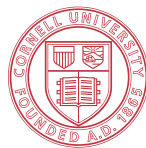


MASTER'S THESIS

Secure Signal Processing for Smart Grid Privacy

submitted to the
Department of Information Technology & Systems Management
at the
Salzburg University of Applied Sciences

by:
Christian D. Peer, BSc



Cornell University

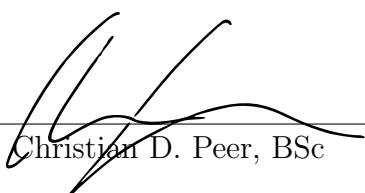
Head of Department: FH-Prof. DI Dr. Gerhard Jöchtl
Supervisor: FH-Prof. DI Mag. Dr. Dominik Engel
Supervisor: Prof. Dr. Stephen B. Wicker

Ithaca, August 2014

Christian D. Peer
Schusteranger 7
5342 Abersee
AUSTRIA

I hereby declare that the following master's thesis was written only by the undersigned and without any assistance from third parties. Furthermore, I confirm that no sources were used in the preparation of this thesis other than those indicated in the thesis itself.

Ithaca NY, August 31, 2014



Christian D. Peer, BSc

1210581049

Personenkennzeichen

Common Information

Name:	Christian D. Peer, BSc
Organization:	Salzburg University of Applied Sciences
Department:	Information Technology & Systems Management
Title:	Secure Signal Processing for Smart Grid Privacy
Keywords:	Smart Grid, Demand Response, Advanced Metering Infrastructure, privacy protection, privacy, load data, multi-resolution load data representation, hierarchical key management, discrete wavelet transform, Public Key Infrastructure
Supervisors:	FH-Prof. DI Mag. Dr. Dominik Engel Prof. Dr. Stephen B. Wicker

Abstract

The Smart Grid helps to reduce peak load, save energy and integrate intermittent energy sources into the grid. It has been shown that information about a consumer's actions, beliefs and preferences can be extracted from high resolution load data collected by Smart Meters. This information can be used in ways that violate consumer privacy. In order to increase consumer control over this information, it has been suggested that load data be represented in multiple resolutions, with each resolution secured with a different key. To make this approach work in the real-world, a suitable key management needs to be employed. In this Master's Thesis, the encryption of multi-resolution load data using hierarchical keys is discussed. A suitable key management scheme as well as a communication protocol is proposed. Emphasis is placed on a privacy-aware design that gives the end-user the freedom to decide which entity is allowed to access user related data and at what granularity. Apart from privacy protection, this Master's Thesis proposes a new protocol for communication within the Smart Grid Infrastructure. The proposed protocol increases grid reliability and stability by protecting the Smart Grid Infrastructure from possible intrusion or Denial of Service attacks.

Acknowledgment

I want to thank my two advisers, Dominik Engel at Salzburg University of Applied Sciences and Steven Wicker at Cornell University for their generous mentorship. Without Dominik I would have been unable to develop the idea for the research presented in this thesis. He was always available for meetings to discuss ideas, and made it possible for me to receive a scholarship to attend Cornell University for five months in my last semester. During that time I had the great fortune to work with Steve Wicker. Steve mentored me as if I was one of his students. We met weekly, wrote a journal article in collaboration with Dominik, and submitted a paper to IEEE, which was accepted and I will thus be able to present it in Venice, Italy in November. Dominik and Steve were generous enough to fund my travels to Venice along with all the necessary tools needed to work productively on this Master's Thesis.

I also want to thank my amazing parents for giving me the possibility to study. Both of them never stopped encouraging me to pursue my goals. They always supported me in all live situations and I know that I can count on them anytime.

Financial support by the Austrian Marshall Plan Foundation as well as by the Austrian Federal Ministry of Economy, Family and Youth and the Austrian National Foundation for Research, Technology and Development is gratefully acknowledged.

Contents

List of Figures	6
1 Introduction	8
2 Related Work	11
2.1 Demand Response	11
2.1.1 The Impact of Different Pricing Models	13
2.1.2 Demand Response Potential in the United States	15
2.1.3 Current Development in the US	16
2.1.4 Demand Response in the European Union	17
2.2 Privacy Invasion and Resulting Effects	22
2.2.1 Privacy Preserving Architecture	23
2.3 Cryptography	24
2.3.1 Symmetric Cryptography	25
2.3.2 Public-Key Cryptography	25
2.3.3 Hybrid Cryptosystems	26
2.3.4 Public-Key Certificates	27
2.3.5 Public-Key Infrastructure	27
2.3.6 Applying a Public-Key Infrastructure to the Smart Grid	28
2.3.7 Hierarchical Key Generation	31
2.4 Multi-Resolution Load Data Representation	32
2.4.1 Discrete Wavelet Transform	32
3 Smart Grid Communication Infrastructure	36
3.1 Data Flows within the Smart Grid	37
3.2 Smart Grid Architecture	38
3.3 Privacy Preserving Methods to Secure Consumption Data	43
3.4 Multi-Resolution Load Data Encryption and Distribution	46
4 Proof Of Concept	50
4.1 The Smart Meter	50
4.1.1 Software Architecture	52

4.1.2	User Interface	64
4.2	The Grid Operator	64
4.2.1	Software Architecture	65
4.2.2	Management User Interface	69
4.3	The Third Party Entity	69
4.3.1	Software Architecture	70
4.4	Public Key Infrastructure	72
4.5	Development Environment	74
5	Conclusion	75
	References	77

List of Figures

2.1	The impact of the different pricing models on peak load reduction. Opt-out scenario. From [1]	14
2.2	The impact of the different pricing models on peak load reduction. Opt-in scenario. From [1]	14
2.3	Peak Demand forecast and possible reduction by scenario. From [2]	16
2.4	Estimates of Advanced Meter Penetration Rates. From [3]	17
2.5	Savings through Demand Response in the EU-15 countries in 2020. * considering that 1kWh saves 500gCO ₂ . ** expressed in equivalent of avoided consumption of large size cities (2 million inhabitants and 150,000 commercials, based on an average consumption of 8.2 TWh/year). *** expressed in equivalent of avoided construction of thermal plants (500 MW). From [4]	19
2.6	Average per capita load reduction potential of each Nuts-3 region in kWh. From [5]	20
2.7	The lifting steps can be concatenated to increase the depth of the discrete wavelet transform. While the high frequency part of each step is preserved, the low frequency part is used as input signal for the next lifting step. The inverse transform works the other way around. Adopted from [6]	34
3.1	Data flows within a privacy-aware demand response architecture. From [7]	37
3.2	The Smart Grid is divided into two different networks: the public network and the Grid Operator's inner network.	39
3.3	The Grid Operator can communicate with the Smart Meter using an encrypted connection.	40
3.4	A third party entity can establish a connection to a Smart Meter using the Grid Operator as a proxy.	41
3.5	The Wavelet transform splits load data into high and low frequency band. The low frequency band equals load data with reduced resolution.	46
3.6	All wavelet coefficients needed for the inverse transformation are encrypted with different keys and transmitted as a single stream.	47
3.7	In order to access load data, the entity has to request the resolution key for the desired resolution.	48

- 4.1 The Software Architecture for the Smart Grid consists of multiple components. The software provides an interface to the consumer/end user user interface as well as to the Grid Operator. 52
- 4.2 The software architecture for the Grid Operator provides an interface to interact with the Smart Meter, the consumer/end user user interface, the third party entity as well as the management user interface. 66
- 4.3 The Third Party Entity software architecture provides an integrated user interface to request access to load data and to display retrieved load data. 70

Introduction

Increasing energy needs accompanied by an emphasis on alternative energy production creates a need for efficient power grid management and regulated power consumption. This so-called Smart Grid enables load balancing and forecasting within a power grid. In addition the Smart Grid is able to influence consumer behavior regarding energy consumption by offering real-time pricing information. Based on this information, consumers can decide when to use devices so as to manage energy costs. Studies show that Smart Grid Infrastructure can reduce peak load during summertime by as much as 20% [2]. To fulfill this task, the Smart Grid relies on Advanced Metering Infrastructure (AMI), a sensor network collecting fine-grained power consumption data. Smart Meters form the core component of an AMI. These devices collect fine-grained consumption data, so-called load data, from a single household. While this data plays an essential part in load balancing and real-time pricing, its collection also creates serious privacy concerns.

It has been shown that apart from information needed for grid operation, other pieces of information can be obtained from fine-grained load data, directly related to sensitive and private information of the end user [8, 9, 10]. Occupancy or sleeping patterns can be determined and certain appliances within the household can be identified and a usage pattern can be drawn. This information can be valuable for targeted marketing as well as criminal purposes. With regard to the former, techniques for matching appliance signatures to load data are called non-intrusive load monitoring (NILM) or non-intrusive appliance load monitoring (NALM) [9].

Acting on privacy and security concerns, customers and governments are rejecting the deployment of Smart Meters and therefore blocking the deployment of the Smart Grid [11]. To address this issue, privacy preserving methods have to be implemented. Two types of approaches show great potential for ensuring privacy within the smart grid: (i) Secure aggregation of encrypted load data and (ii) consumer control over load data in multiple resolutions, each resolution associated with different access levels. In terms of secure aggregation, Erkin et al. give an overview of the recent development in [11]. This leads to

the following research question: How can privacy-preserving methods be used to balance the need for end-user privacy with the information needed for the correct operation of smart grids?

This document presents a new communication architecture for the Smart Grid Infrastructure. The Smart Grid Infrastructure is divided into two separated networks. The inner network consists of the Advanced Metering Infrastructure (AMI) and the sensor network needed to control and balance the grid. The Grid Operator is in charge of controlling and maintaining this network. The public network includes the utilities and any third party companies like energy brokers or companies giving energy saving tips. The public network can communicate with the inner network by using the Grid Operator as a gateway. This allows the Grid Operator to monitor and control the network traffic which enters the inner network. With the Grid Operator acting as a firewall, attacks like hacking or Denial of Service can be prevented. This scheme increases grid stability and reliability. Content encryption is applied whenever privacy is endangered.

Different privacy preserving methods are discussed. This document will focus on the representation and securing of load data in multiple resolutions. NILM/NALM techniques need high resolution load data to gain accurate results. By lowering the resolution of the load data, NILM/NALM techniques can only achieve limited results. While a low resolution on a daily average is sufficient for accounting purposes, applications like load forecasting or energy saving tools require high resolution load data to achieve useful results. This is where multi-resolution load data representation is needed. Each resolution is encrypted with a different key. Trusted services or third parties are only granted access to the resolution level necessary to fulfill their role. Access can be controlled by a trusted authority, or better, by the user. This adds a new degree of freedom, as the user can decide which party gains access to which data.

An approach on how to represent load data in multiple resolutions can be found in [12]. While this approach describes how to split load data into multiple resolutions, it leaves the question about suitable key generation and management unanswered. In this paper, a key management system suitable for accessing multi-resolution load data within the Smart Grid Infrastructure will be introduced. Furthermore this document will suggest the use of hierarchical keys to keep key management efforts as low as possible.

As an alternative to secure aggregation of encrypted load data, load data can be measured directly at substation level. The resulting load data equals the aggregate with the benefit, that no complex neighborhood aggregation is required. Hence, the management effort is reduced. While this approach offers multiple benefits over aggregation, it cannot cover all use cases. To address this problem, this document will suggest to combine measurements from substation level with multi-resolution load data. Using this combination, all required use cases are fulfilled while system complexity remains low.

A software architecture implementing the proposed system is discussed at the end of this document. The software consists of three parts: (i) a Smart Meter, (ii) a Grid Operator and (iii) a Third Party Entity. The software acts according to the proposed system. It implements the following three main ideas: (i) multi-resolution load data representation, (ii) hierarchical encryption and (iii) access management. The modular software design allows an easy implementation of further ideas like load data aggregation or billing.

The research described in this Master's thesis has led to a journal article as well as a conference paper. The conference paper¹ will be presented at the IEEE SmartGridComm 2014. The journal article has not been submitted for publication at the present time.

¹See [13]

Related Work

The following chapter provides background information on the design principles and technologies used within this document. Demand Response is defined and its purpose is revealed. Different pricing models are explained and evaluated based on a study done in the Pacific Northwest of the United States of America (USA). Furthermore, the Demand Response potential in the USA is discussed. It shows that a full deployment of Demand Response has potential to reduce peak demand by up to 20%. To complete the section about Demand Response, the current development in the USA is analyzed. This document focuses on privacy and privacy preserving methods. Hence, the next section defines privacy and discusses how privacy invasion effects the personal development of individuals. By influencing the information provided to an individual, the individual's moral freedom is limited. Furthermore, the awareness of being observed constrains creativity. Privacy invasion can be prevented by designing systems following privacy preserving design patterns. Privacy can be preserved using enhanced cryptographic methods. This chapter gives an overview of the basic concepts of cryptography. It discusses the requirements and the resulting problems of porting a Public Key Infrastructure to the Smart Grid. By choosing the right topology, most of the problems can be addressed. The chapter concludes with describing the concept of multi-resolution load data representation. The concept is based on hierarchical key generation and the Discrete Wavelet Transform. The mathematical background for both concepts is given. More detailed information on any of the discussed topics can be found in the referenced literature.

2.1 Demand Response

The Federal Energy Regulatory Commission (FERC) define Demand Response as:

"Changes in electric usage by end-use customers from their normal consumption patterns in response to changes in the price of electricity over time, or to incentive payments designed to induce lower electricity use at times of high wholesale market prices or when

system reliability is jeopardized." [14]

Demand Response does not aim to reduce the total energy consumption, it just offers possibilities to shift load to avoid or reduce peak loads in the grid [15].

Peak load or peak demand describes a period of time with an electricity demand significantly higher than the average supply level. Peak loads require the utility to fire up additional power stations, so-called peaking power plants, with the ability to provide a high amount of electricity within a short time. Only a few power station types offer the possibility for a fast start up. In terms of renewable energy, only water power plants with pumped storage type dams offer this ability. Wind farms and solar power plants produce electricity depending on the weather condition, hence availability of electricity is not controllable. For the utility, peak loads are expensive, as they require to start up additional power plants. In addition, peak loads can also destabilize the grid if the electricity demand cannot be fulfilled in time.

In [5], Gils argues that load shifting or shedding helps to reduce or avoid peak loads. The goal is to consume electricity when enough of it is available and reduce consumption when electricity supply is low. Demand Response in combination with the Smart Grid helps to achieve this goal by, on the one hand, providing fine granulated, real time load data from the grid and, on the other hand, offering the possibility of targeted load shifting or shedding. Typically, shiftable loads feature at least one of the following characteristics: physical storage (e.g., water supply), demand flexibility (e.g., any kind of washing, ventilation) or heat/cold storage (e.g., central heating, air condition, refrigerator).

According to a report of the Federal Energy Regulatory Commission (FERC) [3], several severe peak loads occurred in 2013. The Smart Grid helped to keep the system alive, balance the grid and distribute the demand. This report also lists some examples, where the Smart Grid helped to minimize grid damage and power outages during several storms and helped to accelerate grid recovery. Utilities used Smart Meters to predict the location and extent of the power outage. Repair crews were deployed to areas, where they were most effective. Automated switches working in tandem with the Smart Meters helped to reduce the total number of customer power outages. During restoration of electric service, Demand Response contributed to power system resiliency allowing a faster and smoother recovery.

Beside improved post-outage power restoration, Demand Response increases distribution-level reliability and supports a reliable integration of renewable energy sources. Demand Response also introduces new pricing models allowing the consumer to reduce their bill by changing their appliance usage behavior. So far, electricity meter reading had to be done by hand requiring utility staff to check each meter in person. As Smart Meters are able to communicate with the utility, meter reading can be done automatically, hence being more cost efficient.

2.1.1 The Impact of Different Pricing Models

Demand Response aims to alter consumption and consumer behavior based on available electricity. There are different programs on how to gain influence on consumption and behavior. In [1], Faruqui et al. examine which demand response programs offer most promising peak reduction potential. This study is based on data from Portland General Electric, a utility in Oregon and, hence aims for the Pacific Northwest of the United States.

Faruqui et al. define the following pricing models to alter consumer behavior and shift loads:

The *direct load control model (DLC)* regulates load by remote shut down of consumer appliances. In return, consumers receive an incentive payment. The payment varies with the level of load savings achieved.

The *time-of-use rates model (TOU)* defines different electricity rates per day. Every day is divided into several time periods with different rates per period. Rates for periods prone to peak loads are higher. The price difference between peak and off-peak periods is suggested to be 2:1.

With the *critical peak pricing rates model (CPP)*, rates are higher during peak hours on a limited number of days of the year. In return, rates are lower during all other hours. The suggested price difference is 4.4:1. By raising the rates only for a few days of the year, the influence of this model on the consumer is little. Still, this model helps to reduce the most severe peak loads on certain days.

Instead of raising the rates for peak hours, the *peak time rebates model (PTR)* pays consumers for load reduction during a peak load period. This model only applies to a limited number of hours a year, too.

In the *curtailable tariffs model*, consumers agree to reduce demand to a predefined level. In return, consumers receive payments that vary with the load curtailment level achieved.

Figure 2.1 shows the impact of the pricing model on the peak load reduction. The scenario assumes that consumers can participate with the possibility to opt-out. In opt-out, consumers are automatically enrolled in a dynamic price model with the option to revert back to the original, static price model. Using opt-out, higher enrollment levels are achieved.

According to Figure 2.1, dynamic pricing with the critical peak pricing rates model (CPP) and the peak time rebates model (PTR) offers the highest potential on reducing peak load. Both models aim to alter consumer behavior when peak loads occur and are therefore most

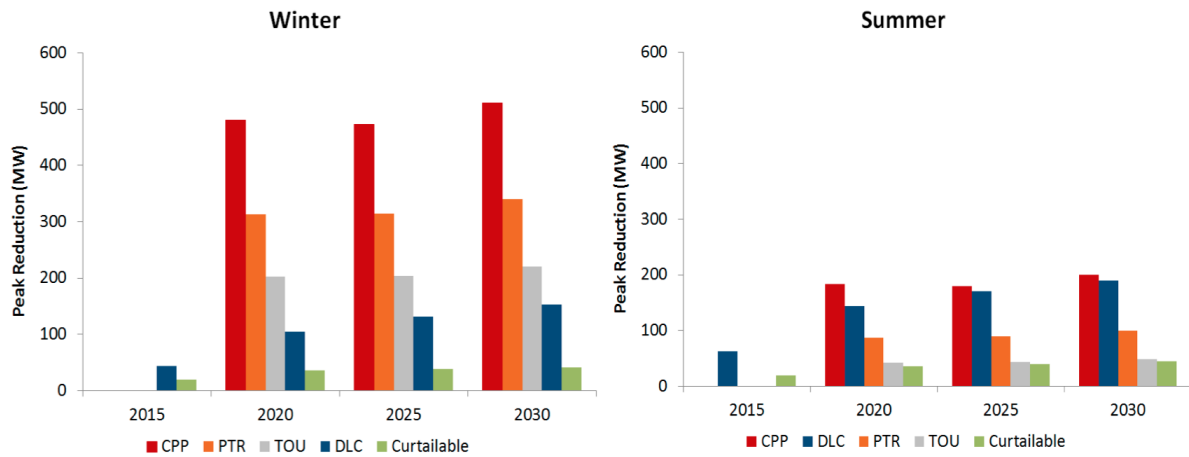


Figure 2.1: The impact of the different pricing models on peak load reduction. Opt-out scenario. From [1]

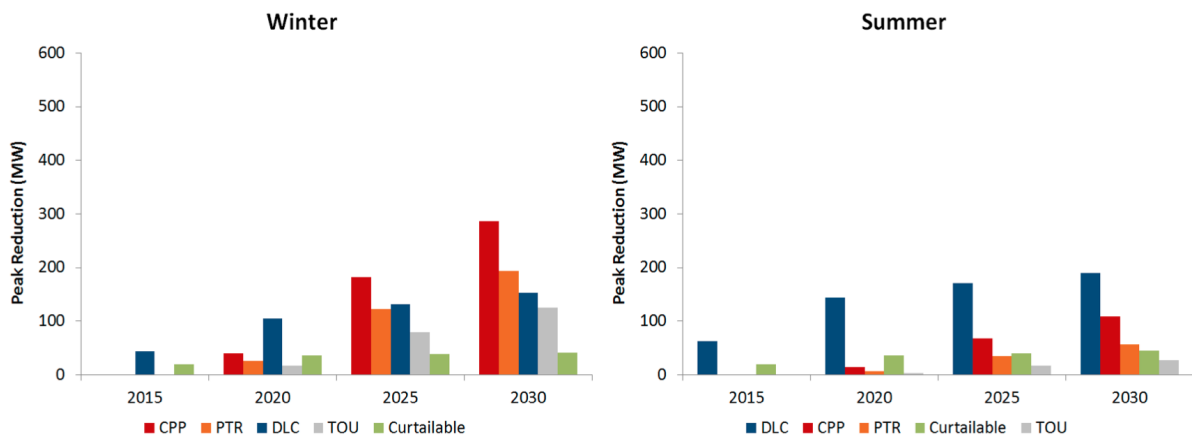


Figure 2.2: The impact of the different pricing models on peak load reduction. Opt-in scenario. From [1]

effective. The time-of-use rates model also shows a good impact on peak load reduction but aims to alter consumer behavior during the whole day. While this model helps balancing the load over the whole day, it does not provide means to react to extraordinary high peak loads on certain days during the year. Interesting is the development of the direct load control model (DLC). The reduction increases steadily over the years due to the distribution of appliances with the capability for remote shut down. This steady influence appears during winter and summer and is consequently a valuable contribution to general load balancing and peak load reduction.

