this feature is not part of the prototype. As discussed, it would be possible to add modules for dealing with sticky policies or applying data content filtering on plaintext messages which are interpretable by the gateway, possibly aided by metadata. Privacy is not completely absent, however, as several features of the system are related to it but do not make up a dedicated enforcement module. First of all, authorisation rules can be used to control access to resources, meaning that privacy-sensitive devices can be shielded off. Furthermore, support for anonymous authentication is included, allowing the smart home to identify itself to service providers without revealing its full identity. Finally, the session set-up protocol results in abstract entity identifiers to which messages are sent. This makes it possible to give a Consumer access to some service without revealing the exact identity and all information about the underlying Resource that is offering it.

Private sessions The private session model discussed in Section 4.9.1 allows private communication between devices without gateway involvement. This model has been added as it may be desirable for some use cases, because it steps away from the default model with session set-up. It should be noted that it does still leave the gateway/resident some control, as the private session needs to be granted. This means there is awareness of the existence of such a session, and some restrictions are possible. For example, it could be possible to set temporary limitations for that device during the private session, such as preventing it from communicating with other home sphere devices. Obviously, most of the gateway's functionality is lost whenever such a private session is allowed, including its ability to enforce and guarantee security. As it effectively shuts out the gateway, this is unavoidable when introducing private sessions. Therefore, the default interaction model with session requests and messages passing through the gateway is still the preferable mode of operation. Choosing the private session model presents, as mentioned, a trade-off between the flexibility of not using the gateway and the missing security which it could offers.

Threat models The next paragraphs review the threats described in Section 3.2.3 and describe how the gateway protects against them. A first threat that was mentioned is the external attacker that is able to eavesdrop on messages and modify their contents. This problem can be solved by the gateway, as it is able to secure messages that travel the internet by providing confidentiality and message authentication. The exact type of protection is configured by setting the policies appropriately. For instance, if confidentiality is not needed and, hence, eavesdropping is not an issue, the user can decide to require authentication only.

There was also the threat of service providers not staying within the limits set for them and agreed in the contract, for example because they were compromised. The gateway has no explicit mechanism for detecting such occasions. In principle it would be possible to add such a system, which would compare declared policies to the actual behaviour of the service provider and what it requests. Such a mechanism was not in the scope of the thesis however, but it could be added as an extension of an authorisation module or as a specific privacy module. The gateway will not simply allow threats of this kind, though. If service providers attempt to take actions that were not agreed upon, authorisation rules for their access will be missing and their requests will not be granted. Removal of compromised service providers needs to be done manually in the current system. Again, an extension to the system could allow this to happen automatically, by receiving such information from a central authority and taking appropriate action in the `Identity Manager` and `Policy Store`.

Inside attackers also pose a threat of eavesdropping and message modification. Hence, the protection mechanisms of confidentiality and authentication which are provided by the gateway also apply here. Again, it should be noted that no specific mechanism is provided for detecting compromised visitors that could execute such an attack, but they can only stay within the limits set for them. Furthermore, a unique key is used between the gateway and each device within the home sphere network. Hence, key material that the compromised visitor device shares with the gateway will not help it in attacking the communication with other devices. Obviously, a compromised device in a group will result in all of this group's communication being compromised. Nonetheless, communication in other groups or between the other devices and the gateway is still secure.

7.3 The policy language

The policy language presented in Chapter 5 allows the user to express requirements for secure connections and for access control.

The former comprises demands for message authenticity and integrity and for confidentiality. Users are able to specify specific mechanisms for achieving these goals, for example, by specifying the need for AES encryption, or they may let the system

decide. These message security demands are always expressed in terms of communication channels. By using this concept, it becomes easy to express the requirements for a specific link between devices. By using groups and wildcards, they also offer the flexibility of making statements regarding communication with all objects in a defined set. The same concept of specifying requirements for communication channels is used to express policies related to privacy.

Access control policies are expressed for the objects (Resources) themselves, instead of for channels. Alternatively, it would be possible to specify them as an `allowAccess` requirement for a communication channel. While this would make authorisation policies less different from other policies, the current approach makes it syntactically clear that access to some Resource is being controlled, and that this access is denied or granted to some Consumer.

The notion of groups in the policy language makes it easier to express policies that apply to a set of devices, without a need for defining them separately. This should also make it easier when policies need to be removed or updated, as this does not need to be done for all of these policies individually. Furthermore, group concepts provide a suitable abstraction to capture the notion of groups, services and users (all the devices belonging to some user).

The Resource and Consumer concepts, on the other hand, make a conceptual distinction between the former entity which offers services and to which access needs to be controlled, and the latter which interacts with the former to use its services. This makes it simple for users to reason about the role of a device in some communication. The disadvantage is that pure peer-to-peer interaction is more difficult to model, as it requires specifying rules for both the Resource and Consumer identity of the devices.

Finally, it should be noted again that the policy language is extensible. Currently, it is already possible to express a wide range of security demands. However, as argued in Section 5.3, it may be desirable to extend the language to express policies for all policy types and to include a wider range of requirements. If more privacy-enforcing modules are introduced in the gateway, there may be a need for more elaborate policies in this domain.

One concept for which a specific format has not been specified in the policy language is entity authentication. While this information is added as a property of a Consumer entity, the exact syntax has not been devised. This would need to include the description of the authentication method, complemented with a link to some specific file containing a password, key pair, … It would be possible to use Priman authentication policies to express all of these, instead of merely for anonymous authentication for which Priman is used at present.

While the current format makes it easy for users to express their policies in a simple text format, it does not include support for a standard description format such as

XML. Of course, this format would make use by non-expert users much harder, but it may be desirable when policies are exchanged between machines. Therefore, it may be desirable to create a conversion between XML and normal text format. Also, being able to convert other standard policy description formats or to handle them by the gateway would be a useful value to add. Another solution that would make policy definition even easier, would be a graphical interface which allows users to select entities and view and specify their policies. However, such a system could easily be wrapped around the language in its current form.

To conclude and summarise, it should be noted that the proposed language allows specification of various security policies. Examples for typical use cases from the smart home in Section 5.2 were based on the use cases in Section 3.1. These make it clear that the language indeed achieves its goal of expressing the policies needed. Possible changes and improvements are extending the language and making it compatible with other standards.

7.4 Other requirements

Offering an interface A chief functional requirement for the gateway that is not directly related to privacy is its ability to offer an interface giving access to resources in the smart home. The concept of a *Resource* has been made very explicit in the gateway's design, and the system allows registration of devices and the services they offer. The `Identity Manager` supports devices as a resource, services and resource groups, making it possible for Consumers to request access to whatever they require. As services of devices are explicitly registered, the gateway is able to act as a repository of available services. Furthermore, aliases can be set for Resources, allowing the flexibility of describing them differently for individual Consumers.

In the presented prototype, the user (resident) itself registers the offered services for each device. However, the design does not prohibit the possibility for the system to execute this configuration itself, by letting the devices themselves specify the services or relying on an online catalogue. This could integrate a standard resource description model, such as the Service Oriented Architecture (SOA) or the Architecture Analysis and Design Language (AADL) described in Section 2.2.4. Another extension would be to let the gateway offer more information about the services available, instead of a simple name that describes it. This could easily be built into the current system, although this extension would provide most value in combination with a standard resource definition language.

Mobility and remote access Another requirement that surfaced in the use cases is support for mobility and remote access. The design has no explicit mechanism for enabling this, although supporting this would, in principle, not be a problem. A device is able to request a session and interact with the gateway from anywhere, so this access, including the authentication check, is not limited to a fixed location. It may

be desirable, however, to extend authentication profiles to allow different authentication requirements based on the context. In that case, it would be possible to specify a different means of authentication for a resident's smartphone on the home network and in a remote session.

Generality and flexibility In order to handle the heterogeneity of devices in the Internet of Things, the gateway aims to be flexible. A first way in which this is achieved is by having a manager for each of the `Enforcement`'s submodules, which registers its own submodules. Hence, this creates a flexible solution which allows support for several providers. This realises the heterogeneity requirement of being compatible with various security mechanisms. In case a new security method needs to be supported for compatibility with a specific IoT device, it is added here. Support for various kinds of networks can be extended by modifications which are limited to the `Network Manager`. Furthermore, flexibility is achieved by dealing with raw streams of byte exchanged between the devices. Hence, the gateway does not depend on the peculiarities of the devices or on the use of the Java programming languages. It will be interoperable with any kind of device, as long as the devices stick to the described message formats and the gateway's protocol, and if security mechanism implementations are used that adhere to the standards. Of course, this may mean that a middleware layer has to be developed for some of the device, in order to support the correct interaction protocol. Another note is that although the gateway can handle raw streams of bytes, it may be beneficial for it to give meaning to the data being exchanged. This is, for example, useful for privacy enforcement.

