Chapter 9

Securing the Home Energy Management Platform

Søren Aagaard Mikkelsen and
Rune Hylsberg Jacobsen

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/62923

Abstract

Energy management in households gets increasingly more attention in the struggle to integrate more sustainable energy sources. Especially in the electrical system, smart grid systems are envisioned to be part in the efforts towards a better utilisation of the energy production and distribution infrastructure. The Home Energy Management System (HEMS) is a critical infrastructure component in this endeavour. Its main goal is to enable energy services utilising smart devices in the households based on the interest of the residential consumers and external actors. With the role of being both an essential link in the communication infrastructure for balancing the electrical grid and a surveillance unit in private homes, security and privacy become essential to address. In this chapter, we identify and address potential threats Home Energy Management Platform (HEMP) developers should consider in the progress of designing architecture, selecting hardware and building software. Our approach starts with a general view of the involved stakeholders and the HEMS. Given the system overview, a threat model is constructed from the HEMP developer's point of view. Based on the threats that have been detected, possible mitigation strategies are proposed taking into account the state of the art of technology for securing platforms.

Keywords: smart grid, IoT, security, privacy, cloud, web service, service-oriented architecture

1. Introduction

A smart grid is a vision of a more intelligent electrical infrastructure. It is envisioned to provide a more reliable and effective electrical grid in the process of integrating more intermittent renewable energy sources, like wind power and solar power. This will demand for systems that will continuously monitor and manage end-points of the grid, for example, an end-point as the residential sector that consumes around 31% of the electricity worldwide [1][1]. These systems are known as Energy Management Systems (EMSs)[2]. EMSs aim at utilising the grid capacity more efficiently in terms of instantaneous demand and generation. This includes flattening the peak demand by giving the demand-side awareness about the time periods it is "smart" to consume energy [2].

Ensuring security and privacy are becoming important issues to address with the deployment of smart meters. Smart meters are devices that record the electric consumption and production and communicates this information automatically to the utility. Research has shown that information from smart meters can reveal personal habits and disclose details about the residents living there. Furthermore, the situation exacerbates if additional meters and actuators are installed on the most consuming devices for Direct Load Control (DLC) in the demand response paradigm. Therefore, security and privacy enhanced system architectures suggest, a Home Energy Management System (HEMS) that runs locally inside the residential home, among other things to resolve privacy issues.

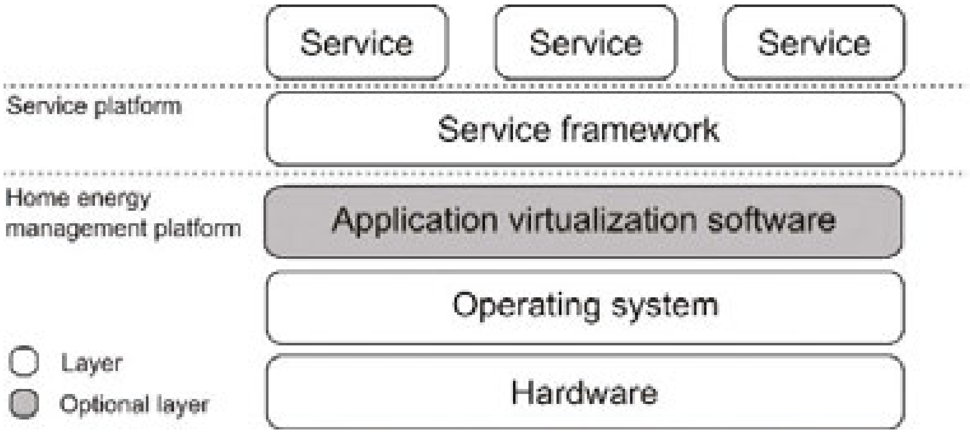


Figure 1. The considered stack for a HEMS.

This chapter describes security and privacy issues for a HEMS in an SOA. It focuses on the Home Energy Management Platform (HEMP) provider [also in other contexts referred to as the Original Equipment Manufacturer (OEM)]. A HEMP provider is responsible for the platform, which the software developers and service providers can deploy their software and services on as seen in **Figure 1**. A HEMP provider creates the platform that facilitates security and privacy in several layers. Following a holistic approach, a threat modelling approach is used that considers the interdependencies between stakeholders of the system before identifying the threats the system faces. A risk assessment is made on the identified threats, and

[1] Percentage based on the average electricity consumption from residential domains in Spain, Norway and Denmark.

[2] EMSs can refer to not only systems that support the operation of the electrical grid on a transmission and distribution level but also systems that automatically control and monitor energy usage in buildings.

possible mitigation strategies are presented based on a review of the state of the art in hardware security. Based on the threat analysis, recommendations to create a secure HEMS are provided.

2. Background and related work

A preliminary review of the domain is presented before developing a threat model for the HEMP. The review includes a description of general threat modelling approaches together with commonly used diagrams to perform the analysis.

2.1. Threat modelling approaches

Threat modelling is a systematic approach that supports the process of finding the appropriate technical solutions for enforcing security and privacy. It considers the potential threats with the greatest impact and addresses these first.

There are generally three approaches to threat modelling [3]:

- **Assets-centric:** Assets define things that need protection, or things the attacker wants or something in between those two. The approach focusses on defining the assets of the system and afterwards, analysing the flow of assets within an organisation. The model can then be applied with an attack set like STRIDE [3], which is a mnemonic for *Spoofing*, *Tampering*, *Repudiation*, *Information disclosure*, *Denial of service* and *Elevation of Privilege*.

- **Attacker-centric:** Envisioned attackers are defined in terms of resources and capabilities. An attacker model identifies a number of scenarios where an attacker can break the system both directly and indirectly. Directly, in terms of exploiting, for example, software vulnerabilities, and indirectly, in terms of exploiting humans that have privileges within the system.

- **System-centric**[3]**:** The approach attempts to look at the designed system as a whole. As the system is being designed, the threat model is continuously being developed or updated. The threat model includes finding threats using, for example, STRIDE and attack trees [3].

When comparing the approaches, the system-centric approach more tightly couples with the development process, whereas the asset-centric approach and attacker-centric approach can be considered more “detached” and are typically performed separate from the software development.

2.1.1. Diagrams for threat modelling

The typical diagrams for revealing threats focus on describing the domain or the data flows within the considered system. The goals of these diagrams are to create a common view of the system and to make the communication paths visible. These diagrams provide an abstraction

[3] Is often called *software-centric*, but for the sake of generality we use the term *system-centric*.

of the system which allows system architects and software developers to create a common view of the system and define the system boundaries.