During summer, the capabilities for peak reduction are less present. This phenomenon is probably due to the hot temperatures and the resulting high use of air conditioning in badly insulated buildings.

The opposite of opt-out is opt-in. With opt-in, consumer remain on their static tariff.

Participation in dynamic rate tariff is on a voluntary basis and requires proactive enrollment. Naturally, opt-in achieves lower enrollment levels than opt-out. Figure 2.2 shows the impact of the pricing model assuming an opt-in participation. General peak reduction is lower as less consumers are assumed to enroll in dynamic pricing rates. Nevertheless, results are quite similar to opt-out dynamic pricing. The results show that over the years, more and more consumers will accept and enroll in the dynamic pricing models. Results for the dynamic load control model (DLC) stay the same. The DLC is operated by the utility, hence no consumer action is required to enroll in any price model.

2.1.2 Demand Response Potential in the United States

Demand Response features a high potential in peak load reduction. In [2], the Federal Energy Regulatory Commission (FERC) gives a forecast on a possible reduction. The forecast distinguishes between four different scenarios.

Business-as-Usual: This scenario considers existing and currently planned demand response programs continuing unchanged without any additional improvements. Medium and Large commercial and industrial customers are provided with interruptible rates and curtailable loads. Direct load control is applied to large electrical appliances and equipment (central heating and air conditioning) of residential and small commercial and industrial customers.

Expanded Business-as-Usual: This scenario expands the Business-as-Usual scenario. It assumes that the current demand response programs are expanded to all states in the United States with higher level of participation. Furthermore, Smart Meters are partially deployed and dynamic pricing is available to customers. The Scenario assumes that 5% of the customers are participating in the dynamic pricing program.

Achievable Participation Scenario: This scenario assumes full-scale deployment of Smart Meters by 2019. A dynamic pricing tariff (real-time pricing) is the default. Customers have the ability to opt-out and therefore participate in other demand response programs like direct load control. 60 to 75% of customers are participating in the dynamic pricing tariff. This scenario also assumes that 60% of the customers will use enabling technologies (e.g., programmable communicating thermostats) in states, where these technologies are cost-effective and available.

Full Participation Scenario: This scenario shows the possible maximum. It assumes full deployment of advanced metering infrastructure with all customers participating in a dynamic pricing tariff. Enabling technologies are used if cost-effective.

Figure 2.3 illustrates the impact of the different scenarios and the estimated peak load reduction. According to [2], Peak Demand without any Demand Response will grow at

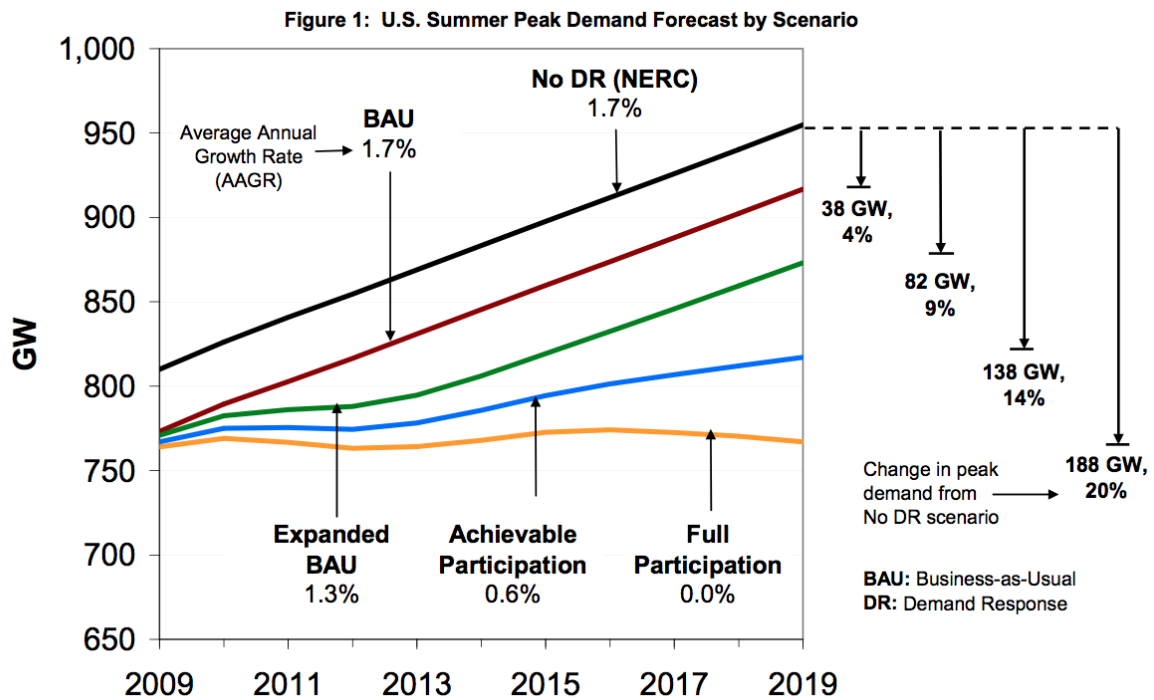


Figure 2.3: Peak Demand forecast and possible reduction by scenario. From [2]

an annual average of 1.7%. Its growth is estimated to increase from 810 gigawatts (GW) in 2009 to approximately 950GW by 2019. Depending on the scenario, the possible peak load reduction varies. Under the full participation scenario, in 2019, the peak load can be reduced by 20% compared to a scenario with no Demand Response programs. The Business-as-Usual scenario shows that even with the current deployed demand response programs and no further changes, peak demand can be reduced by 34 GW or 4% compared to a scenario with no Demand Response programs. The Achievable Participation Scenario estimates a reduction of 100 GW compared to the Business-as-Usual scenario. Comparing the Business-as-usual scenario with the Full Participation Scenario, in 2019, approximately 150GW peak load reduction can be achieved. In comparison, a peaking power plant produces about 75 megawatts. Reducing peak load by 150GW equals the output of 2,000 peaking power plants which will not be necessary to build and operate within the next years.

2.1.3 Current Development in the US

By July 2013, about 151.7 millions electricity meters were deployed in the United States, thereof about 46 million advanced meters (about 30%). Since the begin of 2008, the number of Smart Meters has increased by 580%. This remarkable trend can be seen in Figure 2.4.

Source of No. of Advanced Meters	Reference Date (Month/Year)	Number of Advanced Meters (millions)	Total Number of Meters (millions)	Advanced Meter Penetration Rates (advanced meters as a % of total meters)
2008 FERC Survey	Dec 2007	6.7 ¹	144.4 ¹	4.7%
2010 FERC Survey	Dec 2009	12.8 ²	147.8 ²	8.7%
2012 FERC Survey	Dec 2011	38.1 ³	166.5 ³	22.9%
EIA-861 Annual Survey	Dec 2011	37.3 ⁴	151.7 ⁴	24.6%
Institute for Electric Efficiency	May 2012	35.7 ⁵	151.7 ⁴	23.5%
Innovation Electricity Efficiency	July 2013	45.8 ⁶	151.7 ⁴	30.2%

Sources:
¹ FERC, Assessment of Demand Response and Advanced Metering staff report (December 2008).
² FERC, Assessment of Demand Response and Advanced Metering staff report (February 2011).
³ FERC, Assessment of Demand Response and Advanced Metering staff report (December 2012).
⁴ Energy Information Administration, Form EIA-861 Data File 2 and Data File 8 for 2011 (<http://www.eia.gov/cneaf/electricity/page/eia861.html>).
⁵ Institute for Electric Efficiency, Utility-Scale Smart Meter Deployments, Plans & Proposals (May 2012).
⁶ Innovation Electricity Efficiency, Utility-Scale Smart Meter Deployments: A Foundation for Expanded Grid Benefits (August 2013).

Note: Commission staff has not independently verified the accuracy of EIA or IEE data.

Figure 2.4: Estimates of Advanced Meter Penetration Rates. From [3]

While the growing number of Smart Meters is a welcoming trend, there is still no standardization for Smart Grid Communication and Infrastructure design as well as for consumer privacy protection.

Several institutions are working on guidelines and putting effort into a fast standardization. For the United States, the two most important institutions are (i) IEEE and (ii) NIST. The Institute of Electrical and Electronics Engineers (IEEE) hosts three working groups¹ putting efforts into designing Smart Grid Interoperability Standards. The National Institute of Standards and Technology (NIST), a non-regulatory agency of the United States Department of Commerce released a guideline on how to implement and deploy a Smart Grid Infrastructure with special emphasis on cryptographic means to secure the communication between the Smart Grid participants². Remarkable work has been done by these institutions and numerous publications by companies and universities are driving the development of Demand Response in the right direction. Nevertheless, further research has to be done especially in the sector of security and privacy protection.

2.1.4 Demand Response in the European Union

In [17], John describes that the purpose and goal of Demand Response is different in Europe compared to the United States. In the US, the main purpose is to reduce peak load through load shifting or shedding. In Europe, the main focus lies on cutting down

¹IEEE Std 2030

²NISTIR 7628 [16]

carbon emission and saving energy. Demand Response is also seen as the only possibility to integrate Europe's massive shares of intermittent solar and wind power. Compared to the US, the peak load in Europe is lower due to fewer hot-summer-days and thus less air conditioning installed in the commercial and residential field. In addition, a European home consumes one-third of the power of the average US home. This further reduces the demand response potential and reachable savings. Europe also operates a greater amount of pumped hydro power energy storage which can be used for inexpensive peak load coverage.

While the US had to deploy Demand Response to reduce peak loads occurring in an outdated grid, in Europe the movement towards Demand Response based on a Smart Grid is significantly younger. In 2007, the European Commission proposed the 'Third European Energy Liberalization Package', which entered into force in 2009. It aims to modernize and liberalize the European energy and gas market. Amongst others, it requires a Smart Meter deployment rate of 80% in the whole EU until 2020 [18]. Another goal is to produce 40% of the European electricity using renewable energy [17]. These goals boost the Smart Grid development. This change can be seen in the amount of Smart Grid related projects in the EU. Between 2002 and 2005, there were only a few projects, but from 2006 onwards, the Joint Research Centre (JRC)³ noticed a dramatic increase [19]. Since 2008, the investments in Smart Grid related projects are exceeding €200 millions per year. Until 2012, the JRC registered 281 Smart Grid related projects. Leading countries are the United Kingdom, Germany, France, Italy and Denmark with Denmark being most actively involved in Research and Development projects.

For Europe, the Demand Response potential is more difficult to estimate. The potential varies between the different countries and depends on the production capacity of the energy-intensive industry as well as the residential equipment rates of electric heating, water heating and air condition appliances [5]. In [4], a report from 2008, Capgemini predicts that with Demand Response 25% to 50% of the EU's targets on energy saving and CO_2 reduction can be achieved. Demand Response can save up to 202TWh electricity and about 100 million tons of CO_2 . While these goals sound heroic, the report states that it is unlikely to reach them. Figure 2.5 illustrates the potential savings in the EU-15 countries in 2020. The dynamic scenario is based on the optimal possible adoption of Demand Response to the European market. It assumes a full deployment of Smart Meters and Demand Response programs. The moderate scenario assumes a partial deployment of Smart Meters and a partial implementation of Demand Response programs. In this scenario, only 40% of the measures necessary to reach the EU 2020 goal will be put into place by 2020. The moderate scenario paints an unfortunate picture for demand response. From 2009 onwards, the European Smart Grid landscape changed dramatically. The

³see <http://ses.jrc.ec.europa.eu/>

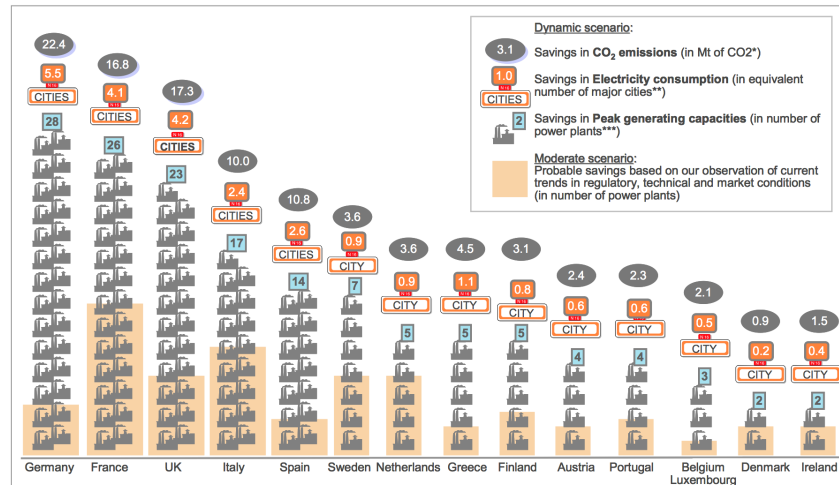


Figure 2.5: Savings through Demand Response in the EU-15 countries in 2020. * considering that 1kWh saves $500gCO_2$. ** expressed in equivalent of avoided consumption of large size cities (2 million inhabitants and 150,000 commercials, based on an average consumption of 8.2 TWh/year). *** expressed in equivalent of avoided construction of thermal plants (500 MW). From [4]

amount of Smart Grid related projects launched within the last years increased and the Smart Meter deployment rate changed. Due to the 'Third European Energy Liberalization Package', the deployment rate increased rapidly. Taking Austria as an example, while the moderate scenario assumes a Smart Meter penetration rate of 50% in 2020, according to a European Commission report from 2014 [20], Austria will reach a penetration rate of over 95%. This is an excellent example for the rapid change in the Smart Grid area. Capgemini points out, that customer targeting based Demand Response with direct load control is 30% to 100% more efficient than a non automated Demand Response system.

In [5], Gils assumes an average 93GW load reduction and a 247GW load increase potential in Europe. Load reduction defines load which is able to delay or shed, whereas load increase stands for advancing demand to an earlier time. Figure 2.6, shows the load reduction potential of each Nuts-3 region in Europe. Nuts stands for Nomenclature of Territorial Units for Statistics and defines regions within the European Union. Gils predicts an average annual load reduction of 7% to 26 % depending on the country.

According to a current report of the European Commission [21], 16 members of the European Union plan a full roll-out of Smart Meters until 2020. These countries are Austria, Denmark, Estonia, Finland, France, Greece, Ireland, Italy, Luxemburg, Malta, Netherlands, Poland, Romania, Spain, Sweden and the United Kingdom. Italy already reached a deployment rate of 100%. In seven member states, there will only be a partial roll-out. The European Commission states in its report, that average energy savings of 3% ($\pm 1.3\%$) of the total electricity consumption will be reached. The costs per metering point are $\text{€}223 \pm 143$ in comparison to the benefits per metering point of $\text{€}309 \pm 170$.

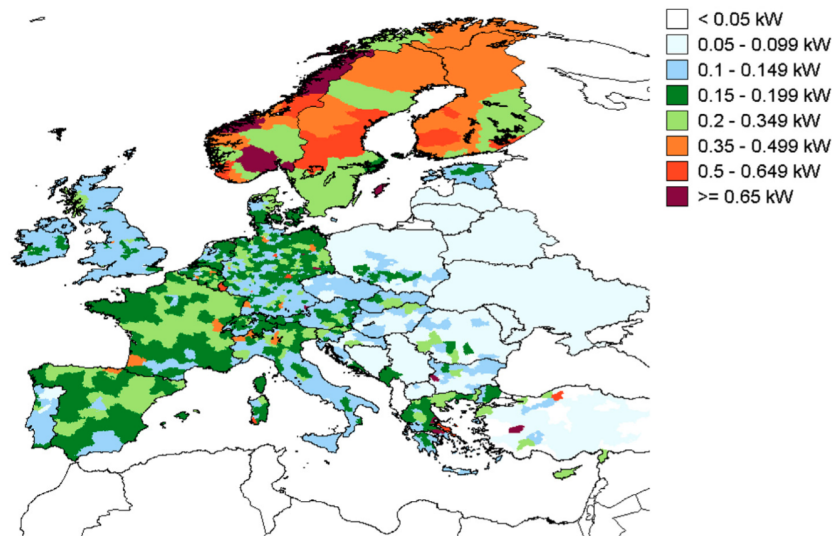


Figure 2.6: Average per capita load reduction potential of each Nuts-3 region in kWh. From [5]

Similar assumptions are made by another report from the JRC [22]. According to this report, the costs per metering point vary widely across the EU member states. Costs are depending on local conditions and the chosen communication technology. 200 million Smart Meters will be deployed until 2020. This equals 75% of EU customers. The communication infrastructure depends on the regional conditions. Power Line Communication (PLC) as well as General Packet Radio Service (GPRS) are the most used communication methods. The report criticizes that there is still no common consensus on the minimum functionality included in the Smart Meter.

According to the JRC [22], most of the current research is targeted to Smart Network Management as well as Smart Customer and Smart Home. Worth mentioning is the ongoing research in Germany and Austria focusing on Electric Vehicle to Grid applications. The main focus hereby lies on the charging process as well as on the communication infrastructure.

For Austria, in [20], the European Commission recommends a full Smart Meter roll-out. The highest net present value (NPV) is reached with a 95% replacement of all gas and electricity meters in 2011 to 2017. The roll-out plan foresees the deployment of 60% of the Smart Meters from 2016 to 2017. According to E-CONTROL [23], in April 2013, 3.4% of the installed electricity meters are Smart Meters with an additional 2% waiting for deployment. The Distribution System Operator (DSO)⁴ is responsible for Smart Meter deployment and ownership. The DSO also regulates third party access to the Smart Meter Infrastructure. PLC is mainly used for communication between metering point and concentrator. From the concentrator on to the Data Management, fiber optics are

⁴referred as the Grid Operator in this Master's Thesis

used for communication. This configuration promises a modern and well equipped grid. According to the European Commission, the main advantages of a full Smart Meter roll-out are energy savings, operational savings as well as the reduction of meter reading costs. Energy savings of 3.5% of the total electricity consumption can be achieved with further 2.5% of peak load shifting potential. The operational savings are due to a more efficient procedure in case of supplier switching. For the customer, the main benefits are reduced bills due to different pricing options and energy savings.

In Germany, the situation looks rather different. The cost benefit analysis of the European Commission is negative [20]. Thus, a full roll-out is not cost-effective. Instead, the European Commission recommends a partial roll-out which foresees a Smart Meter deployment to all customers with an electricity demand exceeding 6000kWh as well as major generation facilities and consumers in new or renovated buildings. The rest of the electricity meters should be updated to so-called intelligent meters which defines an upgradeable measuring system with no external communication link. Communication is mainly based on radio technology (GPRS/UMTS/LTE). Until 2022, the report forecasts a Smart Meter penetration rate of 23%. The potential for energy savings is around 1.2% of Germany's total consumption, the potential for peak load shifting is around 1.3%.

The key barriers in deploying the Smart Grid are rather social or policy and regulatory related rather than of a technical nature. According to [19, 18], the two main problems are consumer acceptance and a lack of standards and regulations. There are still uncertainties over the roles and responsibilities within the Smart Grid. It is still not defined, who is responsible for carrying the costs and in return, earns the benefits. Also, there is still no defined standard for the European Union. A standard would increase compatibility and therefore reduce the costs for deployment. Due to a missing standard, DSOs also risk to invest in technology which might not be used in the future.

Another big problem is the ongoing consumer resistance to participate in the Smart Grid. As a Norwegian pilot study [24] reveals, consumers see electricity as a "low interest" product. The electricity demand is based on habits instead of availability. Consumers do not have the general power situation in mind when using electricity. The JRC experiences a high consumer resistance to participating in Smart Grid related projects [19]. Consumers are skeptic and are not aware of the benefits of the Smart Grid. The JRC points out that it is essential to build a trustful relationship to the consumers. Future Smart Grid Projects need to give the consumers an understanding of the benefits in bill reduction as well as environmental concerns [18]. To increase consumers' trust, the European Commission claims clear regulations on privacy protection [21]. The right of protection of consumers' personal data has to be maintained. Privacy concerns are especially identified in the case of user profiling through high frequency load data readings as well as in the case of protection and access regulation to stored data. This master's thesis addresses this

privacy concerns.

2.2 Privacy Invasion and Resulting Effects

In [25], Warren and Brandeis define privacy as the right "to be let alone" preventing publications of one's "thoughts, sentiments, and emotions". They considered the underlying principle on a right "of an inviolate personality". In [26], Wicker and Schrader are generalizing Warren and Brandeis' conception through a metaphor of a "zone of seclusion" where a person can control access to personal information. Within this zone, the person is "free to experiment, develop relationships, and create an autonomous self without fear of censure or manipulation." Furthermore, Wicker and Schrader discuss the moral impact of privacy invasion [26]. They point out that the collection of personal information become a moral issue when this information is used to interfere and alter personal autonomy, behavior and decision making. Surveillance also limits a person's freedom of self-definition, judgment, choice and action. In the case of marketing, offering a person only choices based on his/her previous behaviors as reflected in the collected data, limits personal freedom and is discriminating. Individuals will get offers, others will never get. Personal freedom of choice and decision-making is influenced and limited. Wicker and Schrader point out, that there is potential for socioeconomic, racial or ethnic discrimination when access to information, health care or educational opportunities are limited. Especially in terms of "new technology" like social media, online messengers or smart phone usage, people are willing to give up their privacy in order to use this technology. According to Wicker and Schrader, most people are unaware of the exposure entailed with giving up their privacy and how this exposure can affect their future life in terms of accessible information, employment opportunities or access to education and services. In addition, in our modern context, it is "required" to interact with others using these "new technologies". Society requires the usage of these technologies in order to stay up to date.