Transparency The presence of the gateway is not completely transparent, as the devices need to be aware that they are communicating via a gateway and need to follow the described interaction protocol. However, as argued in the design chapter, this is necessary for the gateway to achieve its security goals, and hence for the system to be useful. This awareness allows different security measures per subchannel, and the ability to use the gateway for service discovery, both of which were goals of the thesis. On the other hand, the system is transparent for different kinds of devices and programming languages, as discussed before, because it relies on a simple exchange of messages and raw data streams.

User friendliness Finally, the need user friendliness was identified. The policy language allows non-experts to specify security demands, as discussed in the discussion of this language. Furthermore, use of the gateway itself, such as adding devices and registering services, is straightforward. Of course, it would be possible to wrap a graphical interface around several of the configuration parts of the gateway to make this even more intuitive and easier for non-professionals. Such an extension would be a most desirable step in case the system is deployed in realistic settings. Furthermore, it would be possible to set default security rules for devices inside the home sphere or outside of it, or to specifically prompt the user to make security configurations. This decreases the chance of misconfiguration by the user which could cause security risks.

8 Conclusion

This thesis presented a flexible protection mechanism for securing the Internet of Things in a smart home context.

First, a broad literature review was presented, focusing on problems and related solutions in the IoT domain. Next to issues such as networking and addressing, several security-related problems were discussed in more detail, as they are most relevant for this thesis's design. Important findings were the need for a standard security framework and for a suitable key management infrastructure. The related work showed some systems for addressing subproblems such as cryptography and authentication, and a review of description of security policies.

Subsequently, the smart home context and requirements were discussed, followed by the core of the thesis in Chapter 4. That chapter presented the design of the gateway, which stands at the border of the home sphere and protects it. This gateway runs on a device more powerful than the average IoT object and is capable of providing security for devices that are insufficiently able to take care of this themselves.

The system uses the concepts of Resources providing services, Consumers using them, and communication channels between these two entity types. The gateway registers Resources, and hence allows users such as service providers to find a device offering the service they need. This is achieved through a component responsible for registering identity information of devices, which also stores data such as key material. Furthermore, the gateway contains components for managing policies, managing networks and enforcing policies. For imposing security policies, the enforcement module contains submodules for authentication, authorisation, message security and privacy, which in turn contain submodules for specific mechanisms (providers).

The normal mode of operation has a Consumer request a session with a Resource, after which session material is set-up and any kind of raw data can be exchanged in regular messages via the gateway. This makes it possible to use a different security mechanism for the Consumer-gateway and Resource-gateway subchannels, which allows the system to use less powerful encryption for a constrained device within the house and a stronger variant for the connection via the internet. Apart from the normal mode, private sessions between devices which exclude the gateway are possible, eliminating the need to stick to the normal pattern with sessions, but also removing the security features that the gateway provides.

A system for key management is also provided by the gateway. As mentioned, it stores long-term key material shared with each device, which could be either symmetrical or asymmetrical. During session set-up, it creates session secrets which are shared between the gateway and a device. Again, several variants such as authenticated encryption and authentication are supported, and what is chosen depends on the policies. A final note is that the gateway allows devices to request a group key and have it distributed to the members.

Chapter 5 presented an extensible policy language for allowing users to express the demands the gateway should impose. This enables them to express authorisation restrictions and impose requirements for the security level of messages within a session and privacy. It allows users to easily express these requirements for communication channels and devices.

A prototype was presented in Chapter 6. It includes all major modules found in the gateway's architecture and implements multiple providers for several enforcement submodules. The most notable omission in the prototype compared to the design are specific privacy modules for filtering or sticky policies.

An evaluation and discussion of the design were given in Chapter 7. It showed that, on a laptop-class device, the gateway adds a very acceptable overhead. Usually, well over 10 session set-ups can be done per second, unless complex anonymous authentication is required. Furthermore, regular messages within a session adds only a few milliseconds of delay, at most. This means that even on a less powerful device the system will be capable of handling all the traffic entering and leaving the smart home. The chapter also discussed how the gateway meets the other requirements, and looks back at the threat model to see how the functional security requirements are met.

The most important findings are that it is possible to effectively protect the smart home using the system's policy language and gateway. The former allows users to specify restrictions and requirements and hence be in control of their devices. The latter introduces some degree of centralisation, but as argued this is acceptable within the smart home context and makes it possible to effectively enforce security. The gateway offers an interface towards the devices' services and makes it possible to support a heterogeneous set of devices and security mechanisms. It specifies a protocol using a session set-up to give access to a service which then becomes usable in a secure way.

Some possible extensions were identified in the discussion, giving rise to opportunities for future work. A first modification is further integration of the full Priman framework for authentication. If mechanisms such as username/password authentication are included in the framework, it would be possible to rely completely on Priman for compatibility with different authentication providers. All policies governing authentication could then easily be described using the framework's standard.

Furthermore, the gateway could be extended for compatibility with other policy description languages and with standards for describing services. Examples are SAML and the Service Oriented Architecture (SOA). The gateway's modular design, however, means that these changes have limited impact on the rest of the system and that a complete overhaul is not necessary.

Another extension would be to boost the user experience by creating a graphical interface for describing the policies and for configuring the gateway. Such changes would only require isolated modifications to the gateway, but were deferred to future work as this is not the core of this research.

Finally, it should be noted that this thesis focused on creating a mechanism to enforce security rules when given the policies. A related subject is to choose these rules in accordance to preferences of several parties. This may include finding a compromise between desires of smart home residents and service providers. This is a whole different topic for research and was not considered here.

The bottom line and take-away message is that this thesis' gateway contributes to IoT development in a smart home context by supporting a heterogeneous set of devices and providing an interface towards them. Furthermore, it is independent of the exact data exchanged and allows interoperability between several security technologies. As a result, it will help developers to create IoT applications for a smart home in a secure, general and compatible way.

Appendices

Appendix A: Syntax of exchanged messages

This appendix shows the specific syntax set for some of the messages described in Section 4.10 and that are handled by the prototype implementation.

A secured payload looks as follows:

- No encryption: *plaintext*
- Asymmetric (public key) encryption: *plaintext*
- Symmetric encryption: nonce & payload (the & denotes concatenation, no separator is used)

The remainder of this appendix describes the exact syntax for each supported type of message:

- Session request {requested Resource}

  ➔ requestsession:*nameOfresource*|

- Session granted {requested Resource, abstract Resource address ID, session security information}

  ➔ sessiongranted|request:*requestedResource*|
  abstractID:*abstractID*|type:*typeC->GW*,*typeGW->C*|
  secureKeyPayload

  The *abstractID* should henceforth be used in messages exchanged to identify the session. The types describe what message security is used for traffic going to and coming from the gateway, respectively. The *secureKeyPayload* contains session key material. In case a different key is used for both directions, the key for communication from Consumer to gateway goes first, followed by the key for gateway to Consumer traffic (without separator).

- Session denied {requested Resource, description}

  ➔ sessiondenied|request:*requestedResource*|
  description:*description*|

- Session set-up {abstract Consumer address ID, session security information}

  ➔ sessionsetup|abstractID:*consumerID*|type:*typeGW->R*,
  typeR->GW|*secureKeyPayload*

  The *abstractID* should henceforth be used by the Resource to send messages to the Consumer. The types describe what message security is used. The *secureKeyPayload* contains session key material. In case a different key is used for both directions, the key for communication from gateway to Resource goes first, followed by the key for Resource to gateway traffic (without separator).

- `Session end {abstract ID communication, description}`

  ➔ `sessionend|abstractID:`*`abstractID`*`|description:`*`description`*

- `Authentication request {authentication type, challenge}`

  ➔ `authRequest|`*`authType`*`|responseAbstractID:`*`abstractID`*`|`*`challenge`*

  This message is sent by the gateway to another entity to request authentication. The *abstractID* should be used to send a response to the challenge. The *challenge* depends on the type of authentication used. This could simply be a nonce for password authentication, or could be more complicated and contain both an authentication policy and random challenge.

  ➔ `GWauthRequest|`*`authType`*`|abstractID:`*`abstractID`*`|`*`challenge`*

  This message is similar to the above, but is sent by another gateway to request the gateway to authenticate.

- `Authentication response {proof}`

  ➔ `authResponse|`*`abstractID`*`|`*`proof`*

  The *proof* is specific to the authentication mechanism used, and is interpreted correctly by the appropriate authentication module.