The Unified Modelling Language (UML) [4] is general-purpose modelling language that specifies a set of diagrams for modelling systems. It provides a standardised way of modelling systems and provides two fundamental representations of a system: behavioural models and structural models. Behavioural models, such as use case diagrams, can be beneficial to identify stakeholders and describe a proposed functionality. Structural models, such as class diagrams, are useful for describing the domain which includes objects, attributions, operations and relationships.

Data Flow Diagrams (DFDs) have traditionally been used for threat modelling, since the threat problems tend to follow the data flow not the control flow [3]. It is used for describing how data enter and leave the system. The basic elements for modelling the system are shown in **Table 1**. The system is typically modelled with specific scope, where the external system that provides the system with data is modelled using the *external entity* element. The system of interest consists of a number of *processes* and *data stores* with *data flow* between them. To define the boundary between trust elements and non-trusted elements, DFD also includes trust boundaries. Trust boundaries visualise where data flows intersect between the system and a malicious actor.

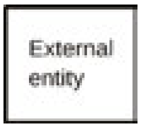
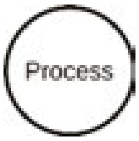
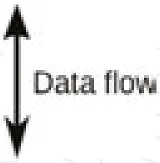
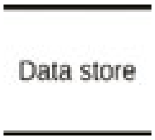
Element	Model	Description	Examples
External entity		Outside scope	A stakeholder delivering or receiving to the system
Process		Execution of system element	Executable code or hardware function
Data flow		Communication between elements	RPC calls, system calls, HTTP
Data store		Entity that stores data	Files, memory, database

Table 1. The basic elements of a DFD.

By modelling the specific system with DFD elements, it is possible to map these elements to the STRIDE threats. For instance, an external entity for the system can generate a *spoofing* threat, that is, an external entity can pretend to be something or someone else than originally thought of. By enforcing an authentication mechanism, it is possible to mitigate the threat if the external entity was trying to *spoof* a person or a thing.

2.2. Home energy management software platforms

As depicted in Figure 1, the HEMS consists of a software platform that provides the middleware for interoperable communication between devices and services. In the following, two open source EMSs are presented. Both use the OSGi (Open Service Gateway Initiative) architecture that specifies a modular and service-based software platform implemented in Java.

2.2.1. OpenHAB

The openHAB project[4] is an open software framework that focusses on enabling home automation by joining systems and technologies available in the smart home domain. It allows for automation rules and offers a single user interface for all such systems. The openHAB software is capable of running on any device that supports the Java Virtual Machine (JVM). The architecture consists of three main components: the OSGi framework, openHAB Core Components and openHAB Add-ons. The OSGi framework and the openHAB Core Components represent the internal communication infrastructure and base libraries. The openHAB Add-ons include a large set of protocol bindings that map between other home automation protocols to an abstract data model. It includes an "item" repository, where an item can be interpreted as an abstraction for a property a device can actuate. The openHAB framework provides mechanisms for securing the communication to the openHAB software platform, but does not have additional security features beside what the OSGi framework can provide.

2.2.2. OGEMA

Open Gateway Energy Management (OGEMA)[5] is an open software framework for smart grid services. It represents a system that supports building automation control and energy management for residential and industrial environments. The framework rests on a hardware-independent platform with a common execution environment for all deployable services. OGEMA uses the OSGi Java framework enabling a modular and dynamic software environment. The OGEMA framework consists of the following entities:

- **OGEMA Services**: Common services like resource administration, user interface, web interface and data logging, but also a common data model and access control.
- **Resources**: Data structures according to the data model representing the connected devices.
- **Applications**: Services for, for example, price-based management and energy analysis.
- **Communication drivers**: Drivers for supporting multiple communication media, for example, ZigBee and EEBus.

[4] http://www.openhab.org/

[5] http://www.ogema.org/

The OGEMA architecture embeds three “modes” of security: standard infrastructure, a controlled environment and proof of security [5]. Furthermore, it isolates third-party applications which can be integrity checked using the public key of the service provider. This is similar to Android's permission handling [5]. Since OGEMA framework is restricted to security capability of the OSGi Java framework and the JVM, it is possible for a malicious component to freeze the platform by allocating too much memory or changing shared variables. A possible solution to this issue is the I-JVM presented in [6].

2.3. Threat analysis of the smart grid domain

Cyber-Physical Systems (CPSs) and Internet of Things (IoTs) are two research domains that overlap and share similar challenges with the smart grid domain. In [7], they identify four key challenges in designing a secure IoT which include data management, identity management, trust management and privacy. Based on these challenges, they propose hardware-based mitigation strategies that include Physical Unclonable Function (PUF) for data provenance, integrity and identity management. However, they present a non-systematic approach where the stakeholders are not clearly identified. In reference [8], they consider threat modelling issues of CPSs. The authors provide a generic model of a CPS and present a case study with a road vehicle. In reference [9], researchers present a survey of security theories, analysis, simulation and application fields but without giving recommendations.

In reference [10], they present a structured method for identifying security threats in smart home scenarios. It is based on a context pattern for the elicitation of domain knowledge for the smart home domain. Using the context pattern for creating DFDs for the smart domain, they identify the entry points and vulnerabilities. This allows them to define the attack paths from entry point to the assets of the system. Their method is sound and structured but lacks possible threat mitigation strategies.

3. Threat modelling of a home energy management platform

Security experts are encouraging designers to make explicit statements about the security assumptions, when designing and implementing systems. Defining the security assumptions is the first step in order to find the vulnerabilities of a system. With a description of the vulnerabilities, it is possible to assess the risks a given design exposes. Security technologies can then be enforced where the vulnerabilities need to be addressed.

3.1. Methodology

A systematic approach is necessary to discover all the vulnerabilities of a system [10]. Our approach for designing a HEMP is shown in **Figure 2**, consisting of five steps. The methodology is based on the work presented in references [3, 10, 11], but adapted to our work.

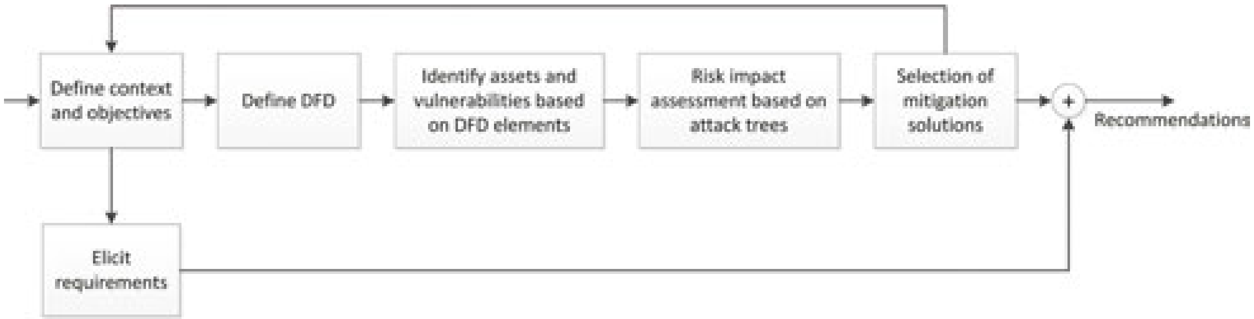

Define context and objectives
Define DFD
Identify assets and vulnerabilities based on DFD elements
Risk impact assessment based on attack trees
Selection of mitigation solutions
+
Recommendations
Elicit requirements

Figure 2. The methodology for performing the threat modelling.