By monitoring, regulating and reintroducing individual's choices in an abridged form, an individual's moral freedom is constrained. its placing limits on the freedom to think, to experiment, to challenge or to make well-considered decisions. By offering a person only information or choices based on its previous interests and decisions, narrows the person's field of vision. There will be a lack of reflective judgment and self-determination. Beside the influence on human behavior through limiting information and choices, the very thought of being constantly observed and monitored changes human behavior. Individuals' thoughts are self-conscious therefore altering the behavior in a self-aware manner [26]. The creativity literature has shown that the awareness of being observed harms the development of creative ideas [27, 28, 29, 30]. Thus raising the question which impact privacy

invasion will have on future innovations and on the development and engineering progress of humanity in general.

This brief review highlights the moral and personal harm caused by privacy invasion. Wicker and Schrader claim, that "there is a moral obligation on the part of engineers to pursue privacy-aware designs for mobile computing and communication systems" [26]. Following this assumption, Section 2.2.1 will focus on privacy preserving architecture, describing major design principles to take into account when designing distributed systems.

2.2.1 Privacy Preserving Architecture

There are different possibilities to enforce privacy protection. One is by regulation and law. While this basic idea is essential for a modern society, it still offers the potential to violate privacy using legal or illegal means. As long as system design and architecture offer the possibility to collect personally identifying information, there is a possibility to violate privacy protection. Therefore a better approach is to ensure privacy protection by design. In [7, 26], Wicker et al. propose a framework for privacy aware design tailored to the development of demand response architectures. They suggest five major elements:

Provide Full Disclosure of Data Collection:

A public statement about data collection should be published by each provider of any service collecting customer data. The type of data collected as well as resolution or granularity of the collected data should be defined. The public statement should also provide information for what the data is used/needed and how long it will be retained. In addition, means by which data will be retained should be listed. By reading the disclosure, the user can gauge the privacy risk and compare the disclosure with the ones from other service providers. Therefore, the disclosure has to be written in a for technically untrained persons understandable way. As the disclosure statement is a valid contract between provider and user, the user can take legal action if the contract is not fulfilled. Authorities can check the fulfillment of the contract, too.

Require Consent to Data Collection:

Similar to a software license agreement, the user has to accept a data collection agreement prior to using the provider's service. This agreement sharpens the user's awareness of the presence of data collection. Furthermore, the user has to be notified of a change in data collection practice. Rather than opt-out for the new agreement, the user can opt-in.

Minimize Collection of Personal Data:

It goes without saying, that collection of personal data should be kept at a minimum. Personal data has to never be used for training or testing purpose without prior aggregation or anonymization. The collection of personal data should be directly related to the functionality of the technology and therefore be essential for the system to work. In addition, the data should be used as close as possible to the collection point. These requirements prevent the collection of data in huge databases, which would allow easier reuse or misuse of data.

Minimize Identification of Data with Individuals:

Considering the system design, it is not necessary to collect personal data to, for example, track a user's movement to generate his running history. It is sufficient to just track the equipment and therefore store/aggregate anonymized location data related to the equipment, not the equipment's user. Functionally and personally identifying records should be separated. Wicker et al. suggest a so-called "Chinese Wall Security Policy". Classes with conflict of interest are defined and data is assigned accordingly. A policy ensures that there is no access to multiple classes at the same time. At most access is only granted to one data set belonging to each class. Hence, it is hard or impossible to draw a relationship between different classes.

Minimize and Secure Data Retention:

As already mentioned, data collection should be directly related to the functionality of the technology. If it is necessary to store data, then it should be retained only as long as absolutely needed. Collecting or storing data for possible future use is not acceptable. Storage of data has to be done in a secure way and in such a manner, that its reuse for a different purpose and its use in an undisclosed manner is impossible or at least difficult.

If there is reasonable suspicion of data leakage, consumers have to be notified immediately, declaring which kind of data was lost or stolen and recommendations for the customer on how to proceed.

2.3 Cryptography

Secrecy always played an important role in human communication. For example, ancient monarch Gaius Julius Caesar used a substitution cipher, called Caesar cipher, to protect confidential messages. Another famous example is the Enigma cipher machine used by

the Germans during World War II. Nowadays encryption is used during our day to day business securing our bank transfers, electronic communication or our online shopping habits. This section describes the basic concepts, modern cryptography is based on.

2.3.1 Symmetric Cryptography

Symmetric Cryptography is based on pre-shared keys. Alice and Bob, willing to communicate in a secure way, have to agree on a cryptosystem and a key. Then, Alice encrypts the plain message using the defined key and sends the ciphertext to Bob. Bob can easily decrypt the ciphertext as he knows the used cryptosystem and key. Symmetric Cryptography is simple and fast and therefore requires less computational power. There is only one drawback: Alice and Bob need to agree on a cryptosystem and a key prior to establishing a secure communication.

A good cryptosystem is one in which all the security is inherent in knowledge of the key and non is inherent in knowledge of the algorithm. [31]

Hence, negotiations about the cryptosystem can be made in public, whereas the negotiations about the key and the key distribution must be done in private, e.g., by couriers hand-carrying the keys. If a key is compromised, Eve can decrypt all message traffic encrypted with that key and also interfere communication by taking over Alice's or Bob's identity.

As a separate key is needed per secure communication, the number of keys to maintain increases rapidly as the number of possible connections increases. For n users, $n(n-1)/2$ keys are required [31]. For example, for secure connection between 100 users, 4950 keys are required. For secure communication within a big network like the World Wide Web (WWW), key management would be impossible.

2.3.2 Public-Key Cryptography

To address the problems occurring using symmetric cryptography, Whitfield Diffie and Martin Hellman developed a completely new approach back in 1976 [31]. They used two different keys for encryption and decryption. Whereas the key for encryption, the public key, is publicly accessible, the key for decryption, the private key, is held in private by the receiver of the message. Anyone can encrypt a message using the public key, but the decryption of the cipher text can only be done using the private key. This system is known as asymmetric cryptography or public-key cryptography.

After Alice and Bob agreed on a suitable public-key cryptosystem, Bob sends Alice his public key. Alice uses this key to encrypt her message and sends the resulting cipher text

to Bob. Bob decrypts the received cipher text using his private key. Any eavesdropper can get Bob's public key and the cipher text, but he cannot decrypt it.

According to Schneier [31], the system is based on trap-door one-way functions. A trap-door one-way function is easy to compute in one direction, but hard to compute in the other. But knowing the secret (the trap-door), the function is easy to compute in the other direction, too. Public and private keys are generated in a way that it is computationally hard to derive the private key from the public key.

Public-key cryptography solves the key-management problem with symmetric cryptography. With symmetric cryptography, the key has to be distributed in private prior to any secure communication. In addition, for every secure connection with a different user, a different key has to be generated, thus making key management and distribution expensive and difficult. In public-key cryptography, each user has to keep only one secret, the own private key, hence simplifying key management.

Schneier points out that, despite this advantage, public-key encryption has two drawbacks. On the one hand, algorithms used for public-key encryption are at least 1000 times slower than algorithms used for symmetric encryption. On the other hand, public-key cryptosystems are vulnerable to chosen-plaintext attacks. As the encryption key is public, an attacker can encrypt all n possible plaintexts and compare the result with the captured ciphertext. The attacker won't be able to recover the decryption key (private key) but he will be able to determine the unencrypted message. The smaller the possible encrypted messages, the more effective the attack is.

2.3.3 Hybrid Cryptosystems

Both, symmetric and public-key cryptography have advantages and disadvantages. Symmetric cryptography is fast and robust against certain attacks whereas it has drawbacks with key management and distribution. Public-key cryptography solves the problem of key distribution but is therefore vulnerable to certain attacks and significantly slower than symmetric encryption.

In most applications, both approaches are combined to benefit from all advantages while minimizing the disadvantages. To establish a secure connection, Alice generates a random session key and encrypts it with Bob's public key. After sending the encrypted session key to Bob, Bob can decrypt it using his private key. Now, Alice and Bob can communicate using fast symmetric encryption and the generated session key. The use of public-key cryptography solves the key distribution problem. If an eavesdropper gets his hands on the session key, he can decrypt the messages encrypted with it. As the session key is only valid for one session and destroyed afterwards, the risk of compromising the session key

is reduced. Details can be found in [31]. This approach is called hybrid cryptosystem and is used in, for example, Transport Layer Security/Secure Socket Layer (TLS/SSL)⁵.

2.3.4 Public-Key Certificates

How can Alice ensure, that the public key received from Bob is Bob's public key and not a substituted one? How can Alice authenticate Bob? In [31], Schneier describes the use of digital certificates in order to authenticate the certificate's owner and his public key. A certificate consists of its owner's personal information, timestamps, expiration date and its owner's public key. A trusted authority signs the certificate or more likely the certificate's hash guaranteeing that the owner of the certificate is the person stated on the certificate. Ideally, the trusted authority requires some kind of authentication process for certificate owners before signing a certificate.

The trusted authority signs a certificate by encrypting the whole certificate or the generated hash value with its private key. The signature can be decrypted using the trusted authority's public key. As the signature can only be generated using the private key and as it is assumed that only the trusted authority holds a copy of its private key, it can be assumed that the signature is valid.

The trusted authority can either be a public certification authority (CA) or (multiple) trusted individuals. In either case, Alice has to trust the signing authority. As there are countless authorities, it is impossible for Alice to know and trust all of them. Thus, a so-called certification chain is generated. A trusted authority is certified by another trusted authority, and so on. If Alice wants to verify a certificate, she has to verify the signatures of all trusted authorities certifying each other until she reaches the certificate of one authority she trusts.

Although certificates are only valid for a certain time period, there must be a mechanism to revoke a valid certificate as a certificate can be stolen/compromised or invalid because of administrative reasons. Certificate revocation is a complicated process and still a big problem as the certificate cannot be removed easily. Most certification authorities provide a database listing all revoked certificates. Alice has to check these databases before trusting a certificate.

2.3.5 Public-Key Infrastructure

In order to ensure a secure communication between the nodes within the Smart Grid Infrastructure, a reliable cryptographic system is required. It must guarantee message

⁵see RFC5246 [32]

integrity and confidentiality as well as provide mechanisms for identification and authentication of nodes. A suitable approach is the use of a Public Key Infrastructure (PKI).

According to [31], a PKI uses public-key cryptography to secure communication. Each entity is required to own two different keys, a public and a private one. As the names indicate, the public key is publicly accessible whereas the private key is kept a secret. Cryptographic functions can be applied using one key but only be undone using the other. Alice can send a cipher to Bob using Bob's public key to encrypt the message. Bob can decrypt the cipher using his private key. As the cipher can only be decrypted with the private key, Mallory cannot decrypt the message. The system also works vice versa. Bob can encrypt a hash calculated from a message and send the message including the cipher to Alice. Alice can decrypt the cipher using Bob's public key. Comparing the decrypted hash with the hash of the received message, Alice can verify if the message was altered during transmission and if Bob was the real sender. This technique is called signing. Public-key cryptography is based on trap-door one-way functions. They are easy to compute in one direction, but without knowing the secret (the trap-door) hard to compute in the other. The key pair is generated in a way that it is computationally hard to derive one key from the other.

Digital certificates are used for identification and authentication. A certificate consists of its owner's personal information, timestamps, expiration dates and the owner's public key. A trusted certification authority (CA) signs the certificate guaranteeing that the owner of the certificate is the person stated on the certificate.

A Public Key Infrastructure is needed for establishing, maintaining and distributing the public/private key pair and its assignment to a certain identity as well as the certificate issuing and management process.

2.3.6 Applying a Public-Key Infrastructure to the Smart Grid

To ensure message integrity and prevent eavesdropping, a secure way for communication between the single nodes is required within a Smart Grid. A system guaranteeing both, integrity and confidentiality for the communication channel and authentication and authorization for accessing provided services has to be implemented. A key management system can be seen as the base of such a system.

Long et al. [33] propose an encryption scheme based on shared secrets. They divide the Smart Grid control architecture into two levels, each with its own key management system, tailored to the computational resources of the devices, respectively. While, at a first glance, shared keys seem to be an easy solution, within a growing infrastructure, the number of keys is growing rapidly. Every entity has to maintain one key per secure connection

to another entity, causing high efforts for key management, renewal and distribution.

To solve this key management issue and to keep the number of secret keys to a minimum, the use of public keys is recommended. As Smith points out in [34], due to the use of digital signatures enabled by public key cryptography, the secret known by each device cuts down to exactly one, its own private key. Public key cryptography needs a Public Key Infrastructure (PKI) used for establishing, maintaining and distributing the public/private key pair and its assignment to a certain identity. According to Smith, a PKI does not have good scalability properties. Therefore, deploying a PKI within a Smart Grid Infrastructure can raise serious issues on how to manage a vast amount of Certification Authorities (CA), maintain the trust path and on how to revoke already issued certificates.

Smith [34] as well as Baumeister [35] point out that within the Smart Grid, there are special requirements for scalability, high availability, compatibility and the ability to update certificates and devices if needed. A vast amount of entities requires a vast amount of issued certificates. High effort to manage these certificates, maintain the trust path and to revoke already issued certificates is needed. In terms of high availability, entities must still be able to verify certificates if the CA is not reachable. PKIs from different utilities with different Policy Enforcements are required to interact, consequently complicating the trust path to verify issued certificates. Entities are designed for a long lifetime, raising the question of the length of the certificate lifetime and the ability to upgrade the entity's cryptographic library.

Issues of scalability and compatibility can be solved using a proper CA topology, and therefore build a proper trust model. Different models and topologies are available.

As described by Buchmann et al. [36], the simplest model is Direct Trust. Every entity receives all the public keys of the entities it has to communicate with at initial set up, and stores them. For secure communication, the entity uses only the stored public keys. While this model requires no certificate verification and therefore no root of trust, it needs no connection to any CA. Therefore it increases high availability. A big drawback is the high effort needed for maintaining the public keys distributed to all entities.

At the Hierarchical Trust Model, public key and identity of an entity are certified by certification authorities (CAs). For improved scalability, multiple CAs are issuing certificates. In the Hierarchical Trust Model, all CAs are arranged in a tree with the inner nodes being CAs and the leaves being the entities. The root of the tree, the root CA, is the trust anchor for all entities and thus fully trusted. Each node signs the certificate of its child nodes. A certificate can be verified following the certification path through the tree until the root CA is reached. While this model offers good scalability, the root CA is totally trusted and hence poses a security vulnerability. Compromising the root CA

leads to a complete loss of security and trust within the model. In addition, considering a nationally or internationally connected grid consisting of many utilities each operating their own CAs, it will be difficult to find one CA trusted by all utilities. More details can be found in [35, 36, 16].

Instead of building a hierarchical tree, in the Mesh Trust Model, all CAs certify each other. The certificate issuer is also the trust root. The trust path is quite short and therefore easy to verify. According to Baumeister, this decentralized model is flexible and robust but lacks in scalability. It might also be difficult for a single CA to verify the trustworthiness of another CA. As the model is decentralized, no common security policies are in force.

The Federated Trust Management Model combines the benefits of the Hierarchical Trust Model and the Mesh Trust Model. A hierarchical PKI is established per domain. Domains are cross certifying each other using a bridge. The bridge, a centralized agent, maintains the cross certification relationships and enforces security policies. This flexible approach allows interoperability within different domains and provides centralized management allowing efficient control and management of the whole PKI. The Bridge CA creates a single point of failure and therefore affects high availability. In [35], Baumeister also notes that cross certification between domains can create inefficient certification paths slowing the certificate verification process and hence hinders the system's real time capability. Nevertheless, both, Baumeister and NIST recommend the Federated Trust Management Model as the only capable trust model for the Smart Grid [35, 16]. Using the Federated Trust Management Model, each entity can deploy and maintain its own PKI, as suggested in [16]. The bridge can be on a regional basis.

Through compromising the private key or changing certificate information, a certificate can become invalid before its lifetime is over, in which case it must be revoked. In [36], Buchmann describes that a PKI can publish revoked certificates in a Certificate Revocation List (CRL). During the verification of a certificate, each entity has to download the CRL to check if the certificate is listed and is therefore revoked. CRLs tend to be large files generating high overhead and hence are hard to process for low resource entities.

A better solution is the implementation of the Online Certificate Status Protocol (OCSP) [36]. During certificate validation, the entity sends a query about the revocation status of the certificate to a OCSP server. The provided information is up to date and communication overhead is reduced. The accessibility of the OCSP server can result in a high availability issue. OCSP stapling⁶ can be used to solve this problem. An entity obtains a OCSP response for its own certificate and provides the cached response to any entity requesting the certificate. The use of OCSP stapling is also recommended by NIST [16].

⁶see RFC 4366 [37]

2.3.7 Hierarchical Key Generation

Already in 1981, Lamport [38] suggested to use a hash chain generating a series of One Time Passwords (OTP) to address the problem of identification by sending a secret password over an insecure communication channel. To construct a hash chain of length N , a one-way hash function F is applied to an initial seed value s N -times.

$$F^2(s) = F(F(s)) \quad (2.1)$$

$$F^N(s) = F(F^{N-1}(s)) \quad (2.2)$$

F^N is used as the initial value and therefore sent to the server in a secure way. The remaining OTPs $F^1 \dots F^{N-1}$ are stored in a secure manner on the client. The client can use F^{N-1} as the next OTP. Knowing F^N , the server can verify the OTP by calculating $F^N = F(F^{N-1})$, but neither the server nor any eavesdropper can determine the next valid OTP as F is a one-way hash function. After a successful authentication, the server stores F^{N-1} as the next value to compare with and F^{N-2} is used for the next authentication attempt. The S/KEY One-Time Password System is one example of how to use OTP for authentication [39].

According to [40], a One-Way Hash Function (OWHF) is defined as:

A function $H()$ that maps an arbitrary length message M to a fixed length message digest MD is a One-Way Hash Function (OWHF), if it satisfies the following properties:

1. *The description of $H()$ is publicly known and should not require any secret information for its operation.*
2. *Given M , it is easy to compute $H(M)$.*
3. *Given MD in the rang of $H()$, it is hard to find a message M such that $H(M) = MD$, and given M and $H(M)$, it is hard to find a message $M'(\neq M)$ such that $H(M') = H(M)$.*

The idea of hash chains can be found in many security systems [41]. Hash chains or hash trees are also used for access control to JPEG2000 coded images or H.264/scalable coded video (H.264/SVC) [42, 43, 44].

Imaizumi et al. propose a scheme for hierarchical access control to JPEG2000 coded images in [42]. Image properties are encrypted with different keys. According to the keys gained, a certain resolution or property can be decrypted. To minimize the number of managed keys, a hierarchical key management is introduced. All keys used are derived from one managed master key using hash chains and cyclic shifts. For decryption, the key for the highest resolution, is used. As the used hash function is no secret, the keys

needed to decrypt the requested resolution can be derived from the one key provided. It is impossible to decrypt the image in a higher resolution, as the needed keys cannot be derived from the one provided. In [43], Wu et al. propose a similar system for access control to JPEG2000 coded images.

In [44], Asghar et al. suggest to use key derivation for encrypting multi-layered coded video (H.264/SCV). The aim is the same as in Imaizumi et al. [42]. A user should be able to watch his/her subscribed layer data when holding just one key. For key generation and distribution, Asghar et al. use the Multimedia Internet Keying Protocol (MIKEY) [45]. Key derivation is done within the MIKEY key generation process. After key generation and distribution, an Advanced Encryption Standard - Counter Mode (AES-CM) Cipher algorithm is used for encryption.

Access control to a multi-resolution representation of load data has similar requirements as for JPEG2000 coded images or H.264/SCV encoded videos. Techniques used for these use cases can be adopted to the Smart Grid. As many successful security systems build on hash chains and one-way hash functions, they can be seen as well-established and secure.

2.4 Multi-Resolution Load Data Representation

To preserve users' privacy, the resolution of load data generated by a Smart Meter can be reduced. As different use cases within the Smart Grid require different resolutions, it is difficult to determine a resolution suitable for all use cases. In addition, according to the framework for privacy aware design proposed by Wicker et al. in [7], there is no need for entities to get access to load data in a higher resolution than actually needed. To solve this problem, Efthymiou [46] and Engel [12] proposes to provide a Smart Meter's load data in multiple resolutions. Efthymiou splits the load data in a high and a low frequency part and anonymizes the high frequency part. Engel proposes a different approach which controls the access to the different resolutions. Access to a certain resolution is only granted according to an entity's need. Furthermore the user can decide, if access to a certain resolution is granted or revoked. Engel [12] suggests to use the wavelet transform based on the Haar wavelet and lifting scheme. This section describes the idea behind the wavelet transformation and the lifting scheme.

2.4.1 Discrete Wavelet Transform

The wavelet transform is a mathematical function splitting a signal into a significant and a less significant part. Compared to the Fourier analysis, the wavelet transform benefits from a better recognition of abrupt changes within a signal. Therefore, it can for example

be used for edge detection. Other use cases are solving partial differential equations, signal compression or resolution reduction.

For analyzing finite signals, the discrete wavelet transform is used (DWT). This document always refers to the DWT.

The simplest wavelet transform consists of two steps: Calculating the average and the difference of two values. The average values form the low frequency band (less significant part) and the differences the high frequency band (significant part). More formally, a finite signal of length N is defined as $x[0], x[1], \dots, x[N-1]$. The above described transform can be written as

$$s[j] = \frac{x[2j] + x[2j + 1]}{2} \quad (2.3)$$

$$d[j] = x[2j] - s[j] \quad (2.4)$$

with $s[j]$ forming the new low frequency signal and $d[j]$ forming the new high frequency signal. To reproduce the original signal, the inverse transform can be performed using

$$x[2j] = s[j] + d[j] \quad (2.5)$$

$$x[2j + 1] = s[j] - d[j] \quad (2.6)$$

[6].

Note that the inverse transform will restore the original transform. Therefore the wavelet transform can be considered as lossless [6].