- `Authentication response & request {proof, authentication type, challenge}`

  ➔ `authResponseWithGWAuthRequest|`*`abstractID`*`|`*`challenge`*`|`*`proof`*

- `Regular message to gateway {abstract ID destination, metadata, payload}`

  ➔ `message|dest:`*`abstractIDOfReceiver`*`|`*`securePayload`*

- `Regular message from gateway {abstract ID sender, metadata, payload}`

  ➔ `message|sender:`*`abstractIDOfSender`*`|`*`securePayload`*

Appendix B: IEEE article

Beveiliging en privacy in het Internet der Dingen

Een overzicht van beveiligingstechnieken voor toestellen, gegevens en privacy in het Internet der Dingen

Dimitri Jonckers
KULeuven
Departement Computerwetenschappen
Leuven, België
dimitri.jonckers@student.kuleuven.be

Abstract -- **Het Internet der Dingen (Internet of Things/IoT) verruimt het internet doordat allerhande objecten er deel van gaan uitmaken. Deze heterogeniteit aan toestellen en bijhorende specificaties betekent echter dat het ontwikkelen van een algemeen raamwerk voor de bescherming van elk type object bemoeilijkt wordt. Dit artikel geeft een overzicht van bestaande oplossingen voor beveiliging van toestellen, data en privacy van gebruikers in een IoT context. De classificatie en selectie poogt technologieën te identificeren die bruikbaar zijn voor de bescherming van consumenten en huishoudens, bijvoorbeeld in het scenario van een slim huis. Dit staat toe om een systeem of raamwerk te ontwikkelen dat technologieën combineert om een complete bescherming van de perimeter van de gebruiker te garanderen.**

Keywords – Internet der Dingen, Internet of Things, IoT, ingebedde systemen, beveiliging, privacy, slim huis

I. Introductie

De impact en het raakoppervlak van het internet zijn in het laatste halve decennium sterk toegenomen. Smartphones maakten hun opmars, en vormen ondertussen in de Verenigde Staten de meest gebruikte methode om toegang tot het internet te verkrijgen [1]. Daarnaast werden *slimme toestellen* [2] op de markt geïntroduceerd, waardoor alledaagse objecten nu deel kunnen uitmaken van het internet. Zo werd deze toetreding al gemaakt door onder andere slimme Tv's [3], slimme thermostaten en slimme verlichting [4].

Dit fenomeen waar allerhande objecten online gaan, wordt het *Internet der Dingen* (*Internet of Things/IoT*) genoemd. Dit paradigma omvat het bouwen van een globale infrastructuur, gebaseerd op het internet, waarin fysische objecten een virtuele tegenhanger krijgen en met elkaar samenwerken om de mensheid bij te staan [5] [6]. Eind 2014 omvatte het IoT al 14 miljard apparaten, en tegen 2020 wordt dit aantal geschat tussen de 20 en 100 miljard toestellen [7].

Momenteel worden verschillende toepassingen van het IoT voorzien, en sommigen zijn al in ontwikkeling of in gebruik [8] [2]. Voorbeelden zijn slimme huizen, slim winkelen en allerhande geavanceerde productieomgevingen zoals slimme kantoren.

Door de verscheidenheid aan toestellen met uiteenlopende specificaties bevat het Internet der Dingen een inherente heterogeniteit. Sommige apparaten zijn computationeel gezien zeer gelimiteerd, met bijvoorbeeld RAM-geheugen van een aantal tientallen kilobytes.

De connectiviteit van al deze apparaten brengt uiteraard een aantal risico's met zich mee. De sterke band tussen de realiteit en de virtuele voorstelling in het Internet der Dingen betekent dat beveiliging en bescherming onmisbaar zijn.

Deze risico's zijn merkbaar voor de gebruiker, zoals getoond in een studie in 2014, waar 57% van de ongeveer 2000 ondervraagde consumenten bezorgd bleek over digitale inbraak en het lekken van data in het IoT [9]. Een andere studie van ongeveer 2000 consumenten in 2015 toonde dat bijna 80% zich zorgen maakt over privacy van hun gegevens bij dienstverleners [10].

Naast de perceptie is er ook de realiteit, waar blijkt dat beveiligingsproblemen inderdaad opduiken. In 2014 werden 10 slimme apparaten zoals tv's, watersproeiers en thermostaten getest, en bleken 8 van deze toestellen ontoereikende authenticatie en autorisatie te hebben. Daarnaast hadden 7 van de toestellen inadequate encryptie [11]. Deze problemen worden gebruikt om effectieve aanvallen uit te voeren op toestellen in omloop [12]. Een recent voorbeeld hiervan is het inbreken in slimme ketels om toegang te krijgen tot het netwerk van het slachtoffer [13]. De heterogeniteit van de toestellen betekent echter dat implementatie van sterke beveiligingsmechanismen niet haalbaar is op alle apparaten [14].

II. Categorisatie van technologieën

Het doel van deze paper is om een overzicht te geven van (beveiligings)problemen en gerelateerde oplossingen in het Internet der Dingen. In sectie III wordt daarom eerst gekeken naar geïdentificeerde problemen.

Bij het bestuderen van oplossingen werd gekeken naar technologieën die ontworpen werden om het IoT te beveiligen door een deel van de beveiligingsproblemen pogen op te lossen.

Sommige van de oplossingen werden ontwikkeld voor de context van draadloze sensornetwerken (WSNs), waarnaar al langer onderzoek bezig is, maar bieden ook mogelijkheden voor gebruik in een ruimere IoT-context. De omschrijving wordt gemaakt met het oog op de ontwikkeling van een algemeen systeem dat de beveiliging verzorgt van een bepaalde perimeter in een context van consumenten of huishoudens. Deze groep zal een belangrijk deel vormen van de IoT-gebruikers, zodat een bruikbare beveiliging zich opdringt. De perimeter wordt verondersteld een heterogene set van toestellen te bevatten, zodat de taak van het systeem wordt om de apparaten en de gegevens en privacy van de gebruikers in de perimeter te beschermen. Concreet willen we nagaan welke technieken bruikbaar zijn en deel kunnen uitmaken van een grotere oplossing. Dit generieke systeem kan een raamwerk zijn dat verschillende beveiligingsconcepten integreert, of kan een meer praktische integratie zijn die de combinatie in praktijk brengt.

In sectie IV wordt een categorisatie van bestaande oplossingen gegeven op basis van een aantal categorieën die bij het bekijken van problemen en beveiligingsvereisten in sectie III naar voor kwamen. Na het overzicht van de oplossingen volgt een discussie met betrekking tot de huidige stand der zaken en de bruikbaarheid van de technieken.

III. Problemen en uitdagingen

Deze sectie van de paper lijst een aantal problemen in het Internet der Dingen op. Eerst volgt een overzicht van algemene problemen, zoals ook terug te vinden in [15] en [8]. Nadien ligt de focus specifiek op beveiliging en privacy.

Een eerste algemeen probleem vindt zijn oorsprong in de veelheid aan toestellen en protocollen. Er bestaan reeds verschillende projecten om dit te harmoniseren, bijvoorbeeld 6LowPAN [16] en EPCGlobal [17], maar standaardisatie dient verder doorgedreven te worden in alle lagen van protocollen. Vooral op hogere lagen blijft de interoperabiliteit een probleem.

Daarnaast zijn er een aantal netwerkproblemen. Ten eerste moet adressering en identificatie van objecten voorzien zijn. Dit betekent bijvoorbeeld het omzetten van identiteiten, zoals van een 64-96 bit RFID-tag naar 128 bit IPv6-adressen. Identificatie moet ook mogelijk zijn aan de hand van bepaalde eigenschappen of groepen. Daarvoor dient het concept van de objectnamendienst (Object Name Service) doorgedreven te worden, om een bidirectionele omzetting te voorzien. De mobiliteit van de toestellen moet ook telkens in acht worden genomen.

Verder is een nieuw protocol voor de transportlaag nodig, aangezien TCP door zijn congestie-controle en buffering niet geschikt is voor het IoT. Dit is te wijten aan het typische patroon van vele korte pakketten die worden uitgewisseld in het IoT, die daarbovenop niet connectie-georiënteerd zijn. De selectie van gepaste protocollen zal een grote impact hebben op modellering van gegevensverkeer en op de kwaliteit van de diensten.

Een ander probleem is energie, met nood aan zuinige of passieve technologie om sensoren in flexibele omgevingen op te zetten. Vooral communicatie verbruikt in relatieve zin veel energie [18].

