A blockchain-based local energy market

Giacomo Gritzan, Torben Petrow, Michelle Jakobi, Sibille Knodel, Richard Sethmann
Hochschule Bremen, Flughafenallee 10, 28199 Bremen

As part of the research project Trusted Blockchains for the Open, Smart Energy Grid of the Future (tbiEnergy), one of the objectives is to investigate how a holistic blockchain approach for the realization of a local energy market could be accomplished and how corresponding hardware security mechanisms can be integrated. This paper provides an overview of the implemented prototype and describes the system and its processes.

1. Introduction

In the course of the energy transition initiated in Germany, energy grid operators are facing the challenge to restructure the traditional energy grid based on central power plants to a decentralized and distributed power grid that is based on renewable energy sources. [1]

The centralized structure of the traditional energy grid provided predictable plannability to grid operators [1]. Through the rising numbers of prosumers [11] and dependence on the availability of localized renewable energy sources such as wind and photovoltaic, the administration effort grows dramatically. This results in an increasing demand for systems that can coordinate and document these processes in a tamper-proof and traceable manner. Furthermore, the German energy industry is striving for the development of smart grids that use intelligent measuring systems, named Smart Meter Gateways (SMGWs), to react more precise and faster to loads. [2], [3], [4], [5], [6]

The research project tbiEnergy demonstrates how a local blockchain-based energy market could be implemented to handle the described problems.

This is achieved by using controllable local systems (CLS) in form of ARMv7 based single board computers (CLS boxes) taking hardware security mechanisms into account. The blockchain infrastructure is implemented and operated in the cloud using the blockchain framework EOSIO. In order to integrate operators of generation, consumption and storage plants in regard of German national regulations, an existing central registry for power and gas generation plant data – the Marktstammdatenregister (MaStR) of the German Federal Network Agency – is used and augmented with additional functionalities. Energy suppliers act as intermediaries, operating a set of cloud-based microservices and managing the CLS. This local energy market differentiates itself from classic power exchanges and uses flexibilities and capacities of local participants to trade energy between customers and provide the energy supplier an additional option to regulate their own balancing groups. These balancing groups are formed in order to be able to control the supply and demand of electricity within the grids without having to define direct relationships between entry and exit points. [10]

This paper presents a concept for the realization of a blockchain-based local energy market and gives an overview of the necessary systems, processes and components.

2. Concept of a local energy market

The local energy market, developed within the scope of the research project, is limited to customers of the energy supplier and thus to its balancing groups. Currently, the focus is on energy facilities that do not receive subsidized feed-in rates under the Renewable Energy Sources Act (EEG). The energy supplier operates the local energy market and is responsible for the corresponding systems. The platform enables customers to buy and sell energy and at the same time it offers the energy supplier the possibility to manage its own balancing groups. The management obligations, remain on the side of the energy supplier as the balancing group manager. Discrepancies between the forecasts and the matching of energy supply and demand are compensated by the energy supplier, for which it receives processing fees. In order to avoid the shutdown of power plants, the local energy market was divided into a primary market and a secondary market.

On the primary market, customers can buy or sell energy from renewable energy production or storage facilities. Customers can choose between a passive and an active marketing model. In the case of active marketing, the plants are managed by the energy supplier and the customer receives a fixed payment for the flexibility it offers. On the other hand, if the customer opts for active marketing, offers for timeslots can be placed on the local energy market after the plants has been registered. Before timeslot offers are placed on the market, power generation and consumption forecasts are calculated based on the power plant data from the MaStR and utility's weather-based energy production forecast data. These calculations ensure that the energy consumption of consumers is matched with the offered generation quantity or storage capacity prior to acquisition in order to prevent the acquisition of very large quantities of energy due to incorrect input by end consumers or malicious market participants. The primary market is closed one hour before the timeslot starts and the sold timeslots go to the highest bidder. This is where the secondary market comes in: The forecasts are recalculated

and the initial forecast is compared with the new forecast. In the next step, the utility is given the opportunity to buy timeslots that previously have not been sold in order to prevent shutdown of facilities. In addition, the differences between the first and second forecasts are compared by the utility and could be used to supply its balancing group.