The two signals x and s are quite similar, with s consisting of only half the amount of values than x and therefore being of less resolution. Comparing s and d to x , the dynamic range is reduced. This feature enables a better compression of the transformed signal.

The easiest implementation of the wavelet transform is the so-called lifting scheme. It consists of a prediction and an update procedure per lifting step. The prediction procedure assumes, that there is a high correlation between two sequenced values within a signal. Consequently, the first value can be seen as a predictor of the second. Instead of calculating the difference between a value and the according average (see equation 2.4), the correction to the prediction is preserved by calculating the difference between the two values. The update procedure extracts the essential feature of the signal. This can be done by calculating the average, as this divides the number of values in half but still preserves the overall structure of the signal.

Prediction and update procedure can be described as following:

$$d[j] = x[2j + 1] - x[2j] \quad (2.7)$$

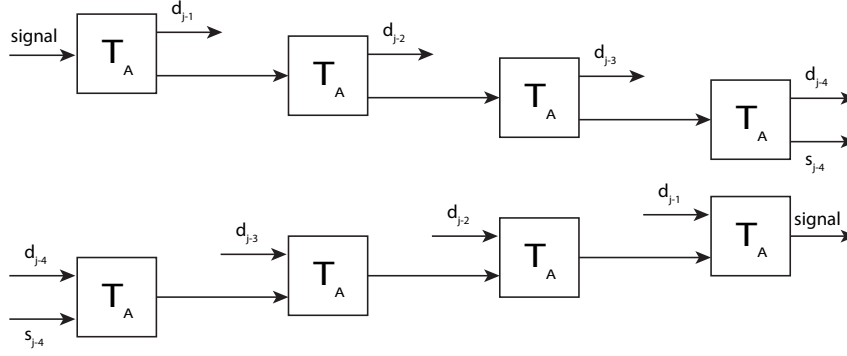


Figure 2.7: The lifting steps can be concatenated to increase the depth of the discrete wavelet transform. While the high frequency part of each step is preserved, the low frequency part is used as input signal for the next lifting step. The inverse transform works the other way around. Adopted from [6]

$$s[j] = x[2j] + \frac{1}{2}d[j] \quad (2.8)$$

and the according inverse transform

$$x[2j] = s[j] - \frac{1}{2}d[j] \quad (2.9)$$

$$x[2j + 1] = d[j] + x[2j] \quad (2.10)$$

Comparing 2.7,2.8 to 2.3,2.4 the main feature of the wavelet transform, namely splitting the signal in a significant and a less significant part, is still preserved. Only the calculation of the difference is defined in a different way. The calculation of the average remains the same, although differently noted.

Using the lifting scheme, the discrete wavelet transform is obtained by concatenating a certain number of lifting steps as shown in Figure 2.7. The resulting low frequency part of one lifting step is used as input signal for the next lifting step. The high frequency part of each step is preserved for later usage or inverse transformation.

The above described wavelet transform using averages and differences is called Haar wavelet. Considering the lifting scheme, the correct notation for the Haar wavelet is

$$d_j[n] = s_{j+1}[2n + 1] - s_{j+1}[2n] \quad (2.11)$$

$$s_j[n] = s_{j+1}[2n] + \frac{1}{2}d_j[n] \quad (2.12)$$

with $0 \leq n < \frac{1}{2}N_j$ for the input signal of each lifting step. N_j is the length of the input signal of lifting step j . j is the index for the current lifting step defined as $0 \leq j < \log_2(N)$ with $\log_2(N) - 1$ defining the first lifting step and $j = \log_2(N)$ defining the original input

signal. Note that the signal must have a length $N = 2^i$ resulting in a maximum of i lifting steps.

The inverse transform of the Haar wavelet using the lifting scheme is defined as

$$s_j[2n] = s_{j-1}[n] - \frac{1}{2}d_{j-1}[n] \quad (2.13)$$

$$s_j[2n + 1] = s_{j-1}[n] + \frac{1}{2}d_{j-1}[n] \quad (2.14)$$

[6]

In [6], Jensen et al. add a normalization step to each lifting step. This step is not further explained within this document. Note that for each lifting step, the mean value of the original signal is preserved.

s_0 and $d_0, d_1, \dots, d_{\log_2(N)-1}$ are called wavelet coefficients. These are needed to restore the original signal using the inverse wavelet transform. In order to achieve some kind of filtering (e.g., noise reduction), $d_0, d_1, \dots, d_{\log_2(N)-1}$ can be altered using e.g a certain threshold [6]. Beside the Haar wavelet, there are other wavelets like the Daubechies wavelet or LeGall wavelet using different functions for the prediction and update process. More details on the Discrete Wavelet Transform can be found in [6]. If only integers can be used as wavelet coefficients, in [47], Engel et al. point out a different formula for the Haar wavelet lifting scheme.

Smart Grid Communication Infrastructure

This chapter discusses a conception approach on how to design a Smart Grid Infrastructure feasible for real-world use cases. Special emphases are placed on a secure communication between the entities as well as preserving consumers' privacy. The proposed approach is based on a simple communication concept. The used technologies are well established, hence guarantee reliability and security. The Smart Grid Infrastructure is split into two separated networks. The networks can communicate with each other using the Grid Operator as gateway or proxy. This concept promises a high protection of the Advanced Metering Infrastructure from hacking attacks or denial of service attacks. This chapter discusses the different data flows within the Smart Grid Infrastructure and analyses the security and privacy risk these data flows are exposed to. Furthermore, different privacy preserving methods to secure consumption data are explained and discussed. These methods are (i) aggregation of consumption data, (ii) consumption measurement from a substation level and (iii) representation of load data in multiple resolutions. The different methods are combined to improve data usability while still preserving consumer privacy. The advantages and disadvantages of each combination are discussed. The combination of multi-resolution consumption data and consumption measurement from a substation level is the most promising. This combination offers a lot of freedom on how data can be used and eliminates the complex protocols needed for neighborhood aggregation. The final section of this chapter shows how to present consumption data in multiple resolutions and how to encrypt each resolution with a different resolution key. Presenting the consumption data in multiple resolution adds a new degree of freedom. Access to privacy relevant information can be controlled on a finer basis than just granting or denying access to certain information. This approach also enables the consumer to decide, which entity gets access to which resolution. The consumer is in charge of controlling his/her privacy. An implementation of the proposed architecture is shown in Chapter 4.

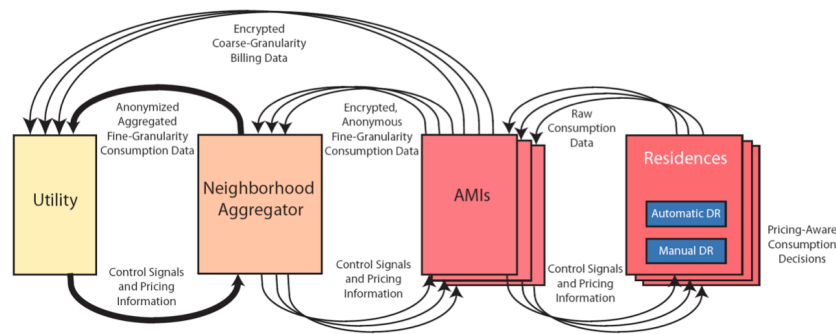


Figure 3.1: Data flows within a privacy-aware demand response architecture. From [7]

3.1 Data Flows within the Smart Grid

The first step towards a privacy preserving Smart Grid Infrastructure is to analyze and classify the different data flows. Figure 3.1 shows a high-level communication architecture proposed by Wicker and Thomas [7]. According to this architecture, Wicker et al. define four different data flows:

Pricing information: In order to alter consumer behavior, real time pricing information must be distributed from the utility to the customer. While pricing information can neither be seen as confidential nor carries any privacy sensitive data, there is no need for special protection. Only integrity must be guaranteed, as manipulating pricing information affects customer behavior and billing.

Billing: In order to achieve correct billing, the utility needs to know fine granulated consumption data to charge the customer with the right real time price. As fine granulated consumption data can be used to deduce customer behavior, customer privacy is affected. Instead of sending the consumption data to the utility, Wicker et al. suggest a different approach. The price-weighted consumption data can be accumulated and stored by the Smart Meter and send to the utility on a daily, weekly or monthly basis. As there is no possibility to retrieve consumer behavior patterns from accumulated consumption data, privacy issues are irrelevant. For this approach, the Smart Meter must be a trusted and tamper proof device. Confidentiality and integrity of the transmitted price-weighted consumption data must be ensured.

Control signals: In Demand Response systems (DR), customer appliances can be controlled by the utility or a third party entity to regulate the electricity consumption. From these control signals, no privacy sensitive data can be retrieved and therefore no special security steps in terms of privacy are needed. Nevertheless, because there is a potential security issue with these control signals, confidentiality and integrity must, once again, be ensured.

Consumption data: For tasks such as grid balancing, demand prediction or price model

calculations, fine granular consumption data is needed by the utility or third party entities. At the same time, this data can be used to draw usage patterns and infiltrate user privacy. Therefore, this data has to be handled with special care. There are several approaches to maintain customer privacy within this task. These are discussed in the Subsection 3.3.

3.2 Smart Grid Architecture

In order to preserve privacy and to ensure secure communication, a system guaranteeing integrity, confidentiality, and authentication is needed within the smart grid. Encrypted communication between two entities must be confidential, therefore no other entity should be capable of decrypting this communication channel. In addition, third party entities should also be able to use services if access is granted to them. It is essential that the system is designed following the framework for privacy aware design proposed in [7]. Each entity should only have access to services and resources on a need to know basis. Information is only stored as long as needed and the user has to be informed how his/her data is being used. Access should be granted on an opt-in basis as opposed to the more prevalent (and less privacy-enabling) opt-out basis.

Possible attacks on the Smart Grid Communication Infrastructure can come from many different sides, namely the user or neighbor, the Grid Operator, Utility or any third party with or without intended access to the Grid. Independent of their origin, attacks can be classified into the following groups: altering/forging messages, eavesdropping, data misuse, altering firmware or stealing private keys and denial of service. The approach proposed in this paper addresses these attacks by relying on well-established techniques for content and communication encryption. Hence, these techniques can be assumed to be safe.

In Section 2.3.6, different approaches on designing a suitable key management system for the Smart Grid have been discussed. A PKI is the only suitable key management system with the capability to manage a big infrastructure with a vast amount of issued certificates. The approach proposed in this paper relies on a certificate based Public Key Infrastructure (PKI). Several PKIs are standardized and well-established, therefore guaranteeing reliability and security. This approach also allows third parties to access the Smart Grid Infrastructure in a secure manner.

The proposed Smart Grid Infrastructure is shown in Figure 3.2. It is split into two separated networks, the grid operator's inner network and the public network. The inner network consist of all AMIs, sensors and other entities managed by the Grid Operator. The public network consist of the utility, networks of other grid operators and any third

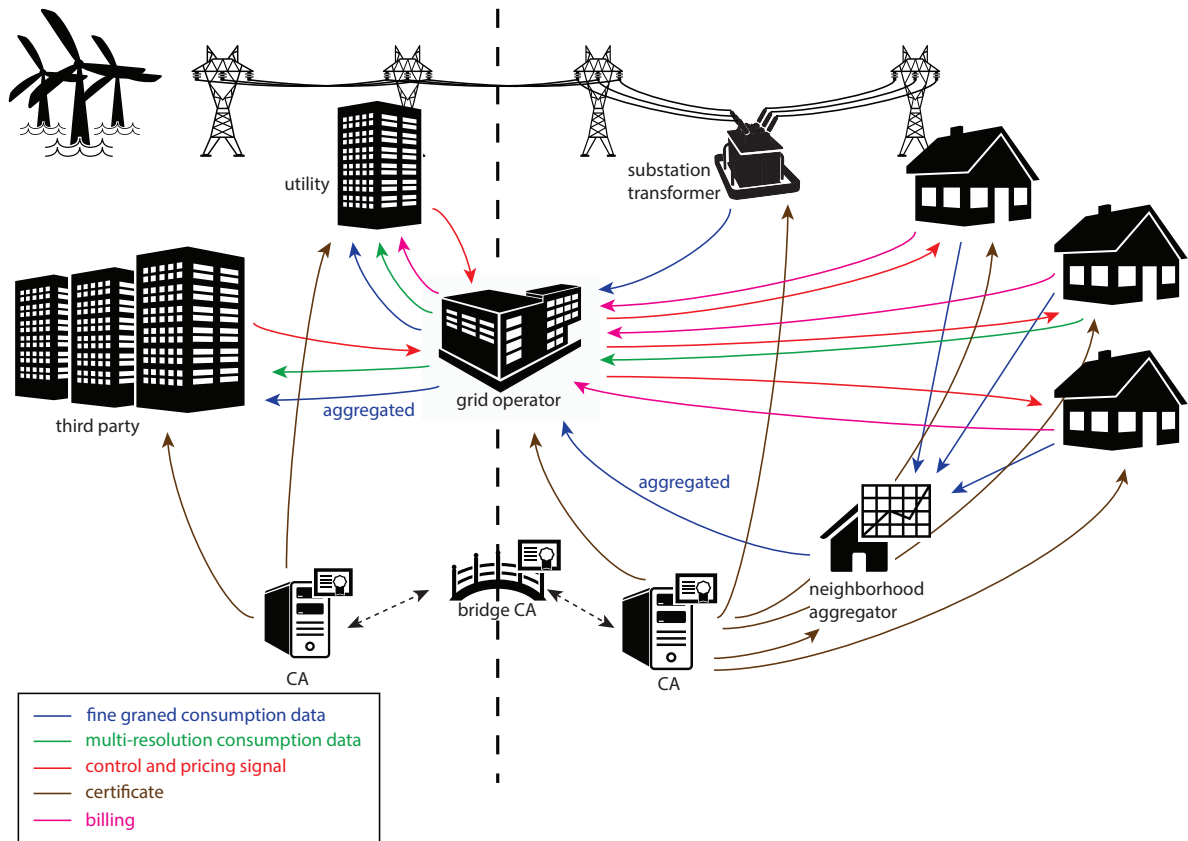


Figure 3.2: The Smart Grid is divided into two different networks: the public network and the Grid Operator's inner network.

party entities. The Grid Operator works as a proxy to enable and also control the communication between these two networks. Apart from the advantage, that the Grid Operator can use its preferred means of communication within the inner network, this separation is necessary to protect the fragile inner network from any possible attack from the public network. Not exposing Smart Meters directly to a public network improves security as the Grid Operator can act as a firewall only allowing authorized entities to communicate with the Smart Meters. AMIs or Smart Meters are devices with low computational power, vulnerable to Denial of Service Attacks (DoS Attacks). Intruding or harming the inner network can lead to serious problems in energy supply. An attack can result in serious issues on grid balancing and pricing. Monitoring and blocking unauthorized traffic by the Grid Operator is an essential part on increasing reliability and availability within the Smart Grid Infrastructure.

This architecture assumes that the Grid Operator can be trusted. This assumption is necessary for grid safety and stability. While the Grid Operator can read possible control signals, content encryption mechanisms ensure that the Grid Operator cannot read any privacy sensitive information.

The proposed Public Key Infrastructure is based on the Federated Trust Management

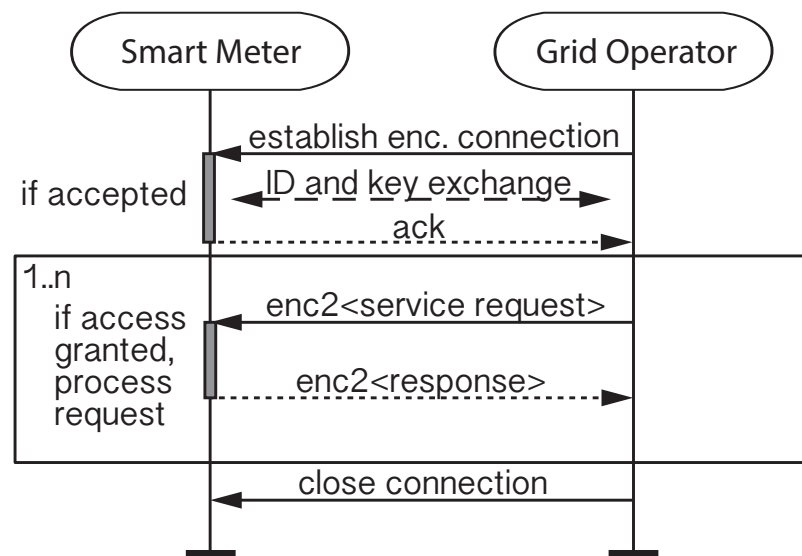


Figure 3.3: The Grid Operator can communicate with the Smart Meter using an encrypted connection.

Model. The CAs within the inner network are operated by the Grid Operator, the ones in the public network are operated by an independent authority. A bridge is used to enable communication with other PKIs, therefore simplifying the certificate management as well as the trust path. Every entity communicating within the Smart Grid Infrastructure obtains a certificate to identify itself. The CAs offer a OCSP Service for fast certificate validation.

The Smart Meter plays a main role in the proposed system and is therefore a trusted device. A Smart Meter must be capable to generate strong keys and store these keys in a manner, that they cannot be read or altered from outside. In addition, a Smart Meter must be able to compute cryptographic functions. As suggested by the United States National Institute of Standards and Technology (NIST) [16] and Wicker et al. [7], a Hardware Security Module (HSM) or a Trusted Platform Module (TPM) can be used to fulfill these requirements. Another requirement is tamper resistance. It must be guaranteed that nobody can intrude or tamper the Smart Meter without authorization. This embraces changes in hardware as well as in software/firmware. For identification and content encryption, each Smart Meter holds a valid certificate including a private/public key pair.

Every message sent in the Smart Grid Infrastructure includes a message hash signed by the sender. This ensures integrity and prevents intruders from sending forged messages. Apart from broadcasts, individual communication is always encrypted using public key cryptography (asymmetric encryption) for initial session key exchange and then changing to symmetric encryption using the prior exchanged session key.

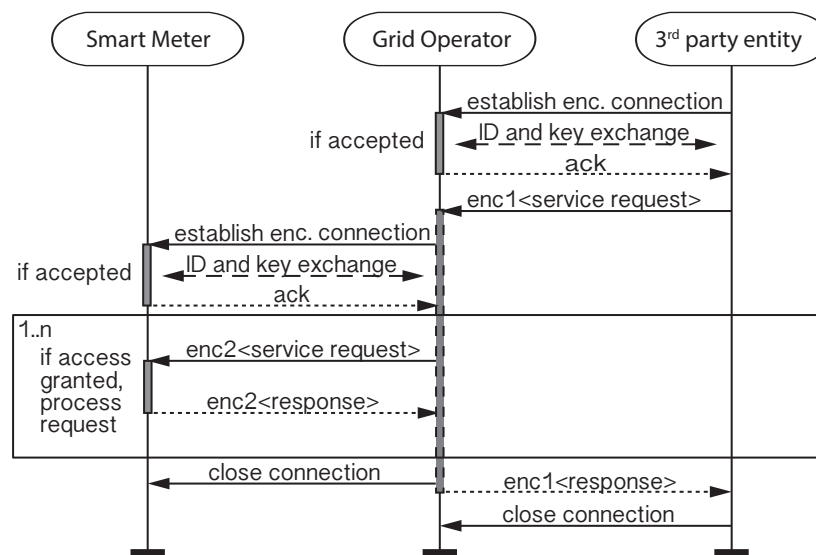


Figure 3.4: A third party entity can establish a connection to a Smart Meter using the Grid Operator as a proxy.

The utility or third party entities can query information from or send control signals to individual Smart Meters using an application programming interface (API) provided by the Grid Operator. Figure 3.3 shows the communication sequence for establishing a connection between the Smart Grid Operator and a Smart Meter. First, the Grid Operator establishes an encrypted connection to the Smart Meter using Transport Layer Security (TLS)¹. The Smart Meter accepts the connection if the Grid Operator provides a valid certificate. As soon as the encrypted connection is established successfully, the Grid Operator can use the Smart Meter's API to place a service request. If the Grid Operator has permission to access the service, the Smart Meter processes the request and sends the result back to the Grid Operator. The Grid Operator can place multiple service requests. The Grid Operator closes the connection as soon as the connection is not needed any more.

Whereas the Grid Operator can connect directly to a Smart Meter, third party entities must connect via the Grid Operator's API with the Grid Operator acting as a proxy. As shown in Figure 3.4, first the third party entity establishes an encrypted connection to the Grid Operator and identifies itself. If the third party entity has permission to access the Smart Grid Infrastructure, the Grid Operator accepts the connection. Now, using the encrypted channel, the third party sends a service request including the target Smart Meter ID to the Grid Operator. After verifying the service request, the Grid Operator establishes an encrypted connection to the Smart Meter and forwards the service request. Based on the third party entity's certificate, the Smart Meter grants or denies access to the requested service. If access is granted, the Smart Meter processes the request and sends

¹see IETF RFC 5246 [32]

the response back to the Grid Operator. The Grid Operator then forwards the response to the third party entity. The third party can place multiple service requests. As soon as the connection is not needed any more, the Grid Operator closes the encrypted connection to the Smart Meter and the third party entity closes the encrypted connection to the Grid Operator. Note that an encrypted communication is established between the third party entity and the Grid Operator as well as between the Grid Operator and the Smart Meter. Since these two connections are independent, the Grid Operator can read the whole communication between third party entity and Smart Meter. The proposed sequence only guarantees communication encryption preventing eavesdropping. For content encryption and hence privacy protection, the Smart Meter can encrypt the response using the third party entity's public key. It is necessary for grid stability and reliability to differ between communication and content encryption. Within the Smart Grid, there are multiple data flows used for load balancing and controlling/managing the grid. Intruding and altering these data flows can cause severe damage to the grid. Hence, it is necessary that the Grid Operator can monitor and control the data flows within the Smart Grid Infrastructure, requiring the Grid Operator to read the sent messages. For data flows containing private information, content encryption has to be applied, preventing the Grid Operator from reading these data flows. However, it must be ensured, that these data flows cannot harm the grid.

