Protecting Data Assets in a Smart Grid SOA

Markus Jung, Thomas Hofer and Wolfgang Kastner
Vienna University of Technology
1040 Vienna

Susen Döbelt
Center for Usability Research & Engineering
1110 Vienna

Abstract

In the future Smart Grid, numerous stakeholders within the electric power grid need to exchange information in order to realize applications like customer energy feedback, billing and invoicing of variable tariffs, demand side management and efficient charging of electric vehicles. Using a Service-oriented Architecture (SOA) based on Web services promises a convenient way to provide a data infrastructure capable to realize the required interactions in a flexible way. A main concern in such an architecture is how access to data can be controlled in order to prevent security or privacy violations. Access control depends strongly on authentication and authorization mechanisms. Therefore, this paper contributes i) a SOA for the Smart Grid and ii) an access control mechanism that is taken into consideration from the early beginning of the system design. Furthermore, a proof of concept implementation and a scalability analysis are used to investigate the requirements on computational resources in a large scale deployment.

1. Introduction

Smart Grids are the next step in the evolution of the traditional power grid. Information and communication technologies (ICT) are added to support integration of variable and renewable energy sources, electric mobility, to provide customers with feedback on energy consumption and to improve the grid stability. Within the future Smart Grid, numerous applications depend on information provided by multiple stakeholders and gathered from heterogeneous technical infrastructures. We suggest a Service-oriented Architecture (SOA) based on Web services as a promising solution to reduce the integration effort needed. Possible use cases for applications that can be realized on top of a SOA are provided below.

- Energy feedback Domestic energy consumption is responsible for about 40 % of the overall energy demands of our society [1]. Providing customers with detailed feedback on their energy consumption has been suggested to support them to save energy. Meta-analysis comparing numerous empirical evaluations on the effectiveness of different types of energy feedback showed possible energy savings ranging from 0% to 20% depending on the quality and the level of detail of provided information [2].
- Demand side management and load management Demand side management and load management both aim to reduce load peaks in the power grid and also to shed loads in situations where grid stability is at stake. Several processes, like heating or cooling, have a certain degree of freedom that allows to manage, shift or curtail the consumption of electricity. Interaction of multiple stakeholders is needed to realize this challenging use case within the Smart Grids.
- Home and building automation A requirement for managing energy consumption within buildings is to employ automation technology, typically providing automation for the domains i) heating, ventilation, air conditioning and cooling (HVAC), ii) security and safety, and iii) consumer electronics. In the residential domain, this technology can provide added value regarding comfort or ambient assisted living.
- E-mobility E-mobility increases and will further increase load in the Smart Grid that needs to be managed through the usage of smart charging mechanisms, but at the same time the accumulators of electric vehicles can act as a flexible storage if there is a local overproduction from renewable energy sources.

Facilitating a SOA based on Web services allows to build a flexible integration layer that makes it possible to reuse existing information sources for a variety of application scenarios. Key concerns within such an information infrastructure are security and privacy. Smart meter values tracked and transmitted in a high temporal resolution offer a detailed insight into the daily life of a household and allow to profile individual behaviour patterns [3]. Therefore, in this

paper we provide an overview of a SOA based on Web services and elaborate how access control can be enforced in order to address the security and privacy concerns. Section 2 summarizes related work and lists relevant technologies for our SOA. Security and privacy requirements are collected in Section 3. Section 4, Section 5 and Section 6 describe the Web service based SOA and the access control concept that addresses the security and privacy requirements. It is followed by the description of a proof of concept implementation and a scalability analysis in Section 7. The results are summarized and a conclusion is drawn in Section 8.

2. Related work

There are several technologies and approaches to realize a service-oriented architecture for Smart Grids. Web services based on SOAP are a popular way to realize such a SOA and for the WS-* stack several access control technologies and protocols are available [4]. WS-Security allows to secure SOAP messages. XML Signature and XML Encryption are key elements of WS-Security and operate on XML documents. WS-Trust and WS-Federation allow to federate multiple security domains. WS-Secure Conversation allows to secure multiple message exchanges. WS-SecurityPolicy can be used to describe the security features that are supported or needed by a Web service. The Security Markup Language (SAML) [5] defines a format and protocol for interoperable exchange of security information about subjects that have to be identified within a certain security domain. Use cases for SAML are Single Sign-On, Identity Federation, Privacy-preserving identification and securing Web service messages using SAML for WS-Security tokens. XACML [6] is an access control language based on XML. It defines a policy language and furthermore provides a request/response model for access decisions.