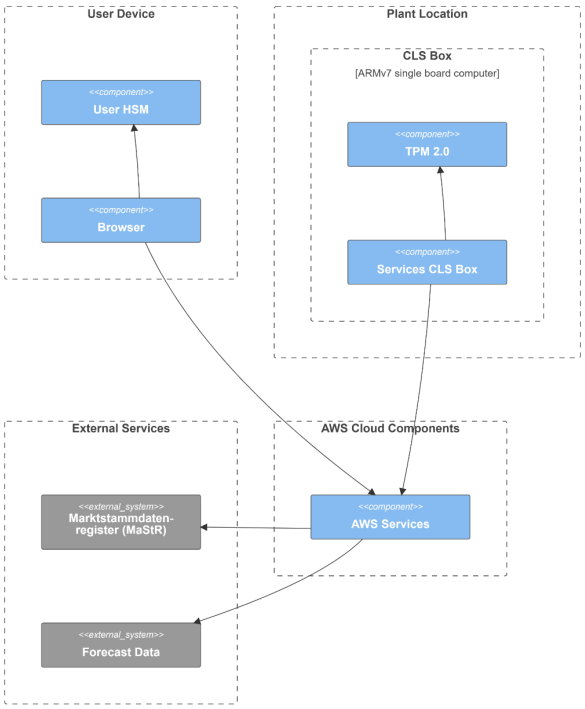

Figure 1 - Architectural Overview

3. Architectural Overview

Figure 1 gives an overview of the designed and prototyped demonstrator. As part of the conceptual design, a microservice approach was developed based on AWS Cloud Services. External services that are utilized are the MaStR of the Bundesnetzagentur and external forecast data e.g., for PV systems and heat pumps. Each plant location is equipped with a CLS box that has a hardware security module (HSM) and is able to communicate with the AWS cloud components using an IoT LTE connection. Interactions between users and the local energy market are processed via a web application. Access to the web application is secured by a user hardware security module. In the following sections, the prototyped components are discussed and an overview of the provided features is given.

4.1 Plant Location / CLS Box

All services on the CLS box are deployed as docker services based on docker-compose files. To allow flexible communication between the service containers they are separated in a docker network. The following subsections describe these various software components running on the CLS box based on *Figure 2*.

4.1.1 CLS-Box-Logic - REST-API

The CLS-Box-Logic is a RESTful web service that is implemented with the Python framework Flask-RESTX. It is designed as the control component of the CLS box. It forms the link between the respective CLS box and the microservice architecture in the AWS cloud. The status of the CLS box can be queried and data from connected devices such as smart meters or CLS devices can be retrieved indirectly via the OpenEMS energy management system (refer section 4.1.4 OpenEMS). In addition, infrastructure services deployable as docker services can be managed. The CLS-Box-Logic ensures that the connection of physically connected CLS boxes and the devices connected to them can be validated before they are added to the local energy market. In addition, it is able to react to events that have been forwarded by the Watcher component of the Blockchain-Connector (refer section 4.1.3 Blockchain-Connector) and process data for example collecting data from OpenEMS or send data to the chain via the Performer component of the Blockchain-Connector.

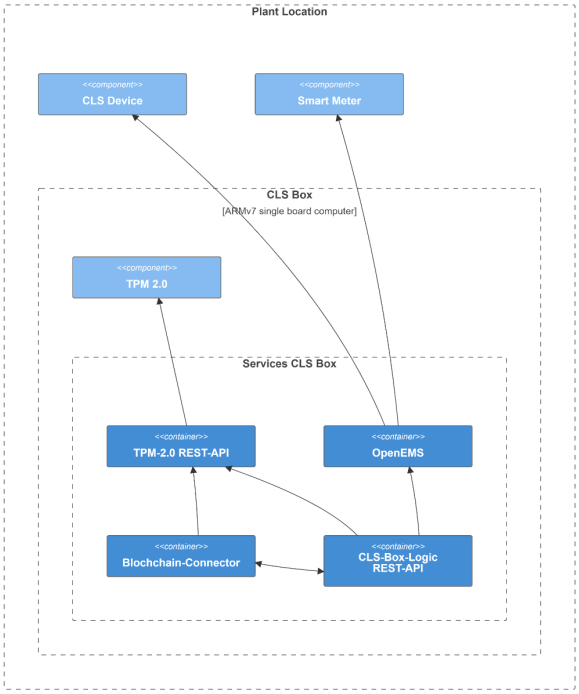

Figure 2 - Plant Location / CLS Box

4.1.2 TPM-2.0 - REST-API

One goal of the project is to secure the key material used for transaction signing on the CLS box. This task is performed by the TPM-2.0 - API. It provides the Trusted Platform Module (TPM) functions for generating key pairs, retrieving public keys and signing transactions via a rest interface secured with AWS Cognito. Currently, a software based TPM is used within the container, which will be replaced by a hardware TPM as part of the further implementation within the scope of the field test.