In Section 3.1, the basic data flows within a Smart Grid Infrastructure are discussed. Figure 3.2 applies these data flows to the proposed architecture. While Wicker and Thomas suggest to send pricing information and control signals to the neighborhood aggregator before distributing it to the AMIs, this document suggests to distribute pricing information and control signals via broadcast. These messages can be split into two different types: messages referring to all AMIs and residences or messages referring only to individual AMIs.

Pricing is no secret and therefore can be sent as broadcast message to every AMI. The utility forwards the information to the Grid Operator, which distributes the information to all AMIs. Within a free-market economy, there are plenty of utilities and energy brokers offering a variety of pricing models to their customers. There must be a way to distribute this pricing information to the according residences and, at the same time, ensure that each residence uses the right price. This could be possible by sending pricing information to the related AMIs or by broadcasting the encrypted pricing information with the related AMIs owning a suitable key. A detailed discussion of this topic would go beyond the scope of this paper.

Control signals are sent by the Grid Operator, utility or any third party in charge of Demand Response. There are two different types of control signals. General ones, addressed to all AMIs and individual ones, only concerning some AMIs participating for example in

a special energy saving program. General control signals can be sent via broadcast and there is no need for encryption. Individual control signals are distributed via single cast. These are encrypted as they might be part of a corporate secret. Nonetheless, the Grid Operator should have the possibility to read and block these control signals because faulty ones can disturb grid stability and therefore cause serious harm to grid and property.

Billing information is sent from the Smart Meter via the Grid Operator to the utility. Information is signed and encrypted. There is no need for the Grid Operator to read this information. The Smart Meter must be a trusted device and also tamper proof.

For aggregation, the data flow is strongly dependent on the implemented protocol. Independent of the chosen protocol, the communication can be wrapped in the hybrid encryption scheme used within this architecture.

For multi-resolution consumption data representation, the protocol is straight-forward and can be found in Section 3.4. The utility or any third-party entity receives the resolution key suitable to decrypt a certain resolution from the Smart Meter. During this key exchange process, content encryption ensures that the resolution key cannot be read by any other entity. Hybrid encryption between utility or third party and Smart Meter can be used for this purpose. The multi-resolution consumption data is encrypted and therefore needs no additional encryption.

3.3 Privacy Preserving Methods to Secure Consumption Data

In the literature, there are two types of approaches showing high potential to resolve the privacy issue in collecting fine granular consumption data: (i) secure aggregation of encrypted data using homomorphic encryption, and (ii) representation of data in multiple resolutions, each associated with different access levels.

As in most of the use cases, fine granular consumption data on a substation level is sufficient and privacy-preserving aggregation can be used to preserve users' privacy. Consumption data of multiple residences is cumulated and the resulting signal is submitted to the utility or any third party entity. Out of the aggregate, it is impossible to determine the consumption data of a single residence. The use of homomorphic encryption allows a secure aggregation of the encrypted consumption data. In the literature there are several different approaches describing how to design aggregation in a privacy preserving manner. Erkin et al. give a good summary of recent developments in this field in [11]. Further proposals can, for example, be found in [48, 49]. The different approaches use techniques like masking, secret sharing, adding noise or differential privacy to blur single

measurements before their aggregation. Depending on the approach, the aggregation is done either by a special entity or by other Smart Meters². In any case, the Smart Meters are required to communicate with each other and to build up a virtual aggregation tree adding communication overhead. The cryptographic algorithms needed for privacy-preserving aggregation are computationally expensive³. As Smart Meters only provide low computational resources, they may struggle with computing these algorithms.

Instead of measuring fine granulated consumption data at the residences and aggregate them later on using an external aggregator, consumption data can be measured directly at a substation level. This approach has the big advantage that no fine granulated consumption data has to leave the residence's Smart Meter and that there is no need to set up and maintain trusted aggregation points. One use case of the Smart Grid is loss detection. Consumption measurements of electrical substations are compared with the sum of consumption measured on residences' Smart Meters to detect electricity loss through theft or faulty devices. Using this approach, loss detection cannot be performed as there is no residential consumption data available. Other use cases are Demand Response and energy saving tips. Both of them can be realized with this approach. Instead of sending fine granular consumption data to the cloud for further processing, this computation can be done directly at the residence's AMI hindering privacy sensitive data from leaving the residence.

A completely different approach is suggested by Eibl and Engel. Non-intrusive load monitoring (NILM) techniques used to analyze user behavior require fine granular data to deliver accurate results. By decreasing time resolution of consumption data, NILM can still generate results suitable for grid balancing and demand prediction while, at the same time, preserving users' privacy. [10] shows the effect of reduced resolution on non-intrusive appliance load monitoring algorithms (NIALM). Reducing time resolution to five minutes already shows a big impact on the accuracy of the result. In [12], Engel and Eibl stick to the conditional access paradigm. Access to a certain resolution is only granted when actually needed and only on behalf of the user. The user can decide which entity has access to which resolution. This puts the user in direct control of his/her privacy. The wavelet transform is used to split consumption data in multiple resolutions. Each resolution is then encrypted with a different hierarchical key. The resulting ciphers are packed into one single stream and transmitted to any entity interested in the consumption data. Each entity granted access to a certain resolution holds the key for this resolution and is thus able to decrypt the desired data. This approach allows an effective and secure way to distribute consumption data. In addition, using hierarchical keys reduces the amount of keys needed to distribute and therefore allows a slim and convenient key management.

²neighborhood aggregation

³see [11] for comparison

Although the time resolution is reduced, using the wavelet transform allows preservation of the original sum. The total consumption can be derived from any resolution. A possible protocol and key generation mechanism can be found in Section 3.4.

Aggregation, measuring at substation level and multi-resolution consumption data representation can be combined in multiple ways and therefore, depending on the combination, offer some new possibilities and benefits.

It is very likely that the utility uses consumption data from aggregation and from substation measurement. The obtained data can be compared and be used for fraud/loss detection within the grid.

The combination of aggregation and multi-resolution consumption data representation is already discussed by Engel and Eibl in [47]. They split consumption data in multiple resolutions before aggregating each resolution using neighborhood aggregation. This approach allows a multi-resolution representation of the aggregate. Access to the aggregate can now be handled on a need-to-know basis. While this approach still preserves the idea of the conditional access paradigm, the user cannot decide anymore, who gets access to the data. As users' privacy is preserved through the aggregation, user decision might not be relevant anymore. Multi-resolution consumption data representation and aggregation can also be used in parallel. Fine granular consumption data for grid balancing and load forecasting is accessed using the aggregate. Multi-resolution consumption data can be used for individual energy saving tips or price modeling. The maximum resolution available per residence can therefore be limited for further privacy enhancement.

An auspicious combination is the parallel use of multi-resolution consumption data representation and measurements on a substation level. It provides the same benefits as using aggregation while eliminating the need of an aggregation network. As discussed earlier, aggregation requires computationally intensive cryptographic algorithms to preserve users' privacy during the aggregation process. These algorithms are not required, if the "aggregate" can be directly measured on a substation level. In addition, aggregation networks require communication and key exchange between individual Smart Meters adding overhead to the communication network. The combination of multi-resolution representation of consumption data and measurements on a substation level require just straight forward communication between Smart Meters and Grid Operator or utility. Splitting consumption data in multiple resolutions requires little computational effort. The cryptographic algorithms needed for secure transmission and access management are well established and, compared to homomorphic encryption, inexpensive⁴. This approach gives access to fine grained consumption data of a whole block and to individual consumption data with reduced resolution. In case of individual consumption data, the user is still in charge of

⁴see [47] for further details

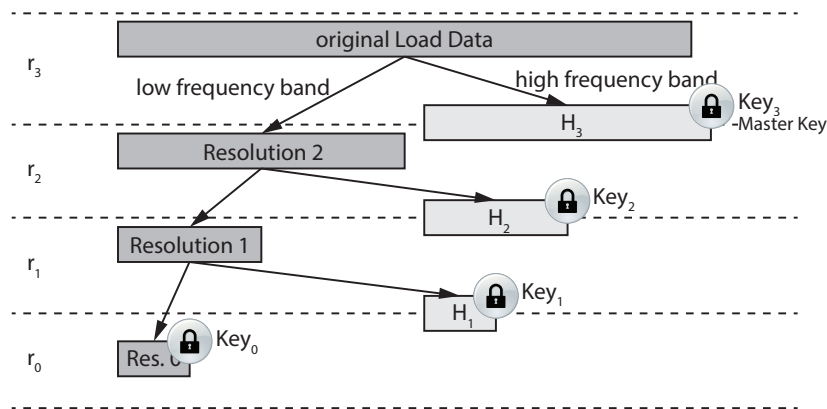


Figure 3.5: The Wavelet transform splits load data into high and low frequency band. The low frequency band equals load data with reduced resolution.

protecting his/her privacy as he/she can decide to whom access is granted for a certain resolution. Loss detection can be performed by comparing measurements from a substation level with the aggregate of individual measurements of the block served by the substation. The aggregate can be calculated by the utility. As each resolution level preserves the total consumption of the measurement, a low resolution level is sufficient for calculating the aggregate. Therefore, no further privacy preserving techniques are required.

3.4 Multi-Resolution Load Data Encryption and Distribution

As discussed in Section 2.4, Engel et al. propose a multi-resolution representation of load data to increase privacy [12, 47]. Access to a certain resolution is based on the conditional access paradigm. A given entity is granted access to a resolution necessary to fulfill its role. As a NILM or NALM algorithm needs high resolution data to achieve accurate results, reducing the resolution of the provided load data reduces the potential for abuse. In addition, the consumer can decide which entity is granted access to a certain resolution. This adds another degree of freedom as entities have to explain their data usage to gain users' trust.

Load data can be represented in multiple resolutions using a suitable wavelet transform, as suggested by Engel et al. in [12]. The Haar wavelet transform suits the requirements best. It consists of calculating averages and deltas, therefore needing few computational resources. The Haar wavelet is a lossless transform; under each resolution, the total consumption over the whole timespan can be derived.

The wavelet transform splits load data into a high and a low frequency band recursively

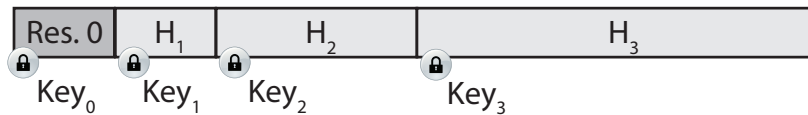


Figure 3.6: All wavelet coefficients needed for the inverse transformation are encrypted with different keys and transmitted as a single stream.

to a certain level. Where the low frequency band is used for the next recursive operation, the high frequency band is preserved. The low frequency band represents the data at a certain resolution with half the number of samples of the next higher resolution. The high frequency band represents the delta of a sample to the according sample of the low frequency band. The values from the high frequency band and the remaining value from the low frequency band are called wavelet coefficients. The wavelet coefficients are needed to do the inverse wavelet transform and restore the load data to a certain resolution. The steps described here can be seen in Figure 3.5.

In order to restore a certain resolution, the inverse wavelet transform is performed on the low frequency band and its corresponding high frequency band. The inverse starts with the coefficients of the lowest resolution and works its way up to the desired resolution.

In order to fulfill the conditional access paradigm introduced prior in this section, wavelet coefficients have to be encrypted with a different key for each resolution (from now on resolution key). Granting access to a certain resolution means to distribute the resolution keys for the certain resolution and for all lower resolutions to the requesting entity. A high number of resolution keys has to be managed and distributed, therefore introducing significant overhead for key management and storage.

In order to address the problem of high key management costs, Hierarchical Keys are introduced. Hierarchical Keys allow the decryption of multiple ciphertexts with a single key although the messages were encrypted with different keys, for example, encrypting three messages m_1, m_2, m_3 each with a different hierarchical key k_1, k_2, k_3 . In terms of decryption, using k_1 just decrypts m_1 , but k_2 or k_3 can be used to decrypt m_1, m_2 or m_1, m_2, m_3 , respectively. Hierarchical Keys therefore simplify key management, as less keys have to be known to decrypt multiple messages. Key generation and sample use cases have already been discussed in Section 2.3.7.

As the use case of multi-resolution representation of load data is quite similar to H.264/SVC and JPEG2000 encryption, techniques proposed in [44, 42, 43] can be adopted. A hierarchical resolution key is generated for each level of resolution. Resolution keys are derived from a master key using hash chains. Any appropriate one-way hash function can be used. Resolution key renewal can be done within a certain time period, e.g., daily. Wavelet coefficients are encrypted using the appropriate resolution key. The wavelet transform itself is

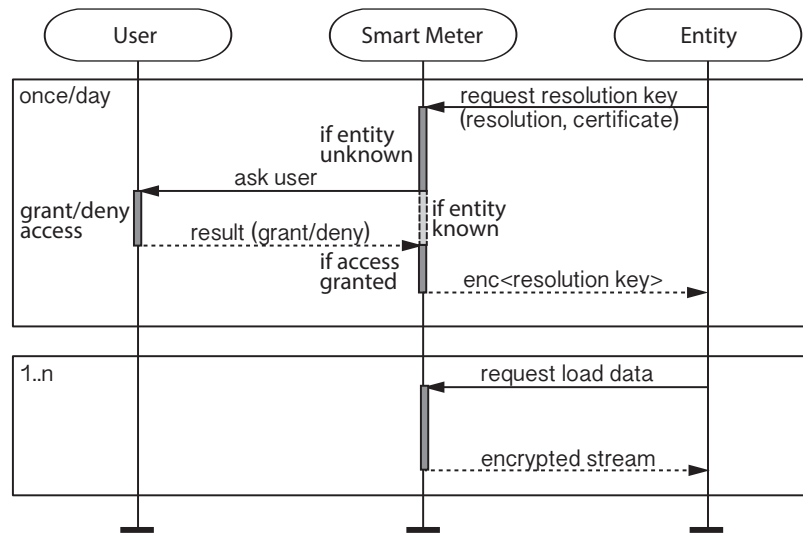


Figure 3.7: In order to access load data, the entity has to request the resolution key for the desired resolution.

performed on a cyclic basic, e.g., hourly, covering a fixed time span, e.g., the last 24 hours. The wavelet coefficients are packed to a single stream (see Figure 3.6) and transferred to any entity requesting it. According to the entity's resolution key, the entity is only able to decrypt the wavelet coefficients of the resolution access was granted to. As the one-way hash function is no secret, the entity can derive the resolution keys to encrypt the wavelet coefficients of a lower encryption but it cannot encrypt any wavelet coefficients of a higher resolution.

Figure 3.7 shows the service requests needed for obtaining load data. This sequence is based on the communication sequence shown in Figure 3.4. Before sending a service request to the Smart Meter, the entity has to establish a connection via the Grid Operator, as described in section 3.2. In order to obtain load data, the entity has to go through two steps, (i) obtaining a suitable resolution key and (ii) retrieving the load data. In order to obtain the resolution key, the entity has to request access for a certain resolution. Therefore, it sends a service request including the certificate and the requested resolution to the Smart Meter. The Smart Meter has to decide, if access is granted. If this is the entity's first access request, the Smart Meter forwards the request to the consumer as he/she can decide, if access for a certain resolution is granted to a certain entity. If the entity is known by the Smart Meter, access can be granted/denied based on the previous consumer decision. In case access is granted, the Smart Meter encrypts the resolution key using the entities public key and sends it to the entity. In a second step, the entity sends a load data request to the Smart Meter. The Smart Meter returns a stream containing the encrypted wavelet coefficients, as shown in Figure 3.6. There is no additional authentication process needed, as the stream is worthless without the resolution

key obtained in step one. By decrypting the wavelet coefficients and performing an inverse wavelet transform, the entity can now restore load data up to the resolution, access was granted. Load data can be obtained as long as the issued resolution key is valid. In order to ensure content security, the resolution key is encrypted using the requesting entity's public key. Hence, only the entity knows its private key, the resolution key cannot be decrypted by the Grid Operator working as a proxy.

Proof of Concept

The previous chapter discusses a conceptual approach to designing a Smart Grid Communication Infrastructure. The approach places emphasis on security and privacy protection. This chapter describes the software implementing this approach. The software is designed as a proof of concept, which demonstrates and evaluates the proposed Smart Grid Communication Infrastructure. It implements the following three main design aspects: (i) multi-resolution load data representation, (ii) hierarchical encryption and (iii) access management. Billing and load data aggregation are not included so far, but can be added in a further step. The component based, flexible software architecture ensures extensibility. The software consists of three parts, each capable to run on a different machine. The first part, the Smart Meter, simulates a Smart Meter. It provides hierarchical encrypted multi-resolution load data to the Grid Operator or any Third Party Entity. A user interface allows the consumer/end user to manage access permissions. The second part simulates the Grid Operator. It offers basic functionality for accessing the encrypted multi-resolution load data and forwards incoming requests from a Third Party Entity to the according Smart Meter. It also provides functionality needed to manage the connected Smart Meters. The third part of the software simulates a Third Party Entity trying to access multi-resolution load data using the Grid Operator as a proxy. The connections between the software parts are encrypted and require certificate authentication. All sent messages are signed by the sender to ensure message integrity. The architecture and design of each part is described in the following sections.

4.1 The Smart Meter

The Smart Meter is the central piece of the proof of concept. The software is designed to simulate a Smart Meter, providing encrypted multi-resolution load data via an Application Programming Interface (API). Each resolution is encrypted with a different hierarchical generated key. The implementation follows the concept described in Section 3.4. Access

to a certain resolution is granted or denied by the Smart Meter, based on a previous user decision. In general there are multiple user roles bonded to different permissions: (i) customer/end user, (ii) grid operator, (iii) utility, (iv) third party. The permissions depend on the usual tasks of each role. For example the customer or end user can access the user interface or administer permissions. The grid operator has access to basic load data and some administrative tasks required to maintain grid stability. The utility only gains access to billing information. The third party can access load data on a certain resolution, defined by the customer/end user. As already stated earlier, the customer/end user is responsible for the access management. He/she receives a notification as soon as a new entity places an access request to a certain resolution. The customer/end user has to decide, which permissions are granted and which role is assigned to the requesting entity. For security reasons, the customer/end user cannot alter the permissions for the grid operator. Access management can be done using the provided user interface. It also offers possibilities to read the current load data or show status information.

The simulation is based on real load data read from a file. The Simulation Engine reads the data set and provides one record per time period to the Smart Meter. The frequency depends on the granularity of the data set. For example, if the data set holds meter readings on a one second basis, one record per second is provided.

In order to allow the Grid Operator to interact with the Smart Meter, the Smart Meter registers itself at the Grid Operator during initial installation. The Smart Meter has a unique name/ID, which is used to identify it. The name must match the name listed on the Smart Meter's certificate.

In order to keep hardware costs at a minimum, Smart Meter is designed to run on platforms with low computational resources. Typical platforms are so-called single-board computers based on an ARM core, like the Raspberry Pi or the Cubieboard¹. The Smart Meter software is implemented using the Java SE Development Kit 8. Java's platform independence allows to run the software on any supported architecture or operating system. For evaluation or demonstration purpose, the software can be run on either real hardware like the Cubieboard or in multiple virtual machines simulating a big grid. As already described in Section 3.2, each Smart Meter receives one certificate to proof its identity and to be capable to use public key cryptography for content encryption. The certificate is stored in the Java `KeyStore` to protect it from unauthorized access or modification. Smart Meter configuration can be done via a config file. This file contains parameters indicating the simulation frequency, the path to the file containing the meter readings, the smart meter id and the Grid Operator's network address. The Java `properties` class can be used to fulfill this task.

¹see <http://www.raspberrypi.org/> and <http://cubieboard.org/>

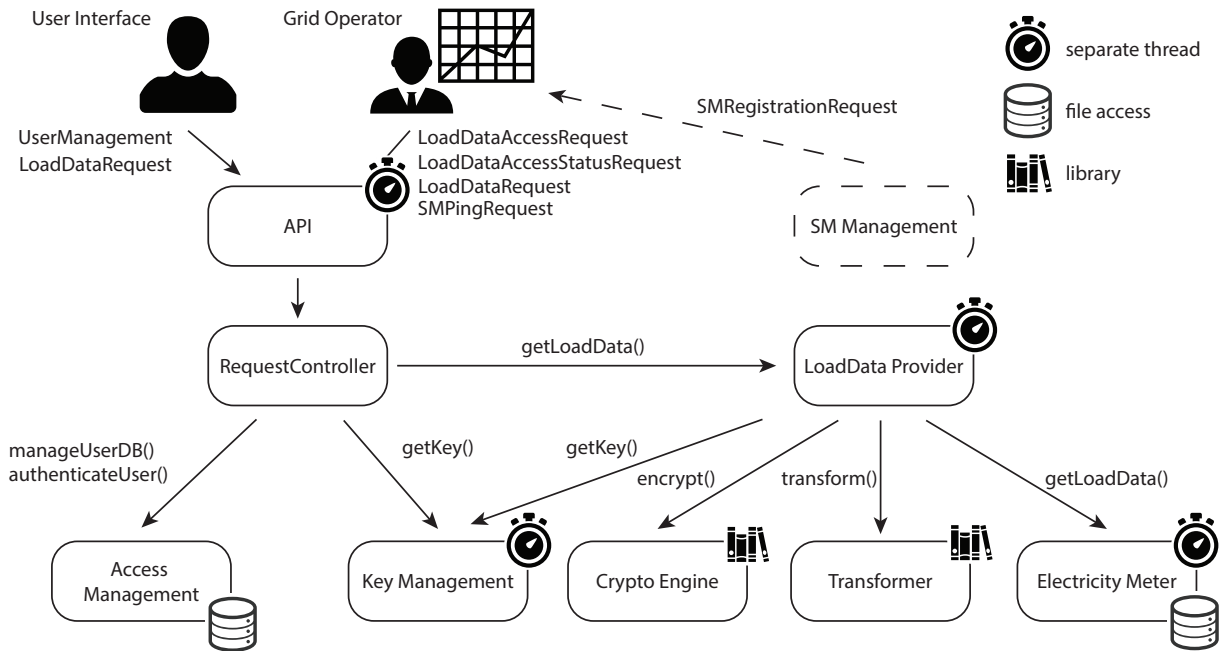


Figure 4.1: The Software Architecture for the Smart Grid consists of multiple components. The software provides an interface to the consumer/end user user interface as well as to the Grid Operator.