The following gives a description of the six steps in our method:

Step 1. Define context and objectives—The context represents the identification of the main stakeholders of the system, the general system architecture and the desired objectives. The purpose of this step is to create the scope of interest and focus on the analysis. Part of the context is the review of regulations with the smart grid area and to describe the considered use cases. This includes also defining the dependency between stakeholders. With the definition of the context and objectives, requirements can be elicited.

Step 2. Define DFD—Using the previously defined context, the flow of information is modelled through DFDs. Context elements are mapped to elements that present input/output, processes, data flow and data storage. In the modelling process, the context can be refined, for example, adding additional stakeholders or modifying the system architecture. Furthermore, it supports the process of modelling the domain knowledge to DFD elements. With the process of defining the context and modelling, the DFD represent the high-level system description.

Step 3. Identify assets and vulnerabilitiesbased on DFD elements—Based on the DFD elements representing the HEMP, the system's assets and threats are identified. Assets represent valuable targets for an attacker and are therefore of interest for the attacker. Assets can be both physical and linked to a process. One approach to identify threats for the identified assets is the STRIDE-per-element or STRIDE-per-interaction approach [3].

Step 4. Risk impact assessment based on attack trees—With explicit threats, the risk impact assessment can be performed using attack trees. The step assesses how noticeable and with what likelihood a threat is. The outcome of this assessment can be either accepted or mitigated using a technical solution. An accepted risk represents a possible attack, but it is typically regarded as unlikely.

Step 5. Selection of mitigation approach—If a threat is unacceptable, a selection of a mitigation solution must be considered. The solution can be based on possible software or hardware technologies that consider an attacker with the same power or more. Mitigation can require a different hardware or software solution. The risk assessment of the threat can help in determining the threat mitigation should be placed in the hardware or software.

3.2. Stakeholders, system architecture and objectives

In **Figure 3**, the major stakeholders of the HEMS are presented. The residential consumer enables the deployment of HEMSs. Trust to the system, reduction of/control of electricity bills, addressing environmental concerns and better comfort (i.e., provision of technical solutions for better control of own energy use) are the main motivational factors for the residential consumers to adopt a smart grid solution [12]. Smart device vendors build sensors and actuators deployed in homes, for example, Wink,[6] SmartThings[7] or Vera[8] represent an envisioned smart device vendor. Often these vendors can be categorised whether or not they provide a total home automation solution or have upgraded an existing product to be IoT-enabled. In the envisioned system, the Distribution System Operator (grid operator) is not directly interacting with the HEMS, instead it depends on the smart grid services developed by service providers and deployed by the service market responsible. Services utilise information retrieved from the residential homes and electric grid, to improve electricity usage for both the residential consumers and grid operator. For connecting the services and the smart devices, a communication service provider (CSP) is needed. The software platform providers create the software platform, where services can be executed.

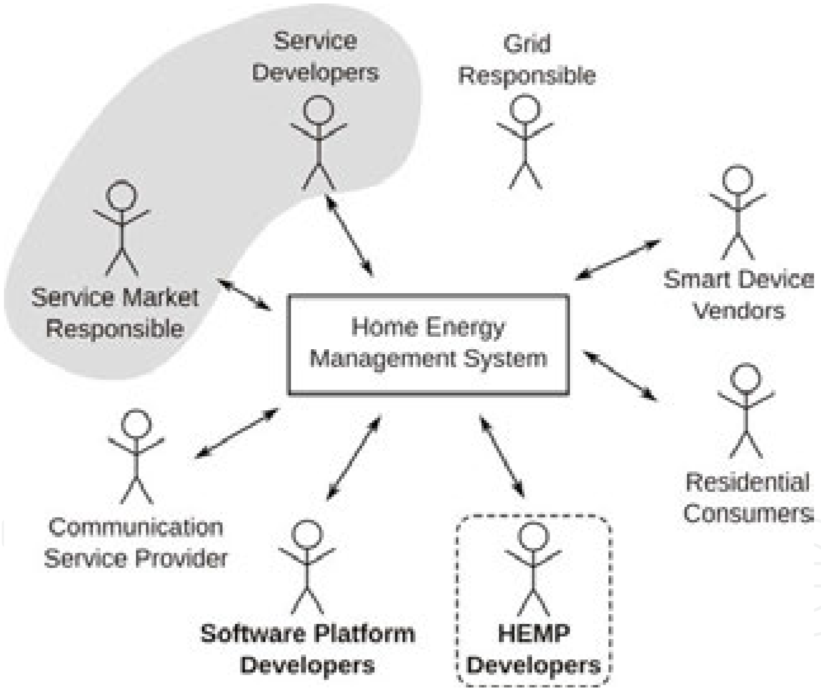

Service Developers
Grid Responsible
Service Market Responsible
Home Energy Management System
Smart Device Vendors
Residential Consumers
Communication Service Provider
Software Platform Developers
HEMP Developers

Figure 3. Overview of the major stakeholders of the HEMS. It is inspired by [32].

The HEMS is considered to be placed in an SOA, where there exist home-oriented and grid-oriented services. The services are implemented as web services based on the REpresentational

[6] http://www.wink.com/

[7] https://www.smartthings.com/

[8] http://getvera.com/

State Transfer (REST) architectural style. The system considers many-to-one mapping between the residential consumer and the DSO, that is, the home-oriented services are executed for each residential consumer, whereas the grid-oriented services utilise the output for optimising the operation for the DSO.

The rationale of the system architecture is that the home-oriented services will optimise according to demands of individual homes, whereas the grid-oriented services will support the operation of the electrical grid. Since grid-oriented services can be highly valuable for the grid responsible (DSO), these will foster the development of the home-oriented services. A fundamental requirement for the approach is that the services can be used as *building blocks* for delivering additional services. For a more detailed insight into the system, the reader is referred to references [13, 14].