4.1.3 Blockchain-Connector

The Blockchain-Connector consists of several components that can be used independently. Within the scope of the project, the RESTful web services Watcher and Performer are deployed and utilized on the CLS box.

The Watcher's task is to react to on chain events and forward these events to the CLS-Box-Logic. The CLS-Box-Logic processes an event and decides whether data must be written to the blockchain. In that case, it uses the TPM-2.0 - API to sign the data and then writes it to the chain via the Performer.

As a result, the Watcher and the Performer form the link between the software components of the CLS box and the blockchain.

4.1.4 OpenEMS

"OpenEMS — the Open-Source Energy Management System — is a modular platform for energy management applications. It was developed around the requirements of monitoring, controlling, and integrating energy storage together with renewable energy sources and complementary devices and services like electric vehicle charging stations, heat-pumps, electrolysers, time-of-use electricity tariffs and more." [9]

In context of the project, it allows the development of manufacturer specific modules that can integrate external devices such as smart meters or CLS devices. The data of the connected devices can be retrieved and stored in a local database. The values collected are provided with the help of a RESTful webservice.

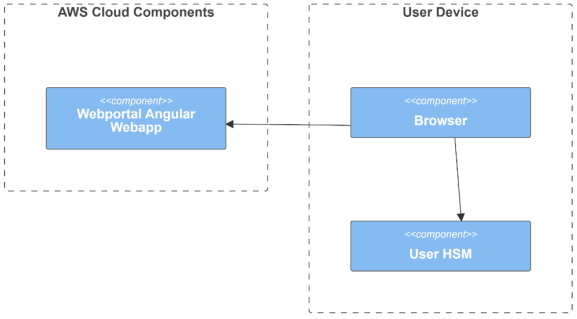

Figure 3 - User Device

5 User Device

The users can interact with the local energy market through the Webportal as shown in *Figure 3*. To authenticate to the Webportal, every participant has a User-HSM that acts as a second factor for the authentication process using the FIDO2 standard and utilizing AWS Cognito in the backend of the Webportal.

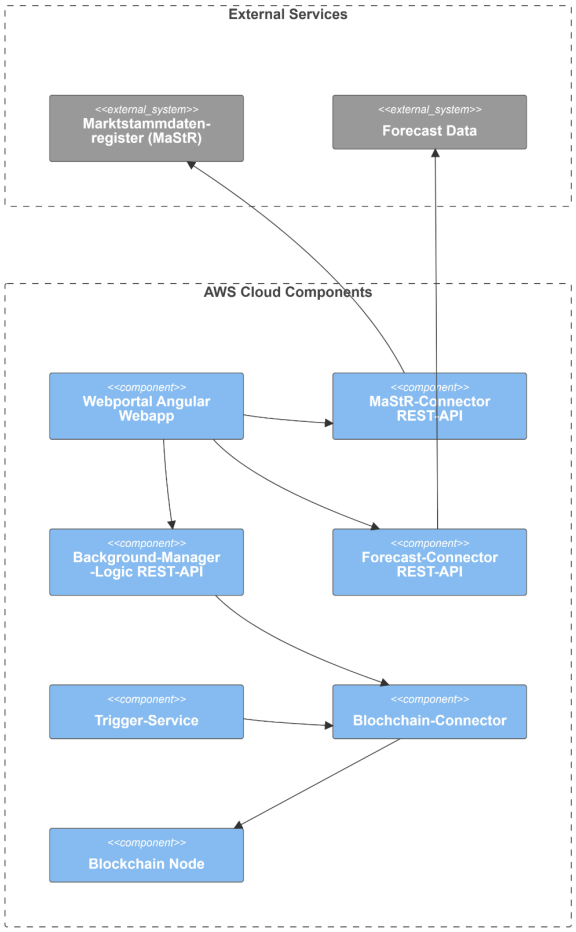

Figure 4 - AWS Cloud Components

6 AWS Cloud Components

The AWS Cloud Components shown in *Figure 4* are based on a set of AWS services. The RESTful webservices and the Webportal are deployed with AWS Fargate, the Trigger Service utilizes AWS Lambda and the Blockchain Nodes are deployed on an AWS -EKS-Cluster. The following subsection will give a description about the functionalities of the individual services.

