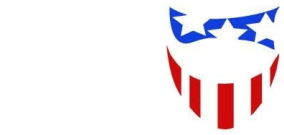
PROJECT REPORT

Privacy-preserving Security Architecture for Smart Grids and Metering

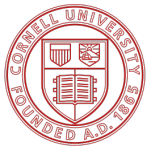
submitted to the
Marshall Plan Scholarship Foundation

submitted by:
Stephan Stadlmair, BSc



Marshallplan-Jubiläumsstiftung
Austrian Marshall Plan Foundation

Fachhochschule Salzburg University of Applied Sciences



Cornell University

Supervisor: FH-Prof. DI Mag. Dr. Dominik Engel
Supervisor: Prof. Dr. Stephen B. Wicker

Ithaca, October 2017

Details

Keywords:	Smart Grids Security Privacy Cryptograhy Architecture
Academic Supervisor:	FH-Prof. DI Mag. Dr. Dominik Engel
Academic Supervisor:	Prof. Dr. Stephen B. Wicker

Abstract

This project report introduces a smart grid architecture, which takes security and privacy aspects into account. First, the exact place of the architecture in the overall smart grid context is defined via analyzing an existing model. Afterwards the boundaries of the architecture are identified, legally and technically. Then fitting technologies are assessed, which can form the basis of the mentioned architecture. These steps pave the path to adapt the technologies for the usage in the smart grid architecture. Technologies presented include Blockchain Technology, Authenticated Encryption with Associated Data, Multi Resolution Representation of Load Data and Aggregation of data based on their privacy relevance. This architecture is then explored and a Proof of Concept implementation is discussed, which introduces useful frameworks and technologies for applying the architectural approaches. Afterwards, gaps that have to be filled for real world applications are presented.

Contents

List of Figures

List of Tables

1

Introduction

Energy grids in prior times just had to satisfy the demand of all the customers in the grid. Therefore, central power plants were the ideal fit and the energy was distributed over the network, coordinated by central entities, overlooking the demand of the participators. The added bidirectional communication of Smart Grids enhances possibilities inside the grid significantly. Through this, integration of so called prosumers [3] is possible. The grid is then able to handle decentralized production, where usually no influence on the grid operators' side is possible. With a Smart Grid in place, much more fine grained control is given to those in charge of the grid stability and communication and coordination can happen from the nuclear power plant to the household, if desired. Another benefit arises through the possibility to quickly react to unforeseen events, like energy shortages or natural influences, e.g. harsh weather conditions, affecting power generation.

Besides all these benefits, certain disadvantages must be addressed. As there are new ways of communication introduced, this communication can be misused internally and externally. Internal misuse would be the usage of consumption data inside eligible organizations, like grid operators, for unauthorized purposes, e.g. targeted advertisement. Those internal misuses are more privacy related concerns. External misuse refers to attacks from outside, e.g. to gain control over parts of the power grid. Both have to be prevented on every level of the Smart Grid and the architecture itself must be resilient against these attacks.

To address the previously mentioned attack vectors, the emerging area of Smart Grids needs to have a proper security and privacy architecture in place. Several stakeholders

with different needs are present, as well as a heterogeneous infrastructure [1]. This infrastructure ranges from computationally weak clients, like Advanced Metering Infrastructure (AMI), to servers in power plants. The Grid Operator needs to have a security and privacy aware architecture for connecting all of the participators, to create broad acceptance of the customers and to install a grid, which is able to withstand even harsh conditions like partial failures of the grid infrastructure. Therefore, a thorough assessment of possible technologies has to happen, especially with certain respect to how they can benefit from each other.

Privacy risks are implied by the usage of smart metering infrastructure. Very fine grained access to the habits and the used appliances of a household can be derived from the load data, as proven in [4]. Most of the technologies, which will be discussed for the application in the proposed Smart Grid architecture, are well known outside of the smart grid area, but have not been assessed in combination and have not been applied to the Smart Grid.

On the security side, the use of Blockchain Technology [5] as a distributed solution for providing identities of participating grid devices like AMIs is proposed. This is done in contrast to usual certificates, which depend highly on Certificate Authorities (CA) [6], like in usual public key infrastructures [7]. Adoptions have to be made for tackling all the requirements in the grid as good as possible. For instance the grid operator is responsible for maintaining grid stability and therefore has a much higher incentive to just have trustworthy devices present in the grid as all the other participators. This can be seen in contrast to Blockchain applications like Bitcoin[1], where every participator is equal. It is assessed how other approaches from the Blockchain Technology context can be used and fit into the concept of a security and privacy aware architecture for the Smart Grid. These include Smart Contracts and Simplified Payment Verification [5]. Together with Authenticated Encryption with Associated Data (AEAD) [8] this approach forms the security basis for the architecture.

In terms of privacy, this report proposes a multi resolution approach introduced by Engel and Eibl in [2], to have several resolutions of load data in place. Those are

[1] http://www.bitcoin.org

encrypted separately and can be decrypted on a need to know basis by other participators, with a household's informed consent in a way like [9] proposes. Furthermore, aggregation is discussed on a neighborhood aggregator level, as introduced in [10]. It is examined if homomorphic encryption is a viable solution [2], or if a usual symmetric approach should be chosen, taking the overall circumstances into account.

The report is structured as follows: First related work will be introduced. The goal is to clarify the boundaries of this thesis. In there, basic technological concepts will be assessed.
Afterwards models and legal requirements for setting up Smart Grid architectures are introduced. This is needed to clarify which properties the architecture has to have and to connect the architecture with existing models displaying the Smart Grid conceptually.
Technical and legal requirements with certain respect to European Union, United States and Austrian legislation are analyzed. Following this and based on the already assessed basic technologies, a look at advanced technology concepts for the actual architecture is done. These concepts are advanced enough to form specific parts of the architecture, although special attention is necessary to connect them and make them benefit from each other. With these technologies set, an architecture is proposed, covering security and privacy, incorporating all affected parties in a modern Smart Grid environment. Afterwards a Proof of Concept (PoC) implementation is done. Finally, it is clarified where room for improvement exists and what should be taken into account, especially if one wants to implement such a system in an environment more sophisticated than a PoC.
A special emphasis will be put on the distinguishing aspects of this architecture, to specify why such a system is needed and where it differs from existing approaches like [10].

2

Related Work

This section covers all basic technologies which will be used in this thesis. It also introduces Smart Grids, why they are useful and where a privacy and security aware architecture would fit.

2.1 Smart Grids and Demand Response Systems

To create a privacy and security aware architecture, it must be clarified in which environment this takes place. The Smart Grid is a development in the area of power grids where bidirectional communication between a utility/grid operator and the consumer is enabled. Today the flow of energy is unidirectional, from big power plants to lots of small consumers. As the amount of local producers is increasing (e.g. solar panels), bidirectional communication is necessary to coordinate energy distribution. This is required to ensure stability of the grid as consumers are not under the direct influence of a utility and too much of decentralized power production could severely influence grid behavior. By the help of these systems, consumer behavior can be influenced via pricing incentives to produce or consume energy at times where the overall grid and other producers can benefit. Therefore there is no direct need of collecting data tightly coupled to the customer [10]. This communication includes a variety of different types of data discussed in Section 3.4.3, which originate from either the utility/grid operator or the household. All of them have different needs in terms of security and privacy. Control signals for devices can and should be treated differently as for instance billing

information of a specific customer.

2.2 Privacy Recommendations for the Smart Grid

With these data flows in place, security and privacy play an important role. Although different, all of these flows need a certain level of protection. Communication from utility/grid operator side should be secured at least to ensure Confidentiality, Integrity and Availability (CIA) [11], together with the extended property of Authenticity from, e.g. control signals for a nuclear power plant or pricing information distribution to the households as those influences consumer behavior and the grid load.

When the consumer comes into play, privacy has to be taken into account. It is crucial to protect consumer data from unwanted targeted advertisement, or even worse robbery, because someone is able to identify when residents are at home, which is possible as shown in [4]. To take privacy into account, the proposed privacy framework of [10] is used. It comprises five parts:

1. **Provide Full Disclosure of Data Collection:** To act accordingly, a proper description of the collected data must be provided. Optimally combined with the granularity when it was collected and how long it will be stored. Doing so should be required with a proper enforcement in place. This should not be revocable and intelligibility must be taken into account as well, to make the customer feel confident about the ownership of the data.

2. **Require Consent to Data Collection:** A user should give consent to the collection, as it creates a certain amount of awareness of what data is collected. To be useful, this should be displayed to the user before the usage of a technology, like the herein proposed Smart Grid architecture, takes place. As changes in the technological basis may occur, a user should be informed properly with an opt-in mechanism in place, to ensure that a consumer understands the changes taking place.

3. **Minimize Collection of Personal Data:** This comes into play when data is concerned, which reveals personal information about the consumer. Collection

of this data must be inevitably connected with the successful operation of the respective technology, which means that if a service is not running correctly, no data collection should take place. Furthermore, the processing of data should be done as close as possible to the place of collection.

4. **Minimize Identification of Data with Individuals:** Data collected from a device should be anonymized, if a direct connection to a consumer is not strictly necessary. Furthermore there should be a different storage for functional and billing information to not be able to get access to both of them, or to make it as difficult as possible for a possible attacker.

5. **Minimize and Secure Data Retention:** There should be a direct link of the collected data to a specific use-case, which is already in place. Storage of data should only be done if necessary and secured. If a security breach occurs and data is stolen, the consumers should be notified as soon as possible, to enable them taking countermeasures. It should also be ensured that data stored cannot be reused in a way not complying to the rules set up formerly.

This serves as the basis of the architecture proposed in this thesis and is kept in mind for design decisions and the selection of technologies used.

2.3 Virtual Power Plants

To be able to suggest the usage of a smart grid architecture, an area has to be found where it is applicable, especially initially, where an application to the whole of the energy grid is not useful. A step by step approach should be favored, to ensure proper implementation, as well as the possibility to have incremental adoption of the system. Therefore risks are minimized and even a smaller scale test environment is possible to convince other parties, like customers and authorities. This would align with the self healing approaches of Smart Grids and adds a layer of resilience. To enable that, the concept of a Virtual Power Plant is presented [12].

A Virtual Power Plant is a combination of several distributed renewable resources,

which act like an independent power plant. It is able to interact with the energy market and has a variety of adjustable parameters to fit well into the grid. A characteristic is the big number of included devices with the need to coordinate their communication. These are coordinated by distribution system operators and are connected to the rest of the grid via an interface [12].

2.4 Cryptographic Primitives

In this section, technologies and concepts will be discussed, which form the foundation of more advanced technologies investigated in Section 2.5, utilized and combined in the proposed architecture in Section 4.2.

2.4.1 Symmetric Cryptography

To transmit information in a secure manner and to ensure the confidentiality, encryption is used. Therefore a message M is encrypted with a key K, using an encryption process E to create a cipher C. The same key K can then be used by the other participating party in this communication, to decrypt the message with the process D. This leads to the following equation for symmetric encryption and decryption (Equation 2.1):

$$M = D(E(M, K), K) \tag{2.1}$$

A common representative of this class of ciphers is the Advanced Encryption Standard (AES) [13]. AES is a blockcipher with 128-bit block length and variable key length, which is usually denoted as AES-{keylength}, e.g. AES-256. The main advantage is that these ciphers are fast compared to e.g. asymmetric ciphers [14] [15].

2.4.2 Asymmetric Cryptography

Instead of sharing the same key for decryption and encryption, in asymmetric cryptography a public and a private key are generated. These are connected via mathematic properties, which are defined in the underlying algorithm. The public key K_{PUB} is

shared with every participator in the network and can be used for encrypting a message M. Afterwards this can be sent to the party, which holds the private key K_{PRIV}. Only this secretly kept private key is able to decrypt the encrypted message (Equation 2.2):

$$M = D(E(M, K_{PUB}), K_{PRIV}) \tag{2.2}$$

Besides the benefits of asymmetric cryptography, these algorithms need longer key lengths [16] to maintain the same level of security and tend to be slower. A common approach is Elliptic Curve Cryptography (ECC), which has the advantage of a smaller key sizes while maintaining the same level of security [14].

2.4.3 Hybrid Cryptography

To enable the symmetric fast approach of encryption, the symmetric key can be exchanged via asymmetric cryptography. Therefore, the exchange can happen on an untrusted network, but further encrypted information exchange can use the benefits of fast symmetric encryption as shown in Figure 2.1 with first a public key transmission to send the symmetric (session) key. Afterwards faster symmetric encryption, especially with fewer key size, is possible. Combining symmetric and asymmetric approaches like

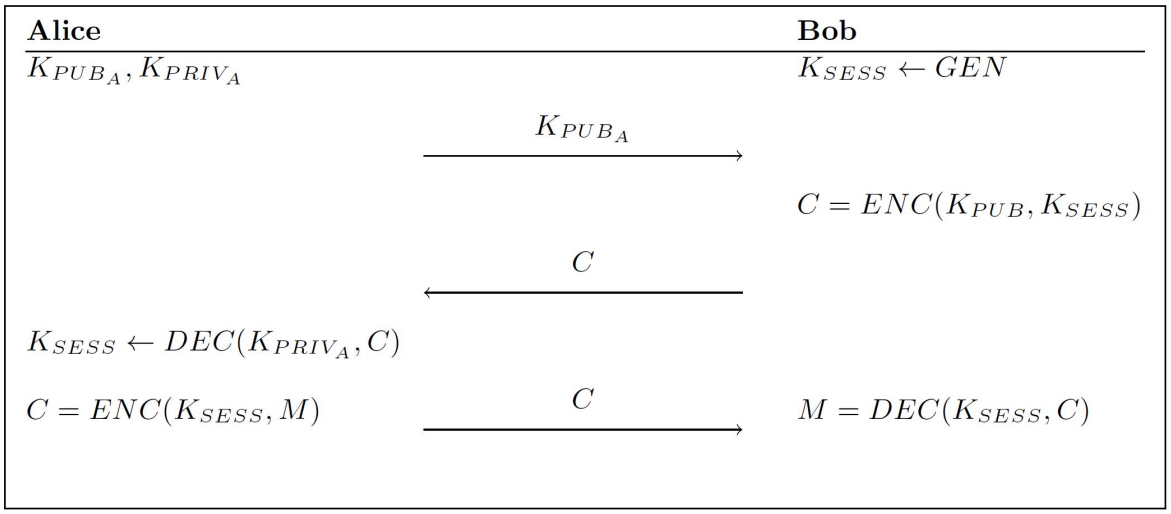


Figure 2.1: Hybrid Cryptography example

that form hybrid crypto-systems [14].