Detailed description of the functionality and architecture of the software can be found in Section 4.1.1. The used communication sequences are illustrated in Figures 3.3, 3.4 and 3.7 as well as described in Sections 3.2 and 3.4

So far, the described Proof of Concept includes only capabilities for privacy preserving load data representation using multi-resolution load data. Nevertheless, the component based software architecture allows to extend the software by capabilities for billing or load data aggregation.

4.1.1 Software Architecture

The Smart Meter consists of several core components, which can be seen in Figure 4.1. Several components run as own threads. Hence, all components have to be implemented in a thread-safe manner. In order to interact with the consumer and the Grid Operator, the software provides an API. The API is accessible via an encrypted communication channel. Every API request includes an identification tag based on the requester's certificate. The identification tag is used to identify the requester and determine the requester's permissions.

The API: The API provides a RESTful interface and can be accessed via HTTPS using SSL/TLS as encryption protocol. For authentication purposes, the certificate of the opposite party is required during connection establishment. In order to ensure message

integrity, all API calls and responses need to be signed. In order to provide the API, an application server is needed. `Jetty`² is a Java based web server and Java servlet container. Its lightweight design makes it perfect for machine to machine communications. `Jetty` can be found in several software frameworks and products including Apache ActiveMQ or the Google App Engine. For an enhanced API design and development, `Jersey` is used. `Jersey`³ is a RESTful Web Service framework used to generate Java API for RESTful Services (JAX-RS) compatible Interfaces. It offers an easy and comfortable way of implementing Java Servlets using annotations. Each API function is mapped to one servlet. The servlet container takes an incoming API call and forwards it to the according servlet. Therefore, each request is handled by a separate thread. Each servlet forwards the API call to the `RequestController`. JavaScript Object Notation (JSON) is used to transmit data objects. Compared to the Extensible Markup Language (XML), JSON offers reduced communication overhead and allows a human readable data representation.

The API provides the following functions:

- `LoadDataAccessRequest`: This function is used by any entity to request access to load data in a certain resolution. The request includes the requester's identification tag, the identification tag of the sender (as the requester could be a third party entity where as the sender is the Grid Operator) and the requested resolution.
- `LoadDataAccessStatusRequest`: This function is used by any entity to check the status of an access request. Requests can be accepted, declined or pending. In case of pending, the consumer/end user has not made a decision on the access request. The request includes the requester's identification tag, the identification tag of the sender and the requested resolution.
- `LoadDataRequest`: This function is used to request the multi-resolution load data. The request includes only the identification tag of the sender.
- `SMStatusRequest`: Using this request, the Smart Meter's status information can be accessed. This information can contain firmware version, connection status, health status, etc. Special permission is needed to access this information.
- `UserManagementRequest`: The customer can decide which entity gains access to which resolution. The `UserManagementRequest` is used for access management to the multi-resolution load data.
- `SMPingRequest`: This request is used by the Grid Operator to check, if the Smart Meter is still reachable under the registered address. Beside the Grid Operator's identification tag, this request includes a random number. The response to this request includes the Smart Meter's identification tag and the random number again.

²see <http://www.eclipse.org/jetty>

³see <https://jersey.java.net>

Request Controller: The request controller provides an interface to the API. The servlets use this interface to query the required data. Independent of the request type, the procedure to process the request always stays the same. First, the request controller checks whether the requesting entity has the permission to query or alter the requested data. The request controller uses the interface of the Access Management Component to authenticate the requesting entity and to check its roles and permissions. The Request Controller queries the requested data Only if access is granted and hands the result back to the calling servlet. If no access is granted, the Request Controller throws a `NoPermissionException`. The request controller is the software main entrance point and implemented according to the Singleton design pattern. It holds references to all components within the software.

Access Management: This component is responsible for managing all the users connecting to the Smart Meter. A user can be the consumer, the grid operator, the utility or any other third party entity. Depending on the user's role, different permissions are granted. Grid operator and consumer obtain special permissions for controlling and managing the Smart Meter and the permissions of the other users. Access Management provides an interface which allows user authentication and managing the users and their permissions and roles. This includes adding new users, viewing and altering permissions and roles as well as deleting users. New users placing an access request are also managed by Access Management. Their data is temporarily stored until the consumer accepts or declines the request. Access Management requires a persistent storage to recover the known users and their assigned permissions and roles after a restart of the Smart Meter. Data can be stored in a single file or a lightweight database. Care should be taken to protect the permission data from unauthorized access.

Key Management: The encryption of multi-resolution load data requires several keys in a hierarchical relation to each other. Further details on hierarchical key generation are discussed in Section 2.3.7 and Section 3.4. The keys are generated and managed by the Key Management component. For each resolution to encrypt, a different key is required. For example, if the load data is provided in four different resolutions, one master key and three hierarchically derived keys need to be generated. Key Management generates new keys periodically, for example once a day. All entities with granted access to a certain resolution share the same key. In order to have permission revocation take place, key renewal is required. This implies, on a daily key renewal cycle, an entity can encrypt the resolution up to 24 hours before key revocation takes place. The periodical task can be done using a `Java TimerTask`. The `Java Timer` class allows to schedule the execution of `TimerTasks`. The Key Management component provides an interface which allows to query the current keys. Locking mechanisms are required to ensure that no key can be queried during key generation.

Key generation can be done using the Java crypto library. Listing 4.1 shows how to generate a 128 bit key suitable for AES encryption. By default, Java supports a maximum key size of 128 bit, due to US export laws. In order to increase the key size, the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files have to be installed on every machine running the software. An alternative approach is to use a third party crypto library like Bouncy Castle⁴. For a proof of concept, a 128 bit key is assumed to be safe.

```
1 import javax.crypto.KeyGenerator;
2
3 KeyGenerator keyGenAES = javax.crypto.KeyGenerator.
    getInstance("AES");
4 keyGenAES.init(128);
5 SecretKey masterKey = keyGenAES.generateKey();
```

Listing 4.1: Key generation in Java

To generate hierarchical keys, multiple keys are derived from each other. After generating the master key following Listing 4.1, keys can be derived following Listing 4.2. It uses the SHA-256 algorithm to create a digest with a size of 256 bits. So far, there are no known succeeding attacks on this algorithm⁵. As the `MessageDigest` class only takes bytes as input, the key must be converted from a `SecretKey` object to a byte array prior to derivation. The SHA-256 algorithm generates a 32 byte hash from the initial key. The `SecretKeySpec` class can be used to rebuild a valid key from the generated hash. The required key needs to be of 16 bytes (128 bits) length, hence only the first 16 bytes of the hash are needed.

```
1 import java.security.MessageDigest;
2
3 MessageDigest md = MessageDigest.getInstance("SHA-256");
4     //create Instance of message digest using SHA-256 als
5     algorithm
6
7 byte[] keyRaw = key.getEncoded(); //Transform key to byte
8     array
9 md.update(keyRaw); //Apply one-way hash function
10
11 byte[] newKeyRaw = md.digest(); //collect result of hash
12     function
```

⁴see <https://www.bouncycastle.org>

⁵Summer 2014


```
10 SecretKey newKey = new SecretKeySpec(newKeyRaw, 0, 16, "AES");  
    //create a valid key out of the first 16 bytes of the  
    hash.  
11  
12 md.reset(); //reset the one-way hash function
```

Listing 4.2: Key derivation in Java

LoadData Provider: As the name suggests, this component provides the encrypted multi-resolution load data. It is responsible for retrieving the load data from the electricity meter, splitting the load data into multiple resolutions and finally encrypting the resolutions respectively. The whole process is repeated in a certain time interval, for example every fifteen minutes or on a hourly basis. For this purpose, the LoadData Provider is derived from the Java `TimerTask` class. As a Java `TimerTask` runs in a separate thread, special emphases have to be put on concurrency and thread safety. The load data update process contains several steps: The LoadData Provider collects the load data of a certain time span, e.g., the last 24 hours, from the electricity meter. Each record contains the load value and the corresponding timestamp. In order to split the load data in multiple resolutions, the LoadData Provider passes the load data to the Transformation Library. Then, the LoadData Provider queries the current hierarchical keys from the Key Management component and encrypts the transformation coefficients with the corresponding keys. The highest resolution is encrypted using the master key. Each lower resolution is encrypted using a key derived from the one used for the next higher resolution. See Figure 3.5 for a proper illustration of the correct key assignment. During the whole process, the timestamps remain unchanged. The encrypted coefficients and the timestamps are packed together in a data object and temporarily stored for further delivery. The data object can be accessed via the interface, which is provided by the LoadData Provider.

Crypto Engine: This library provides basic cryptographic functionality for symmetric and asymmetric encryption as well as digital signature verification. For symmetric encryption, the library uses the Advanced Encryption Standard (AES) and supports keys of 128, 192 and 256 bits length. Listing 4.3 shows how the Java Cryptographic Extension (JCE) framework can be used to encrypt a byte array using AES. The `Cipher` object is initialized with the parameter `AES/CBC/PKCS5Padding`. The parameter defines to use AES as cryptographic algorithm in Cipher Block Chaining Mode (CBC) using the PKCS5 padding scheme⁶. The cipher requires an initialization vector for initialization. The vector should be randomly generated. The vector used for encryption is also needed for decryption. A `IvParameterSpec()` object passes the initialization vector to the `Cipher`. The `Cipher` object is initialized with the initialization vector and a

⁶RSA Laboratories, "PKCS #5: Password-Based Encryption Standard", version 1.5, November 1993

suitable key. The generation of suitable keys is done by the KeyManagement component. Cipher.ENCRYPT_MODE sets the Cipher object into encryption mode. The message is encrypted by calling Cipher.doFinal(). For readability and better transmission, the resulting cipher message is transformed into a Base64 encoded string.

```
1 import javax.crypto.Cipher;
2 import javax.crypto.SecretKey;
3 import javax.crypto.spec.IvParameterSpec;
4 import org.apache.commons.codec.binary.Base64;
5
6 byte[] message = {1,2,3,4}; //message to encrypt
7 byte[] iv = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}; //
   initialization vector, randomly filled
8 IvParameterSpec ivspec = new IvParameterSpec(iv); //
   generate parameters
9 Cipher encryptor = Cipher.getInstance("AES/CBC/PKCS5Padding
   ");
10 encryptor.init(Cipher.ENCRYPT_MODE, key, ivspec);
11 byte[] cipher = encryptor.doFinal(message); //encrypt
12 String result = new String(Base64.encodeBase64(cipher));
```

Listing 4.3: Symmetric encryption in Java using AES

The decryption of the cipher is analog to the encryption and can be seen in Listing 4.4. The Cipher object is created as described earlier. Cipher.DECRYPT_MODE sets the Cipher object into decryption mode. The same initialization vector as for the encryption has to be used. Before decrypting the message, the string cipher has to be transformed to a byte array again. Decryption is done by calling Cipher.doFinal().

```
1 import javax.crypto.Cipher;
2 import javax.crypto.SecretKey;
3 import javax.crypto.spec.IvParameterSpec;
4 import org.apache.commons.codec.binary.Base64;
5
6 byte[] iv = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
7 IvParameterSpec ivspec = new IvParameterSpec(iv);
8 byte[] raw = Base64.decodeBase64(cipher);
9 Cipher decryptor = Cipher.getInstance("AES/CBC/PKCS5Padding
   ");
10 decryptor.init(Cipher.DECRYPT_MODE, key, ivspec);
11 byte[] result = decryptor.doFinal(raw);
```

Listing 4.4: Symmetric decryption in Java using AES

Asymmetric encryption is achieved using RSA as algorithm. For asymmetric encryption, a private and a public key are required. These keys are included in the certificate and therefore stored in the Java `KeyStore`. Certificate generation is described in Section 4.4. A private/public key pair can also be generated following the steps in Listing 4.5. It generates two keys with a length of 1024 bits. The keys can be accessed using `KeyPair.getPublic()` and `KeyPair.getPrivate()`.

```
1 import java.security.KeyPair;
2 import java.security.KeyPairGenerator;
3 import java.security.PrivateKey;
4 import java.security.PublicKey;
5
6 KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA
7     ");
8 keyGen.initialize(1024);
9 KeyPair key = keyGen.generateKeyPair();
10 PrivateKey private = key.getPublic();
11 PublicKey public = key.getPrivate();
```

Listing 4.5: Generate a public/private key pair suitable for encryption using RSA in Java

Listing 4.6 shows how to encrypt a message using RSA. The `Cipher` object is initialized with the right algorithm to use. `Cipher.ENCRYPT_MODE` sets the `Cipher` object into encryption mode. Depending on the application, either the public or the private key is used as encryption key. Calling `Cryptor.doFinal()` finally encrypts the message. For readability reasons, the resulting cipher is transformed to a string using a Base64 encoder.

```
1 import javax.crypto.Cipher;
2 import javax.crypto.SecretKey;
3 import org.apache.commons.codec.binary.Base64;
4
5 byte[] message = {1,2,3,4}; //message to encrypt
6 Cipher encryptor = Cipher.getInstance("RSA");
7 encryptor.init(Cipher.ENCRYPT_MODE, key);
8 byte[] cipher = encryptor.doFinal(message); //encrypt
9 String result = new String(Base64.encodeBase64(cipher));
```

Listing 4.6: Asymmetric encryption using RSA in Java

In order to decrypt an asymmetric encrypted message, the `Cryptor` object has to be set into decryption mode. For decryption, the counter key to the one used for encryption has to be used. If the private key was used for encryption, the public key has to be used for

decryption and vice-versa. `Cryptor.doFinal()` decrypts the cipher. The single steps can be seen in Listing 4.7.

```
1 import javax.crypto.Cipher;
2 import javax.crypto.SecretKey;
3 import org.apache.commons.codec.binary.Base64;
4
5 byte[] raw = Base64.decodeBase64(cipher);
6 Cipher decryptor = Cipher.getInstance("RSA");
7 decryptor.init(Cipher.DECRYPT_MODE, key);
8 byte[] result = decryptor.doFinal(raw);
```

Listing 4.7: Asymmetric decryption in Java using RSA

Every message sent within the Smart Grid Infrastructure is required to be signed to ensure message integrity. The Crypto Engine library provides the functionality to sign these messages and to verify the signature of the messages. The signature for an object is created by calculating the hash of the object and encrypting the hash with the entity's private key. Listing 4.8 shows the required steps. The hash is generated using a `MessageDigest` object. The SHA-256 algorithm is used to generate a 32 byte hash from the object.

```
1 import javax.crypto.Cipher;
2 import java.security.MessageDigest;
3
4 MessageDigest md = MessageDigest.getInstance("SHA-256");
5 Cipher encryption = Cipher.getInstance("RSA");
6 encryption.init(Cipher.ENCRYPT_MODE, privateKey);
7
8 String message = "This_is_a_test_message";
9 byte[] byteMessage = message.getBytes("UTF8");
10 md.update(byteMessage);
11 byte[] hash = md.digest();
12 byte[] signature = encryption.doFinal(hash);
```

Listing 4.8: Creating a signature for an object in Java using SHA-256 and RSA

The signature can be used to verify if the object was altered after the signing process. In order to verify the signature, the object as well as the signature and the signer's public key are required. The required process is shown in Listing 4.9. First, the hash of the object is generated. Then, the signature is decrypted using the signer's public key. If the decrypted signature equals the calculated hash, the object has not been altered.

```
1 import javax.crypto.Cipher;
2 import java.security.MessageDigest;
```

```

3 import java.util.Arrays;
4
5 MessageDigest md = MessageDigest.getInstance("SHA-256");
6 Cipher decryptor = Cipher.getInstance("RSA");
7 decryptor.init(Cipher.DECRYPT_MODE, publicKey);
8
9 String message = "This_is_a_test_message";
10 byte[] byteMessage1 = message1.getBytes("UTF8");
11 md.update(byteMessage1);
12 byte[] newHash = md.digest();
13 byte[] hashToCompare = decryptor.doFinal(signature);
14
15 if(Arrays.equals(newHash, hashToCompare))
16 {
17     System.out.println("hash_equal");
18 }

```

Listing 4.9: Verifying the signature of an object in Java using SHA-256 and RSA. Signature is the calculated object signature from Listing 4.8

Transformer: The Discrete Wavelet Transform (DWT) is performed by this library. The interface provides methods to perform transform and inverse transform using the Haar wavelet. The mathematical background and the formula are given in Section 2.4.1. The used algorithm is straight forward and implemented recursive. Following Listing 4.10, two methods are needed to perform the wavelet transform. Method `transform()` takes the initial signal, checks whether it has an appropriate length and sets the required variables. Then, the method starts the recursive transformation algorithm. When the algorithm is done, the results are assembled and returned. The `WaveletCoefficients` class acts as data object to store the resulting wavelet coefficients. The coefficients consist of the higher frequency bands (called details) and the final lower frequency band (approximation). See Figure 3.5 for an explaining illustration. The method `recursiveTransform()` performs the recursive algorithm. It requires the algorithm's current approximation and the list holding all details calculated so far as parameters. It returns the final approximation value. In order to perform the wavelet transform, the algorithm calculates the mean of two following values of the approximation and stores them for the next step. Then, it calculates the delta between the mean value and the first approximation value and adds the delta to the list with details. This procedure is repeated for all values of the approximation list. The method then calls itself again, passing on the new approximation values.

```

1 public static WaveletCoefficients transform(List<Float>
    signal) throws WrongFormatException

```

```
2 {
3     if(!IsPowerOfTwo(signal.size())) //check whether
4         signal has 2^n elements
5     {
6         throw new WrongFormatException("Wrong_signal_length._
7             Lenght_must_be_2^n");
8     }
9     List<List<Float>> details = new ArrayList<List<Float
10         >>();
11     WaveletCoefficients result = new WaveletCoefficients();
12
13     float res = recursiveTransform(signal,details); //start
14         recursive transformation
15
16     result.setApproximation(res);
17     result.setDetails(details);
18     return result;
19 }
20
21 private static float recursiveTransform(List<Float>
22     approximation, List<List<Float>> details)
23 {
24     if(approximation.size()==1) //termination
25     {
26         return approximation.get(0);
27     }
28     List<Float> newApproximation = new ArrayList<Float>();
29     List<Float> newDetails = new ArrayList<Float>();
30     for (int i = 0; i < approximation.size(); i+=2) {
31         float mean = (approximation.get(i) + approximation.
32             get(i+1))/2;
33         newApproximation.add(mean);
34         newDetails.add(mean-approximation.get(i));
35     }
36     details.add(0,newDetails);
37     return recursiveTransform(newApproximation,details);
38 }
```

Listing 4.10: Discrete

Wavelet Transform using the Haar Wavelet, recursive implementation in Java

The inverse transform is performed similar to the transform. Two methods are required, `inverseTransform()` and `recursiveInverseTransform`. The methods are shown in Listing 4.11. Method `inverseTransform()` requires a `WaveletCoefficients` object as argument. This object is the result of method `transform()` described above. Method `inverseTransform()` returns a list containing the resulting signal. Before calling the recursive algorithm, it sets the according data fields. Method `recursiveInverseTransform()` requires several parameters: (i) the list with the algorithm's current approximation values, (ii) the list containing all details, (iii) the target level which should be reached by the algorithm and (iv) the algorithm's current level. Level defines the amount of algorithm calls or the resolution of the signal. Level zero indicates the lowest resolution and the first call of the recursive algorithm. The recursive algorithm terminates as soon as the target level is reached. The algorithm takes an approximation value and adds or subtracts the according details value. The resulting two values form the approximation values for the next algorithm call. If the desired level is reached, the values form the signal. This procedure is repeated until all approximation values are processed. The method calls itself again, passing the new approximation values and an incremented level value.

```
1 public static List<Float> inverseTransform(  
    WaveletCoefficients coefficients)  
2 {  
3     List<Float> approximation = new ArrayList<Float>();  
4     approximation.add(coefficients.getApproximation());  
5     List<List<Float>> details = coefficients.getDetails();  
6     List<Float> result = recursiveInverseTransform(  
        approximation,details,details.size(),0);  
7     return result;  
8 }  
9 private static List<Float> recursiveInverseTransform(List<  
    Float> approximation, List<List<Float>> details, int  
    targetLevel, int currentLevel)  
10 {  
11     if(currentLevel>=targetLevel) //termination  
12     {  
13         return approximation;  
14     }  
15     List<Float> currentDetails = details.get(currentLevel);  
16     List<Float> newApproximation = new ArrayList<Float>();  
17     for (int i = 0; i < approximation.size(); i++) {  
18         newApproximation.add(approximation.get(i) -  
            currentDetails.get(i));
```

```
19         newApproximation.add(approximation.get(i) +
20             currentDetails.get(i));
21     }
22     return recursiveInverseTransform(newApproximation,
    details, targetLevel, currentLevel+1);
23 }
```

Listing 4.11: Discrete Wavelet Inverse Transform using the Haar Wavelet, recursive implementation in Java

For the Smart Meter, only the transformation functionality is required. Nevertheless, functionality for inverse transformation is included in the library, too. On the one hand, inverse transformation is used for testing purposes, on the other hand, the library can be reused for other software like ThirdParty (see Section 4.3).