The main characteristics of a service-oriented system are [15]:

- Heterogeneous platform and execution environment.
- Communication is handled through standards.
- Advances the concept of component-based software by reducing the coupling between services.
- Encourage continuous and independent (re)deployment of software.

The domain model is depicted in **Figure 4** using a UML class diagram together with the main stakeholders. It is composed of classes (rectangular icons), packages (folder icons) and actors (stick figures). It considers the domain to consist of two major parts: a *service market* and a *residential home*. Both parts include a number of elements (represented as classes) that represent the implementation by a stakeholder. At this stage, only elements that are relevant for an identified stakeholder are considered. The relationship between the elements indicates the

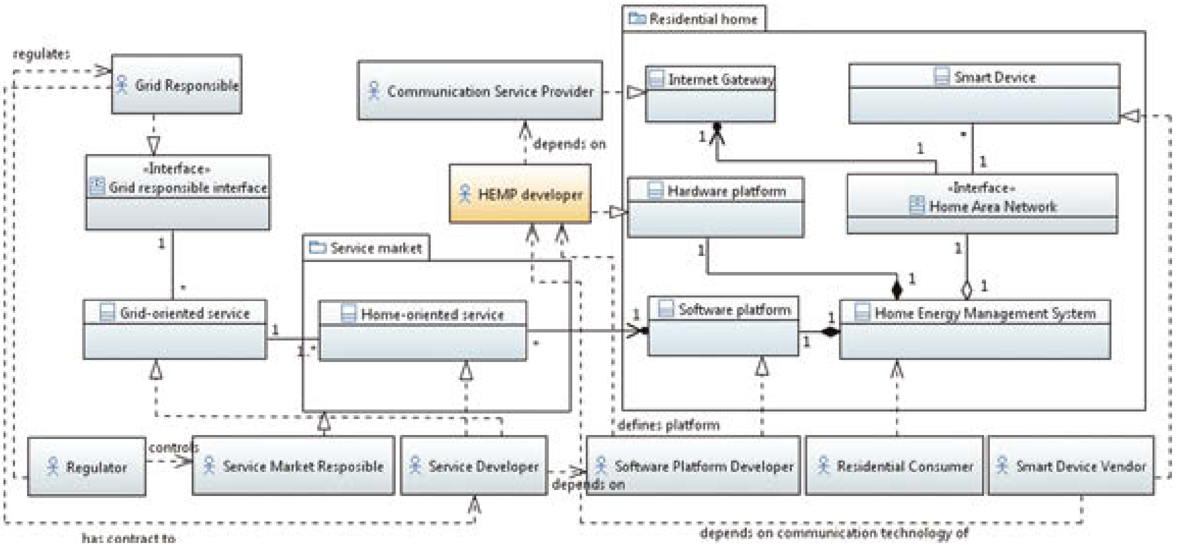


Figure 4. Domain model of the system.

multiplicity and type of association. Besides the basic UML arrows representing the associations, two other arrows are drawn. The dependency between the stakeholders is modelled with an open arrow to visualise how stakeholder's business objectives are associated with other stakeholders. A closed arrow indicates the realisation of the element.

In the process of defining hardware recommendations for the HEMP developer, focus is set on the HEMP developer as depicted and highlighted in **Figure 3**. As seen in Figure 4, the HEMP developer has several dependencies—either directly or indirectly.

To this end, the objective of the HEMP developer for the HEMS is the following:

> *The HEMP will provide a security and privacy enhanced platform for service developers to deploy on web service on, while being a trustworthy platform for the residential consumers, service providers and the grid responsible.*

3.3. Requirements

To identify the security threats and vulnerabilities of the HEMP (see Section 3.5), the necessary requirements, which the platform must adhere to, have to be explicitly stated. An explicit set of requirements can guide the process of designing a platform which is resistant to possible attacks. Furthermore, it allows for a risk assessment of third-party services installed on the device.

Requirements are usually understood as stating *what* a system is supposed to do—contrary to *how* it should do it [16]. It characterises the desired functional behaviour and performance, whereas non-functional requirements aim at fulfilling a desired property. Non-functional requirements are usually judged by the operator of the system and relate more to the system architecture, whereas functional requirements are testable.

In the following, both the functional and non-functional requirements for the HEMP will be presented. The requirements are classified according to the direct stakeholders for the HEMP developer. The rationale is that the requirements and dependencies from the stakeholders will have an impact on the threats and thereby on the mitigation strategies that technical solutions can provide. The listed requirements are derived from use cases and requirements specified in references [14, 17–19].

3.3.1. Software platform developer

An essential feature of the HEMS is to provide an environment for executing services relevant for external actors in the smart grid. This includes services with real-time dependencies or not. These services can be manufactured by different service providers and therefore depend on an *isolated execution environment*, if a single service should not crash the system. Moreover, software trends dictate a continuous development and maintenance process to follow an agile development life cycle and fix newly discovered vulnerabilities. Thus, doing updates to the services is vital for its sustainability. Finally, a software platform is responsible for delivering the “knobs” of the smart devices for services and external system to interact with in a secure way. The requirements are as follows:

R1. The HEMS shall support execution of real-time dependent services, like services for controlling a residential battery according to external signals.

R2. The HEMS employ separation mechanisms to securely isolate services.

R3. The HEMS shall be able to upgrade existing services.

R4. The HEMS shall support remote software updates from the software framework developers for managing services.

R5. The HEMS shall provide security measures appropriate for the protection of integrity and confidentiality of services.

These requirements relate to service providers which represent the main stakeholder of the software platform developer.

3.3.2. Smart device vendor

End devices in the smart grids often represent the load of the electrical system. Giving these devices the ability to communicate with other devices facilitates information to be disseminated to all stakeholders. For the smart devices to be part of the smart grid, a communication infrastructure in the Home Area Network (HAN) that incorporates a heterogeneous set of communication protocols is essential [20, 21]. Furthermore, for smart devices to provide a secure communication path, it is necessary to enable end-to-end encryption. The smart device vendors' requirements are as follows:

R1. The HEMS shall support multiple communication protocols for the HAN.

R2. The HEMS shall provide smart device vendors with the ability to extend the set of communication drivers easily.

R3. The HEMS shall facilitate end-to-end encryption between smart devices and the services that utilise and control the smart devices.

Besides depending on the physical communication technology implemented on the HEMS, software drivers are also necessary to provide an interface for a service.

3.3.3. Residential consumer

The intelligent automation of the smart grid depends on the residential consumers. They will be an integral part that will ensure reliability of the electrical grid by modifying the way energy is used. A HEMS can support the behavioural change but requires the acceptance of the residential consumer. It is generally believed that a user-centric approach is vital, where trust to the system remains one of the priorities [12]. The requirements for a residential consumer are:

R1. The HEMS must secure the confidentially and integrity of meter data.