De nood voor beveiliging werd reeds aangehaald in de introductie, net als de heterogeniteit die in acht moet worden genomen. Dit heeft als effect dat er beperkingen zullen zijn, zoals het ontbreken van geheugenbescherming op sommige apparaten [15], en dat verschillende toestellen een andere set beveiligingsmechanismen zullen ondersteunen. Daarnaast wordt ook de nood aangehaald om de toestellen bruikbaar te houden voor gebruikers zonder technische expertise [19]. Een aantal nieuwe aanvalsmodellen zal waarschijnlijk opduiken [6], doordat digitale opponenten op hetzelfde moment zowel inwendig als uitwendig aan een systeem kunnen zijn [20]. Aangezien vele objecten zich op open locaties bevinden, zijn fysieke aanvallen denkbaar.

Cryptografie is een cruciale bouwsteen voor beveiliging [6], maar deze dient in een IoT context uiteraard niet enkel krachtig, maar ook licht te zijn. Hieraan gerelateerd, blijft sleutelbeheer nog steeds een probleem binnen het IoT.

Authenticatie en autorisatie zijn nodig om toegang tot apparaten en data te beheren, maar een uitgebreid algemeen raamwerk ontbreekt nog [21]. Verder blijven er uitdagingen wat betreft vertrouwen tussen partijen, bijvoorbeeld een mechanisme om over vertrouwen te onderhandelen [8], en foutentolerantie en aansprakelijkheid [5].

Daarnaast is er nood aan privacy, aangezien het door aanwezigheid van vele sensoren en goedkope opslag, voor een gebruiker moeilijk wordt om controle te bewaren over de eigen data [5]. Belangrijk is om op te merken dat de gebruiker niet alle sensoren rond zich controleert [15].

Er blijft nood aan een standaard beveiligingsinfrastructuur voor het Internet der Dingen [14], net als aan een globaal raamwerk en mechanisme voor privacy [8]. Voor privacy betekent dit zowel een aanpassing van de wetgeving, als ontwikkeling van technologieën om deze regelgeving af te dwingen [22]. Oplossingen zijn nodig, want deze problemen veroorzaken een grote vertraging voor het grootschalig gebruik van het IoT [23].

IV. Bestaande oplossingen voor beveiliging

Een aantal belangrijke concepten en noden voor beveiliging in een informatiesysteem, dus ook voor het Internet der Dingen, zijn confidentialiteit, authenticatie en autorisatie. Aangezien hiervoor eerst nood is aan ondersteuning voor versleuteling en sleutelbeheer, worden technieken en bestaande oplossingen in dit gebied eerst besproken. Nadien bekijken we oplossingen voor confidentialiteit en toegangscontrole. Privacy kan al naar voor komen bij andere beveiligingsconcepten, maar wordt hier apart beschreven om het als een volwaardige uitdaging naar voor te brengen en om technieken te beschrijven die specifiek op privacy gericht zijn. Een laatste groep oplossingen heeft te maken met omschrijving van beveiligingsvoorwaarden in de vorm van beleidsuitdrukkingen. Dit speelt een belangrijke rol in het ondersteunen van heterogene apparaten, waarvoor de

beveiligingseisen en –voorkeuren op een hoog niveau moeten kunnen worden omschreven dat interoperabiliteit toestaat en gebruiksvriendelijker is.

A. Cryptografie & sleutelbeheer

Het belang van encryptie werd al aangehaald als een belangrijke vereiste voor beveiliging. Zowel symmetrische als asymmetrische varianten zijn mogelijk. Die laatste is flexibeler en vergemakkelijkt de verdeling van sleutels, maar heeft een nadeel qua performantie [14].

Aangezien de beperkte hardware het uitvoeren van encryptie in software zwaar maakt, is er gekeken naar hardware-encryptie als alternatief [24]. Er werd een vergelijking gemaakt tussen een hardware-variant, op een CC2420 chip, en verschillende software-varianten voor het uitvoeren van symmetrische versleuteling. De hardware-versie bleek significant sneller en heeft een licht voordeel wat betreft geheugengebruik, maar dit is niet beschikbaar op alle platformen en biedt dus geen universele oplossing.

Een overzicht van sleutelbeheersystemen voor het IoT is beschikbaar in [25]. Deze bron geeft aan dat cryptografie op basis van publieke sleutels niet geschikt is door performantieproblemen. Verder geeft het verschillende varianten van vooraf verdeelde sleutels (pre-shared keys) weer, afhankelijk van het aantal sleutels aan client- en server-kant. Daarnaast worden mathematische systemen besproken die betere beveiliging bieden dan vooraf verdeelde sleutels, maar ook zwaarder zijn.

In [26] worden de varianten geclassificeerd als gedeelde meestersleutel, vooraf verdeelde en publieke sleutels. De gedeelde meestersleutel is weinig bruikbaar, aangezien één gedeelde sleutel zorgt dat heel het netwerk leesbaar is wanneer deze onthuld wordt. De publieke sleutel-aanpak wordt opnieuw onbruikbaar genoemd vanwege de performantie.

Daarentegen wordt publieke sleuteltechnologie wel gebruikt in het IoT, bijvoorbeeld in Sizzle [27], waar lichtere elliptische curve cryptografie (Elliptic Curve Cryptography /ECC) varianten worden gebruikt. Het gebruik hiervan zorgt voor een uitvoeringstijd van ongeveer 1 seconde voor de onderhandelingsfase van sleutels.

De aanpak met vooraf verdeelde sleutels is al vaker gebruikt in systemen. Een mogelijkheid is om een unieke gedeelde key te hebben met elke node, die op voorhand wordt verdeeld [28].

In het PAKA systeem voor WSNs [29] wordt een gateway met vooraf gedeelde sleutels gebruikt. Het is ook mogelijk om directe communicatie tussen nodes op te zetten, door een pad op te bouwen doorheen andere nodes. PAKA biedt bescherming tegen het overnemen van zowel een node als van de gateway, maar is niet uitgebreid getest op realistische systemen.

Andere systemen met vooraf verdeelde sleutels geven probabilistische bescherming, die veilig zijn zolang een bepaald aantal nodes niet overgenomen is [26].

Over het algemeen geeft een aanpak met vooraf verdeelde sleutels een betere performantie, maar biedt het niet dezelfde schaalbaarheid als publieke sleutelvarianten.

B. Confidentialiteit

Aangezien confidentialiteit gebruik maakt van encryptie, wat reeds besproken werd in de vorige sectie, kijken we onmiddellijk naar een aantal systemen.

Leap+ [30] gebruikt symmetrische sleutels om efficiënte confidentialiteit te garanderen. Dit systeem is echter enkel bruikbaar in statische omgevingen en is onveilig in het geval van overgenomen nodes tijdens de initiatiefase.

TinySec [31], en het nieuwere MiniSec, geven confidentialiteit op linkniveau in TinyOS [32].

Het Datagram Transport Layer Security (DTLS) [33] protocol geeft TLS functionaliteit, onder andere encryptie op het beveiligingsniveau van het internet, voor UDP-verbindingen. Dit kan worden gebruikt om op hoger niveau een punt-tot-punt architectuur op basis van de publieke sleutelinfrastructuur op te zetten [34], maar dit werd niet getest op de meest beperkte toestellen.

Sizzle biedt ook punt-tot-punt encryptie [27] met ECC publieke sleutelvarianten voor de initialisatie van sleutels. Dit systeem gebruikt een gateway, maar de nodes zelf zijn verantwoordelijk voor encryptie en het opzetten van sleutels.

Er zijn dus zowel punt-tot-punt als linkniveau oplossingen beschikbaar. Sommige systemen gebruiken een gateway, maar geen enkel hiervan maakt daar gebruik van om het werk van de nodes te optimaliseren.

C. Toegangscontrole

Voor authenticatie is het mogelijk om, op het laagste niveau, gebruik te maken van fysiek niet-dupliceerbare functies om unieke keys te genereren op zwakke toestellen. Dit maakt gebruik van unieke hardware eigenschappen die ontstaan in het productieproces, en die daardoor op een niet-dupliceerbare manier sleutelberekeningen kunnen doen [35]. Aangezien dit zich op erg laag niveau bevindt, is het moeilijk om te gebruiken, tenzij er ondersteuning vanuit de hardware ontwikkelaars wordt voorzien.