6.1 RESTful Webservices

The RESTful Webservices are based on the Python framework Flask-RESTX, which is an extension of the Flask framework and integrates e.g., the documentation of interfaces with the help of the so-called OpenAPI specification. It generates OpenAPI documents based on annotations that contain a standardized, language-independent description of the functions provided by the RESTful web services. The OpenAPI document is also used to generate TypeScript clients for the web application and the possibility to display the available functions using Swagger-UI. Each webservice uses the service AWS Cognito for authentication and authorization purposes.

6.1.1 MaStR-Connector

In order to utilize existing databases that store Informations about facilities and their operators, the development of the MaStR -Connector was started. The MaStR contains information about power and gas production plants which must be registered on a mandatory basis and are assigned to clearly identifiable market players. The MaStR provides its data via a SOAP web interface, but this does not allow filtering of all units regarding market actors. The MaStR-Connector was therefore designed as a RESTful webservice that manages synchronized copy of the units of the MaStR, while providing a more flexible way to access the data. Other advantages are the control over the availability of the service and the avoidance of rate limits for data retrieval. The key functionalities of the MaStR-Connector are the synchronization via the MaStR SOAP web interface and the possibility to query units for a specific market actor. The MaStR-Connector retrieves the MaStR data of electricity and gas production plants, market actors such as plant operators, network operators and energy suppliers via the MaStR database and then synchronizes them with its own database. Two methods are used for this purpose:

When initializing the database for the first time, the MaStR Connector populates the database with entries read from XML files that are exported from the MaStR database. After this first initialization, new entries or changes since the last update are retrieved via the SOAP web interface of the MaStR. Subsequently, the MaStR data is parsed from XML into internal data types and stored or updated in a DocumentDB hosted in the AWS cloud environment. These functions are outsourced to Redis jobs to run in the background asynchronously. The current status of the jobs can be inspected by querying corresponding job IDs. The MaStR-Connector allows to query details of the stored units by their MaStR number. Furthermore, all units of a market actor can be retrieved via the MaStR-Connector.

6.1.2 Background-Manager-Logic

Along with the Trigger-Service and the Blockchain-Connector the Background-Manager-Logic represents the management sector within the microservice architecture. In addition to executing background tasks, this area bridges the blockchain network and the other microservices. The RESTful Background-Manager-Logic web service uses an Amazon DocumentDB for data storage. With the help of the collected data foundation of the database, the Background-Manager-Logic can provide the necessary auxiliary functions for the different user roles. For the administrative user, these functions include adding and querying energy-related components from the database and retrieving the status of the service. Users without extended usage rights for the background-manager-logic are given the opportunity to register their own units on the local energy market, place offers and bid on offers. These requests are forwarded to the Performer component of the Blockchain-Connector that executes the corresponding smart contracts to write the data to the blockchain.

6.1.3 Forecast-Connector

The Forecast-Connector is a RESTful web service that processes and provides forecasts and secures the balancing group of the local energy market against manipulations. These forecasts are calculated estimates of the power consumption and production over time periods (timeslots) of 15 minutes for a set of units from the MaStR. The forecast data is created by the energy supplier and uploaded daily to an Amazon Elastic File System (EFS) volume in the AWS cloud. The Forecast-Connector then extracts the forecast data and migrates the contained forecasts and timeslots into its own database. The imported timeslots and their affiliated forecasts can then be retrieved via REST routes.

6.2 Webportal

The Webportal is a responsive web UI based on web application framework Angular, which acts as an interface to the local energy market for consumers, plant operators, the energy supplier as well as administrators. Users are authenticated using credentials and a user HSM. Depending on their authorization level, users are offered different functionalities. Users holding administrative privileges are able to check the status of all associated microservices. They are also able to link physically attached hardware components such as CLS boxes, meters and units in order to provide users access to the local energy market as shown in Figure *5*.