R2. The HEMS should provide residential consumers the ability to install and uninstall services on demand.

3.4. Data flow diagram of the home energy management platform

The main purpose of DFDs is to identify the key *data flows* in the system. Modelling the data flows, contrary to the *control flows*, reveals how information is exchanged throughout the system. A DFD model explains who takes part in a process for delivering the information, the information needed to complete a process and what information is stored.

Figure 5 provides an overview of the internal data flows on a HEMS and the HEMS interactions with the external entities based on reference [14]. Besides the DFD symbols presented in **Table 1**, the DFD is annotated with dotted lines indicating the trust boundaries. These are quantified three trust levels from the HEMP provider's point of view. Furthermore, it contains dotted arrows between processes, to indicate how a process affects another process's runtime. These are useful for expressing hardware-related issues.

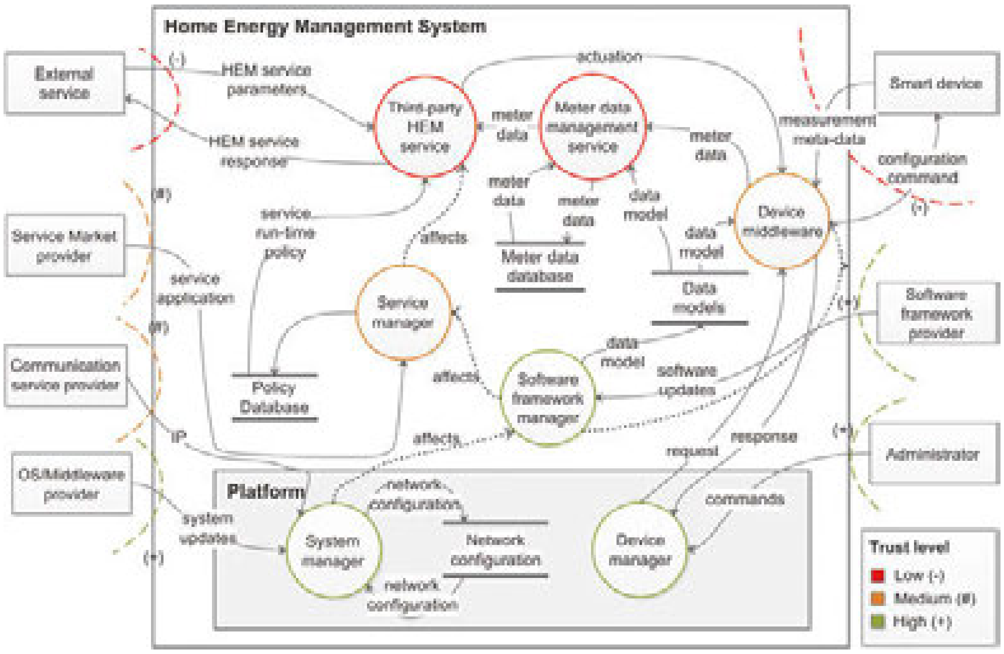


Figure 5. DFD of the HEMS.

As seen in Figure 5, seven processes are included in the model. This includes two services: third-party HEM service and a Meter Data Management (MDM) service. The MDM service is explicitly included to show the data flow of meter data and storage of the smart device configurations. The third-party HEM service presents any high-level service useful for smart grid operations. The service manager and software framework manager relate to the continuous maintenance of the software on the HEMS. Lastly, there is a system manager and device manager for the physical interaction with the HEMS.

3.5. Threat analysis

A threat model is a depiction of a system's attack surface. An attack surface has numerous threats towards a set of assets within the system. The purpose of mitigation techniques is to protect the assets from a possible attacker. The envisioned attacker can be assigned capabilities, such as being able to have physical access to the system or only have access through the network. Other attackers include social engineers which through "social" interaction can gain unauthorised privileges to the system. This could be attackers using phishing emails to exploit vulnerable code in installed software.

The threats for the HEMP provider are a combination of the indirect threats that the software platform provider faces and the external entities that have physical access to the HEMS. In the following, we classify the considered attacker, identify the assets of the HEMS and elicit a subset of vulnerabilities.

3.5.1. Assets of the HEMS

Assets represent valuable targets for an attacker, and therefore naturally represent the system susceptibility for threats. In the following, the key assets are identified with a description of the reason for its inclusion:

Services: These represent both value-added services for the residential consumer, but also account for the foundation of service market generation. A compromised service can lead to cyber attacks on homely property. Furthermore, it can also lead to violation of the intellectual property for the service provider, thus undermining a service market. The services are dependent on the software platform provider to secure the services, where it assumes the underlying software stack (e.g., virtual machine, Operating System (OS), etc.) will not be compromised. The services can contain a range of sensible information regarding the residential consumer, for example, granular meter data, personal identifiable information, cryptographic key sets and so on.

Service manager: This process is responsible for presenting, installing, updating and uninstalling services on the HEMS. It is in direct contact with the residential consumer and service market provider. An attack on the service market manager can lead to undermining the service ecosystem with services unable to be update if a critical software patch needs to be installed. It depends on the given privileges from the software platform. It contains the audit of installed services and cryptographic keys to allow for a secure connection to the smart market provider.

System manager: This manager administrates the OS stack including middleware software (JVM, Python interpreter), system libraries and kernel (network interface, memory management, I/O interface, etc.), and therefore performs privileged operations. The entire software framework stack depends on its implementation. Therefore, it is assumed that it is configured and maintained reliably.

3.5.2. Identifying vulnerabilities

Vulnerabilities represent an attacker's entry points to the system. It is therefore crucial to identify these entry points in order to mitigate possible attacks. Here the vulnerabilities are presented using the STRIDE-per-interaction approach between the service and service manager. Other interactions identified in the DFD are omitted for brevity.

3.5.2.1. Services/service manager

T1. Spoofing: The HEMS contacts the service market provider for exploring, installing, updating the services on Smart Market provider. If the attacker is able to insert spoofed IP packets or DNS packets when the service manager sends requests to the original Service Market site, the service manager can be sent to a site with malicious services.

T2. Tampering: An attacker can inject code for replacing integrity check of installed services. This will lead to malicious services being installed, which were not verified by the *service market controller*.

T3. Repudiation: Someone[9] rejects installation of a service in order to deny possible payment for energy bill.

T4. Information disclosure: An attacker is able to see the intellectual property of a service.

T5. Denial of service: If an attacker can block any communication with the service manager, the service manager cannot update installed services.

T6. Elevation of privileges: The HEMS supports third-party services to be installed through the *service market provider*. If the *service market provider* is able to install a service that is able to access outside the boundary of its privileges, an attacker can access the meter data, smart device configurations.