Op een hoger niveau zijn ook al een aantal authenticatie-systemen ontwikkeld. TinySEC werd reeds aangehaald als een oplossing op linkniveau, en er bestaan ook punt-tot-punt publieke sleutelvarianten [34] [27], soms gebruik makend van ECC [36]. In [28] wordt een authenticatiesysteem met vier stappen gepresenteerd, waarbij een gebruiker in interactie treedt met een node, die de gateway gebruikt om te authentiseren. De performantie van dit systeem wordt echter enkel theoretisch geanalyseerd.

Leap+ [30] gebruikt symmetrische sleutels en laat clusters toe voor flexibiliteit.

Wat betreft autorisatie in het Internet der Dingen, is attribuut-gebaseerde toegangscontrole een variant die vaak

verkend wordt, aangezien een klassieke aanpak moeilijk te beheren is in gedistribueerde toepassingen. Een studie van de performantie [37] toont echter aan dat attribuut-gebaseerde technieken zelfs voor smartphone-achtige toestellen te zwaar is, en veel IoT objecten vallen in een minder krachtige klasse.

Toch zijn een aantal architecturen ontwikkeld, zoals fdac, dat gedetailleerde toegangscontrole biedt met regels in een boomstructuur en logische (EN, OF, NIET) uitdrukkingen [38]. Dit systeem werd enkel getest op relatief krachtige objecten.

Een ander ontwerp [39] is specifiek op het IoT gericht, maar blijft theoretisch. Een implementatie en bijbehorende tests van onder andere performantie ontbreken.

Het is dus duidelijk dat een rijke expressiviteit nodig is, maar dat het afdwingen van de regels voor de toestellen zelf vaak niet haalbaar is.

D. Privacy

Volgens sommige bronnen moeten privacy- en beveiligingsdoelstellingen worden gebalanceerd [19], en werken voordelen op het ene gebied de het andere gebied vaak tegen. De *Privacy by design* principes [40] stellen echter dat dit soort valse afwegingen moet worden vermeden en dat integratie van de twee streefdoelen mogelijk is.

Privacy dient wel in acht te worden genomen bij het selecteren van beveiligingsmechanismen. Een voorbeeld is te vinden bij authenticatie, waar een klassieke aanpak met certificaten geen oplossing biedt voor privacy. Daarentegen bestaan er attribuut-gebaseerde technologieën, die toelaten om anoniem te authentiseren en autoriseren, door te bewijzen dat een gebruiker aan bepaalde voorwaarden voldoet zonder exacte waardes te onthullen [41]. Dit is bijvoorbeeld geïntegreerd in het ABC4Trust project [42], en in het voor IoT ontworpen ePass [43]. Zoals reeds gezegd, zijn de toestellen vaak zelf niet in staat de computationeel zware mechanismen toe te passen.

Om bepaalde eigenschappen, zoals gevoeligheid, van data aan te geven wanneer deze de eigen beveiligde perimeter verlaat, is het mogelijk om metadata toe te voegen. Deze wordt dan gebruikt door de ontvanger, bijvoorbeeld een dienstverlener, om gepaste beveiliging van de gegevens te voorzien. Het is mogelijk om datalabels (data tagging) te gebruiken om metadata toe te voegen aan datastromen [44], waardoor het mogelijk wordt om informatiestroomcontrole te introduceren [45]. Met 8 bits aan metadata verkrijgt men reeds heel wat vrijheid om data te verrijken, en dit systeem is haalbaar op zwakke PIC apparaten. Deze aanpak is dus efficiënt, maar is enkel effectief als de metadata nuttig gebruikt wordt aan de kant van de vertrouwde computerbasis (trusted computing base) bij de dienstverlener.

Verder is het gebruik van toegangscontrole belangrijk om de gebruiker te laten bepalen hoe zijn gegevens mogen worden gebruikt. In [46] specifieert een gebruiker dat gegevens van bepaalde apparaten toegankelijk zijn voor zichzelf, voor vrienden, of andere groepen via een website interface. In het scenario van een slim huis is dit soort toegangslijsten een haalbare aanpak.

Ten slotte is het mogelijk de gegevens zelf te anonimiseren. Bij *k-anonimiteit* zijn de gegevens niet te onderscheiden van data van (k-1) anderen. CASTLE [47] gebruikt het clusteren van data om stromen dynamisch te anonimiseren. Dit soort technieken is geschikt voor dienstverleners, die data van verschillende bronnen binnenkrijgen, maar is niet nuttig voor de kant van de consument, waar geen meerdere stromen beschikbaar zijn.

E. Beleidsuitdrukkingen

Verschillende manieren om beveiligingspreferenties uit te drukken zijn ontwikkeld. Sommige hiervan zijn specifiek voor toegangscontrole, terwijl anderen informatiesystemen omschrijven samen met beveiligingseisen voor componenten.

Het reeds vernoemde fdac staat uitgebreide expressies in boomstructuren van logische combinaties toe [38], en attribuut-gebaseerde systemen zoals ePass [43] staan door combinaties van attributen eveneens een rijke expressiviteit toe. ABC4Trust [42] ondersteunt attribuut-gebaseerde expressies voor authenticatie en autorisatie in XML-formaat.

De eXtensible Access Control Mark-up Language [48] (XACML) biedt eveneens omschrijving voor toegangscontrole-uitdrukkingen aan als een uitbreiding van XML.

Al deze systemen bieden expressiviteit voor een subset van uit te drukken beveiligingsvoorkeuren, waardoor integratie nodig is om een volledige set van abstracties toe te staan. Verder zijn veel van deze systemen niet voor de IoT-context ontwikkeld.

De Security Assertion Mark-up Language [49] (SAML) is een andere uitbreiding van XML, die bedoeld is om algemene beveiligingsvoorkeuren uit te drukken. Deze is niet specifiek voor het IoT ontworpen, maar probeert wel algemene expressies toe te staan, weliswaar met minder diepgaande expressiviteit voor bijvoorbeeld toegangscontrole.

De Service Component Architecture [50] (SCA) werd ontworpen voor omschrijving van componenten, en bevat een beleidsraamwerk waarin enkele beveiligingseisen kunnen worden uitgedrukt. Dit is een universele architectuur, en dus niet specifiek voor het IoT. De Architecture Analysis and Design Language [51] (AADL) is een andere manier om componenten te omschrijven, maar is wel specifiek ontworpen voor ingebedde systemen. Het bevat een aantal taaluitbreidingen voor onder andere beveiliging.

F. Discussie

De opdeling die gemaakt is, toont aan dat er op elk gebied al inspanningen zijn geleverd en dat concrete systemen ontwikkeld zijn.

Bij de problemen werd reeds omschreven dat een algemeen raamwerk ontbreekt, en dit wordt bevestigd wanneer de oplossingen worden bestudeerd. Zelfs al combineren sommige systemen oplossingen in meerdere categorieën, bijvoorbeeld

door integratie van toegangscontrole en confidentialiteit, ontbreekt een globaal systeem dat oplossingen samen brengt voor elke categorie en geschikt is voor alle IoT-toestellen. Dit zien we bijvoorbeeld bij authenticatiemechanismen, waar aanpak van de interoperabiliteit tussen toestellen met ondersteuning voor uiteenlopende authenticatiemethodes nog ontbreekt. Net zozeer zijn veel van de systemen niet of onvoldoende getest op apparaten met beperkte kracht, of blijkt dat ze niet geschikt zijn in deze situaties. Voor cryptografie en sleutelbeheer is vooral het efficiënt ondersteunen van publieke sleutels nog een probleem. Voor privacy is attribuut-gebaseerde toegangscontrole een belangrijk onderwerp, maar is de uitvoering er van op zwakke toestellen niet haalbaar. Metadata (labels) vormen een andere manier om privacy te verkrijgen, maar er zijn te weinig concrete systemen om het effect aan te tonen. Wat betreft beleidsomschrijvingen, is het vooral belangrijk de principes van technologie-onafhankelijke omschrijvingen van algemene informatiesystemen door te trekken naar het Internet der Dingen.

In een algemeen raamwerk voor beveiliging moet zowel confidentialiteit als toegangscontrole worden aangeboden, en dit betekent dat versleuteling en sleutelbeheer moeten worden ondersteund. Verder moet het mogelijk zijn privacy-vereisten af te dwingen. Dit komt overeen met de eerste 4 categorieën in het overzicht, waarvan technologieën moeten worden samengevoegd om een complete bescherming te verkrijgen. De beveiligingseisen op deze gebieden moeten op een eenduidige manier kunnen worden omschreven, agnostisch ten opzichte van de heterogeniteit van de toestellen. Dit komt overeen met categorie E.