2.4.4 Hash functions

Hash functions are special mathematical functions which map a given input to a certain output with a fixed size. In cryptography this is used to uniquely identify inputs without having the input itself in place. For instance, in modern applications hash values of passwords are stored instead of passwords, so even if the stored values get stolen, no real passwords can be found. The big advantage of these functions is that one cannot get back to the input with just the output given. A very important property of cryptographic hash functions is their resistance against collisions. This means that two different inputs do not lead to the same output and ensure the ability to uniquely identify a certain input in a non predictable way [14]. A well known member of the family of cryptographically secure hashes is the Secure Hash Algorithm-2 (SHA-2) with variable hash length as described in [17].

2.5 Relevant Cryptographic and Privacy Concepts

As the basis of used technologies is now set, we can go further and define the concepts which are used in the building blocks for the architecture.

2.5.1 Haar Wavelet

The Haar Wavelet can be used to reduce the resolution of a discrete signal. In doing so, each application of the wavelet reduces the amount of data by half. The main benefit in using the Haar Wavelet is its lossless property. To perform this transformation, only basic operations are necessary, which is especially an advantage when having an environment like the Smart Grid, with lots of weak devices like smart meters, in place. It splits a signal in a high pass and a low pass band, where the high pass is used to reconstruct the original signal [2].
The following steps are performed:

1. Saving the difference of X_{2i} and X_{2i+1} which forms the high band H_i.
2. Calculating the average between X_{2i} and X_{2i+1} which forms the low band L_i.

To reconstruct the signal these steps are necessary:

1. To compute X_{2i} perform $\frac{1}{2}L_i - \frac{1}{2}H_i$
2. For creating X_{2i+1} calculate $H_i + X_{2i}$

Those operations are performed very efficiently and maintain the same average of values [2].

2.5.2 Merkle Trees

A useful application for hash functions are Merkle Trees [18]. An example for such a tree is shown in Figure 2.2.

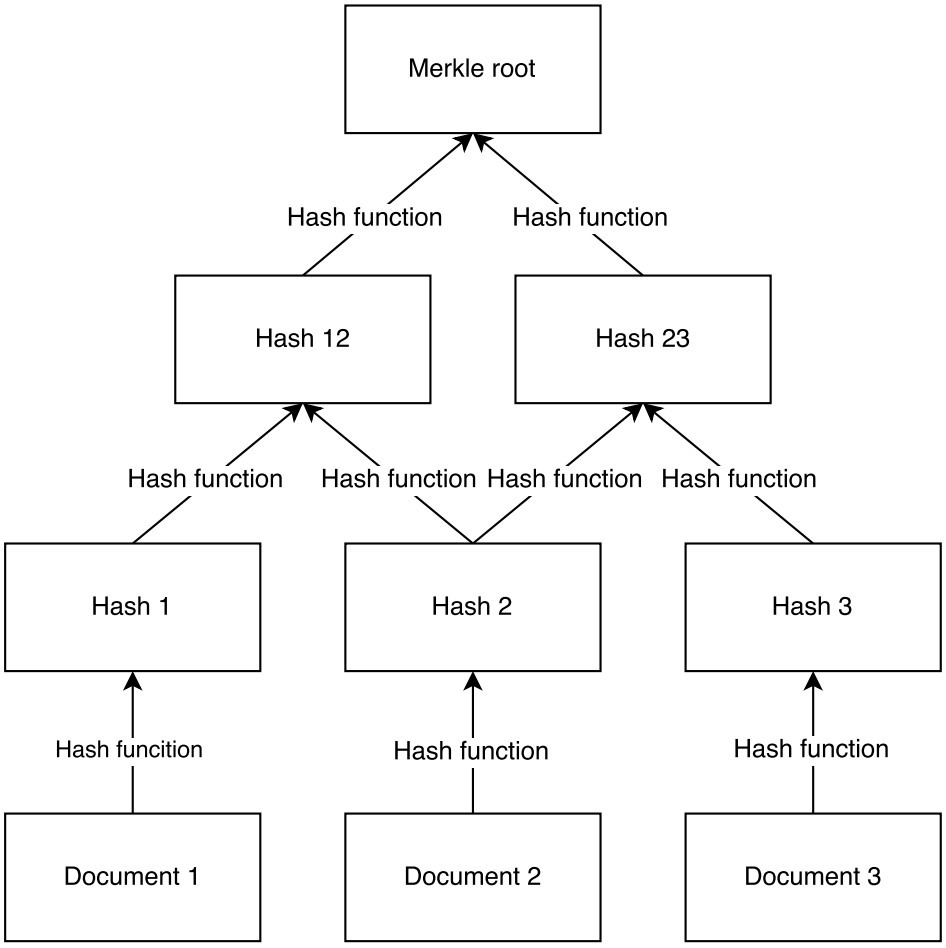


Figure 2.2: Merkle Tree example

As shown in the picture everything starts with a set of documents and their respective hash values, created with a function like SHA-256. Then two of those are combined

and hashed again until a single hash value is formed. This final value is called *Merkle Root*. An important property of such a tree is the ability of verifying if a document and therefore its hash, is included in the tree without actually having all the documents in place. This enables a device to not be in the need of trusting another one, by comparing the actual Merkle Root and the calculated one by recalculating the Merkle Tree with the received and probably questionable data.

2.5.3 Hash-Based Message Authentication Code

A Hash–Based Message Authentication Code (HMAC) [19] is a method to ensure message integrity. Besides usual cryptograpic hash functions, this approach uses a secret key as well to increase security. Therefore, HMAC can withstand attacks, which would be successful on the used hash function alone.
It is applied to a message m as follows [19] (Equation 2.3):

$$HMAC(K, m) = H((K' \, XOR \, opad)||H((K' \, XOR \, ipad)||m)) \tag{2.3}$$

This means that there is a cryptographically secure hash function H applied to a key K' padded with zeros and XOR with *opad* (0x5c repeated until the hash functions block length is reached), chained together with the hash of the key XOR with another value *ipad* (0x36 repeated until the hash functions block length is reached) with the message m.

2.5.4 Certificates

A digital certificate is used to connect a public key to a certain identity. It usually consists of several identifying parameters about the certified party, in combination with its specific public key [20]. Other information, like expiration date, used ciphers and more is included as well. These certificates are usually signed by a trusted Certifcation Authority (CA). Certificates can be self signed, but commonly they are signed by a CA especially in applications which are Internet-based. The certificates which are used to verify these CAs are called *Root Certificates* and are pre-installed on modern computer

systems, which introduces a variety of trust issues [21]. First a CA is a single authority, which can be forced by public authorities to act in unforeseen ways, or even worse be hacked. The pre-installation by manufacturers on systems is as well problematic as they could not act as desired and agreed (e.g. installing root certificates other than the usual ones).

2.5.5 Public Key Infrastructure

A Public Key Infrastructure (PKI) is a cryptographic system, which ensures message confidentiality and integrity. This is done through public key cryptography as briefly described in Section 2.4.2. The infrastructure ensures that a public key in a network is pinned to an identity. There are CAs in place which ensure this [7]. It is possible to set them up in different topologies to form trust models. Examples for these are:

1. **Direct Trust** is the simplest one and probably the closest to the actually targeted solution of this thesis. It describes that every participator holds every certificate of all parties. This can be cumbersome in terms of keeping all the nodes up to date. There is no need for a CA, as all the entities are able to keep track of the certificates by themselves [7], [22], [23]. Several unresolved questions are open with this approach. For instance how to handle weak clients with low computational power or limited storage abilities.

2. **Hierarchical Trust** has CAs in place to ensure the integrity of certificates. It works with a tree structure set up to increase performance. All the inner nodes are CAs and the leaves are the actual devices to verify. The root of this tree is a Root CA, which is universally trusted. This makes an architectural approach like Hierarchical Trust prone to attacks, as it is just necessary to capture the root CA for being able to impersonate all the devices identified by its certificates. [7], [22], [23].

3. **Mesh Trust** is an approach where each participating CA is certified by each other. This keeps the trust path short, but lacks in terms of scalability. A

benefit is the decentral approach, so it is very robust against attacks [7], [22], [23].

4. **Federated Trust** is a combined approach between the Hierarchical and the Mesh Trust model. This is done via the introduction of domains, which could be defined by regional borders, or per company influence area. These are connected via bridge CAs, but those again form single point of failures and can introduce significant speed issues [7], [22], [23].

None of the well-known approaches is flawless and a new way of managing certificates in the Smart Grid domain is desirable.

2.5.6 Homomorphic encryption

Homomorphic encryption is a way of encrypting data in a way to be able to perform certain operations in the encrypted domain as well. This could be addition or multiplication for instance. The big advantage of using such an approach is that the need for decryption would vanish. Therefore operations on datasets can be performed in untrusted environments as well. On the other hand encryption following e.g. Paillier cryptosystem [24], is computationally very expensive, although it is not fully homomorphic, meaning that only certain operations are possible in the encrypted domain, in this case addition. Especially compared with common symmetric ciphers like AES and following the assumption that 1024 modules have to be used for sufficient security in place. The following table (2.1) shows the results of Eibl and Engel from [2], where they applied it to load curve data:

	AES	PAI-1024
Exec. time	1.91	85,355
Std. dev	0.03	133

Table 2.1: Execution time of encryption of wavelet transformed load curves. (400 curves with 100 encryptions each) [2]

It can be seen that Pailler, which is a not fully homomorphic cipher (which are faster than fully homomorphic ones) is slower than AES by the factor of 44,688 in this application, where load curves with 96 values are used. This could still be an acceptable

time for certain applications, but must be kept in mind.

2.5.7 Blockchain

The Blockchain as emerging technology forms the basis of distributed systems like the cryptocurrency Bitcoin[1] [5]. Its main advantage is the ability of coordinating a highly distributed network for storing data, or even executing programs (Smart Contracts) in a distributed way, as for instance in Ethereum[2] together with vanishing the need for trusting the other participating nodes and therefore creating a "trustless" environment.

A Blockchain comprises several blocks, which are linked together via the hash of the previous block. The structure of a block as it exists in, e.g. Bitcoin is shown in Figure 2.3.

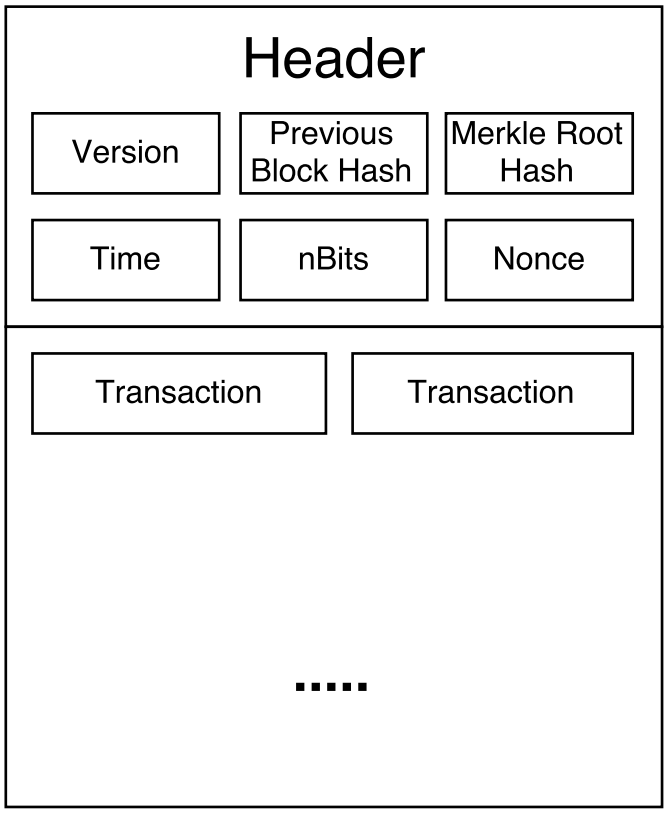


Figure 2.3: Block example

A block has a header with information important for the chain itself. This comprises a version, the hash of the previous block and the Merkle Root Hash, as discussed

[1] https://bitcoin.org/
[2] https://ethereum.org/

in Subsection 2.5.2. There is as well a timestamp included and a nBits field, which adjusts the difficulty of the mining process. This is necessary to adapt to different computational power available in the grid dynamically, as in a Blockchain application like Bitcoin, the number of participants and therefore computational power can change quickly. The *Nonce* field is used to adjust the hash-value of the block during the mining process [5].

The body is made of smaller information chunks which need to be saved. This can be any information, in case of Bitcoin these are the transactions. During designing such a Blockchain it should be kept in mind to limit the stored information as much as possible, to limit the size of the overall chain, as it has to be stored on each participating device if there are no measurements like Simplified Payment Verification (SPV) or similar in place. A consideration could be to save just hashes (or HMACs) and deliver the plaintext from a central storage only on a need to know basis.

Different operation modes for a Blockchain exist, depending on the level of equality in the environment [25]:

1. **Public Blockchains** let every participator of the Blockchain vote for the incorporation of new Blocks. This must be coordinated through some sort of consensus mechanism which is discussed in Subsection 2.5.7.1. All the entries are publicly readable and every participator has the same rights.