3.5.3. Attack trees

Attack trees are useful for visualising the escalation of attacks (see **Figure 6**). The attack tree shows the root cause of an attack and what an attack subsequently after would allow an attacker to do. What would seem to be a minor attack on the system can propagate through the system, leaving assets exposed for an attacker.

In the following, the threats towards the process with the lowest trust boundary is focussed on for brevity; the "*Third-party HEM service*" (where the MDM service is a special case). The threats **T1**, **T2**, **T4**, **T5** and **T6** can be collected from the root threat "*Install of malicious service*". The root of the attack tree is set to this threat. It is presented with the problematic state as root. Each edge in the attack tree is labelled with "{noticeable/likelihood}" measure, to indicate if the attack is believed to be noticeable and the likelihood of such an approach (**Figure 6**).

4. Mitigation strategies

Often the time spend on judging the level of the risk should be compared to the time for addressing the threat. When addressing the threat, it is possible to mitigate the risk by redesigning of system architecture, implementing a mitigation feature or simply ignoring it. In the following, the chapter will review for hardware security and privacy mechanisms that mitigate the "*install of malicious service*" attack.

[9] The term "someone" instead of attacker is used deliberately, since it can represent the user of the system as well.

4.1. Security and privacy technologies

The process of developing a secure hardware platform has recently intensified. Previously, computer security primarily focussed on creating secure software architectures and protocols with the assumption that the lower-layer applications (e.g., the OS) would be secure. However, flaws in an OS implementation can likely lead to additional vulnerability of the application on top of it. Therefore, researchers have proposed systems for running trusted code on an untrusted OS, but there exists still pitfalls, for example, Iago attacks [22]. In the following, the concept of a Trusted Execution Environment (TEE) will be reviewed as well as technologies that use this concept.

4.1.1. Trusted execution environments

In its essence, a TEE is an environment that you can *choose* to rely upon to perform sensitive tasks. The goal of TEE is to provide an isolated execution environment, secure storage, remote attestation, secure provisioning and a trusted path [23]. In hardware, it usually defines a distinguished part of the hardware architecture which is encrypted and integrity protected. It isolates an area of the processor, memory and peripherals for performing privileged operations. Next to the TEE, a Rich Execution Environment (REE) is considered outside the Trusted Computing Base (TCB), where both an untrusted OS and untrusted third-party services can be executed. Despite its realisation in AMD Secure Execution Environment, ARM's TrustZone technology [24], and Intel's SafeGuard Extensions (SGX) [25], it is not widely used by service

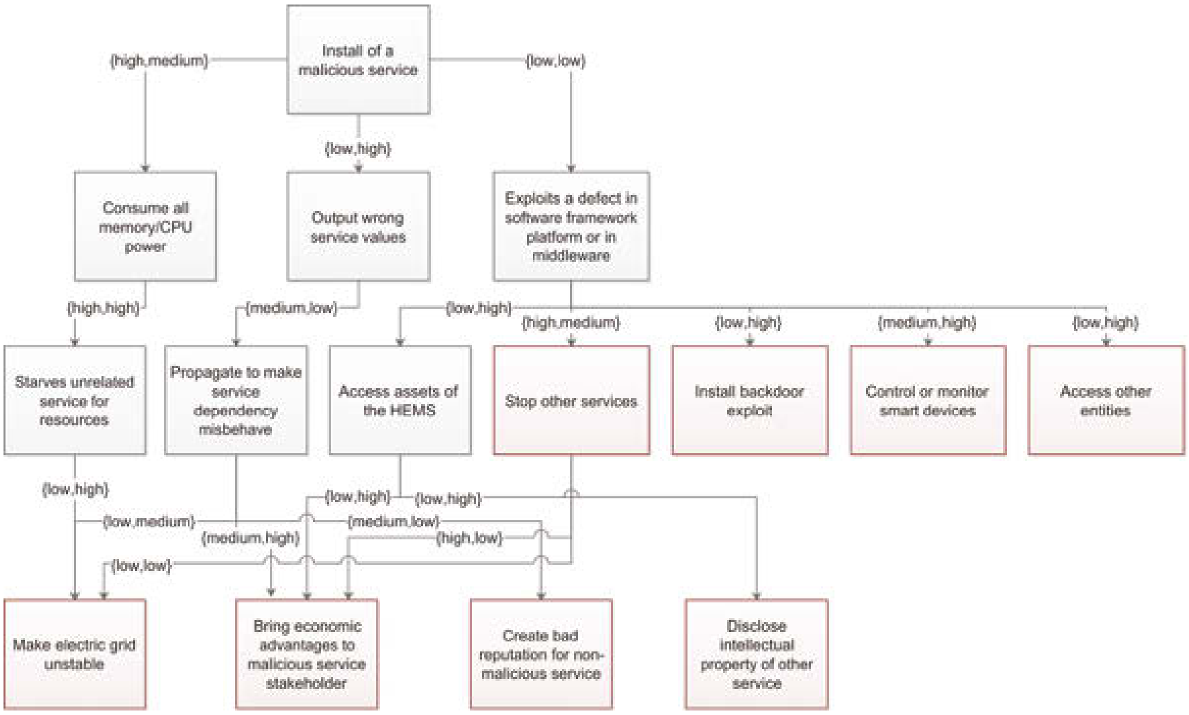

Install of a malicious service
{high,medium}
{low,low}
{low,high}
Consume all memory/CPU power
Output wrong service values
Exploits a defect in software framework platform or in middleware
{high,high}
{medium,low}
{low,high}
{high,medium}
{low,high}
{medium,high}
{low,high}
Starves unrelated service for resources
Propagate to make service dependency misbehave
Access assets of the HEMS
Stop other services
Install backdoor exploit
Control or monitor smart devices
Access other entities
{low,high}
{low,high}
{low,high}
{low,medium}
{medium,low}
{medium,high}
{high,low}
{low,low}
Make electric grid unstable
Bring economic advantages to malicious service stakeholder
Create bad reputation for non-malicious service
Disclose intellectual property of other service

Figure 6. Attacks based on threats towards “Third-party HEM service”.

developers. In the following, we will examine the hardware security technologies for mitigating the identified risk in Section 3.4.

4.1.1.1. ARM TrustZone technology

The security extensions embedded in the specification of ARMv6 and later are called the TrustZone technology [24]. It provides two operational worlds: a *normal world* and *secure world*. This allows for different execution privileges for applications. For instance, an application responsible for handling confidential data can be executed in the secure world, without a normal application being aware about; even with vulnerabilities in the normal world's OS. The two worlds are executed through two virtual processors with hardware access controls to switch between the two worlds. The hardware access switch defines the active components in the hardware when switched. Traditionally, ARM TrustZone has primarily been used for Digital Rights Management (DRM) or banking applications, but are not restricted to those types of applications.