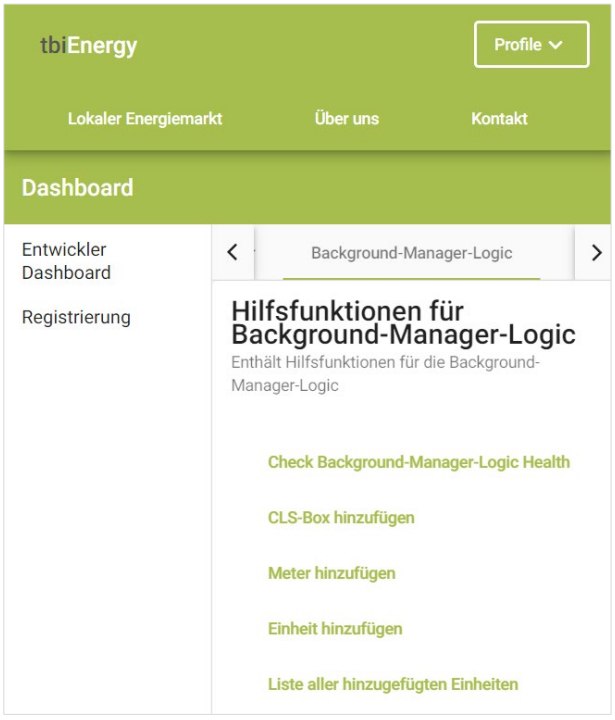

Figure 5 - Developer Dashboard Webportal

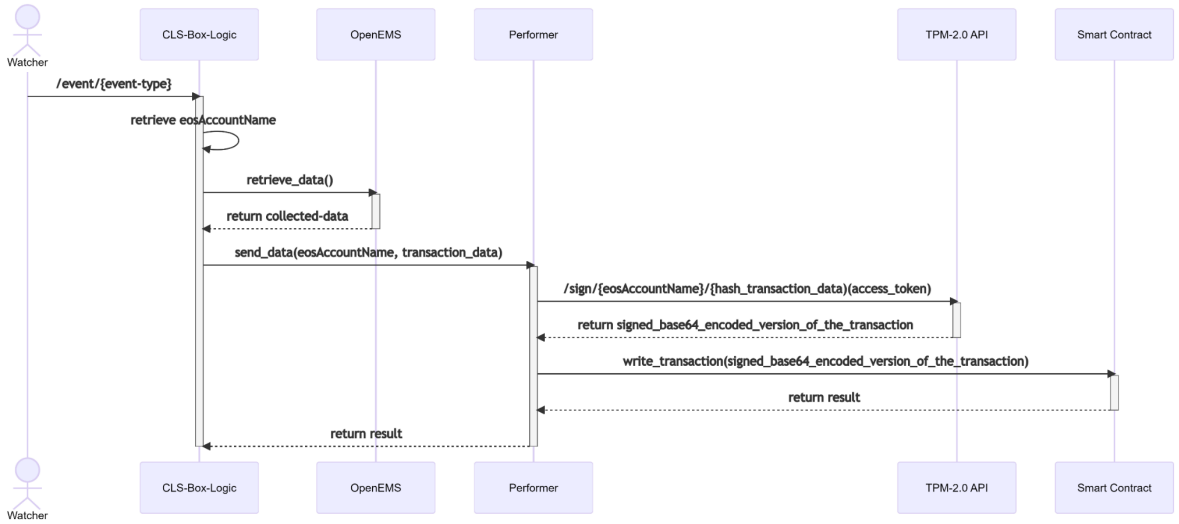

Figure 6 - Process of signing and sending data based on a triggered event

Based on these created links, users are able register their own units and place offers on the local energy market. Also, the energy supplier is able to get an overview over its balancing group. The required functionalities are provided by the web portal using the connected microservices in the backend (Forecast-Connector, Background-Manager-Logic, MaStR-Connector, CLS-Box-Logic instances).

6.3 Trigger-Service

The Trigger-Service is tasked with triggering the execution of functions of other system components under certain conditions. In specified time intervals it requests the synchronization operations of the MaStR-Connector, the update of the Forecast-Connector and generation of events through the Blockchain-Connector. The requests are sent by AWS Lambda functions which in turn are executed by AWS EventBridge rules that check for the preset time periods expiration.

6.4 Blockchain Node

The blockchain used within the project is the open-source platform EOSIO which uses the consensus mechanism “delegated Proof of Stake (dPoS)”. EOSIO executes industrial-scale blockchains with the support of smart contracts and the possibility to build private blockchain networks. [7], [8]

7 Securing the signing of transaction data

Figure *6* describes the process when a specific send_data event is triggered by the Trigger-Service. The Watcher on the CLS box reacts to this event and calls the CLS-Box-Logic that loads the eosAccountName generated within an initial registration process. During the next step the data for the corresponding meter or CLS device is received from the OpenEMS system and sent to the Blockchain Connector component Performer. The Performer sends the hash of the transaction data and the corresponding eosAccountName to the TPM-2.0 API which signs the transaction with the connected TPM 2.0. The signed and encoded version of the transaction is then passed back to the Performer which writes it to the chain. Doing this, the transaction signing process on CLS box has been secured by the utilization of the HSM. The private-key of the keypair initially created within the TPM-2.0 API thus, never leaves the TPM. A malicious attacker would therefore have to gain physical access to the system locations in order to transfer malicious data in the name of an CLS box. Consumption or generation data of the Smart Meters or CLS devices connected to the CLS boxes can thus only be transmitted to the blockchain with the key material generated on the CLS box within the TPM 2.0 REST-API and it is not possible to extract the private keys from the TPM.