2. **Consortium Blockchains** have dedicated nodes of specific entities in place, which are responsible for incorporating new blocks. Coordination in terms of a consensus mechanism has to happen if this approach is used, but only for the entities eligible to vote. The possibility to read the chain might remain public.

3. **Private Blockchains** are under the supervision of a single entity. This single entity decides if a new block is incorporated or not. It might also be not publicly readable and usual applications are within an organizations boundary.

Which one to choose is highly dependent on the actual use case. Public Blockchains qualify for situations where there is no physical representation of the stored good (e.g. Bitcoin) and a big number of participators with equal rights is desired. By design there

can't be a single entity which controls what is going on in the chain, so equality in real world as well as in the Blockchain domain is a necessity. Configuration should be thought through very well too, as changes afterwards are very hard to enforce if not explicitly planned.
Consortium Blockchains have their benefits when certain entities have to maintain a physical representation, like an energy infrastructure. They also fosters collaboration without being limited to company borders as deciding nodes in the chain can be supplied by all of the participators.
Private Blockchains come into play if there is exactly one entity, like an insurance, which wants to be sure that only certain entries are added and wants to verify those, but wants to maintain a public readability of those entries.

2.5.7.1 Consensus Mechanisms

As a Blockchain is a highly decentralized system, if it is used with a public or consortium approach as discussed in the previous chapter, coordination of the participators has to take place. There are several possibilities available how such a coordination can be done.

The first ones discussed are those available for the Public Blockchain:

1. **Proof of Work** is a mechanism where the amount of accomplished work determines if a node of a chain is allowed to add a new block to the chain. This is usually done through a cryptographic puzzle. The protocol supplies for instance a desired hash puzzle like "it should end with 'aa'". Because of the unpredictability of hashes, the nodes have to brute force the value. The first which finds the value is allowed to add the block. A disadvantage is the usage of energy used to determine a fitting hash, as computation can get resource intensive, if the difficulty is set accordingly. Proof of Work is sometimes called mining, especially in the Bitcoin context and as reward the protocol gives the "miners" the respective currency of the chain to create an incentive for mining and foster block creation [5].

2. **Proof of Stake** on the other hand focuses on participators with the highest stake of the overall good, as it can be assumed that those parties have the greatest incentive to have the chain up and running in a trustworthy way. Therefore a pseudo random process is used where the participators are weighted with their respective wealth. In the context of Bitcoin this would be the Bitcoins itself, so wealthier nodes are more probable to be chosen. This has the great advantage of no unnecessarily used energy, but fosters monopolization as a party gotten lots of stake once might not want to share it again. This again could lead to a shortage of money in the system and harm trade or other services built on top of the chain. A further requirement is that there is a currency in place, which the participators desire to have [26].

The next ones apply for Consortium Chains and partially for Private Blockchains:

1. **Proof of Authority** requires entities to be in the possession of certain private keys to sign blocks. There is no coordination happening, so it is a viable solution for private chains. It is not very resistant to the failure or malicious behavior of nodes, as a single node could simply incorporate new blocks on it's own behalf [25].

2. An alternative is the **Practical Byzantine Fault Tolerance** (PBFT) Consensus. Initially there is a leading *Validation Node* defined. This one is responsible for broadcasting a "to be made" decision to all of the other participators. The decision is made, if more than 2/3 of the participators agree e.g. appending a new block to the chain [27].

3. A further alternative is **Delegated Proof of Stake Consensus** (DPOS) which is comparable to PBFT but not all participators are eligible to vote, just selected *witnesses* which are chosen leading nodes called *Stakeholders*. Witnesses are reelected on a daily basis. The algorithm ensures that every witness voted at least once before a new witness-selection-process starts [28].

The approach used depends very much on the actual application. Where Proof of Authority qualifies very well for private chain applications, PBFT and DPOS are more

viable for at least somehow distributed chains. A voting process with PBFT gets increasingly expensive in terms of contacting all the nodes with increasing number of the participators. DPOS might be a solution viable for arbitrary chain sizes.

2.5.7.2 Smart Contracts

Smart Contracts are applications that can be executed on the Blockchain by participating nodes, which qualifies them as distributed programs [29]. Their logic is often oriented on real contracts, e.g. to pay a certain amount of currency if certain events occur, like an insurance. On public Blockchains they might be brought in by every participating party, in Private or Consortium Chains a dedicated group of nodes might have the reserved ability of creating such contracts. They usually need fuel to be executed, which e.g. is called "Gas" in Ethereum, which is directly connected to the actual currency of the chain. Therefore it is ensured that the computational power of the chain is not exploited. Outcomes of Smart Contracts are publicly readable.

3

Design Constraints

To be able to set up an appropriate architecture for smart grids, which takes security and privacy into account, the affected parts have to be defined. Existing architectural models will be assessed and the parts which are affected by security and privacy will be identified. Afterwards legal requirements in the European Union, the United States and Austria will be analyzed. Interconnections between the different architectural parts will be shown and the types of data streams, which have to be handled, are listed. This will help to establish a common view on where the different architectural technologies have to be placed and to distinguish when they are appropriate and when they are not.

3.1 The Smart Grid Architecture Model

The Smart Grid Architecture Model (SGAM) is a 3-dimensional architectural model for the smart grid, which is shown in Figure 3.1 and proposed in [1].

The dimensions in the figure are defined as follows:

Domains: This is where the different sites involved in the grid are defined. They must be considered for the architecture on a general level. Also the different computational abilities on each level must be taken into account.

Zones: These are used for adding the concept of power system management and are describing data flow aggregation towards the market. As this understanding

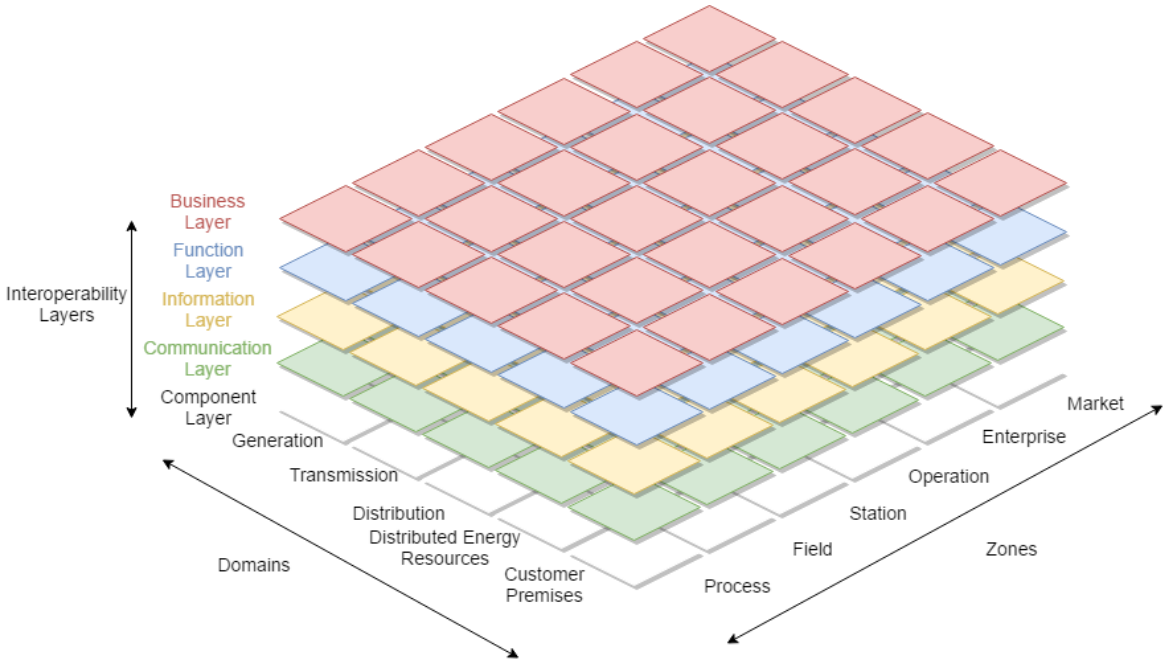


Figure 3.1: SGAM framework [1]

of data flow and management is different from the one this thesis is going to address, the zones are conceptually and intentionally left out for this proposed architecture.

Interoperability Layers: Herein relies the definition of cross-cutting concerns between the SGAM-Basis-elements which are defined via the Domains and Zones. Those are used to define the interaction between different systems and are from inherent importance for the definition of a privacy and security aware architecture.

This model tackles every single aspect of the smart grid and might not be optimal for an architecture with a foremost technical focus. Therefore a reduction of complexity happens in the next section, to be able to focus just on the most relevant parts of this architectural approach.

3.2 Simplified SGAM Model

As the SGAM acts as a basis for the proposed architecture of this thesis, some adaption has to be done to fit for the needs of focusing on privacy and security. As mentioned

beforehand, especially the zones can be left out as they don't provide required information. In Table 3.1 the relevant parts from SGAM are displayed and described. Those parts, where a special emphasis is put on in the thesis, are marked bold.

	Third party (Extension to SGAM)	**Generation**	**Transmission**	**Distribution**	**Distributed Energy resources (DER)**	**Customer Premis**
Component layer	Services for the customer (optimizing energy usage, rented charging stations etc.)	Power plant, Grid operator	Grid, transformer	Neighborhood aggregator, Substation transformer	E-Mobility, solar, wind	**Smart meter and household infrastructure.**
Comm. layer	**Here the communication between the different devices of the grid takes place and it is the main part of this thesis. Table 3.2 specifies the technologies which will be assessed to form a suitable Smart Grid Architecture**					
Information layer	Can be seen as predefined, as this is expected to be required by stakeholders. Must be considered in terms being able to be carried via the communication layer.					
Function layer	Will not be discussed in detail in the thesis. Just for verification purposes of the underlying use-cases.					
Business layer	**Also important as here are the regulatory requirements located.**					

Table 3.1: Adopted parts of the architectures.

As the supporting parts of SGAM have been identified, it is possible to transform the remaining parts into a framework for setting up a privacy and security aware architecture. The information present in the business layer will be addressed in Section 3.3. Table 3.2 covers the *Communication Layer* of the SGAM Model and shows the technologies which will be assessed through this thesis and if and where they are a good fit for a privacy and security aware Smart Grid architecture.

Architectural overview				
Level	**Devices**	**Security**	**Privacy**	
Household	AMI	PKI based on Blockchain, Authenticated Encryption with Associated Data (AEAD)	Data aggregation in Smart Meters	Multi resolution conditional access
Neighborhood	Aggregator		Aggregation on neighborhood level	
Grid operator	Server			
Utility	Server			
3rd party	Server			

Table 3.2: Architectural parts to be covered technically

To begin, legal requirements in the European Union, Austria and the United States are addressed.

3.3 Legal Requirements

There are several legal requirements in place in different countries regarding energy grids. This Section discusses the implications in the European Union, the United States and Austria on a high level and how they affect the architectural decisions.

3.3.1 Requirements in the European Union and Austria

The European Union created a recommendation for implementing and rolling out of Smart Metering systems [30]. Therein the member states are required to install such intelligent energy networks. Privacy is a clearly addressed concern within the recommendation. There is as well regulated, that after a data breach a notification within 24 hours has to happen to inform the affected customers. This underlines the necessity of having a security aware architecture in place to prevent such events.
In this document common minimum requirements are also listed, which include a 15 minutes data reading interval. Aligned with this EU recommendation, in Austria the requirements for grid operators and utilities are regulated by the "Elektrizittswirtschafts- und organisationsgesetz" [31]. This law states that a grid operator has to ensure grid access. Furthermore certain quality requirements have to be met. If those quality criteria cannot be matched, the grid operator is eligible to disconnect certain parts of the grid to maintain stability. This is especially important when cooperation with other European grid systems comes into play. If stability cannot be ensured by any reason, claims for damages may occur. This puts special emphasis on a decentralized approach in terms of the underlying architecture.
Concerning smart meters (intelligent metering), it is stated that power consumption information must be made available for the customer. Thereafter the law is very specific about the handling of more fine grained metering information. This more specific information (referred to as 15 minutes interval data) should be available for the customer and must not be submitted to, or requested by the grid operator or utility without the consent of the customer, unless it is directly coupled to maintain grid stability and just for that purpose. Afterwards it has to be deleted and the access has to be recorded. The customer has to give explicit consent for any other usage. Usually just consumption data on a daily basis should be transmitted [31].

3.3.2 Requirements in the United States

The U.S. has similar regulations in place as stated in [32] (Federal Regulation and Development of Power). There the grid operators and energy providers are as well

forced to deliver power in a constant and reliable manner. The law is not as specific as it is a federal law and meant to be applicable for the whole of the U.S. in contrast to state level jurisdiction. In New York state a development plan is on its way to set up a smart grid in the state [33], where a Smart Meter is defined as a Device which has at least the ability to track consumption in 15 minute intervals, therefore a more fine grained resolution can be expected. The current state of this is a bill and not yet a law. Dates in the document refer to mid of 2018 for an expected implementation.

3.4 Technical Requirements

Several technical requirements are given in terms of grid components and data which is sent and used in the grid. The following subsections will address those requirements and explain the certain implications set up by them.

3.4.1 Existing Energy Grid Infrastructure

As seen in Table 3.1 there is a number of different devices out in the Smart Grid. Those are initially shown in Figure 3.1. In this section only those which have an impact on security and privacy related topics are taken into account. Assumptions which can be basically done about them are:

1. **AMI:** Lots of them exist in the grid as every participating household has to have one. Those are very limited devices in terms of performance, but must be taken into account as they are forming the majority of the network.

2. **Neighborhood Aggregator/Substation transformer:** Those devices are responsible for aggregation usage data of customers out in the field and cannot be trusted in most cases [10].

3. **Power Plants / Grid operator:** These are trusted places under the direct control of grid operators and utilities.

4. **3rd party companies:** Those are untrusted but it can be assumed that they have an incentive to keep the infrastructure up and running.

As we can see from the prior listing, the main bottleneck are the AMIs on household level as untrusted device.