4.1.1.2. Intel SGX

Intel's SGX extension is a set of instructions and mechanisms for managing the memory access to the Intel Architecture processors [25]. The main principle relies upon the concept of a protected memory container, also referred to as an *enclave*. The enclave can be created through application code, where sensitive data are explicitly marked. When the application is executed, a sensitive part of the application's memory space is encapsulated within an enclave. This enclave ensures that the confidentially and integrity of that memory space are sustained even with the presence of privileged malware (i.e., super-user capabilities). Furthermore, to ensure the integrity of the application inside the enclave, Intel SGX supports CPU-based attestation and sealing. An audited process monitors the built process of the enclave and assesses the identity of the hardware from where the enclave should be executed. Before the application is executed of the CPU, the identity of the hardware is verified through the sealing identity. It contains a digital signature (known as report) of the enclave's initial state. The enclave is then capable of receiving a report of the state, verifying its correctness. To verify that the correct software has been instantiated on the platform to remote system, it can perform an attestation with the SGX-supported processor (known as quote). This can prove to a remote system that application has been sent to a genuine SGX implementation. The reader is referred to [26] for a thorough description of the Intel SGX technology.

4.1.1.3. TrustLite and TyTAN

TrustLite [27] is a security architecture for resource constrained embedded systems that generally have to be cost effective in terms of development and production costs. The architecture allows for software isolation with execution-aware memory protection. The memory protection enforces a strict access control on memory by considering the program counter. Furthermore, it includes a secure exception engine that protects tasks from unauthorised exception handling. It has a secure loader that enables update of the security policy as well as prevents memory leakages after resetting the platform. However, TrustLite is static in terms of loading software components and their isolation, since this is required at boot time. TyTAN [28] leverages the TrustLite architecture for providing dynamically loading software components together with an integrated real-time system.

4.1.1.4. Haven

The objective of Haven security architecture [29] is to protect the confidentiality and integrity of a user's unmodified application in the cloud from an untrusted cloud provider. It is assumed that the processor is implemented correctly, but otherwise it is assumed that the adversary can access everything else, including memory and I/O devices. On software level, it is assumed that the adversary has full access to the entire software stack, including the OS, hypervisor, Basic Input/Output System (BIOS), device firmware and so on.

The inventors of Haven solve the confidentiality and integrity problem by using an *inverse sandboxing* technique also known as a *shielded execution*. Their solution is called Haven, and it leverages the Intel SGX and Microsoft's Drawbridge project [30]. Intel SGX allows application developers to protect their data from unauthorised access or modification by software that have highest privilege levels (e.g., a super-user).

The deployment process of the Haven application is similar to the deployment of a regular cloud application, with the additional step of verifying that the process was correctly performed (the attestation from the Intel CPU). It is assumed that the cloud provider provides an Infrastructure as a Service (IaaS), that delivers the storage, hardware and networking components on a virtualisation platform (e.g., KVM, Xen, etc.).

4.2. Recommendations and conclusion

Recommendations for a HEMP provider are given based on the vulnerabilities discovered from the threat model and the review of hardware security technologies based on TEE. Some of these recommendations reflect the challenges the security community and the hardware manufacturers are facing today, thus implementation details are still unexplored. Therefore, this chapter limits the thoroughness of recommendations to a problem description and possible approaches that need to be adapted for the HEMS. The problem description is linked to the identified requirements and threats, where the approaches are linked to the hardware technology review. The list of recommendations is as follows:

- **The HEMP should support isolation of services in terms of data and resources**: At the service layer envisioned for the HEMS, software frameworks for energy management already provide isolation of services (e.g., OSGi-based software frameworks). However, as services become mission critical in relation to energy management, the computational resources must also be isolated. Furthermore, as services need different privileges for accessing data, the HEMS should provide a data isolation mechanism. An application isolation mechanism based on the TyTAN security architecture [28] could comply with such demands. Furthermore, it allows the use of real-time-dependent intelligent automation services and allows for securely installing additional communication drivers (see **R1, R2, R6, R7 R9, R10, R11**).

- **The HEMP could provide an inverse sandboxing mechanism, if the deployment of a service is sensitive for access or modification by highest privileged users:** Acting as a platform for intelligent automation services to be deployed on, the service provider might have contractual agreement with the residential consumer about their electricity consumption. For achieving this, the service providers must ensure the integrity and confidentially of the intelligent automation services. This can be achieved, for example, by using the inverse sandboxing mechanism that the Haven [91] security architecture provides (see **R5**).
- **The HEMP should place the device authentication process and the update process in a trusted environments and vulnerable data in a secure data storage (e.g., containing private keys)** Authentication becomes a larger problem in the smart grid because of the necessity of self-organisation. A possible solution is presented in [31] which is based on TEE (see **R3, R4, R7, R8**).

Constructing a HEMP, which is both secure and ensures the privacy of considered data assets, is challenging. But in order to do so, an important property for enforcing security is to *build it in* the system and not *onto* the system. This chapter contributes with a threat modelling of HEMS based on the requirement and design phases of the Microsoft Security Development Lifecycle. A domain model is constructed using UML, and the requirements are elicited from use cases for HEMS under research. A DFD models an abstract HEMS platform based on a combination of the architecture of OGEMA framework and reference architecture of a mobile platform [32]. Based on a threat analysis of the DFD, an attack tree is constructed with the focus on a malicious service attacker. Given the threats, mitigation strategies are reviewed for giving recommendations for HEMP manufacturers in the future smart grid.

Acknowledgements

The research leading to these results has received funding from the EU Seventh Framework Programme (FP7/2007–2013) under grant agreement no. 317761 (SmartHG).

Author details

Søren Aagaard Mikkelsen and Rune Hylsberg Jacobsen*

*Address all correspondence to: rhj@eng.au.dk

Department of Engineering, Aarhus University, Denmark

References

[1] T. H. Christensen, A. Ascarza, and W. Throndsen, *Country–specific factors for thedevelopment of household smart grid solutions: Comparison of the electricity systems, energypolicies and smart grid R&D and demonstration projects in Spain, Norway and Denmark*. København: SBIforlag, 2013.

[2] P. Palensky and D. Dietrich, "Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads," *Ind. Informatics, IEEE Trans.*, vol. 7, no. 3, pp. 381–388, 2011.

[3] *A. Shostack, Threat modeling: Designing for security*. Indianapolis, IN: Wiley, 2014.