8 Conclusion

Within the project, a concept for a local energy market was successfully developed and hardware security could be securely integrated into the CLS box and the Webportal. To validate the technical feasibility, a prototype was created, and its components have been described. During the implementation phase, we learned, that using the blockchain also requires some traditional components, such as RESTful web services and databases, to manage the infrastructure components and process the data before it could be written to the blockchain. Data tampering security has also been implemented using TPM 2.0 on the CLS box and authentication with the user HSM within the web frontend. The presented prototype implements the critical system processes that will be validated in the next steps of the project within a field test. In conclusion, the developed concept of a local energy market and application of hardware security can contribute to the desired energy transition in the German smart grid.

Acknowledgements

The authors would like to thank the German Federal Ministry of Economic Affairs and Climate Action (BMWK) for the funding and all tbiEnergy project partners for the good cooperation.

References

[1] Oliver D. Doleski. Die Energiebranche am Beginn der digitalen Transformation: aus Versorgern werden Utilities 4.0. Oliver D. Doleski Fiduiter Consulting Ottobrunn, Deutschland, 2017, S. 842. ISBN: 978-3-658-15737-1. DOI: 10.1007/978-3-658-15737-1. URL: http://dx.doi.org/10.1007/978-3-658-15737-1_38.

[2] Florian (Blockchainbundesverband) Glatz. Stellungnahme des Blockchain Bundesverband: Fragen für das Fachgespräch zum Thema Blockchain im Ausschuss Digitale Agenda. 2018. URL: https://www.bundestag.de/resource/blob/580950/6f592a83b376199a092e1616eaba5402/A-Drs-19-23-028-Glatz-data.pdf (visited 21. 02. 2019)

[3] H Leopold OVE und T Bleier OVE. "Innovationsdynamik durch Sicherheit im Smart Grid". In: 131 (2014), S. 79–84. DOI: 10.1007/s00502-014-0202-4. URL: https://link.springer.com/content/pdf/10.1007%2Fs00502-014-0202-4.pdf (Visited 17. 02. 2019).

[4] Dennis Laupichler, Stefan Vollmer, Holger Bast und Matthias Intemann. "Das BSI-Schutzprofil: Anforderungen an den Datenschutz und die Datensicherheit für Smart Metering Systeme". In: DuD 35.8 (2011), S. 542–546. ISBN: 1614-0702. DOI: 10.1007/s11623-011-0134-7. URL: https://link.springer.com/content/pdf/10.1007%2Fs11623-011-0134-7.pdf.

[5] Arbeitsgruppe 2 des Nationalen IT-Gipfels (AG2). Nutzen und Anwendungen intelligenter Energienetze Arbeitsgruppe 2 Vernetzte Anwendungen und Plattformen für die digitale Gesellschaft. 2015. URL: https://div-konferenz.de/app/uploads/2015/12/141218_AG2_UAG- IN_Nutzen_Anwendung_Energie.pdf (visited 02. 03. 2019)

[6] Bundesamt für Sicherheit in der Informationstechnik (BSI). Das Smart-Meter-Gateway - Cyber-Sicherheit für die Digitalisierung der Energiewende. Bonn, 2018. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Broschueren/Smart-Meter-Gateway.pdf?__blob=publicationFile&v=6 (Visited 19. 04. 2019).

[7] EOSIO. EOS.IO Technical White Paper v2. 2018. URL: https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md#consensus-algorithm-bft-dpos (visited 04. 08. 2022).

[8] EOSIO. EOSIO for developers. 2022. URL: https://eos.io/for-developers/ (visited 04. 08. 2022).

[9] OpenEMS // Docs. 2022. URL: https://openems.github.io/openems.io/openems/latest/introduction.html (visited 05. 08. 2022).

[10] A. Schwintowski, Hans-Peter; Scholz, Frank; Schuler, Handbuch Energiehandel, 4., völlig. Erich Schmidt Verlag, 2018.

[11] BMWI. Was ist ein "Prosumer"?. 2016. URL: https://www.bmwi-energiewende.de/EWD/Redaktion/Newsletter/2016/06/Meldung/direkt-erklaert.html (visited 09. 08. 2022)