3.4.2 Requirements Imposed by the Infrastructure

As shown in Table 3.2 there are several different architectural parts which have to be taken into account. All of them imply certain constraints in terms of computational power in the smart grid, which will be discussed here in contrast to the trustworthiness in Subsection 3.4.1.

1. **AMI** Advanced Metering Infrastructures are in every participating household and therefore out of physical reach of grid operators and utilities. It must be ensured that those devices are tamper-proof, especially when they are responsible for aggregating billing data. Furthermore it can be assumed that those devices are especially computationally weak and will form the majority of the network.

2. The **Aggregator** is a device located in neighborhoods and owned by the grid operator. Computationally better equipped than an AMI it is possible to executed more complex operations and store more data there.

3. The hardware of the **Grid Operator** and the **Utility** has virtually no boundaries and can be made fit to the needs of a possible architecture. Also those devices under direct control of the operator can be seen as trustworthy.

4. Servers of **Third Parties** should only be allowed to have access to data where either the user has given consent, or no privacy issue at all is in place. Execution of critical computational tasks should be omitted. Besides that, the same infrastructural assumptions for grid operators and utilities can be made.

3.4.3 Smart Grid Data Streams

To be able to discuss how to secure communication in a Smart Grid, a distinction between the different data types occurring in the grid must be done. It is proposed by Wicker and Thomas in [10] to differentiate between four types of data in a Smart Grid:

1. **Pricing information:** This is neither confidential nor introduces privacy issues as pricing information is just sent from the utility to the consumer to affect its behavior. Nevertheless integrity is a crucial part, as altered pricing could have an impact on consumer behavior.

2. **Billing:** To be able to bill people for their exact energy usage, their exact consumption must be known. Although this would reveal a lot of privacy concerning information, it is not necessary to transmit this fine grained data to the utility. Wicker and Thomas suggest to aggregate this kind of data already in the Smart Meter and just transmit the price weighted information to the utility. Confidentiality and integrity must be ensured during transmission.
   It is crucial to have tamper-proof devices as Smart Meters in place, to be sure billing works as expected.

3. **Control Signals:** When having a *Demand Response System* in place, one capability is to send control signals to devices for optimizing the energy usage in the grid. As this communication is unidirectional and only from the utility to the household, no privacy issues are exposed. But as devices are controlled by that signals, integrity and confidentiality must be ensured.

4. **Consumption data:** Most of the benefits of a Smart Grid depend on the retrieved consumption data. Prediction of price models or grid balancing are tasks which are just possible due to the usage of this fine grained data. Therefore it is essential to have some measurements in place for protecting the consumers privacy as well as to ensure data integrity and data confidentiality.

This distinction is crucial in terms of defining measurements for their protection as cryptographic tools can be time and resource intensive. Their application should be thought through very well and focus on the necessary parts.

3.4.4 Communication Interfaces

In order to setup a meaningful architecture, all possible interfaces between participators of a smart grid must be identified. This is done in Figure 3.2

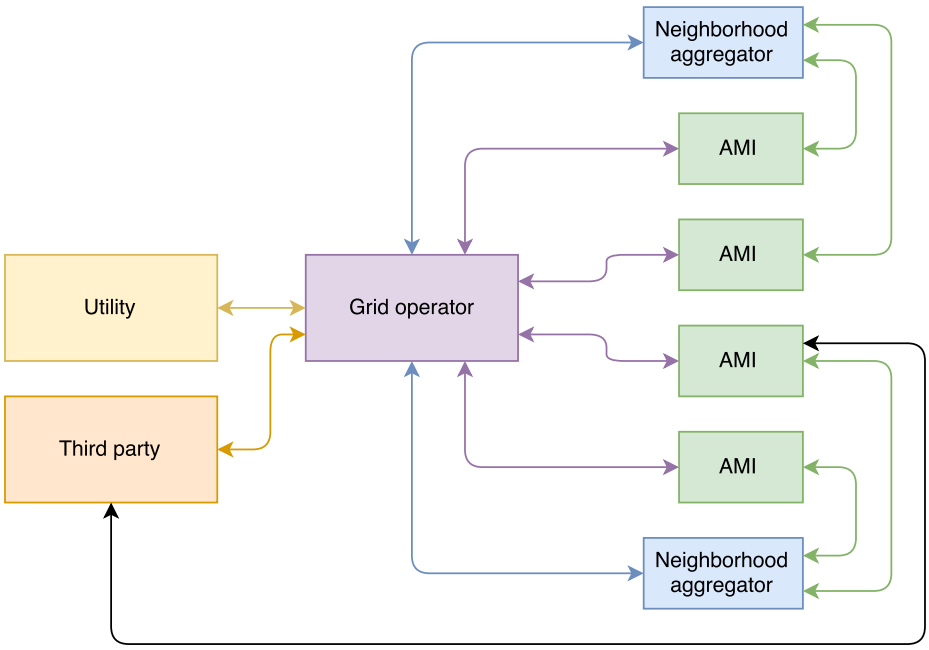


Figure 3.2: Interfaces

It can be seen, that there happens a communication from AMIs to neighborhood aggregators and furthermore to the grid operator and the utility. AMIs are able to communicate with third parties as well, without a grid operator or a utility involved. Therefore all the security and privacy measures must be in place and working already at the level of a household. The application of the proposed architecture would fit very well in Virtual Power Plant environments, like those described in Section 2.3, which are connected to the rest of the grid via well defined interfaces to enable controlling and to operate independently of the the grid, if necessary.

4

Security and Privacy Aware Architecture

After pointing out the benefits of the mentioned concepts and why a Smart Grid is able to take advantage of them, a propose will be presented how to combine them into a meaningful architecture, which fulfills the confidentiality, integrity and authentication requirements [11]. Furthermore it tackles privacy concerns of the consumers.

4.1 Architecture Building Blocks

In this section all the underlying technologies are discussed, which are needed to form the proposed architecture for the Smart Grid. These are more specific for architectural use than the ones discussed in Section 2.4 and Section 2.5 and comprise of them.

4.1.1 DPKI

Through using the abilities of the *Blockchain*, it is possible to set up a new way of storing identity information of a X.509 certificate [20] inside a then decentralized PKI [34] and inside the blocks of the chain. A common PKI relies heavily on Certification Authorities (CAs), as they have control over root certificates and possess their private keys. With this set, there are several single points of failure in terms of every single CA. If it is possible for a hacker or governmental organization to infiltrate a CA,

authenticity can no longer be ensured in the network. Through the approach of saving identity information in the *Blockchain*, the participants can take advantage of the several benefits provided.

First of all, the identity information is automatically copied to every participant of the chain. Therefore there is no need for a central CA, as every node is able to prove the authenticity of communication peers. Even if large portions of the network fail, it would not affect the communication abilities. Furthermore a very decentral approach is fostered. The majority of the network decides over the trustworthiness of nodes [34]. Every node is able to verify the identity of all the other participators as each node possesses all the information or at least can verify it by the help of Blockchain mechanisms like Simplified Payment Verification, which is introduced in [5].

To be sure to have the right communication counterpart in place, it is crucial to have a proper incorporation mechanism set up, especially if one is not facilitating a *Public Blockchain* as the basis for the DPKI.

4.1.2 Authenticated Encryption

Authenticated Encryption [35] is a technique to encrypt and authenticate at the same time. This is different to the approach used by a common PKI (decentralized or not) because it initially only provides confidentiality and authentication but no integrity as a hash function must be applied to the message in different steps. Another benefit arising from this is the gain in terms of speed. As the following results of a Python[1] implementation show (Table 4.1), Authenticated Encryption is much faster than asymmetric encryption by using 2048 bit RSA keys.

	Time [s]	
Encryption rounds	AEAD	RSA
10	0.019	1.070
100	0.222	10.631

Table 4.1: Encryption with AEAD

Therefore it is suggested to use the underlying PKI for symmetric key exchange and to

[1] www.python.org

use Authenticated Encryption for further communication to have a hybrid cryptosystem [7] in place.
There are different operation modes available for Authenticated Encryption, which have been assessed in [36]. Based on the recommendations in this resource CCM, which is basically Cipher Block Chaining (CBC) with a Message Authentication Code (MAC), is proposed as the preferred operation mode as computational power is a scarce resource in Smart Grids and CCM uses few resources. This mode is memory efficient and provides the possibility to attach associated data (AEAD).

By the help of this associated data it is possible to have information attached to the encrypted data which is not encrypted, like a header. Only the integrity is ensured which has benefits in terms of routing or further processing of the data.

4.1.3 Multi Resolution Conditional Access

By the help of Multi Resolution Conditional Access as proposed in [2], it is possible to split the load data into different resolutions by the usage of, for instance the Haar Wavelet. Afterwards these resolutions can be encrypted separately and access to them can be provided on a need-to-know basis by the owning entity of the load data, which is usually a household in the context of a Smart Grid. This provides several benefits in terms of privacy as the desired load data resolutions for service providers can be adjusted at will. Therefore the customer knows, which parties are able to see which details of their energy consumption.
As described in [2] the transformation is not resource intensive and would therefore just add a very small footprint to the overall time needed for preprocessing load data at consumer level. This is especially beneficial because of the huge advantages in terms of privacy.

4.2 Proposed Architecture

After assessing possible usable technologies in the context of Smart Grid, this section combines them into an architecture as shown in Figure 4.1. The different participators in there are interacting with each other through the previously defined interfaces. Main initial application for such an architecture would be a Virtual Power Plant as discussed in Section 2.3. This forms a closed system with clearly defined interfaces to other participators of the energy grid. Therefore this would be a good fit for every new architectural approach, especially because both systems are designed to be applicable incrementally. The herein proposed solution fits well for such a system as it is designed to interact with a big number of participators.

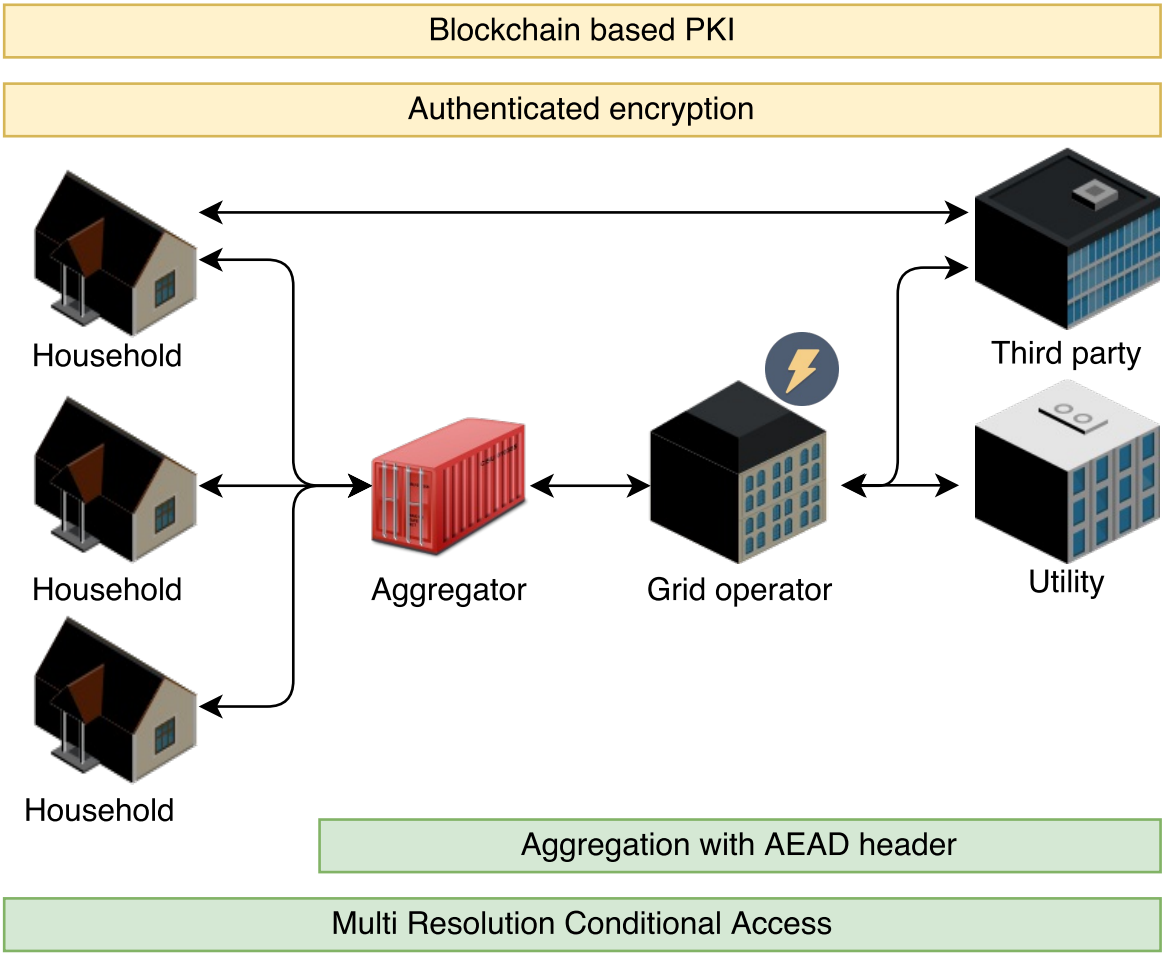


Figure 4.1: Proposed grid architecture

4.2.1 Security

The basis of the proposed architecture will be formed by a DPKI, which uses a Consortium Chain approach as discussed in Section 2.5.7. The main reason to do so is that there is no trust needed within the whole Smart Grid when using a DPKI. There is no CA in place, which could act unforeseen. Even if the majority of the grid fails, if just two nodes remain, they still would be able to verify the authenticity of the other device. Speed is increased too, as no verification request must be sent to a CA which might be located elsewhere. This makes a DPKI a much better fit as a common PKI. The Consortium Chain approach is necessary as the grid operator or the utility, whoever is in charge of maintaining the infrastructure, has a much higher need to secure the physical infrastructure as well as the transmitted data, because those are forced by law to do so. This is presented in Section 3.3. It is highly recommended to set up elliptic curve cryptography as the asymmetric cipher as it ensures greater security than for instance RSA with fewer bits used for the key as shown in [16]. The huge number of expected devices enforces a strong emphasis on optimization in every aspect of the architecture.