Idealiter zou dit uitgebreide systeem gebruik kunnen maken van publieke sleutels voor flexibiliteit, en zou er ondersteuning zijn voor attribuut-gebaseerde toegangscontrole voor privacy. Wel is duidelijk dat deze versleuteling en toegangscontrole te zwaar zijn om op de toestellen zelf uit te voeren. Verschillende systemen gebruiken een gateway die soms bepaalde taken van de nodes overneemt, maar de gateway wordt nooit optimaal gebruikt om het werk op de nodes zelf te verlichten. In een uitgebreid beveiligingsraamwerk zou het gebruik hiervan de werklading van de toestellen zelf kunnen verlagen. Dit betekent een vorm van centralisatie, maar de efficiëntie-problemen van bestaande technieken zorgen dat dit gateway-concept zich opdringt [19].

V. Conclusie

Deze paper besprak beveiligings- en privacyproblemen en geassocieerde oplossingen in het Internet der Dingen. Een opdeling van ontwikkelde oplossing werd gemaakt in categorieën van cryptografie en sleutelbeheer, confidentialiteit, toegangscontrole en privacy. Deze werden besproken en er werd een voorstel gedaan over hoe deze categorieën samen komen in een universeel raamwerk, dat verschillende aspecten van beveiliging van een perimeter van toestellen en gegevens van consumenten verzorgt. Een laatste besproken categorie waren beleidsomschrijvingen voor systemen, dewelke ook gaan nodig zijn in dit raamwerk om een beschrijving van beveiligingseisen te geven op een hoog niveau.

VI. Referenties

[1] "2015 U.S. Digital Future in Focus," comScore, 2015.

[2] J. Gubbi, R. Buyya, S. Marusic en M. Palaniswami, „Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems,* vol. 29, nr. 7, pp. 1645-1660, 2013.

[3] "16 Best Smart TVs of 2015," (16 feb 2015). [Online]. Available: http://www.digitaltrends.com/home-theater/best-smart-tvs/.

[4] "The Best Smart Light Bulbs of 2015," (6 apr 2015). [Online]. Available: http://www.pcmag.com/article2/0,2817,2483488,00.asp.

[5] R. H. Weber, „Internet of Things: New security and privacy challenges," *Computer Law \& Security Review,* vol. 26, nr. 1, pp. 23-30, 2010.

[6] R. Roman, P. Najera en J. Lopez, „Securing the internet of things," *Computer,* vol. 44, nr. 9, pp. 51-58, 2011.

[7] "The Internet of Things: Making the most of the Second Digital Revolution," U.K. Government, 2014.

[8] D. Miorandi, S. Sicari, F. De Pellegrini en I. Chlamtac, „Internet of things: Vision, applications and research challenges," *Ad Hoc Networks,* vol. 10, nr. 7, pp. 1497-1516, 2012.

[9] "The Internet of Things: the Future of Consumer Adoption," Acquity Group, 2014.

[10] J. Groopman, "Consumer perceptions of privacy in Internet of Things," Altimeter, 2015.

[11] "Internet of Things Research Study," HP, http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA5-4759ENW.pdf, 2014.

[12] "Hacking the internet of things: from smart cars to toilets," (22 jul 2014). [Online]. Available: http://www.alphr.com/features/389920/hacking-the-internet-of-things-from-smart-cars-to-toilets.

[13] "Cracking WiFi Passwords By Hacking into Smart Kettles," (24 okt 2014). [Online]. Available: http://thehackernews.com/2015/10/hacking-wifi-password.html.

[14] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu en D. Qiu, „Security of the Internet of Things: perspectives and challenges," *Wireless Networks,* vol. 20, nr. 8, pp. 2481-2501, 2014.

[15] L. Atzori, A. Iera en G. Morabito, „The internet of things: A survey," *Computer networks,* vol. 54, nr. 15, pp. 2787-2805, 2010.

[16] "IPv6 over Low power WPAN (6lowpan)," [Online]. Available: https://datatracker.ietf.org/wg/6lowpan/.

[17] "EPCGlobal," GS1, [Online]. Available: http://www.gs1.org/epcglobal.

[18] M. A. Simplicio, B. T. De Oliveira, C. B. Margi, P. S. Barreto, T. C. Carvalho en M. N{\"a}slund, „Survey and comparison of message authentication solutions on wireless sensor networks," *Ad Hoc Networks,* vol. 11, nr. 3, pp. 1221-1236, 2013.

[19] T. Polk en S. Turner, „Security challenges for the internet of things," in *Workshop on Interconnecting Smart Objects with the Internet, Prague*, 2011.

[20] R. Roman, J. Zhou en J. Lopez, „On the features and challenges of security and privacy in distributed internet of things," *Computer Networks,* vol. 57, nr. 10, pp. 2266-2279, 2013.

[21] S. Sicari, A. Rizzardi, L. Grieco en A. Coen-Porisini, „Security, privacy and trust in Internet of Things: The road ahead," *Computer Networks,* vol. 76, pp. 146-164, 2015.

[22] M. Langheinrich, „Privacy by design: principles of privacy-aware ubiquitous systems," in *Ubicomp 2001: Ubiquitous Computing*, 2001.

[23] "2012 Hype Cycle for the Internet of Things," Gartner, 2012.

[24] M. Healy, T. Newe en E. Lewis, „Analysis of hardware encryption versus software encryption on wireless sensor network motes," in *Smart Sensors and Sensing Technology*, Springer, 2008, pp. 3-14.

[25] R. Roman, C. Alcaraz, J. Lopez en N. Sklavos, „Key management systems for sensor networks in the context of the Internet of Things," *Computers \& Electrical Engineering,* vol. 37, nr. 2, pp. 147-159, 2011.

[26] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz en A. Khalili, „A pairwise key predistribution scheme for wireless sensor networks," *ACM Transactions on Information and System Security (TISSEC),* vol. 8, nr. 2, pp. 228-258, 2005.

[27] V. Gupta, M. Wurm, Y. Zhu, M. Millard, S. Fung, N. Gura, H. Eberle en S. C. Shantz, „Sizzle: A standards-based end-to-end security architecture for the embedded internet," *Pervasive and Mobile Computing,* vol. 1, nr. 4, pp. 425-445, 2005.

[28] M. Turkanovic, B. Brumen en M. Hölbl, „A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion," *Ad Hoc Networks,* vol. 20, pp. 96-112, 2014.

[29] S. Tripathy, „Effective pair-wise key establishment scheme for wireless sensor networks," in *Proceedings of the 2nd international conference on Security of information and networks*, 2009.

[30] S. Zhu, S. Setia en S. Jajodia, „LEAP+: Efficient security mechanisms for large-scale distributed sensor networks," *ACM Transactions on Sensor Networks (TOSN),* vol. 2, nr. 4, pp. 500-528, 2006.

[31] „TinySec: a Link Layer Security Architecture for Wireless Sensor Networks," 2004.

[32] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler en K. Pister, „System architecture directions for networked sensors," in *ACM SIGOPS operating systems review*, 2000.

[33] IETF, „Datagram Transport Layer Security Version 1.2," (2012). [Online]. Available: https://tools.ietf.org/html/rfc6347.

[34] T. Kothmayr, C. Schmitt, W. Hu, M. Brünig en G. Carle, „DTLS based security and two-way authentication for the Internet of Things," *Ad Hoc Networks,* vol. 11, nr. 8, pp. 2710-2723, 2013.

[35] A. Cherkaoui, L. Bossuet, L. Seitz, G. Selander en R. Borgaonkar, „New paradigms for access control in constrained environments," in *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2014 9th International Symposium on*, 2014.

[36] J. Liu, Y. Xiao en C. P. Chen, „Authentication and access control in the internet of things," in *2012 32nd International Conference on Distributed Computing Systems Workshops*, 2012.

[37] X. Wang, J. Zhang, E. M. Schooler en M. Ion, „Performance evaluation of attribute-based encryption: Toward data privacy in the IoT," in *Communications (ICC), 2014 IEEE International Conference on*, 2014.

[38] S. Yu, K. Ren en W. Lou, „FDAC: Toward fine-grained distributed data access control in wireless sensor networks," *Parallel and Distributed Systems, IEEE Transactions on,* vol. 22, nr. 4, pp. 673-686, 2011.

[39] N. Ye, Y. Zhu, R.-c. Wang, R. Malekian en L. Min, „An efficient authentication and access control scheme for perception layer of internet of things," *Int. J. Appl. Math. Inf. Sci,* vol. 8, pp. 1617-1624, 2014.

[40] A. Cavoukian en others, „Privacy by design: The 7 foundational principles," *Information and Privacy Commissioner of Ontario, Canada,* 2009.