Electricity Meter: The Electricity Meter component features the interface to the electricity meter. It provides the Smart Meter with accurate real time consumption data. The component buffers the readings of the last 24 hours, needed by the LoadData Provider to transform and encrypt the load data. The buffer itself is a data structure similar to a queue or a ring buffer. The buffer's size is defined by the amount of records it has to store for the defined time span. For example, if real-time load data is provided with a granularity of one second, to buffer all records of the last 24 hours, a buffer capable to store 86.400 records is needed. The buffer's size is fixed. In order to prevent a buffer overflow, the oldest record is removed as soon as the buffer is full and a new record is added. For data access, the buffer provides two methods. Either the whole buffer or several recent records can be read at once. Reading of records does not remove records from the buffer. Hence, the buffer always provides the electricity readings from the last 24 hours. The LoadData Provider always reads the whole buffer at once, whereas the User Interface for example can read only the reading of the last 10 minutes to update a consumption chart. The buffer must be thread safe as read and write access can occur at the same time. The Electricity Meter component has to collect meter readings from the electricity meter periodically. Hence, it is implemented as a Java `TimerTask`. As this proof of concept acts as a simulation, there is no physical electricity meter connected. The electricity meter is simulated using prerecorded meter readings as the data basis. The data is read from a file on software startup. Only one meter reading is available per time unit and updated at a certain frequency. The length of the time unit and the frequency are defined by the provided data granularity. For example, if the data provides one meter reading per second, a time unit is well as the frequency is one second or one hertz. In order to speed up simulation, the frequency can be altered. For later usage and demonstration purpose, the prerecorded data can be exchanged with a real electricity meter. For this scenario, the component has to be adopted to access the readings of the real electricity meter.

SM Management: The Smart Meter (SM) Management is an administrative component needed for registering the Smart Meter at the Grid Operator. In order to establish a connection to a Smart Meter, the Grid Operator needs to know, if the Smart Meter is online and under which address it can be reached. On software startup, the SM Management component establishes a connection to the Grid Operator's API and registers the Smart Meter using the Smart Meter's ID and network address. The ID is defined in the config file and must match with the name stated on the Smart Meter's certificate. The network address is a suitable value to contact the Smart Meter within the used communication network. For an Internet Protocol (IP) based network, the address would be the IP address of the Smart Meter's network port. The Grid Operator's address is stored in the config file, too. If the Smart Meter's address is assigned dynamically and the address is prone to be changed while the software is running, the SM Management component must provide functionality to detect an address change and to update the address at the Grid Operator. On software shutdown, the SM Management component connects to the Grid Operator's API once again to deregister the Smart Meter and to inform the Grid Operator, that the Smart Meter is no longer available.

4.1.2 User Interface

A user interface is provided to the consumer. The user interface allows the consumer to read the smart meter's health status, change the permissions for accessing the multi-resolution load data and read the current load data. The user interface is an important management tool to help the consumer monitor his or her current consumption and also to manage the access permissions to the load data. As the Smart Meter is providing all the information via the API, the User Interface can be available in different forms. For example, the user interface can be accessible via a smart phone app, a desktop application, a web interface or an in-house display. The grid operator can also forward the requests for the user interface making the user interface accessible via the Internet.

4.2 The Grid Operator

The Grid Operator manages the Smart Grid Infrastructure and provides an interface to third party entities to enable access to the Smart Grid Infrastructure. The Grid Operator can be seen as a proxy or gateway. A third party entity can access the API of a Smart Meter following the sequence diagram shown in Figure 3.4. In order to control communication within the Smart Grid Infrastructure and to prevent attacks on the Smart Grid Infrastructure, the Grid Operator monitors and restricts third party access. Only known third party entities are allowed to access the Smart Grid Infrastructure. Access

is further limited as only functions provided by the Grid Operator's public API can be used to interact with the Smart Meters. The Grid Operator software implements its own access management module to maintain the user database and corresponding permissions. Using a management user interface, permission settings can be added or altered. In order to establish a connection to a Smart Meter, the Grid Operator needs to know the Smart Meter's ID as well as its address. Smart Meter Management provides an API to the Smart Meters which allows the Smart Meters to register themselves as soon as they are online. In order to allow the consumer/end user to access the Smart Meter User Interface via the Internet, the Grid Operator can provide functionality to forward the corresponding API calls to the Smart Meter. For grid balancing and load forecasting, the Grid Operator has access to load data on a basic resolution. The Grid Operator needs to access Load Data for grid balancing and load forecasting. For the proof of concept, this functionality is not included. The basic sequence on accessing load data can be evaluated via the Third Party Entity software. Hence, there is no need to include this functionality in the Grid Operator software. This software is implemented in Java SE Development Kit 8. In contrast to the Smart Meter, the Grid Operator software can run on any kind of x64 platform. There are no hardware limitations. The Grid Operator holds a certificate allowing identification and the usage of public key cryptography. The certificate is stored in the `Java KeyStore` to protect it from unauthorized access or modification.

4.2.1 Software Architecture

The software architecture for the Grid Operator is similar to the one for the Smart Meter. The software consists of multiple components, shown in Figure 4.2. The API receives requests from other software via an encrypted communication channel. Each request includes an identification tag required to authenticate the requester. The API forwards the requests to the Request Controller which processes the request. Every API call is processed in a different code. Hence, all components need to be implemented in a thread safe manner. The API provides functionality for different user groups and might be accessible via multiple network interfaces. Furthermore, as the Grid Operator works as a proxy/gateway, it needs the ability to perform remote API calls at the Smart Meters. The following paragraphs give a further description of the single software components.

The API: The API is the central part of the software. It provides a RESTful interface which can be accessed via HTTPS using SSL/TLS. API requests are handled by servlets. For each API function, a separate servlet exists. Each API call is forwarded to a new instance of the corresponding servlet. As every API call runs in its own thread, a thread safe design is necessary. Jetty and Jersey are used as web server and servlet

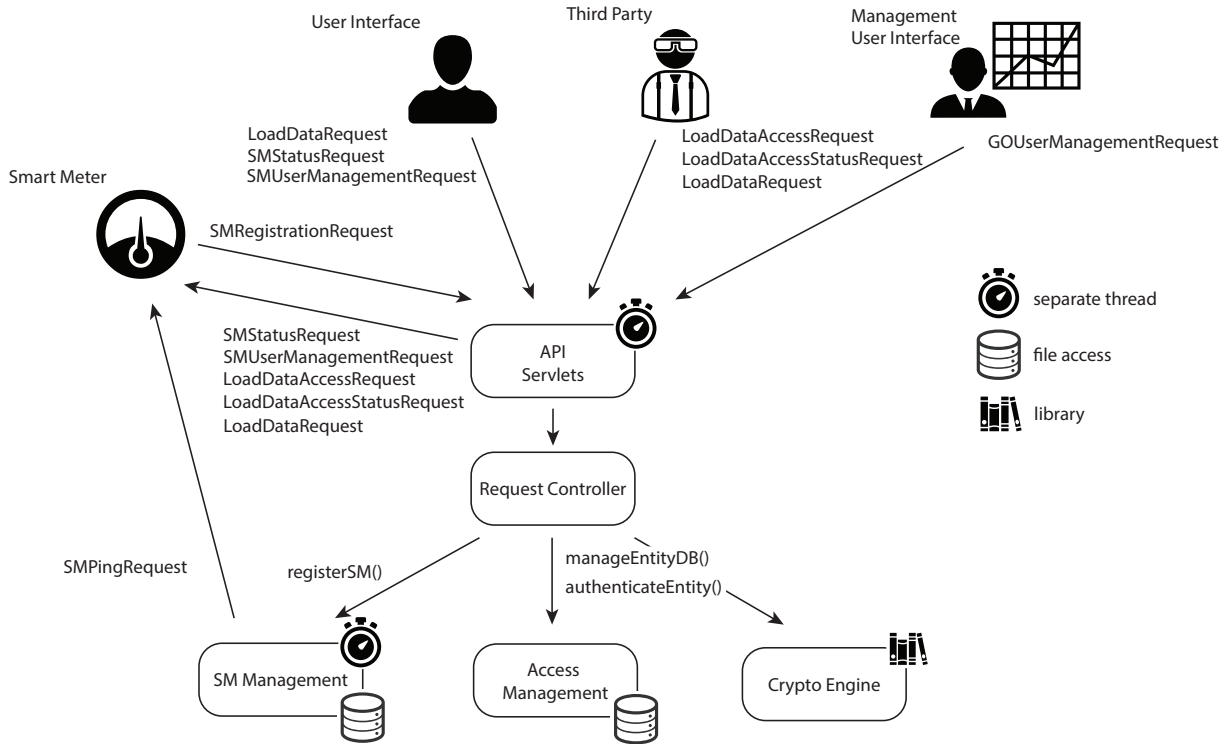


Figure 4.2: The software architecture for the Grid Operator provides an interface to interact with the Smart Meter, the consumer/end user user interface, the third party entity as well as the management user interface.

container⁷. JSON is used to exchange data objects. For authentication, the certificate of the opposite party is required during connection establishment. In addition, to ensure message integrity, all API calls and responses need to be signed with the senders certificate. The servlets are responsible for handling the incoming API request. Independent of the request, the servlet first checks, if the requesting entity owns permission to place the request. Then it queries the requested data from the Request Controller or it forwards the request to the corresponding Smart Meter. In order to forward API requests, the servlets are capable to do remote API calls. A forwarded request is also-called a tunneled request. When sending a tunneled request to the Smart Meter, the Grid Operator's ID as well as the ID of the requesting entity are included. Some tunneled requests carry privacy sensitive information. Hence, the payload of these requests is separately encrypted between the two logical communication nodes (e.g., third party entity and Smart Meter). This process is also referred to as content encryption. An example is shown in Figure 3.7.

The API provides the following functions:

- **LoadDataAccessRequest:** This function is used by a third party entity to request access to load data in a certain resolution from a certain Smart Meter. The request includes the requester's identification tag, the target Smart Meter's ID and the

⁷see Smart Meter Software Architecture, Section 4.1.1 for further details

requested resolution. This is a tunneled request and therefore forwarded to the corresponding Smart Meter. Content encryption is needed to secure the payload.

- **LoadDataAccessStatusRequest:** This function is used by a third party entity to check the status of an access request for a certain Smart Meter. The request includes the requester's identification tag, the target Smart Meter's ID and the requested resolution. This is a tunneled request and therefore forwarded to the corresponding Smart Meter. Content encryption is needed to secure the payload.
- **LoadDataRequest:** This function is used to request the multi-resolution load data from a certain Smart Meter. The request includes only the identification tag of the requester and the target Smart Meter's ID. This is a tunneled request.
- **SMRegistrationRequest:** This request enables a Smart Meter to register or deregister itself at the grid operator. The request contains the Smart Meter's ID and its network address.
- **GOUserManagementRequest:** This function is needed by the Management User Interface to administer the Grid Operator's user database. Special permissions are required to call this function.
- **SMStatusRequest:** Using this request, a Smart Meter's status information can be accessed. This request is needed by the Smart Meter User Interface and only accessible by a consumer/end user. This is a tunneled request with content encryption.
- **SMUserManagementRequest:** This request is needed by the Smart Meter User Interface. It allows the consumer/end user to administer the user management and user permissions. This is a tunneled request with content encryption.

The requests used by the Smart Meter User Interface are optional and not implemented in the proof of concept.

Request Controller: The Request Controller component provides an interface for the API servlets to query the needed data from other components. The component is implemented using the Singleton pattern to only generate a single valid instance of the component. The component itself holds references to all other components. Its interface provides methods to query the Access Management component and the Smart Meter Management component.

Access Management: The Access Management component is similar to the one used for the Smart Meter software. It lists all third party entities which have access to the Smart Grid Infrastructure. There are no special permissions or user groups required. Access permission is only granted or denied. As reference, the third party entity ID is stored. The ID must match the entity's certificate. Access Management is administered via the Management User Interface using the corresponding API functions. The Access Management component provides an interface to the Request Controller which allows

user authentication and user management. User management includes querying, adding and revoking users and permissions. Access Management is also responsible to store the permissions for users accessing the Management User Interface. As this feature is not relevant for a proof of concept, it is not implemented so far. In order to prevent data loss on software termination, persistent storage is required. Users and permissions can be stored in a file or a small database.

SM Management: In order to forward requests to a specific Smart Meter, the Grid Operator needs to know whether the Smart Meter is online or not and under which address it is reachable. On software startup, the Smart Meter sends a `SMRegistrationRequest` to the Grid Operator's API. The request contains the Smart Meter's ID as well as its network address. The Smart Meter Management component stores the Smart Meter's ID, the network address as well as the timestamp of the registration. Using the SM Management interface, the network address can be queried by the Request Controller and the API servlets. The interface also provides functionality to deregister a Smart Meter when it is not available anymore.

In order to clean up the database and to prevent outdated data records, the SM Management component has to validate the records periodically. Invalid or outdated data records can occur if a Smart Meter goes offline without deregistration, for example in case of a software crash or connection problem. Periodically, a so-called ping is sent to all registered Smart Meters. The ping consists of a random number. The Smart Meter replies with the exact same number. As all messages within the Smart Grid Infrastructure are signed by their sender, the SM Management component can use the reply to verify if the Smart Meter is still active under the registered address and if the responding Smart Meter is the same as the registered one. As the validation process is performed periodically, a `Java TimerTask` can be used to fulfill this task.

When a Smart Meter registers at the Grid Operator, the Smart Meter Management component must check the database for duplicate entries. Both, Smart Meter ID and network address must be unique within the database. If any duplicate entries are detected, the SM management component pings the duplicate Smart Meter. If an invalid reply is received, the duplicate record is removed and the new Smart Meter can register. If a valid reply is received, the new Smart Meter is not allowed to register. The incident is reported to the Grid Operator. This feature helps to prevent attacks within the Smart Grid Infrastructure.

The database needs to be restored persistently to prevent data loss due to software termination. The data can be stored in a file or a lightweight database.

Crypto Engine: This library provides basic cryptographic functionality and is equal to the Crypto Engine used within the Smart Meter software. The Grid Operator software

needs this library to verify and sign the messages received from an entity or sent to an entity. See Section 4.1.1 for a further description.

4.2.2 Management User Interface

The Management User Interface is an extra piece of software. It uses the Grid Operator's API to administer the third party entities allowed to access the Smart Grid Infrastructure. The User Interface is only accessible by the Grid Operator.

4.3 The Third Party Entity

The Third Party Entity software is used by a third party entity to access hierarchical encrypted multi-resolution load data from several Smart Meters. In order to be able to access the Smart Grid Infrastructure, the software has to use the Grid Operator's API to be able to communicate with the Smart Meters. The detailed communication sequence is shown in Figure 3.4. In order to call the Grid Operator API, the Third Party Entity needs to know the Grid Operator's network address. The sequence chart shown in Figure 3.7 illustrates the steps needed to access the load data. First, the Third Party Entity software has to place a `LoadDataAccessRequest` including the requested resolution. The consumer/end user has to decide, if he/she accepts the request. A `LoadDataAccessStatusRequest` can be used to check the status of the pending access request. If access is granted by the user, the Third Party Entity software can query the corresponding resolution keys. The Smart Meter encrypts the key using the third party entity's public key and sends it back to the Third Party Entity software. In this case, content encryption is needed to prevent the Grid Operator from reading the keys. The Third Party Entity software can now query the desired load data placing a `LoadDataRequest`. The Key Management component takes care of all received keys and updates expired keys. The Crypto Engine library and the Transformer library are used to decrypt the resolution key as well as the load data and to perform an inverse transform on the load data. The Third Party Entity software has an integrated user interface. The user interface allows the user to place and manage Load Data Requests as well as to view the request status. If access is granted, the user interface shows the load data queried from the Smart Meters. The software is able to access and show load data of multiple Smart Meters. This software is implemented in Java SE Development Kit 8 and designed to run on any platform with graphical user interface. Like any other entity within the Smart Grid, the Third Party Entity software holds a certificate for identification. The certificate is also used for content encryption. It is stored in the Java `KeyStore`. The name on the certificate is used as the Third Party Entity's identification tag.

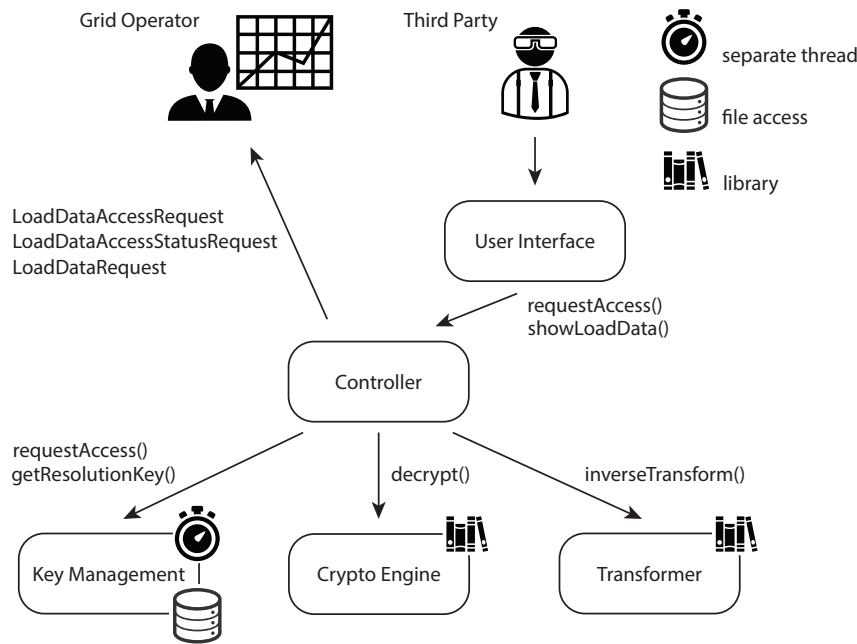


Figure 4.3: The Third Party Entity software architecture provides an integrated user interface to request access to load data and to display retrieved load data.

4.3.1 Software Architecture

The software architecture for the Third Party Entity is based on a classical, straight forward component based design. There is no need to provide an API. The software calls remote API to query the desired resources. The integrated user interface is used to interact with the user. Several libraries are reused from the Smart Meter software. The components and their basic interaction can be seen in Figure 4.3. The Controller is the center part of the software. It holds references to all components and delivers the information to the User Interface. The User Interface itself uses the controller to place any requests triggered by the user. The Key Management component keeps track of all access requests as well as the obtained keys. Below, the single components are described in detail.

Controller: The Controller is the main entrance point of the software. It starts the User Interface and holds references to all components. It provides an interface to the User Interface allowing management of the access requests, request or update resolution keys from a certain Smart Meter and to query load data in a certain resolution. The Controller is the only component within the Third Party Entity software capable to call the Grid Operator's API.

The following API calls are used:

- **LoadDataAccessRequest:** This function is used to request access to load data in a certain resolution from a certain Smart Meter. The request includes the Third Party Entity's identification tag, the target Smart Meter's ID and the requested

resolution. If the request is accepted by the Smart Meter, the response includes the resolution key which is required to encrypt the load data in the requested resolution. As the resolution key is privacy sensitive information, it is encrypted with the Third Party Entity's public key.

- **LoadDataAccessStatusRequest:** An access request is pending until the consumer/end user decides to accept or decline the access request. The `LoadDataAccessStatusRequest` can be used to query the status of an access request. The request includes the Third Party Entity's identification tag as well as the requested resolution and the target Smart Meter's ID. The response shows if the access request is still pending, accepted or declined.
- **LoadDataRequest:** This function is used to request the multi-resolution load data from a certain Smart Meter. The request includes the Third Party Entity's identification tag of the requester and the target Smart Meter's ID. The response shows the encrypted stream of wavelet coefficients needed to rebuild the load data.

In order to rebuild the load data out of the encrypted stream of wavelet coefficients, the controller first queries the required hierarchical keys from the Key Management component. Then it decrypts the wavelet coefficients up to the desired resolution using the Crypto Engine library. Finally, using the Transformer library, the load data is restored from the wavelet coefficients and forwarded to the User Interface.

The controller's interface provides methods to manage the access requests, update resolution keys and to query load data of a certain Smart Meter. The interface is mainly used by the User Interface. The management of access requests is done by the Key Management component.

User Interface: This component provides a graphical user interface to the user. It allows the user to place and alter access requests and view the status of pending access requests. If access is granted, the User Interface shows a chart representing load data in a certain resolution. Via a list menu, the user can select the load data from different Smart Meters. The User Interface uses the controller's interface to query the required data. This includes managing the access requests and querying load data of a certain Smart Meter.

Key Management: The Key Management component holds all information concerning the access requests and received resolution keys. The Smart Meter ID, the requested resolution and the access request status are saved in a persistent manner. This can be done using a file or a lightweight data base. The received resolution key and the corresponding expiration date is only stored temporarily. After software startup, the resolution keys have to be retrieved again. As the resolution keys are only valid for a short period of time, there is no need for persistent storage. The Key Management interface provides functionality to place new access requests and to view, alter or delete existing ones. The

interface also allows access to the hierarchical resolution keys. The Key Management component derives the required hierarchical keys from the retrieved resolution key. If a requested resolution key is not available but access to the resolution key is granted, the Key Management component requests the resolution key using the controller's interface. A periodical task validates all temporarily stored resolution keys and removes expired keys. This task is implemented using a Java `TimerTask`. Due to the periodical task, concurrent read/write access on the data can occur. Hence, a thread-safe implementation is necessary.