Therefore there will be *Validator Nodes* in place which are responsible for changes like adding or removing trustworthy devices to and from the grid. Furthermore no public mining process or similar will take place. This is necessary as not every aspect of the power grid can vanish in favor of digital representations and Proof of Work algorithms are not viable.

Different security levels of incorporation should be set up. An approach could consist out of the following two integration strategies:

1. **Option A** is for devices with high and medium security requirements. The process would consist out of the following: The Validation Server creates an asymmetric registration keypair. A transmission of the public key to the bearing device is initiated. Here a differentiation between medium and high security devices can happen. This should be done either in person (high security) or via a established TLS connection (medium security) prior to the communication. This key is then used to encrypt the public key of the device and sent back to the

Validation Server. This one forwards the key and further transmitted information about the device to the consortium of the Validator Nodes, to initiate a voting process.

2. **Option B** is reserved for devices with low security concerns. Therefore a registration process like with Domain Validated Certificates is used [37]. The Validation Server tells the bearing device to create a specific directory to ensure control over the device itself. This is then checked by the Validation Server. The bearing device is requested to sign a certain value as well, to ensure the possession of the corresponding private key. Afterwards the information is sent to the consortium chain validators for voting.

This approach ensures to have a balance between importance of the device and administrative effort in place to verify the trustworthiness.

After having a new device ready for voting, a decision can be made via DPOS as discussed in Section 2.5.7.1. Therefore several identity information submitted via Option A or B are collected to form a new block. This block is then handed over to the voting mechanism, which takes place according to DPOS. If the result of the voting is positive, the block is signed by the according devices and incorporated into the chain. Mirroring of identity information happens then across the participating devices.

4.2.1.1 Computationally Weak Clients

As one can see in Table 4.2 the resulting Blockchain can reach a significant size, especially when it should be saved on devices like Smart Meters. The actual amount of devices to be managed should be considered even bigger.

The implementation was done using Python 3.6 [2], without any special library and a data structure which can be seen in Table 4.3. Verification is done by simply processing every block and measuring this time, like a Validation Node would do.

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

Identities	Blocks	Block Ids	Size (KB)	Header (KB)	Verif. time (s)
1,000	10	100	643	3	0.053
1,000	100	10	665	23	0.054
10,000	100	100	6,422	23	0.508
10,000	1,000	10	6,647	226	0.524
1,000,000	1,000	1,000	639,899	226	43.167
1,000,000	10,000	100	642,149	2,256	43.747
1,000,000	100	10,000	639,674	23	43.614

Table 4.2: Blockchain size simulation results

The saved header information is pretty much equal to the one in Bitcoin Blocks, except the parts needed for mining which are skipped. The saved identity information is close to the one saved in X.509 certificates [20].

As we can see, the overall size of the Blockchain header is very much dependent on how much identity information is stored per block. A good threshold should be chosen to ensure block creation can take place at sufficient speed but as well keeps the header size low.

Performance of the chain on computationally weak clients is directly coupled to the size of the overall header. The steps to be performed for identity verification in such clients are the following [5]:

1. At first the weak clients (e.g. a smart meter) download the header of the chain without the identity information of the blocks.

2. The client wants to verify if a certain identity is part of the Blockchain and therefore a trustworthy communication counterpart. It requests the block in which the identity is stored from a node, which has the whole Blockchain (to lookup the corresponding header), together with the Merkle Path of the identity. This looks like Figure 4.2. The Merkle Path would be all hashes marked with blue color. *ID4* is the one to be verified and already in possession of the client as well as the green marked *Merkle Root*. Via hashing of *ID4* and hashing this together with H3 and so on, the client should be able to construct a *Merkle Root* matching the one it already possesses. If that is the case, the client knows the identity is valid and can initiate data exchange.

Header	
Version	Version of header information
Time	Creation time of the block
Previous Block Hash	Double SHA-256 hash of previous block
Merkle Root Hash	Root hash of the Merkle Tree formed by the identity entries of this block
Identity Information	
Id	UUID which uniquely identifies an entity in the grid
Version	Version of identity information used
Serial Number	Used to refer to explicitly this identity entry and all related things like e.g. logging. This field should be at least unique in combination with the issued Validation Node.
Algortihm	The algorithm used to generate the keypair
Validity Period	Signals how long this identity information is valid and when a new one is necessary. This should be chosen with care as this affects the chain length.
Issuer	Which Validation Node created this entry
Subject	More identifying information regarding the identified subject. This should be a hash of some textual information saved at other places, as space is crucial.
Subject Public Key	Public key for this entity
Revoked	A boolean flag to determine if this entry is used to revoke this entry.

Table 4.3: Block field description

4.2.1.2 Architecture Configuration

The configuration of the *Blockchain* should be saved in Smart Contracts [34]. Saving other configurations of the architecture, like used ciphers, is possible as well. This enables the benefit of a constant ability to adapt for changing needs, e.g. broken ciphers, which can be changed for the whole Blockchain or the overall architecture. Smart Contracts could be executed in the same way as a voting takes place. With the signature of the majority of *Validator Nodes* in place, coordinated through using DPOS, security or privacy relevant technologies can be exchanged. This ensures resistance against future attacks and is able to work within the boundaries of a Consortium Chain.

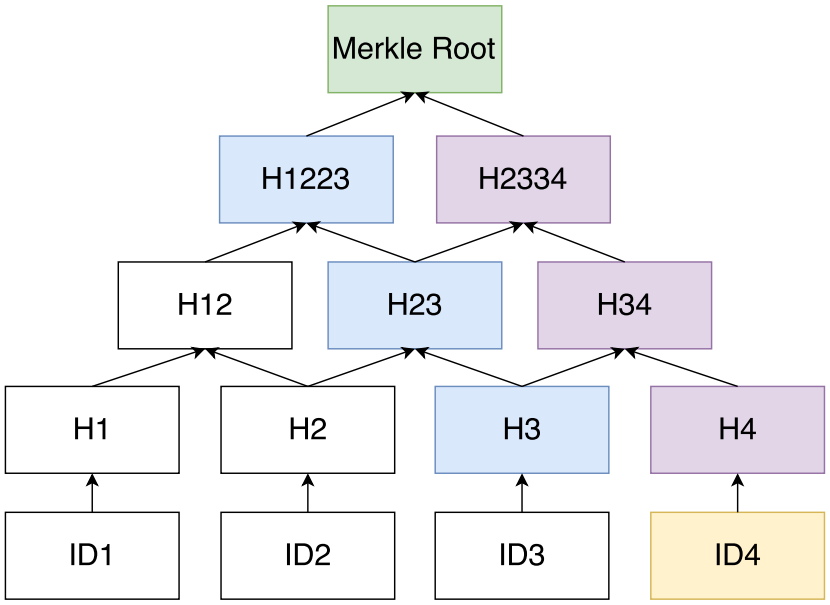


Figure 4.2: Merkle Path

4.2.1.3 Securing of Data Transmission

As there are only trusted devices in place, the transmission of data is the next thing in the architecture to secure. As discussed in Section 3.4.3 there will be four major types in the grid. The only application for low security integration of devices is the pricing information and every device which is just concerned by that. The confidentiality of the data is not crucial, but ensuring the integrity and securing this data streams with just HMAC as described in Section 2.5.3 is suggested. HMAC has better security in place as just hashing the data and encryption of the data is not necessary.
Households transmit data either to third parties, which are part of the DPKI as well if desired, for achieving some service or to a neighborhood aggregator. If the device or organization is out of the context of the DPKI for the Smart Grid, communication can take place via common TLS connections or HTTPS. The DPKI does not limit such a hybrid approach if one endpoint identity is not part of the Blockchain. To ensure confidentiality and integrity inside the Blockchain-wide communication for every data except pricing information, Authenticated Encryption with Associated Data (AEAD) is used.

AEAD has the big advantage to include unencrypted information where just the integrity is ensured, which makes it superior to a usual approach with just AES or a

similar symmetric cipher in place. This enables the network to gain processing speed, as no resource intensive decryption has to be done every time information has to be processed or routed, which is especially a benefit in this Smart Grid context. Benefits in terms of security are as well in place as there is no necessity to decrypt information at other places than the receiver. To be sure to have a secure transmission in in the grid, a symmetric cipher like AES with at least 128-bit key-length should be used.

4.2.2 Privacy

Besides security, privacy is the second major part of the proposed architecture. To the best knowledge of the author there are two approaches available in the grid to ensure privacy and both are used in this architectural approach. First there is Multi Resolution Conditional Access to access load data. The second option is secure aggregation. Multi Resolution Conditional Access fits very well together with Authenticated Encryption with Associated Data to ensure privacy aspects of the grid. As proven in [2] there happens a lossless sub-sampling of the energy consumption by usage of the Haar-Wavelet as discussed in Section 2.5.1. The different resolutions are thereafter encrypted with different keys, which can be achieved by the usage of Authenticated Encryption and sent over the network. Detailed unencrypted meta-information can be attached via the according property of AEAD, which enables enhanced aggregation at the level of the aggregator. The constant split of data into high and low pass data to create different resolutions would form a structure as shown in Figure 4.3. Access to the highpass parts of the data can then be protected via different keys as proposed in [9].

Aggregators can then decide based on the not encrypted but integrity ensured header information, which encrypted information should be aggregated. To aggregate the data, decryption is required if no homomorphic cipher is used. To use a homomorphic approach (even partially like a Pailler Cryptosystem as described in [2]), it must be weighted if such a big delay in terms of processing time is acceptable. At least from a legal perspective it would be viable, as the defined resolution therein is 15 minutes.

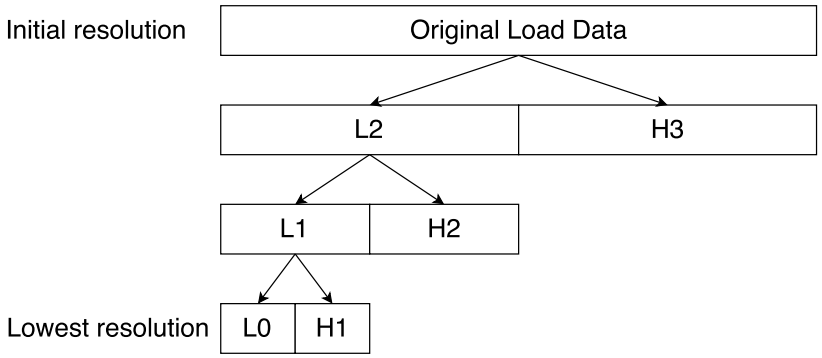


Figure 4.3: Different levels of wavelet transformation

Because the aggregator has knowledge about the level of detail included in the encrypted load data via the header information, just the packets relevant for privacy can be identified. This enables another speed advantage. One can think about special marking of information packets if aggregation should be strictly prohibited. This could be necessary for certain use-cases specifically designed for one household e.g. there is a third party attached in the line after the aggregator, which is a specific service provider for that household.

If this aggregator is not a trusted device, the approach using homomorphic encryption for transmission of the respective data to the aggregtor should be thought through. The Smart Meter in the household would therefore decide if a specific resolution is prone to a privacy infringement. Based on the contextual privacy of this resolution (e.g., is the sleep-cycle to a certain degree inferable? [4]), AES could be omitted and replaced with homomorphic ciphers such as Paillier Cryptosystems as stated in [2]. Herein Engel and Eibl show as well, that encryption done with this approach, together with a 1024 bit module, leads to an enormous increase in terms of processing time. Therefore it is proposed to use this approach very selectively. Another aspect which should be taken into account is that encrypted load data could be altered. This is possible through the homomorphic property of Pailler Cryptosystems and integrity should be always ensured by attaching signed hashes of the actual encrypted data or by the use of HMACs.

After aggregation a household looses control over their respective data as no clear distinction is given anymore. This control is given to the grid operator or the utility,

whoever is responsible (and required by law). Afterwards a transmission to the grid operator, utility or third party is possible. These parties are enabled to decrypt the respective information with the resolution they need and can simply decrypt just those junks as they have again access to the unencrypted header information via AEAD. Furthermore they can be given access to just those junks, which enables selective security to take place. This is another big advantage of using multi resolution data.

5

Proof of Concept

To proof the viability of the architecture, which was introduced in Section 4.2, a Proof of Concept implementation is presented in this Section and analyzed. First the used environment is described and justified. Afterwards the actual code which is responsible for the different parts of the architecture will be assessed. In the end possible improvements will be covered and steps are introduced, which would be needed for a real world application. Some parts of the code are omitted and marked with comments. The full code is available in Appendix A.

5.1 Computational Environment

To implement the different parts of the architecture, the programming language Python in version 3.6 [1] is chosen. Python is a programming language with a considerably big community [38] and therefore lots of libraries are available to tackle different tasks. Another reason to choose this language is because it can be run on a variety of platforms, ranging from embedded devices to web servers, which enables a coherent development environment. This is especially helpful in the context of smart grids as a wide spectrum of computational power is present, ranging from low power AMIs to dedicated servers. All the implementations are done on a standard Windows 10 Home PC with 16GB of RAM and an Intel i7 5500U processor with 2.4 GHz.

[1] www.python.org

5.2 Architecture parts

Each of the following subsections covers the implementation of one of the architectural parts and introduces the used libraries and how they connect to the other parts of the architecture. Communication happens through a REST-interface between common nodes and validator nodes, which have different targets. Figure 5.1 pictures this situation.