[41] A. Alcaide, E. Palomar, J. Montero-Castillo en A. Ribagorda, „Anonymous authentication for privacy-preserving IoT target-driven applications," *Computers \& Security,* vol. 37, pp. 111-123, 2013.

[42] P. Bichsel, J. Camenisch, M. Dubovitskaya, R. R. Enderlein, S. Krenn, I. Krontiris, A. Lehmann, G. Neven, J. D. Nielsen, C. Paquin en others, „D2. 2 Architecture for Attribute-based Credential Technologies-Final Version," *ABC4Trust-Deliverable to the European Commision,* 2014.

[43] J. Su, D. Cao, B. Zhao, X. Wang en I. You, „ePASS: An expressive attribute-based signature scheme with privacy and an unforgeability guarantee for the Internet of Things," *Future Generation Computer Systems,* vol. 33, pp. 11-18, 2014.

[44] R. V. Nehme, E. A. Rundensteiner en E. Bertino, „Tagging stream data for rich real-time services," *Proceedings of the VLDB Endowment,* vol. 2, nr. 1, pp. 73-84, 2009.

[45] D. Evans en D. M. Eyers, „Efficient data tagging for managing privacy in the internet of things," in *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, 2012.

[46] X. Huang, R. Fu, B. Chen, T. Zhang en A. Roscoe, „User interactive internet of things privacy preserved access control," in *Internet Technology And Secured Transactions, 2012 International Conference for*, 2012.

[47] J. Cao, B. Carminati, E. Ferrari en K.-L. Tan, „Castle: Continuously anonymizing data streams," *Dependable and Secure Computing, IEEE Transactions on,* vol. 8, nr. 3, pp. 337-352, 2011.

[48] T. Moses en others, „Extensible access control markup language (xacml) version 2.0," *Oasis Standard,* vol. 200502, 2005.

[49] N. Ragouzis, J. Hughes, R. Philpott, E. Maler, P. Madsen en T. Scavo, „Security assertion markup language (saml) v2. 0 technical overview," *OASIS Comittee Draft,* vol. 2, 2008.

[50] M. Beisiegel, H. Blohm, D. Booz, J.-J. Dubray, A. C. Interface21, M. Edwards, D. Ferguson, J. Mischkinsky, M. Nally en G. Pavlik, „Service component architecture," *Building systems using a Service Oriented Architecture. BEA, IBM, Interface21, IONA, Oracle, SAP, Siebel, Sybase, white paper, version,* vol. 9, 2007.

[51] P. H. Feiler, D. P. Gluch en J. J. Hudak, „The architecture analysis & design language (AADL): An introduction," 2006.

Appendix C: Poster

KU LEUVEN

A security mechanism for the Internet of Things in a smart home context

FACULTEIT
INGENIEURSWETENSCHAPPEN

Master
Computer Science
Engineering

Master's thesis
Dimitri Jonckers

Promotor
Bart De Decker

The Internet of Things

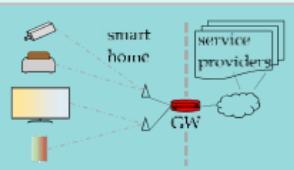
- 20 – 100 billion IoT devices by 2020
- Many aimed at consumer market
- Many lack appropriate security
- Resource-constrained devices make security difficult

Goals

- Develop a system to enforce security and privacy
- Support heterogeneity of the IoT
- Offer an interface towards the smart home's IoT services

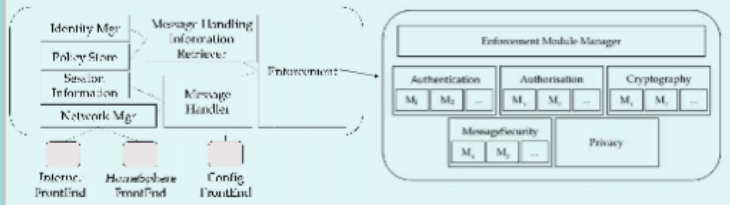
Policy language

- Communication channel requirements (confidentiality, authentication, privacy)
- Resource access rules



The gateway

- Security enforcement based on policies



- Messages *Resources – Consumers*
  - Session set-up: service mapping, key material for devices individually
  - Followed by regular messages within a session
- Modular *Enforcement* component
  - Integrate several security providers
  - Extensible with more protection mechanisms

Results

- Performance: small overhead, feasible for smart home
  - > 10 set-ups/second
  - 100s of regular messages/second
- Interface towards devices, service discovery
- Support IoT heterogeneity
  - Several security mechanisms
  - Specify different requirements for subchannels

Bibliography

[1] "2015 U.S. Digital Future in Focus," comScore, 2015.

[2] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems,* vol. 29, no. 7, pp. 1645-1660, 2013.

[3] "16 Best Smart TVs of 2015," (2015, feb 16). [Online]. Available: http://www.digitaltrends.com/home-theater/best-smart-tvs/.

[4] "The Best Smart Light Bulbs of 2015," (2015, apr 6). [Online]. Available: http://www.pcmag.com/article2/0,2817,2483488,00.asp.

[5] R. H. Weber, "Internet of Things: New security and privacy challenges," *Computer Law & Security Review,* vol. 26, no. 1, pp. 23-30, 2010.

[6] R. Roman, P. Najera and J. Lopez, "Securing the internet of things," *Computer,* vol. 44, no. 9, pp. 51-58, 2011.

[7] L. Atzori, A. Iera and G. Morabito, "The internet of things: A survey," *Computer networks,* vol. 54, no. 15, pp. 2787-2805, 2010.

[8] "The Internet of Things: Making the most of the Second Digital Revolution," U.K. Government, 2014.

[9] "2015 Hype Cycle for Emerging Technologies," Gartner, 2015.

[10] D. Miorandi, S. Sicari, F. De Pellegrini and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks,* vol. 10, no. 7, pp. 1497-1516, 2012.

[11] "The Internet of Things: the Future of Consumer Adoption," Acquity Group, 2014.

[12] J. Groopman, "Consumer perceptions of privacy in Internet of Things," Altimeter, 2015.

[13] "Internet of Things Research Study," HP, http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA5-4759ENW.pdf, 2014.

[14] "Hacking the internet of things: from smart cars to toilets," (2014, jul 22). [Online]. Available: http://www.alphr.com/features/389920/hacking-the-internet-of-things-from-smart-cars-to-toilets.

[15] "Cracking WiFi Passwords By Hacking into Smart Kettles," (2014, oct 24). [Online]. Available: http://thehackernews.com/2015/10/hacking-wifi-password.html.

[16] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu and D. Qiu, "Security of the Internet of Things: perspectives and challenges," *Wireless Networks,* vol. 20, no. 8, pp. 2481-2501, 2014.

[17] M. Langheinrich, "Privacy by design: principles of privacy-aware ubiquitous systems," in *Ubicomp 2001: Ubiquitous Computing*, 2001.

[18] R. Roman, J. Zhou and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks,* vol. 57, no. 10, pp. 2266-2279, 2013.

[19] T. Polk and S. Turner, "Security challenges for the internet of things," in *Workshop on Interconnecting Smart Objects with the Internet, Prague*, 2011.

[20] "IPv6 over Low power WPAN (6lowpan)," [Online]. Available: https://datatracker.ietf.org/wg/6lowpan/.

[21] "EPCGlobal," GS1, [Online]. Available: http://www.gs1.org/epcglobal.

[22] M. A. Simplicio, B. T. De Oliveira, C. B. Margi, P. S. Barreto, T. C. Carvalho and M. N{\"a}slund, "Survey and comparison of message authentication solutions on wireless sensor networks," *Ad Hoc Networks,* vol. 11, no. 3, pp. 1221-1236, 2013.

[23] "2012 Hype Cycle for the Internet of Things," Gartner, 2012.

[24] S. Sicari, A. Rizzardi, L. Grieco and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Computer Networks,* vol. 76, pp. 146-164, 2015.

[25] M. Healy, T. Newe and E. Lewis, "Analysis of hardware encryption versus software encryption on wireless sensor network motes," in *Smart Sensors and Sensing Technology*, Springer, 2008, pp. 3-14.

[26] R. Roman, C. Alcaraz, J. Lopez and N. Sklavos, "Key management systems for sensor networks in the context of the Internet of Things," *Computers & Electrical Engineering,* vol. 37, no. 2, pp. 147-159, 2011.

[27] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," *ACM Transactions on Information and System Security (TISSEC),* vol. 8, no. 2, pp. 228-258, 2005.