Crypto Engine: The Third Party Entity software queries privacy sensitive data from the Smart Meter. As the Grid Operator is needed as proxy/gateway to establish a logical connection to the Smart Meter, the Grid Operator can also read the whole communication between the Third Party Entity and the Smart Meter. Content encryption can be used to secure the payload of the message from eavesdropping. The payload of a message is encrypted using the opponent's public key. Hence, only the opponent can decrypt the payload using its private key. The wavelet coefficients are symmetric encrypted using hierarchical keys. The Crypto Engine library provides functionality for asymmetric and symmetric encryption. The library is reused from the Smart Meter software. Further details on the library can be found in Section 4.1.1.

Transformer: The Transformer library offers functionality to perform a wavelet transform as well as an inverse wavelet transform using the Haar wavelet. The inverse wavelet transform restores the load data from the wavelet coefficients and is thus needed by the Controller. As the Transformer library is reused from the Smart Meter software, further explanations can be found in Section 4.1.1.

4.4 Public Key Infrastructure

A Public Key Infrastructure (PKI) is required to generate and distribute digital certificates and to maintain the trust path. Furthermore, the PKI is responsible for certificate revocation. Several open source solutions are available, for example EJBCA or OpenX-PKI⁸. While these solutions are well established, they go beyond the scope of a proof of concept.

For a proof of concept with only little nodes involved, certificates can be generated by hand. This can be done using the Java `KeyStore` and `OpenSSL`. The following steps are required to create a Certificate Authority (CA) and to create and sign certificates:

1. **Create CA:** This initial step creates a Certificate Authority and the corresponding CA root certificate using `OpenSSL`. It can be done issuing the following command:

⁸see <http://www.ejbca.org/> and <http://www.openxpki.org/>

```
openssl req -new -x509 -days 3650 -extensions v3_ca /  
-keyout cakey.pem -out cacert.pem
```

The command generates a root certificate and the corresponding 1024 bit RSA key pair. The certificate is valid for 10 years. The private key is written to the file `cakey.pem` whereas the certificate can be found in the file `cacert.pem`. During certificate generation, OpenSSL will ask to enter a new PEM pass phrase. This pass phrase is required every time the root certificate is used to sign a user certificate. The pass phrase and the private key have to be kept at a secure place.

2. **Create client keys:** Each entity in the Smart Grid needs an individual certificate. The following command can be used to create a client certificate and the corresponding keys:

```
keytool -keystore keystorename -genkey -alias  
clientname
```

`keystorename` defines the filename of the Java KeyStore, `clientname` the name of the entity. Note that every entity needs its own Java KeyStore file. The file must be distributed to the entity in order to be used by the corresponding software. This command creates a new Java KeyStore. Hence, it asks to enter a keystore password. This password is needed every time the new Java KeyStore is accessed.

3. **Create Certificate Signing Request (CSR):** After creating the client certificate, the certificate must be signed by the CA to be valid. A Certificate Signing Request is generated to transmit the new certificate to the CA. The CSR is exported to the file `client.csr`.

```
keytool -keystore keystorename -certreq -alias  
clientname -keyalg rsa -file client.csr
```

4. **Sign certificate:** The CSR must be signed by the CA. This can be done by executing the following command.

```
openssl x509 -req -CA cacert.pem -CAkey cakey.pem -in  
client.csr -out client.cer -days 365
```

The command will ask for the PEM pass phrase created in Step 1. The signed client certificate is valid for 365 days and can be found in the file `client.cer`.

5. **Import CA root certificate:** In order to recognize the trust path and to trust the certificate issuer, the Java KeyStore needs to know the CA root certificate. This certificate can be added using the following command.

```
keytool -import -keystore keystorename -file cacert.  
pem -alias RootCA
```

The command prompts the keystore password created earlier.

- 6. Import signed certificate:** Finally, the signed client certificate can be imported into the Java KeyStore again.

```
keytool -import -keystore keystorename -file client.  
cer -alias clientname
```

Steps 2 to 6 can be repeated for every entity participating in the proof of concept. As all certificates are signed by the same CA and the root certificate is included in the Java KeyStore, the trust path is maintained. The entities can validate the different certificates.

4.5 Development Environment

This section briefly describes the tools used to develop and implement the described software. As mentioned earlier, the software is implemented using Java SE Development Kit 8. Eclipse Kepler was used as Integrated Development Environment (IDE). The software architecture is based on a component based design. For each component, several unit tests are implemented. The unit tests are used to test and check basic functionality as well as the component's interface. JUnit is used as the testing framework. In order to provide status information during software execution, the Simple Logging Facade for Java (SLF4J) is used as the logging framework. For build automation and dependency management, Apache Maven is used. It offers means to define the dependencies for each software component, takes care of the right build order and manages external dependencies as well. Apache Maven also allows to execute the unit tests after the software build. For version and revision control as well as for backup purposes the source files are stored in an online repository using Apache Subversion (SVN). In order to ensure consistent software testing and buildable code, JetBrains TeamCity is used as build server. The build server frequently checks the repository for pending changes. If pending changes are detected, it builds the software and runs all unit tests. Any occurring errors are immediately reported to the developer. The software is designed to run on multiple target platforms. In order to ensure correct functionality, the build server can invoke builds and tests on different platforms. This software is build and tested on x64 as well as ARM.

Software documentation is done using L^AT_EX and Javadoc. Diagrams and figures are drawn using Adobe Illustrator.

Conclusion

Demand Response in combination with the Smart Grid helps to reduce peak demand, save energy and integrate intermittent energy sources into the electricity grid. Although the purpose of Demand Response differs between the European Union and the United States, the potential for Demand Response is high for both. Security as well as privacy protection play an important part in a successful deployment of Demand Response and the Smart Grid.

Secure means of communication are required within the Smart Grid. It is essential to ensure authentication, authorization and integrity to prevent unauthorized parties from eavesdropping or altering communication. Attacks on the Smart Grid can cause serious damage to property and society. It is essential to protect the Smart Grid from intruders in order to increase and guarantee grid stability and reliability.

The Smart Grid collects and transfers consumer related information. Using this information, appliances within the household can be identified and usage patterns can be drawn. This privacy invasion causes customers and governments to reject the deployment of the Smart Grid.

In this Master's Thesis, a secure way of communication, suitable to be used within a real-world Smart Grid Infrastructure, has been introduced. The approach divides the Smart Grid Infrastructure into two separate networks, the public network and the Grid Operator's inner network. Communication between these two networks is only possible through the Grid Operator acting as a proxy. The Grid Operator monitors and controls the traffic. Hence, the Grid Operator acts as a firewall and protects the Smart Grid Infrastructure from possible attacks. Communication as well as privacy are secured by means of connection encryption as well as content encryption. A bridged PKI ensures reliable and efficient key management and distribution.

In order to preserve the consumer's privacy, the combination of measurements from substation level with multi-resolution load data is proposed as an alternative to neighborhood

aggregation. Neighborhood aggregation is a complex process adding additional communication overhead to the network. Measurements from the substation level already provide aggregated load data and do not require an additional aggregation step. In combination with multi-resolution load data, all Smart Grid use cases can be fulfilled. The combination provides aggregated load data as well as load data from individual consumers in multiple resolutions. The proposed protocol allows fine-grained access management based on the actual need of the requester. The consumer can decide which entity gains access to his/her data and at which specific resolution. For multi-resolution representation, the wavelet transform is used as it adds just a small computational overhead and the transformation process is lossless. Each resolution is encrypted using a different resolution key. Key management efforts are reduced by introducing a hierarchical key management scheme using one-way hash functions for key derivation.

A software architecture for a suitable proof of concept is given. The software consists of three parts, each simulating a different entity within the Smart Grid Infrastructure. The software is used to demonstrate and evaluate the proposed scheme.

The proposed scheme offers a secure way of communication within the Smart Grid. Methods are used to preserve the consumer's privacy. A new degree of consumer freedom is added, as the consumer can decide to whom and at what level his/her personal data can be provided.

References

- [1] A. Faruqui and R. Hledik, “An Assessment of PGE’s Demand Response Potential,” 2012, Last Accessed: August 29, 2014. [Online]. Available: http://www.portlandgeneral.com/our_company/energy_strategy/resource_planning/docs/irp_demand_response.pdf
- [2] “A national assessment of demand response potential,” in *Staff Report, Federal Energy Regulatory Commission*, June 2009, Last Accessed: August 29, 2014. [Online]. Available: <http://www.ferc.gov/legal/staff-reports/06-09-demand-response.pdf>
- [3] M. P. Lee, O. Aslam, B. Foster, D. Kathan, J. Kwok, L. Medearis, R. Palmer, P. Sporborg, and M. Tita, “Assessment of demand response and advanced metering,” in *Staff Report, Federal Energy Regulatory Commission*, Federal Energy Regulatory Commission, Oct 2013, Last Accessed: August 29, 2014. [Online]. Available: <http://www.ferc.gov/legal/staff-reports/2013/oct-demand-response.pdf>
- [4] A. Chardon, O. Almén, E. Lewis, Philip, J. Stromback, and B. Château, “Demand Response: a decisive breakthrough for Europe,” *Capgemini Energy, Utilities and Chemicals*, 2008, Last Accessed: August 29, 2014. [Online]. Available: http://www.vaasaett.com/wp-content/uploads/2010/01/0805_Demand-Response_PoV_Final.pdf
- [5] H. C. Gils, “Assessment of the theoretical demand response potential in europe,” *Energy*, vol. 67, no. 0, pp. 1 – 18, 2014, Last Accessed: August 29, 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0360544214001534>
- [6] A. Jensen and A. la Cour-Harbo, *Ripples in Mathematics: The Discrete Wavelet Transform*. Berlin: Springer-Verlag Berlin Heidelberg, 2001.
- [7] S. Wicker and R. Thomas, “A privacy-aware architecture for demand response systems,” in *Proceedings 44th Hawaii International Conference on System Sciences (HICSS’11)*, Jan 2011, pp. 1–9, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1109/HICSS.2011.24>
- [8] M. Lisovich and S. Wicker, “Privacy concerns in upcoming residential and commercial demand-response systems,” in *Clemson University Power Systems*

- Conference 2008*. Clemson University, March 2008, Last Accessed: August 29, 2014. [Online]. Available: <http://www.truststc.org/pubs/332.html>
- [9] G. Hart, “Nonintrusive appliance load monitoring,” *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, Dec 1992, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1109/5.192069>
- [10] G. Eibl and D. Engel, “Influence of data granularity on nonintrusive appliance load monitoring,” in *Proceedings of the Second ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '14)*. Salzburg, Austria: ACM, 2014, pp. 147–151, Last Accessed: August 29, 2014. [Online]. Available: <http://doi.acm.org/10.1145/2600918.2600920>
- [11] Z. Erkin, J. Troncoso-Pastoriza, R. Lagendijk, and F. Perez-Gonzalez, “Privacy-preserving data aggregation in smart metering systems: an overview,” *IEEE Signal Processing Magazine*, vol. 30, no. 2, pp. 75–86, 2013, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1109/MSP.2012.2228343>
- [12] D. Engel, “Wavelet-based load profile representation for smart meter privacy,” in *Proceedings IEEE PES Innovative Smart Grid Technologies (ISGT'13)*, Washington, D.C., USA, Feb. 2013, pp. 1–6, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1109/ISGT.2013.6497835>
- [13] C. D. Peer, D. Engel, and S. Wicker, “Hierarchical Key Management for Multi-resolution Load Data Representation,” in *Proceedings of 5th IEEE International Conference on Smart Grid Communications (SmartGridComm 2014)*. Venice, Italy: IEEE, Nov. 2014, to appear.
- [14] V. Balijepalli, V. Pradhan, S. Khaparde, and R. M. Shereef, “Review of demand response under smart grid paradigm,” in *Innovative Smart Grid Technologies - India (ISGT India), 2011 IEEE PES*, Dec 2011, pp. 236–243, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1109/ISET-India.2011.6145388>
- [15] S. Feuerriegel and D. Neumann, “Measuring the financial impact of demand response for electricity retailers,” *Energy Policy*, vol. 65, no. 0, pp. 359 – 368, 2014, Last Accessed: August 29, 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S030142151301032X>
- [16] NIST, “Smart grid cybersecurity strategy, architecture, and high-level requirements,” in *NISTIR 7628 Guidelines for Smart Grid Cybersecurity*. National Institute of Standards and Technology, U.S. Department of Commerce, 2013, vol. 1, Last Accessed: August 29, 2014. [Online]. Available: http://csrc.nist.gov/publications/drafts/nistir-7628-r1/draft_nistir_7628_r1_vol1.pdf

- [17] J. S. John, “Is europe ready for automated demand response?” online, greentechgrid, Tech. Rep., October 2011, Last Accessed: August 29, 2014. [Online]. Available: <http://www.greentechmedia.com/articles/read/is-europe-ready-for-automated-demand-response>
- [18] E. Giglioli, L. Senni, and C. Panzacchi, “How Europe is approaching the smart grid,” *McKinsey and Company*, 2010, Last Accessed: August 29, 2014. [Online]. Available: http://www.mckinsey.com/~media/mckinsey/dotcom/client_service/epng/pdfs/mck%20on%20smart%20grids/mosg_europe_vf.ashx
- [19] V. Giordano, F. Gangale, G. Fulli, M. S. Jiménez, I. Onyeji, A. Colta, I. Papaioannou, A. Mengolini, C. Alecu, T. Ojala *et al.*, “Smart grid projects in europe: lessons learned and current developments,” *JRC Scientific and Policy Reports*, no. JRC79218, 2013, Last Accessed: August 29, 2014. [Online]. Available: http://www.smartgridnews.com/artman/uploads/2/Smart_Grid_projects_in_Europe_Lessons_Learned.pdf
- [20] European Commission, “Country fiches for electricity smart metering,” *Commission staff working document*, no. SWD(2014) 188 final, July 2014, Last Accessed: August 29, 2014. [Online]. Available: <http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52014SC0188&from=EN>
- [21] —, “Benchmarking smart metering deployment in the EU-27 with a focus on electricity,” *Reprot from the Commission*, no. COM(2014) 356 final, July 2014, Last Accessed: August 29, 2014. [Online]. Available: <http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52014DC0356&from=EN>
- [22] C. F. Covrig, M. Ardelean, J. Vasiljevska, A. Mengolini, G. Fulli, and E. Amoiralis, “Smart Grid Projects Outlook 2014,” *JRC Scientific and Policy Reports*, no. JRC90290, 2014, Last Accessed: August 29, 2014. [Online]. Available: http://ses.jrc.ec.europa.eu/sites/ses.jrc.ec.europa.eu/files/u24/2014/report/ld-na-26609-en-n_smart_grid_projects_outlook_2014_-_online.pdf
- [23] W. Boltz and M. Graf, “Smart Meter 2013, Einführung von intelligente Messgeräten in Österreich,” *Energie-Control Austria*, Tech. Rep., 2012, Last Accessed: August 29, 2014. [Online]. Available: http://www.e-control.at/portal/page/portal/medienbibliothek/strom/dokumente/pdfs/Smart%20Meter_Monitoringbericht_FINAL.pdf
- [24] H. Saele and O. Grande, “Demand response from household customers: Experiences from a pilot study in norway,” *IEEE Transactions on Smart Grid*, vol. 2, no. 1, pp. 102–109, March 2011, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1109/TSG.2010.2104165>

- [25] S. D. Warren and L. D. Brandeis, “The right to privacy,” *Harvard Law Review*, vol. 4, no. 5, pp. pp. 193–220, 1890, Last Accessed: August 29, 2014. [Online]. Available: <http://www.jstor.org/stable/1321160>
- [26] S. Wicker and D. Schrader, “Privacy-aware design principles for information networks,” *Proceedings of the IEEE*, vol. 99, no. 2, pp. 330–350, Feb 2011, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1109/JPROC.2010.2073670>
- [27] T. M. Amabile, “Effects of external evaluation on artistic creativity,” *Journal of Personality and Social Psychology* 37, no. 2, pp. 221–233, February 1979.
- [28] W. Diehl, Michael; Stroebe, “Productivity loss in brainstorming groups: Toward the solution of a riddle.” *Journal of Personality and Social Psychology*, vol. 53, no. 3, pp. 497–509, Sep 1987. [Online]. Available: <http://dx.doi.org/10.1037/0022-3514.53.3.497>
- [29] P. B. Camacho, L. Mabel; Paulus, “The role of social anxiousness in group brainstorming.” *Journal of Personality and Social Psychology*, vol. 68, no. 6, pp. 1071–1080, Jun 1995.
- [30] R. B. Matlin, Margaret W.; Zajonc, “Social facilitation of word associations.” *Journal of Personality and Social Psychology*, vol. 10, no. 4, pp. 455–460, Dec 1968.
- [31] B. Schneier, *Applied cryptography: protocols, algorithms, and source code in C*, 2nd ed. New York: John Wiley & Sons, Inc, 1996.
- [32] T. Dierks and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” in *IETF Request for Comments*, no. 5246, August 2008, Last Accessed: August 29, 2014. [Online]. Available: <http://tools.ietf.org/html/rfc5246>
- [33] X. Long, D. Tipper, and Y. Qian, “An advanced key management scheme for secure smart grid communications,” in *Proceedings IEEE International Conference on Smart Grid Communications (SmartGridComm’13)*, Oct 2013, pp. 504–509, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1109/SmartGridComm.2013.6688008>
- [34] S. Smith, “Cryptographic scalability challenges in the smart grid (extended abstract),” in *Proceedings IEEE PES Innovative Smart Grid Technologies (ISGT’12)*, Jan 2012, pp. 1–3, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1109/ISGT.2012.6175564>
- [35] T. Baumeister, “Adapting PKI for the smart grid,” in *Proceedings IEEE International Conference on Smart Grid Communications (SmartGridComm’11)*, Oct 2011, pp. 249–254, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1109/SmartGridComm.2011.6102327>

- [36] J. Buchmann, E. Karatsiolis, and A. Wiesmaier, *Introduction to Public Key Infrastructures*. Springer Berlin Heidelberg, 2013.
- [37] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright, “Transport Layer Security (TLS) Extensions,” in *IETF Request for Comments*, no. 4366, April 2006, Last Accessed: August 29, 2014. [Online]. Available: <http://www.ietf.org/rfc/rfc4366.txt>
- [38] L. Lamport, “Password authentication with insecure communication,” *Commun. ACM*, vol. 24, no. 11, pp. 770–772, Nov. 1981, Last Accessed: August 29, 2014. [Online]. Available: <http://doi.acm.org/10.1145/358790.358797>
- [39] N. Haller, “The S/KEY One-Time Password System,” in *IETF Request for Comments*, no. 1760, 1995, Last Accessed: August 29, 2014. [Online]. Available: <http://www.ietf.org/rfc/rfc1760.txt>
- [40] S. Bakhtiari, R. Safavi-naini, J. Pieprzyk, and C. Computer, “Cryptographic hash functions: A survey,” Department of Computer Science, University of Wollongong, Tech. Rep., 1995, Last Accessed: August 29, 2014. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.8428>
- [41] V. Goyal, “How to re-initialize a hash chain,” Cryptology ePrint Archive, Report 2004/097, 2004, Last Accessed: August 29, 2014. [Online]. Available: <https://eprint.iacr.org/2004/097.pdf>
- [42] S. Imaizumi, M. Fujiyoshi, H. Kiya, N. Aoki, and H. Kobayashi, “A key derivation scheme for hierarchical access control to JPEG2000 coded images,” in *Advances in Image and Video Technology*, ser. Lecture Notes in Computer Science, Y.-S. Ho, Ed. Springer Berlin Heidelberg, 2012, vol. 7088, pp. 180–191, Last Accessed: August 29, 2014. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-25346-1_17
- [43] Y. Wu, D. Ma, and R.-H. Deng, “Progressive protection of JPEG2000 codestreams,” in *International Conference on Image Processing (ICIP '04)*, vol. 5, Oct 2004, pp. 3447–3450 Vol. 5, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1109/ICIP.2004.1421856>
- [44] M. Asghar and M. Ghanbari, “Cryptographic keys management for H.264/scalable coded video security,” in *8th International ISC Conference on Information Security and Cryptology (ISCISC'11)*, Sept 2011, pp. 83–86, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1109/ISCISC.2011.6062331>
- [45] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman, “MIKEY: Multimedia Internet KEYing,” in *IETF Request for Comments*, no. 3830, 2004, Last Accessed: August 29, 2014. [Online]. Available: <http://tools.ietf.org/html/rfc3830>

- [46] C. Efthymiou and G. Kalogridis, “Smart grid privacy via anonymization of smart metering data,” in *First IEEE International Conference on Smart Grid Communications (SmartGridComm’10)*, Oct 2010, pp. 238–243, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1109/SMARTGRID.2010.5622050>
- [47] D. Engel and G. Eibl, “Multi-resolution load curve representation with privacy-preserving aggregation,” in *Proceedings of IEEE Innovative Smart Grid Technologies (ISGT) 2013*. Copenhagen, Denmark: IEEE, Oct. 2013, pp. 1–5, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1109/ISGTEurope.2013.6695231>
- [48] L. Chen, R. Lu, and Z. Cao, “PDAFT: A privacy-preserving data aggregation scheme with fault tolerance for smart grid communications,” *Peer-to-Peer Networking and Applications*, pp. 1–11, 2014, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1007/s12083-014-0255-5>
- [49] F. Li, B. Luo, and P. Liu, “Secure information aggregation for smart grids using homomorphic encryption,” in *First IEEE International Conference on Smart Grid Communications (SmartGridComm), 2010*, Oct 2010, pp. 327–332, Last Accessed: August 29, 2014. [Online]. Available: <http://dx.doi.org/10.1109/SMARTGRID.2010.5622064>