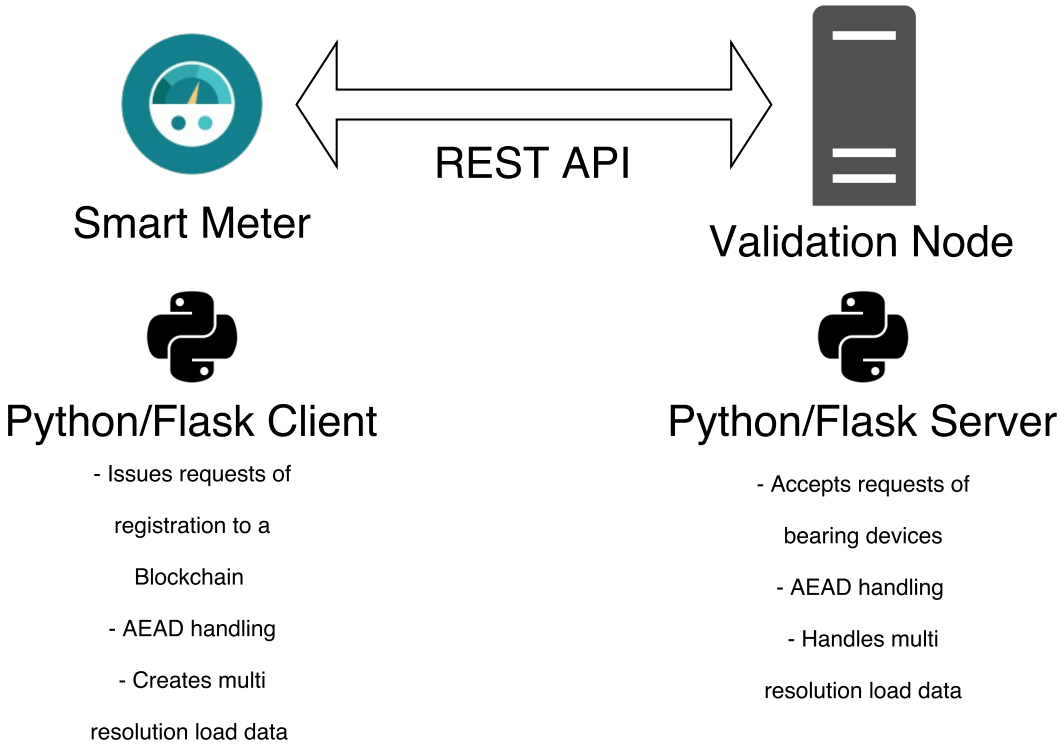


Figure 5.1: Communication in Proof of Concept implementation

5.2.1 Blockchain based DPKI

In this Proof of Concept the library flask[2] in version 0.12.1 is used to setup participating validator and client nodes as web servers. Both of them use a shared library, which supplies methods to work with RSA encryption in a convenient way in the whole project. At first the listing for the initial common node setup is shown:

[2]http://flask.pocoo.org/

```
app = Flask(__name__)
app.config['id']  = id
keyPair = dpki.getRSAKeypair()
app.config['privKey'] = keyPair["privKey"]
app.config['pubKey'] = keyPair["pubKey"]
```

Listing 5.1: Initialization of a participating node

The code for setting up a server is created with the ability to start multiple instances simply through calling the above method several times. When the port number is specified, this is even possible on the same machine. The supplied id should be unique in the overall network to identify the node. Afterwards a RSA-keypair is created with the previously mentioned helper library. Those keys are then saved in the configuration of the specific server for later usage.

```
data = {}
data["nodeid"] = app.config["id"]
#Other common X.509 fields omitted
data["encryptedKey"] = dpki.encryptPubKeyWithRegKey
        (app.config['pubKey'], reqData['regkey']).hex()
data["revoked"] = False

json_data = json.dumps(data)

requests.post(reqData['valUrl']+"/api/register",
        data=json_data)
```

Listing 5.2: Initiate the registration of a client

In Listing 5.2 we can see the method which is called for initiating the registration process at a known validator node. Therefore a POST request has to be sent (in this case via a GUI), which supplies the url of the validator node via 'valUrl' in the body of the request and the registration public key in 'pubKey', to encrypt public keys of the devices. Afterwards a JSON object is created to submit the specified identification data for the new chain participator. A client node shares the same initial startup as the common node, as it is a superset in terms of functionality.

```
data = request.get_json(force=True)
#File opening amd reading skipped
entryToValidate = {}
entryToValidate["nodeid"] = data["nodeid"]
#Other common X.509 fields omitted
```

```
6  entryToValidate["publicKey"] = dpki.
7  decryptPubKeyWithRegKey
8          (regprivkey, bytes.
9          fromhex(data["encryptedKey"])),
10 entryToValidate["revoked"] = data["revoked"]
11
12 candidatePath = "candidates/"+data["nodeid"]+'.json';
13
14 #Saving data in files skipped.
```

Listing 5.3: Endpoint for registering a new bearing candidate

Listing 5.3 shows the method which gets activated by a registration request. Following the information extraction and decryption from the JSON, the identifying information is saved to a shared folder, ready for initiating a voting process.

```
1  #Load all the bearing candidates from pool
2  for filename in glob.glob("candidates/*.json"):
3          partialContent = open(filename, 'r')
4                  .read().replace("\n","")
5          newBlockContent.append(partialContent);
6
7  #Lines for loading old chain files omitted
8
9  newBlockHeader["version"] = "1"
10 newBlockHeader["time"] = str(datetime.now())
11 newBlockHeader["previousBlockHash"] =
12         hashlib.sha256(prevHash.encode("utf-8"))
13         .hexdigest()
14 #Placeholderoperation as it is just needed for
15 #weak clients which are not monitored in this test
16 newBlockHeader["merkleRootHash"] =
17         hashlib.sha256(b'Hello World').hexdigest()
18
19 newBlock["header"] = newBlockHeader
20 newBlock["content"]  = newBlockContent
21
22 blockchain.append(newBlock)
23 blockchainheader.append(newBlockHeader)
```

Listing 5.4: Create a block and add it to the chain

The shown code Listing 5.4 is responsible for forming a new block. Loading existing data from files is omitted in this listing. In this Proof of Concept it is triggered by a

POST-Command. In a real world application this would happen automatically after a threshold value of bearing devices is reached. This threshold should be chosen carefully as it has an immediate effect on the header size and therefore how fast verification can happen on weak clients as shown in Subsubsection 4.2.1.1. On the other hand it affects incorporation speed of new devices and how fast they are available for communication. A specific voting process is skipped in this example as it would very much depend on the requirements of the company actually implementing the architecture. Afterwards it is checked if a local copy of the Blockchain exists and if not, a new one is created. An extra file for just the headers of the chain is created to make exchange easier. Afterwards all candidates are read and appended to the according files. The header created is a valid one, just the Merkle Root is a mock implementation as it is not used in this PoC.

5.2.2 Authenticated Encryption with Associated Data

In Python there is a library available (python-aead)[3], which handles the AEAD approach through AES cipher with 128 bit in combination with SHA-256 for integrity checks. It is used in this implementation to encrypt slices of load data with different resolutions. To identify those afterwards, specific header information can be added.

```
1 cryptor = AEAD(AEAD.generate_key())
2 ct = cryptor.encrypt(bytes(body, "utf-8"),
3 bytes(header, "utf-8"))
4 cryptor.decrypt(ct, bytes(header, "utf-8"))
```

Listing 5.5: Apply AEAD to the data

The listing above shows how this can be performed using the supplied library. At first a helper method is called to generate a secure key. Afterwards the bytes of the body are encrypted and the header information is attached as unchangeable data. Decryption can happen in the same way with a properly initialized cryptor object (in terms of the key) and when the according unaltered associated data is in place.

[3]https://github.com/Ayrx/python-aead

5.2.3 Multi Resolution Conditional Access

To perform a multi resolution conditional access approach, a basic Haar-Wavelet implementation is done. Because of the simplicity of the Haar-Wavelet in this application a self created implementation is done, without relying on a third party library. The according code can be seen in the following listing:

```
1  ...
2  bands.lowband = [];
3  bands.highband = [];
4  for (i, item) in enumerate(values):
5          if (2*i+1) > (len(values)-1):
6                  return bands;
7
8          highband = values[2*i] - values[2*i+1];
9          lowband = values[2*i] + highband;
10
11         bands.lowband.append(lowband);
12         bands.highband.append(highband);
```

Listing 5.6: Basic Haar Wavelet implementation

As one can see, at first there is an object created to hold the respective values of the high and the low band. Verification happens through testing if the average value stays the same. Afterwards those values are processed by the AEAD algorithm and can be sent to an aggregator from a client.

5.2.4 Aggregation

As aggregation in this context is simply averaging of values, a specific implementation is skipped. It makes especially few sense to implement this without a real world application, missing a solid basis for a decision if aggregation should happen.

5.3 Discussion

This Proof of Concept is reduced to a very basic set of functionality to ensure a focus on the most important parts. Things like SPV and Smart Contracts, as well as aggregation

have been omitted to lower complexity and especially implementation of aggregation is, in terms of programming, not demanding, but needs a solid basis of decision rules when performed and when not. Extending it to have the mentioned parts would be the next step. In terms of Blockchain technology, building upon a blueprint basis like Hydrachain[4] or Hyperledger Fabric[5] is very much recommended, as those have industry backing and a considerable amount of supporters in the open source community, which can be seen on their respective Github pages.

[4]https://github.com/HydraChain/hydrachain
[5]https://github.com/hyperledger/fabric

6

Conclusion

A lot of research has been done focusing on certain subareas of Smart Grids, like specific technologies to secure parts of this new communication layer in energy networks. To the best knowledge of the author, little research has been done related to the overall interplay and viable combinations of a broader group of technologies.

The overall goal is to provide an architectural approach with a broader group of technologies which is in line with all identified requirements with special respect to privacy and security. Combining privacy and security preserving measures gives valuable insight in setting up an architecture, which can be applied in a real world scenario. The proposed architecture incorporates security as well as privacy preserving aspects and uses state-of-the-art technology to comply with technological as well as with legal requirements, together with taking several participating devices like AMIs, third parties and aggregators into account. Additionally several legal documents for the United States and the European Union were screened and restrictions, which are implied by the used technologies were assessed. Possible interconnections were shown to create a well fitting architecture out of existing components. These components all have proven their applicability in industry and daily usage in other environments, but to the best knowledge of the author not in this combination.

A Proof of Concept implementation has shown the viability of the proposed architecture and how all the approaches can be implemented using the Python programming language. This is especially a good fit, because it is wide spread and available for a big amount of devices ranging from embedded ones to fully fletched servers creating a

heterogeneous environment.

A further new aspect of this work is the usage of blockchains as a distributed identity storage. Through using this technology and the inherent decentral approach therein, it is possible to add further stability and reliability to the grid. The Blockchain itself has proven itself in a broad range of applications, with Bitcoin[1] and Ethereum[2] as their probably most popular representatives. This combined with the opportunity to get rid of CAs and added benefit in terms of SPV and Smart Contracts enables further possibilities for the Smart Grid. Especially as there is no single point of failure left in the system, at least if the implementation of the architecture utilizes more than one validation server. Afterwards no further connection is necessary to any validation instance if a full node is concerned. The only exception would be if a certificate gets invalid, for instance if a private key gets exposed. Identity checks can be performed by each participator of the network on its own, so it gets extremely hard for an attacker to control bigger portions of the network. Therefore the grid benefits from a remarkable improvement in terms of resilience against failures, as every single node has gained a new level of independence.

A weak client can contact any trusted full node, regardless if validation or usual client, to request advice and follow the SPV approach, so weak clients need at least a trustworthy device to communicate with. There is no full trust required as the client can verify the information of the other node via its Merkle Root. If the hash does not match the expected outcome, the client could simply contact another participator and send an alert elsewhere. Areas of improvement would be the usage of faster homomorphic ciphers as for instance the Pailler-Cipher to be able to have a quicker communication in place. Nevertheless using Pailler would be possible because of the suspected transmission time of 15 minutes per household in the United States, the EU and Austria. With AES-128 and SHA-256, widely acknowledged standards are in place, as well as their combination to AEAD. Asymmetric cryptosystems like ECC should be enforced for having the best speed benefits present, while using the smallest possible key size. The security of the DPKI in place is highly dependent on the amount of *Validation Servers* available. The more the better for the system, as it is secure as long as the

[1] www.bitcoin.org

[2] www.ethereum.org

decision threshold of DPOS (usually 2/3 of the witnesses) is made up of trustworthy devices. The incorporation process is another improvable section, but has to deal with the same shortcomings as every initial identity verification known from issuing common certificates for HTTPS communication in the Internet.
The most likely real world application for the architecture is inside a virtual power plant (VPP). This emerging infrastructures were defined recently and can be seen as a power plant from the outside world. As those are new and smaller scaled than the overall grid, they can act as an optimal Proof of Concept area for first real world applications.
Because of the decentral approach of the Blockchain, it would be possible to extend the network easily. Even if there is no big global or network wide Blockchain desired. Similar to a Federated Trust Model, the Virtual Power Plants could form Blockchain Domains by themselves and could then cross certify them. A possible approach for doing so would be to submit the blockchain header from domain to domain (or from VPP to VPP). This would even be possible across each domain of a grid because of the considerably small size of the different header files. Whenever inter-domain communication is desired, a communication endpoint of another domain could seek support from any full node of the initiating domain to verify a blockchain entry. This is as secure as many full nodes are available per domain. Figure 6.1 illustrates that inter domain approach.

A viable architectural approach has been proposed which takes all known stakeholders into account and is ready to be used. Performance and legal requirements were taken into account as well as technical ones.