[28] V. Gupta, M. Wurm, Y. Zhu, M. Millard, S. Fung, N. Gura, H. Eberle and S. C. Shantz, "Sizzle: A standards-based end-to-end security architecture for the embedded internet," *Pervasive and Mobile Computing,* vol. 1, no. 4, pp. 425-445, 2005.

[29] M. Turkanovic, B. Brumen and M. Hölbl, "A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion," *Ad Hoc Networks,* vol. 20, pp. 96-112, 2014.

[30] S. Tripathy, "Effective pair-wise key establishment scheme for wireless sensor networks," in *Proceedings of the 2nd international conference on Security of information and networks*, 2009.

[31] S. Zhu, S. Setia and S. Jajodia, "LEAP+: Efficient security mechanisms for large-scale distributed sensor networks," *ACM Transactions on Sensor Networks (TOSN),* vol. 2, no. 4, pp. 500-528, 2006.

[32] "TinySec: a Link Layer Security Architecture for Wireless Sensor Networks," 2004.

[33] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler and K. Pister, "System architecture directions for networked sensors," in *ACM SIGOPS operating systems review*, 2000.

[34] IETF, "Datagram Transport Layer Security Version 1.2," (2012). [Online]. Available: https://tools.ietf.org/html/rfc6347.

[35] T. Kothmayr, C. Schmitt, W. Hu, M. Brünig and G. Carle, "DTLS based security and two-way authentication for the Internet of Things," *Ad Hoc Networks,* vol. 11, no. 8, pp. 2710-2723, 2013.

[36] A. Cherkaoui, L. Bossuet, L. Seitz, G. Selander and R. Borgaonkar, "New paradigms for access control in constrained environments," in *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2014 9th International Symposium on*, 2014.

[37] J. Liu, Y. Xiao and C. P. Chen, "Authentication and access control in the internet of things," in *2012 32nd International Conference on Distributed Computing Systems Workshops*, 2012.

[38] X. Wang, J. Zhang, E. M. Schooler and M. Ion, "Performance evaluation of attribute-based encryption: Toward data privacy in the IoT," in *Communications (ICC), 2014 IEEE International Conference on*, 2014.

[39] S. Yu, K. Ren and W. Lou, "FDAC: Toward fine-grained distributed data access control in wireless sensor networks," *Parallel and Distributed Systems, IEEE Transactions on,* vol. 22, no. 4, pp. 673-686, 2011.

[40] N. Ye, Y. Zhu, R.-c. Wang, R. Malekian and L. Min, "An efficient authentication and access control scheme for perception layer of internet of things," *Int. J. Appl. Math. Inf. Sci,* vol. 8, pp. 1617-1624, 2014.

[41] A. Cavoukian and others, "Privacy by design: The 7 foundational principles," *Information and Privacy Commissioner of Ontario, Canada,* 2009.

[42] A. Alcaide, E. Palomar, J. Montero-Castillo and A. Ribagorda, "Anonymous authentication for privacy-preserving IoT target-driven applications," *Computers & Security,* vol. 37, pp. 111-123, 2013.

[43] P. Bichsel, J. Camenisch, M. Dubovitskaya, R. R. Enderlein, S. Krenn, I. Krontiris, A. Lehmann, G. Neven, J. D. Nielsen, C. Paquin and others, "D2. 2 Architecture for Attribute-based Credential Technologies-Final Version," *ABC4Trust-Deliverable to the European Commision,* 2014.

[44] J. Su, D. Cao, B. Zhao, X. Wang and I. You, "ePASS: An expressive attribute-based signature scheme with privacy and an unforgeability guarantee for the Internet of Things," *Future Generation Computer Systems,* vol. 33, pp. 11-18, 2014.

[45] R. V. Nehme, E. A. Rundensteiner and E. Bertino, "Tagging stream data for rich real-time services," *Proceedings of the VLDB Endowment,* vol. 2, no. 1, pp. 73-84, 2009.

[46] D. Evans and D. M. Eyers, "Efficient data tagging for managing privacy in the internet of things," in *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, 2012.

[47] X. Huang, R. Fu, B. Chen, T. Zhang and A. Roscoe, "User interactive internet of things privacy preserved access control," in *Internet Technology And Secured Transactions, 2012 International Conference for*, 2012.

[48] J. Cao, B. Carminati, E. Ferrari and K.-L. Tan, "Castle: Continuously anonymizing data streams," *Dependable and Secure Computing, IEEE Transactions on,* vol. 8, no. 3, pp. 337-352, 2011.

[49] W. R. Claycomb and D. Shin, "A novel node level security policy framework for wireless sensor networks," *Journal of Network and Computer Applications,* vol. 34, no. 1, pp. 418-428, 2011.

[50] T. Moses and others, "Extensible access control markup language (xacml) version 2.0," *Oasis Standard,* vol. 200502, 2005.

[51] N. Ragouzis, J. Hughes, R. Philpott, E. Maler, P. Madsen and T. Scavo, "Security assertion markup language (saml) v2. 0 technical overview," *OASIS Comittee Draft,* vol. 2, 2008.

[52] A. Taherkordi, F. Loiret, A. Abdolrazaghi, R. Rouvoy, Q. Le-Trung and F. Eliassen, "Programming sensor networks using REMORA component model," in *Distributed Computing in Sensor Systems*, Springer, 2010, pp. 45-62.

[53] M. Beisiegel, H. Blohm, D. Booz, J.-J. Dubray, A. C. Interface21, M. Edwards, D. Ferguson, J. Mischkinsky, M. Nally and G. Pavlik, "Service component architecture," *Building systems using a Service Oriented Architecture. BEA, IBM, Interface21, IONA, Oracle, SAP, Siebel, Sybase, white paper, version,* vol. 9, 2007.

[54] P. H. Feiler, D. P. Gluch and J. J. Hudak, "The architecture analysis & design language (AADL): An introduction," 2006.

[55] A. Put, I. Dacosta, M. Milutinovic and B. De Decker, "PriMan: Facilitating the Development of Secure and Privacy-Preserving Applications.," in *SEC*, 2014.

[56] "Bouncy Castle Crypto APIs," [Online]. Available: http://www.bouncycastle.org.

[57] Oracle, "Java Cryptography Architecture (JCA) Reference Guide," [Online]. Available: http://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html.

[58] IBM, "IBM: Identity Mixer," [Online]. Available: http://research.ibm.com/labs/zurich/idemix.

[59] D. McGrew and J. Viega, "The Galois/counter mode of operation (GCM)," *Submission to NIST. http://csrc. nist. gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec. pdf,* 2004.

Master thesis filing card

Student: Dimitri Jonckers

Title: A security mechanism for the Internet of Things in a smart home context

Nederlandse titel: Een beveiligingsmechanisme voor het Internet der Dingen in de context van een smart home

UDC: 681.3

Abstract: The Internet of Things (IoT) broadens the scope of the internet to tens of billions of devices. Because of the heterogeneity of the connected objects and their specifications, it becomes difficult to craft a general framework for the IoT and its security. This thesis aims to provide security and privacy for Internet of Things devices in a smart home setting.

At first, a literature review was conducted which yielded an overview of problems in the Internet of Things, with a specific focus on security. Furthermore, this study investigated several existing solutions for securing devices, data and privacy in an IoT context.

The core contribution is the development of a gateway which stands at the border of the smart home, between the home's devices and outside users such as service providers. It is capable of providing confidentiality, authentication, authorisation and privacy and can take care of this on behalf of constrained devices which are incapable of securing themselves. The modular architecture includes several providers for each security domain, and can easily be extended in order to support more mechanisms.

The interaction model starts with a session set-up between Consumers and Resources. This gives the gateway the capability of service discovery, as it provides an interface to services offered by the devices. Afterwards, messages are exchanged within a session and the gateway ensures the security of the communication subchannels.

The gateway enforces security based on policies which the user configures. A second contribution of this thesis is a policy description language designed for this purpose. It allows users to specify requirements for their devices and communication channels with other objects and parties, possibly located outside of the smart home.

Performance test results show a limited impact on performance, allowing tens of session set-ups per second and several hundreds of messages per second to be exchanged within a session. Hence, the gateway provides security in the IoT in a performant manner. The flexibility in supporting several security providers and the possibility to address services uniformly imply that the gateway will aid developers to securely create applications for the heterogeneous Internet of Things.

Thesis submitted for the degree of Master of Science in Engineering: Computer Science, specialisation Distributed Systems

Thesis supervisor: Prof. dr. ir. Bart De Decker

Assessors: Prof. dr. ir. Y. Berbers
Dr. A. Kimmig

Mentor: Ir. A. Put