[4] Object Management Group (OMG), "Unified Modeling Language." [Online]. Available from: http://www.omg.org/spec/UML/ [Accessed: 30-Jan-2016].

[5] M. Zillgith, F. Institut, S. Energiesysteme, and D. Nestle, "Security Architecture of the OGEMA 2.0 Home Energy Management System System Concept and Security Requirements," *Int. ETG-Kongress*, vol. 9, pp. 1–6, 2013.

[6] N. Geoffray, G. Thomas, G. Muller, P. Parrend, S. Frenot, and B. Folliot, "I-JVM: A Java Virtual Machine for Component Isolation in OSGi," *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, pp. 544–553, 2009.

[7] A. Kanuparthi, R. Karri, and S. Addepalli, "Hardware and Embedded Security in the Context of Internet of Things," in Proceedings of the 2013 *ACM workshop on Security, Privacy & Dependability for Cyber Vehicles - CyCAR '13*, pp. 61–64, 2013.

[8] J. Zalewski, S. Drager, W. McKeever, and A. J. Kornecki, "Threat Modeling for Security Assessment in Cyberphysical Systems," *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop on - CSIIRW '13*, 2013, p. 1.

[9] T. Lu, J. Lin, L. Zhao, Y. Li, and Y. Peng, "A Security Architecture in Cyber-Physical Systems: Security Theories, Analysis, Simulation and Application Fields," *Int. J. Secur. Its Appl.*, vol. 9, no. 7, pp. 1–16, 2015.

[10] K. Beckers, S. Faßbender, and M. Heisel, *A threat analysis methodology for smart home scenarios*, vol. 8448, no. 256980. Cham: Springer International Publishing, 2014.

[11] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, and W. Joosen, "A Privacy Threat Analysis Framework: Supporting the Elicitation and Fulfillment of Privacy Requirements," *Requir. Eng.*, vol. 16, no. 1, pp. 3–32, 2011.

[12] F. Gangale, A. Mengolini, and I. Onyeji, "Consumer Engagement: An Insight from Smart Grid Projects in Europe," *Energy Policy*, vol. 60, pp. 621–628, 2013.

[13] R. H. Jacobsen and S. A. Mikkelsen, "Infrastructure for Intelligent Automation Services in the Smart Grid," *Wirel. Pers. Commun.*, vol. 76, no. 2, pp. 125–147, 2014.

[14] S. A. Mikkelsen and R. H. Jacobsen, "Consumer-Centric and Service-Oriented Architecture for the Envisioned Energy Internet," in *2015 Euromicro Conference on Digital System Design*, 2015, pp. 301–305.

[15] Microsoft – Developer Network, "Chapter 1: Security Fundamentals for Web Services." [Online]. Available from: https://msdn.microsoft.com/en-us/library/ff648318.aspx [Accessed: 25-Jan-2016].

[16] E. S. K. Yu, "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering," in *Proceedings of ISRE '97: 3rd IEEE International Symposium on Requirements Engineering*, 1997, pp. 226–235.

[17] Home Gateway Initiative (HGI), "*Use Cases and Architecture for a Home Energy Management Service*", 2011 [Online].Available: http://tinyurl.com/hposzxd .

[18] Energy@home, "*Use cases V 3.0*," 2015 [Online]. Available: http://tinyurl.com/hfctt9e

[19] U.S. National Institute of Standards and Technology, *Guidelines for smart grid cybersecurity.* Gaithersburg, MD, 2014.

[20] C. Greer, D. A. Wollman, D. E. Prochaska, P. A. Boynton, J. A. Mazer, C. T. Nguyen, G. J. FitzPatrick, T. L. Nelson, G. H. Koepke, A. R. Hefner Jr, V. Y. Pillitteri, T. L. Brewer, N. T. Golmie, D. H. Su, A. C. Eustis, D. G. Holmberg, and S. T. Bushby, "*NIST Framework and Roadmap for Smart Grid Interoperability Standards*, Release 3.0," Nist Spec. Publ., vol. 0, Gaithersburg, pp. 1–90, Oct. 2014

[21] M. Pichler, A. Veichtlbauer, and D. Engel, "*Evaluation of OSGi-based architectures for customer energy management systems*," 2015 IEEE Int. Conf. Ind. Technol., vol. 0, pp. 2455–2460, Mar. 2015.

[22] S. Checkoway and H. Shacham, "Iago attacks," in *Proceedings of the eighteenth international conference on Architectural support for programming languages and operating systems - ASPLOS '13*, 2013, vol. 41, no. 1, p. 253.

[23] A. Vasudevan, E. Owusu, Z. Zhou, J. Newsome, and J. M. McCune, "Trustworthy Execution on Mobile Devices: What Security Properties Can My Mobile Platform Give Me?," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7344 LNCS, pp. 159–178, 2012.

[24] ARM Security Technology, "Building a Secure System using TrustZone Technology." [Online]. Available from: http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf [Accessed: 28-Jan-2016].

[25] Intel, *Software guard extensions programming reference*, 2013. [Online]. Available from: https://software.intel.com/sites/default/files/329298-001.pdf [Accessed: 27-Jan-2016].

[26] V. Costan and S. Devadas, *Intel SGX explained*, 2016. [Online]. Available from: http://ia.cr/2016/086.

[27] P. Koeberl, S. Schulz, A.–R. Sadeghi, and V. Varadharajan, "TrustLite: A Security Architecture for Tiny Embedded Devices," *Proc. Ninth Eur. Conf. Comput. Syst. – EuroSys* '14, vol. 0, pp. 1–14, 2014.

[28] F. Brasser, B. El Mahjoub, A. Sadeghi, C. Wachsmann, and P. Koeberl, "TyTAN: Tiny Trust Anchor for Tiny Devices," in *Proceedings of the 52nd Annual Design Automation Conference on - DAC '15*, 2015, pp. 1–6.

[29] A. Baumann, M. Peinado, and G. Hunt, "Shielding Applications from an Untrusted Cloud with Haven," *ACM Trans. Comput. Syst.*, vol. 33, no. 3, pp. 1–26, 2015.

[30] Microsoft Research, "Drawbridge." [Online]. Available from: http://research.microsoft.com/en-us/projects/drawbridge/ [Accessed: 28-Jan-2016].

[31] A. J. Paverd and A. P. Martin, "Hardware Security for Device Authentication in the Smart Grid," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7823 LNCS, pp. 72–84, 2013.

[32] N. Asokan, L. Davi, A. Dmitrienko, S. Heuser, K. Kostiainen, E. Reshetova, and A.-R. Sadeghi, "Mobile Platform Security," *Synth. Lect. Inf. Secur. Privacy, Trust*, vol. 4, no. 3, pp. 1–108, 2013.