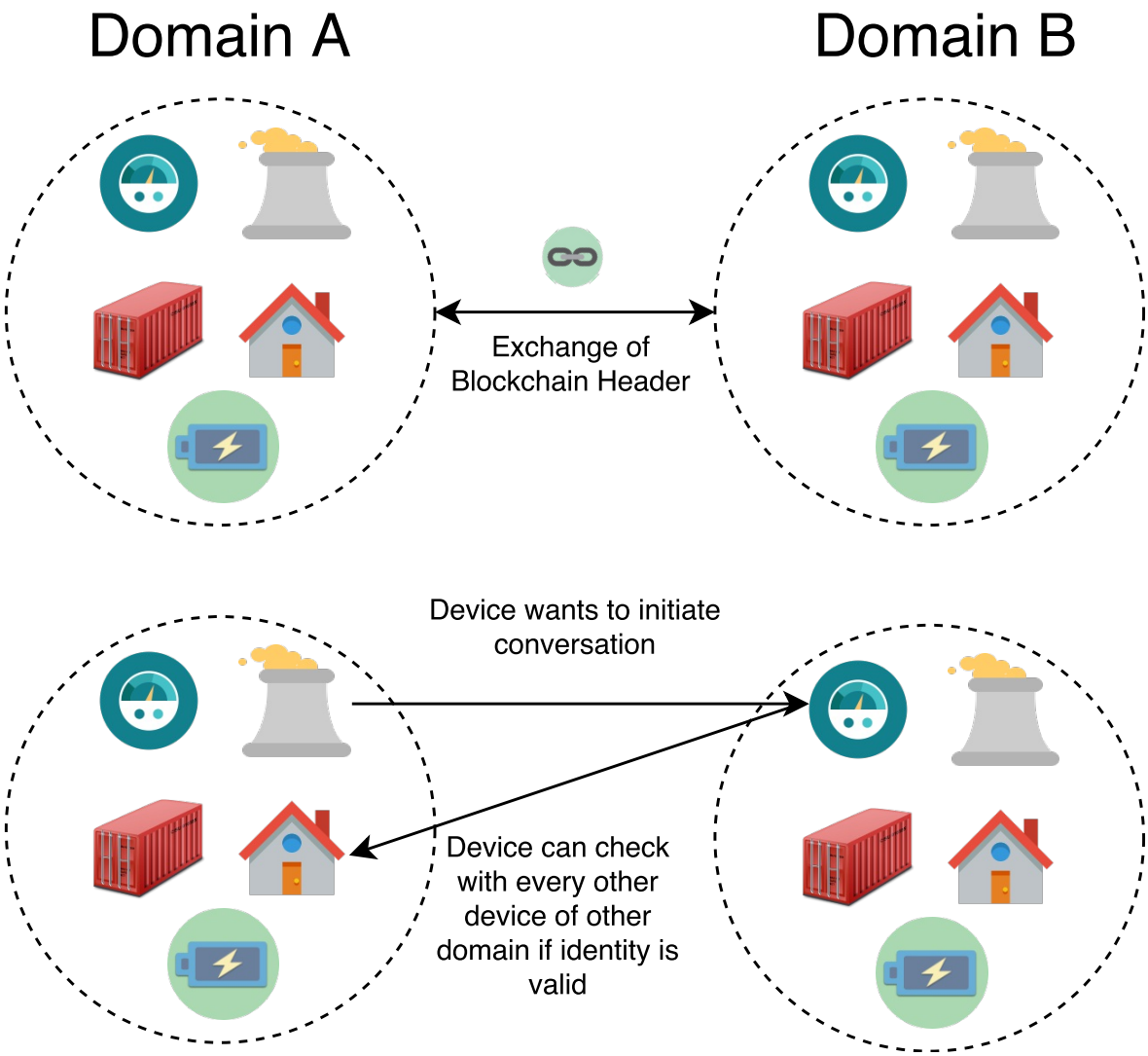


Figure 6.1: Proposed inter domain communication

Bibliography

[1] J. Bruinenberg, L. Colton, E. Darmois, J. Dorn, J. Doyle, O. Elloumi, H. Englert, R. Forbes, J. Heiles, P. Hermans *et al.*, "Cen-cenelec-etsi smart grid coordination group smart grid reference architecture," *CEN, CENELEC, ETSI, Tech. Rep*, 2012.

[2] D. Engel and G. Eibl, "Multi-resolution load curve representation with privacy-preserving aggregation," in *Innovative Smart Grid Technologies Europe (ISGT EUROPE), 2013 4th IEEE/PES*. IEEE, 2013, pp. 1–5.

[3] G. Ritzer and N. Jurgenson, "Production, consumption, prosumption the nature of capitalism in the age of the digital prosumer," *Journal of consumer culture*, vol. 10, no. 1, pp. 13–36, 2010.

[4] M. Lisovich and S. Wicker, "Privacy concerns in upcoming residential and commercial demand-response systems," *IEEE Proceedings on Power Systems*, vol. 1, no. 1, pp. 1–10, 2008.

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[6] C. Ellison and B. Schneier, "Ten risks of pki: What you're not being told about public key infrastructure," *Comput Secur J*, vol. 16, no. 1, pp. 1–7, 2000.

[7] J. A. Buchmann, E. Karatsiolis, and A. Wiesmaier, *Introduction to public key infrastructures*. Springer Science & Business Media, 2013.

[8] P. Rogaway, "Authenticated-encryption with associated-data," in *Proceedings of*

the 9th ACM conference on Computer and communications security. ACM, 2002, pp. 98–107.

[9] C. D. Peer, D. Engel, and S. B. Wicker, "Hierarchical key management for multi-resolution load data representation," in *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on.* IEEE, 2014, pp. 926–932.

[10] S. Wicker and R. Thomas, "A privacy-aware architecture for demand response systems," in *System Sciences (HICSS), 2011 44th Hawaii International Conference on.* IEEE, 2011, pp. 1–9.

[11] M. E. Whitman and H. J. Mattord, *Principles of information security.* Course Technology Ptr, 2011.

[12] D. Pudjianto, C. Ramsay, and G. Strbac, "Virtual power plant and system integration of distributed energy resources," *IET Renewable Power Generation*, vol. 1, no. 1, pp. 10–16, 2007.

[13] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard.* Springer Science & Business Media, 2013.

[14] B. Schneier, *Applied cryptography: protocols, algorithms, and source code in C.* John Wiley & Sons, 2007.

[15] Y. Kumar, R. Munjal, and H. Sharma, "Comparison of symmetric and asymmetric cryptography with existing vulnerabilities and countermeasures," *International Journal of Computer Science and Management Studies*, vol. 11, no. 03, 2011.

[16] J. Lopez and R. Dahab, "An overview of elliptic curve cryptography," 2000. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.2771

[17] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Secure hash standard (shs)," *Federal Information Processing Standards (FIPS) Publications (PUBS)*, 2015.

[18] R. C. Merkle, "Protocols for public key cryptosystems," in *Security and Privacy, 1980 IEEE Symposium on.* IEEE, 1980, pp. 122–134.

[19] H. Krawczyk, R. Canetti, and M. Bellare, "Hmac: Keyed-hashing for message authentication," 1997. [Online]. Available: http://dl.acm.org/citation.cfm?id=RFC2104

[20] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet x.509 public key infrastructure certificate and certificate revocation list profile," Internet Requests for Comments, RFC Editor, RFC 5280, May 2008. [Online]. Available: https://tools.ietf.org/rfc/rfc5280.txt

[21] Microsoft Corporation. Microsoft trusted root certificate: Program requirements. Accessed: March 28th, 2017. [Online]. Available: https://technet.microsoft.com/en-us/library/cc751157.aspx

[22] T. Baumeister, "Adapting pki for the smart grid," in *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*. IEEE, 2011, pp. 249–254.

[23] NIST Smart Grid, "Introduction to guidelines for smart grid cyber security," *Guideline, Sep*, 2010. [Online]. Available: https://www.nist.gov/sites/default/files/documents/smartgrid/nistir-7628_total.pdf

[24] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1999, pp. 223–238.

[25] V. Buterin. On public and private blockchains. Accessed: March 28th, 2017. [Online]. Available: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/

[26] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, 2012. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf

[27] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," *Operating Systems Review*, vol. 33, pp. 173–186, 1998. [Online]. Available: http://people.eecs.berkeley.edu/~kubitron/cs162/hand-outs/BFT.pdf

[28] D. Larimer, "Delegated proof-of-stake consensus," Bitshares, Tech. Rep., 2014, accessed: March 28th, 2017. [Online]. Available: https://bitshares.org/technology/delegated-proof-of-stake-consensus/

[29] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, no. 9, 1997. [Online]. Available: http://ojphi.org/ojs/index.php/fm/article/view/548/469

[30] European Commission, "Commission recommendation," *Official Journal of the European Union*, vol. 148, 2012.

[31] Austrian National Council, "Elektrizitätswirtschafts- und organisationsgesetz," 2010. [Online]. Available: https://www.ris.bka.gv.at/GeltendeFassung/Bundesnormen/20007045/ElWOG%202010%2c%20Fassung%20vom%2028.03.2017.pdf

[32] U. S. Government, "United states code," *Federal Regulation and Development of Power*, 2011. [Online]. Available: https://www.law.cornell.edu/uscode/text/16/chapter-12

[33] T. A. Felix W. Ortiz, William Colton. Assembly bill a4223. [Online]. Available: https://www.nysenate.gov/legislation/bills/2017/A4223

[34] C. Allen, A. Brock, V. Buterin, J. Callas, D. Dorje, C. Lundkvist, P. Kravchenko, J. Nelson, D. Reed, M. Sabadello, G. Slepak, N. Thorp, and H. T. Wood. (2015,) Decentralized public key infrastructure. [Online]. Available: http://www.weboftrust.info/downloads/dpki.pdf

[35] D. Maimut and R. Reyhanitabar, "Authenticated encryption: Toward next-generation algorithms," *IEEE Security & Privacy*, vol. 12, no. 2, pp. 70–72, 2014.

[36] P. Svenda, "Basic comparison of modes for authenticated-encryption (iapm, xcbc, ocb, ccm, eax, cwc, gcm, pcfb, cs)," 2016. [Online]. Available: https://www.fi.muni.cz/~xsvenda/docs/AE_comparison_ipics04.pdf

[37] Lets Encrypt. Let's encrypt - how it works. Accessed: March 29th, 2017. [Online]. Available: https://letsencrypt.org/how-it-works/

[38] Stackoverflow. Developer survey results 2016. Accessed: June 15th, 2017. [Online]. Available: https://insights.stackoverflow.com/survey/2016

List of Abbreviations

AEAD Authenticated Encryption with Attached Data

AES Advanced Encryption Standard

AMI Advanced Metering Infrastructure

CA Certification Authority

CBC Cipher Block Chaining

CCM Counter with CBC-MAC

DER Distributed Energy Resource

DPKI Decentralized Public Key Infrastrucure

DPOS Delegated Proof of Stake

ECC Elliptic Curve Cryptography

MAC Message Authentication Code

NILM Non-Intrusive Load Monitoring

PBFT Practical Byzantine Fault Tolerance

PKI Public Key Infrastructure

POC Proof Of Concept

RSA Rivest, Shamir and Adleman

SGAM Smart Grid Architecture Model

SHA Secure Hash Algorithm

SPV Simplified Payment Verification

VPP Virtual Power Plant

Appendix

Herein supporting information is provided for this thesis, for reference and clarification purposes.

A

Proof of Concept Code

This Chapter contains the code necessary for setting up the Validation Server and different clients in a DPKI with certain respect to the Smart Grid as Python implementations. At first the code of the DPKI helper script is shown which has some basic methods for handling operations in the DPKI environment:

```
1  import os.path
2  from cryptography.hazmat.backends import default_backend
3  from cryptography.hazmat.primitives import serialization
4  from cryptography.hazmat.primitives.asymmetric import rsa
5  from cryptography.hazmat.primitives.asymmetric import padding
6  from cryptography.hazmat.primitives import hashes
7  import re
8
9  def hasChain(id):
10         return os.path.isfile(id+"/chain.json")
11
12 def encryptPubKeyWithRegKey(pubKey, regKey):
13
14         #Workaroud to get the library to load the keys
15         regKey = regKey.replace("-----BEGIN PUBLIC KEY-----", "")
16         regKey = regKey.replace("-----END PUBLIC KEY-----", "")
17         regKey = re.sub("(.{64})", "\\1\n", regKey, 0, re.DOTALL)
18         regKey = "-----BEGIN PUBLIC KEY-----\n" + regKey
19         regKey = regKey + "\n-----END PUBLIC KEY-----"
20
21         encryptionKey = serialization.load_pem_public_key
22                 (bytes(regKey,'utf-8'), backend = default_backend())
23
24         encryptedKey = encryptionKey.encrypt(
25         bytes(pubKey,'utf-8'),
26         padding.OAEP(
27         mgf=padding.MGF1(algorithm=hashes.SHA1()),
28         algorithm=hashes.SHA1(),
29         label=None
30         )
31         )
32
```

```
33          return encryptedKey
34
35  def getRSAKeypair():
36          return getDefinedRSAKeypair(2048)
37
38  def getLongRSAKeypair():
39          return getDefinedRSAKeypair(4096)
40
41  def getDefinedRSAKeypair(length):
42          keypair = rsa.generate_private_key(
43          public_exponent=65537,
44          key_size=length,
45          backend=default_backend()
46          )
47          keys = {}
48          keys["privKey"] = keypair.private_bytes(
49          encoding=serialization.Encoding.PEM,
50          format=serialization.PrivateFormat.TraditionalOpenSSL,
51          encryption_algorithm=serialization.NoEncryption()
52          ).decode("utf-8").replace("\n", "")
53
54          keys["pubKey"] = keypair.public_key().public_bytes(
55          encoding=serialization.Encoding.PEM,
56          format=serialization.PublicFormat.SubjectPublicKeyInfo
57          ).decode("utf-8").replace("\n", "")
58
59          return keys
60
61  def decryptPubKeyWithRegKey(regPrivKey, encryptedPubKey):
62          #Workaroud to get the library to load the keys
63          regPrivKey = regPrivKey
64                  .replace("-----BEGIN RSA PRIVATE KEY-----", "")
65          regPrivKey = regPrivKey
66                  .replace("-----END RSA PRIVATE KEY-----", "")
67          regPrivKey = re
68                  .sub("(.{64})", "\\1\n", regPrivKey, 0, re.DOTALL)
69          regPrivKey = "-----BEGIN RSA PRIVATE KEY-----\n"
70                  + regPrivKey
71          regPrivKey = regPrivKey
72                  + "\n-----END RSA PRIVATE KEY-----"
73
74          private_key = serialization.load_pem_private_key(
75          bytes(regPrivKey,'utf-8'),
76          password=None,
77          backend=default_backend()
```

```
78          )
79
80          decryptedKey = private_key.decrypt(
81          encryptedPubKey,
82          padding.OAEP(
83          mgf=padding.MGF1(algorithm=hashes.SHA1()),
84          algorithm=hashes.SHA1(),
85          label=None
86          )
87          )
88
89          return decryptedKey.decode("utf-8").replace("\n", "")
```

Listing A.1: DPKI helper script

The following script is used to setup a client in the DPKI environment:

```
1
2  from flask import Flask
3  from datetime import datetime
4  from flask import render_template
5  from flask import request
6  import requests
7  import dpki
8  import json
9
10 def start_common_node(portNumber, id):
11          app = Flask(__name__)
12          app.config['id']  = id
13
14          keyPair = dpki.getRSAKeypair()
15
16          app.config['privKey'] = keyPair["privKey"]
17          app.config['pubKey'] = keyPair["pubKey"]
18
19 @app.route('/')
20 @app.route('/home')
21 def home():
22          """Renders the home page."""
23          return render_template(
24          'index.html',
25          title='Home Page for node ' + app.config['id'],
26          year=datetime.now().year,
27          )
28
29 @app.route('/contact')
```

```
30 def contact():
31         """Renders the contact page."""
32         return render_template(
33         'contact.html',
34         title='Contact',
35         year=datetime.now().year,
36         message='Your contact page.'
37         )
38
39 @app.route('/about')
40 def about():
41         """Renders the about page."""
42         return render_template(
43         'about.html',
44         title='About',
45         year=datetime.now().year,
46         message='Your application description page. ID: '
47                 + app.config['id']
48                 + " and the public key is "
49                 + app.config['pubKey']
50         )
51
52 @app.route('/backend/initiateregistration', methods = ['POST'])
53 def initiateregistration():
54
55         reqData = request.get_json(force=True)
56         data = {}
57         data["nodeid"] = app.config["id"]
58         data["version"] = "version"
59         data["validity"] = "1.1.2020"
60         data["issuer"] = "Cornell"
61         data["subject"] = "A smart device"
62         data["encryptedKey"] = dpki
63                 .encryptPubKeyWithRegKey(app.config['pubKey'],
64                 reqData['regkey']).hex()
65         data["revoced"] = False
66
67         json_data = json.dumps(data)
68
69         r = requests.post(reqData['valurl']
70                 +"/api/register", data=json_data)
71
72         return r.text
73
```

```
74          app.run(host='localhost', port = portNumber)
```

Listing A.2: Common Node in Python

This script is the corresponding code for setting up a validator node:

```
1
2  """
3  The flask application package.
4  """
5
6  from flask import Flask
7  from datetime import datetime
8  from flask import render_template
9  from flask import request
10 from datetime import datetime
11 import hashlib
12 import dpki
13 import json
14 import os, shutil
15 import glob
16 import time
17
18 def start_common_validator(portNumber, id):
19          app = Flask(__name__)
20          app.config['id']  = id
21
22          keyPair = dpki.getRSAKeypair()
23
24          app.config['privKey'] = keyPair["privKey"]
25          app.config['pubKey'] = keyPair["pubKey"]
26
27          @app.route('/')
28          @app.route('/register')
29          def register():
30          """Renders the home page."""
31          return render_template(
32          'index.html',
33          title='Register Page for validator ' + app.config['id'],
34          id=app.config['id'],
35          genesisValidator = app.config['id']
36                  == "68162d43-4d41-43b7-b8aa-f22325ff10b2"
37          )
38
39          @app.route('/contact')
40          def contact():
```

```
41              """Renders the contact page."""
42              return render_template(
43              'contact.html',
44              title='Contact',
45              message='Your contact page.',
46              id=app.config['id'],
47              genesisValidator = app.config['id']
48                      == "68162d43-4d41-43b7-b8aa-f22325ff10b2"
49              )
50
51      @app.route('/setup')
52      def setup():
53              """Renders the about page."""
54              return render_template(
55              'setup.html',
56              title='Setup',
57              id=app.config['id'],
58              message='Setup for ' + app.config['id'],
59              genesisValidator = app.config['id']
60                      == "68162d43-4d41-43b7-b8aa-f22325ff10b2",
61              chainExists = dpki.hasChain(app.config['id'])
62              )
63
64      @app.route('/api/register', methods = ['POST'])
65      def apiregister():
66              regprivkey=""
67              data = request.get_json(force=True)
68
69              #Grab private key for registration:
70              with open("bearer/private/" +
71                      data["nodeid"] + ".txt", 'r') as myfile:
72              regprivkey=myfile.read().replace('\n', '')
73
74
75
76              entryToValidate = {}
77
78              entryToValidate["nodeid"] = data["nodeid"]
79              entryToValidate["version"] = data["version"]
80              entryToValidate["validity"] = data["validity"]
81              entryToValidate["issuer"] = data["issuer"]
82              entryToValidate["subject"] = data["subject"]
83              entryToValidate["publicKey"] = dpki.
84                      decryptPubKeyWithRegKey(regprivkey,
85                      bytes.fromhex(data["encryptedKey"])),
```

```
86                 entryToValidate["revoced"] = data["revoced"]
87
88                 candidatePath = "candidates/"+data["nodeid"]+'.json';
89
90                 if not os.path.exists(os.path.dirname(candidatePath)):
91                         os.makedirs(os.path.dirname(candidatePath))
92
93                 with open(candidatePath, 'w+') as outfile:
94                 json.dump(entryToValidate, outfile)
95
96                 return "Registered for voting"
97
98         @app.route('/api/generateKeypair', methods = ['POST'])
99                 def generateKeypair():
100                """Renders the about page."""
101                data = request.get_json(force=True)
102
103                keyPair = dpki.getLongRSAKeypair()
104
105                pubKeyFile = "bearer/public/" + data["bearerId"] + ".txt"
106                if not os.path.exists(os.path.dirname(pubKeyFile)):
107                        os.makedirs(os.path.dirname(pubKeyFile))
108
109                with open(pubKeyFile,"w+") as f:
110                f.write(keyPair["pubKey"])
111
112                privKeyFile = "bearer/private/" + data["bearerId"] + ".
                      txt";
113
114                if not os.path.exists(os.path.dirname(privKeyFile)):
115                        os.makedirs(os.path.dirname(privKeyFile))
116
117                with open(privKeyFile,"w+") as f:
118                f.write(keyPair["privKey"])
119
120
121                return keyPair["pubKey"]
122
123        @app.route('/api/formblock', methods = ['POST'])
124        def formblock():
125                #Skip voting procedure for now as
126                #we assue that every blockshould be created
127
128                #Prepare filepath
129                chainpath = "chain"+app.config["id"]+"/chain.json";
```

```
130                 chainheaderpath = "chain"+
131                         app.config["id"]+"/chainheader.json";
132
133                 if not os.path.exists(os.path.dirname(chainpath)):
134                         os.makedirs(os.path.dirname(chainpath))
135                 if not os.path.exists(os.path.dirname(chainheaderpath)):
136                         os.makedirs(os.path.dirname(chainheaderpath))
137
138                 #Be sure the file exists
139                 open(chainpath, 'a').close()
140                 open(chainheaderpath, 'a').close()
141
142
143                 newBlockContent = []
144                 for filename in glob.glob("candidates/*.json"):
145                         partialContent = open(filename, 'r')
146                                 .read().replace("\n","")
147                         for x in range(0, 10000):
148                         newBlockContent.append(partialContent);
149
150
151
152                 with open(chainpath) as data_file:
153                 try:
154                         blockchain = json.load(data_file)
155                 except ValueError:
156                         blockchain = []
157
158                 with open(chainheaderpath) as data_file:
159                         try:
160                                 blockchainheader = json.load(data_file)
161                         except ValueError:
162                                 blockchainheader = []
163
164
165                 if len(blockchain) == 0:
166                         prevHash = "Hello"
167                 else:
168                         prevHash = str(blockchain[-1])
169
170                 newBlockHeader = {}
171                 newBlockHeader["version"] = "1"
172                 newBlockHeader["time"] = str(datetime.now())
173                 newBlockHeader["previousBlockHash"] = hashlib
174                         .sha256(prevHash.encode("utf-8")).hexdigest()
```

```
175                 #Placeholderoperation as it is just needed
176                 #for weak clients which are not monitored in this test
177                 newBlockHeader["merkleRootHash"] = hashlib
178                         .sha256(b'Hello World').hexdigest()
179
180                 newBlock = {}
181                 newBlock["header"] = newBlockHeader
182                 newBlock["content"]  = newBlockContent
183
184                 for x in range(0, 100):
185                         blockchain.append(newBlock)
186                         blockchainheader.append(newBlockHeader)
187
188                 with open(chainpath, 'w') as outfile:
189                         json.dump(blockchain, outfile)
190
191                 with open(chainheaderpath, 'w') as outfile:
192                         json.dump(blockchainheader, outfile)
193
194                 #shutil.rmtree('candidates')
195
196                 return "Block formed, and candidates cleaned"
197
198         @app.route('/api/validate', methods = ['POST'])
199         def validate():
200                 #read every entry and check for
201                 start_time = time.time()
202                 #data = request.get_json(force=True)
203                 #id = data["validatorId"]
204                 #key = data["validatorPubKey"]
205
206
207                 chainpath = "chain"
208                         +app.config["id"]+"/chain.json";
209                 with open(chainpath) as data_file:
210                         try:
211                                 blockchain = json.load(data_file)
212                         except ValueError:
213                                 blockchain = []
214
215                 for block in blockchain:
216                         for information in block["content"]:
217                                 informationJ = json.loads(information)
218                                 if informationJ["publicKey"] == ""
```

```
219                                         and informationJ["revoced"] ==
                                                False:
220                                         pass
221                 #just go through to simulate search
222
223
224
225         return "---␣%s␣seconds␣---"
226                 % (time.time() - start_time)
227
228 app.run(host='localhost', port = portNumber)
```

Listing A.3: Validator Node in Python

With this code segment, AEAD was compared to RSA:

```
1
2  from aead import AEAD
3  import time
4  from cryptography.hazmat.backends import default_backend
5  from cryptography.hazmat.primitives import serialization
6  from cryptography.hazmat.primitives.asymmetric import rsa
7  from cryptography.hazmat.primitives.asymmetric import padding
8  from cryptography.hazmat.primitives import hashes
9  import re
10 import sys
11
12 plainText = "...."
13 additionalData = "...."
14 times = 99
15
16 print("Data␣size␣" + str(sys.getsizeof(plainText))
17         + "Bytes")
18 print("Header␣size␣" + str(sys.getsizeof(plainText))
19         + "Bytes")
20
21 #Performing AEAD
22 start = time.time()
23 for x in range(0, times):
24         cryptor = AEAD(AEAD.generate_key())
25         ct = cryptor.encrypt(bytes(plainText, "utf-8"),
26                 bytes(additionalData, "utf-8"))
27         cryptor.decrypt(ct, bytes(additionalData, "utf-8"))
28 end = time.time()
29 print("Elapsed␣time␣for␣AEAD␣[s]:␣" + str(end - start))
30
```

```
31
32 #Performing RSA
33 start = time.time()
34 for x in range(0, times):
35         privkey = rsa.generate_private_key(
36         public_exponent=65537,
37         key_size=2048,
38         backend=default_backend()
39         )
40
41         pubkey = privkey.public_key();
42
43         ciphertext = pubkey.encrypt(
44         (plainText + additionalData).encode('UTF-8'),
45         padding.OAEP(
46         mgf=padding.MGF1(algorithm=hashes.SHA1()),
47         algorithm=hashes.SHA1(),
48         label=None
49         )
50         )
51
52         plaintext = privkey.decrypt(
53         ciphertext,
54         padding.OAEP(
55         mgf=padding.MGF1(algorithm=hashes.SHA1()),
56         algorithm=hashes.SHA1(),
57         label=None
58         )
59         )
60
61 end = time.time()
62 print("Elapsed time for RSA[s]: "
63         + str(end - start))
```

Listing A.4: AEAD vs. RSA

The next listing shows the complete Haar implementation together with verification steps of identifying the same average:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 class Object(object):
5         pass
6
7         def applyHaar(values):
```

```
8                bands = Object();
9                bands.lowband = [];
10               bands.highband = [];
11               for (i, item) in enumerate(values):
12                       if (2*i+1) > (len(values)-1):
13                               return bands;
14
15                       highband = values[2*i] - values[2*i+1];
16                       lowband = values[2*i] + highband;
17
18                       bands.lowband.append(lowband);
19                       bands.highband.append(highband)
20
21       mu, sigma = 1, 0.25  # mean and standard deviation
22       s = np.random.normal(mu, sigma, 50)
23       plt.plot(s)
24       plt.title('Haar Wavelet example - Input '
25               + str(sum(s) / float(len(s))))
26       plt.ylabel('kwh')
27       plt.xlabel('time[s]')
28       plt.show()
29
30       bands = applyHaar(s);
31
32
33       plt.plot(bands.lowband)
34       plt.title('Haar Wavelet example - After Haar transform '
35               + str(sum(s) / float(len(s))))
36       plt.ylabel('kwh')
37       plt.xlabel('time[s]')
38       plt.show()
```

Listing A.5: Haar wavelet