2.1. Smart Grids and Service-oriented Architectures

Interoperability among heterogeneous technologies and stakeholders within the smart grid is a key issue which has been addressed within the NIST Framework and Roadmap for Smart Grid Interoperability Standards [7]. Web services using SOAP for message exchange with HTTP used as transport layer and XML for message encoding are key technologies to provide interoperability. Open and interoperable message interfaces are one aspect but for true interoperability also the information models and semantics on the application layer need to be addressed. For this purpose, technologies and standards like OPC Unified Architecture (OPC UA) and Open Building Information Exchange (oBIX) provide standardized service interfaces, information models and data representations. OPC UA provides a core information model but can be extended with custom information models for certain domains or to map other technologies. The focus of OPC UA resides in the area of industrial and building automation systems and the importance for Smart Grids is outlined in [7]. OPC UA provides beside a custom binary protocol also a protocol binding to SOAP for message exchange and XML for data encoding. oBIX can be seen as an alternative to OPC UA which focuses on the domain of building automation system. It favors a RESTful protocol design and comes with a standard object model that can be used to represent appliances found within home and building automation systems. It provides also a powerful way to extend the existing object model with custom types using the concept of contracts. For interoperable information models in the Smart Grid, the Common Information Model (CIM) [8] provides a common vocabulary and basic ontology using UML for the electric power industry.

In [9], different views on Service-oriented Architectures in the context of Smart Grids are outlined. Within the inter-enterprise view a SOA based on OPC UA, CIM and semantic Web services is presented. More details on the created SOA are provided within [10].

[11] presents use cases for the application of OPC UA in the Smart Grid and shows the feasibility of using OPC UA within an ICT infrastructure for the Smart Grid. Implementation details regarding the integration with business process engines are described.

2.2. Access control in federated Service-oriented Architectures

A Web Service Architecture for Enforcing Access Control Policies is described in [12]. Within this paper, a concept is presented how authorization for Web service requests can be handled. It is based on user access rights and uses XACML and WS-Policy.

How access control can be realized in a cross-organizational context is shown in [13] where Web services of different service providers are composed in order to realize business functionality. Different access control frameworks for a SOA are evaluated and a 2-level access control architecture is proposed that overcomes the limitations of the evaluated frameworks.

2.3. Access control in a Smart Grid SOA

Challenges regarding the authentication and authorization for Smart Grid application interfaces are addressed by [14]. Key questions are outlined regarding the possibility of losing the authenticated user identity if various application interfaces are in

use, privilege escalation and the challenge of defining and enforcing consistent authorization policies. Furthermore, the paper describes interoperable standards that can be used in a reference architecture to address these challenges. PKI trust models are evaluated and a Smart Grid PKI is presented in [15]. The presented PKI model can complement our SOA and access control concept. A privacy preserving authentication scheme for the Smart Grid is contributed by [16]. For our further work, we plan to investigate how this authentication scheme can be incorporated in our architecture.

3. Security and privacy requirements

3.1. Consumer-driven requirements

Security and privacy issues are a key concern in the large scale deployment of smart meters in Europe. Currently, smart meter security is discussed intensively by national authorities and researchers, but this discussion covers merely one aspect of the future smart grid ICT infrastructure. Threatening scenarios of malicious access to smart meters and smart grid infrastructure worry about billion dollar damage that can be caused through cyber-attacks. Several civil rights organizations started to form resistance against smart meter deployments in different countries and lead to a cancelation of smart meter rollouts or an expensive justification of installments. Therefore, taking into account the customers' wishes and fears about privacy and security issues is of highest importance to ensure future acceptance of smart grid [17]. First guidelines for a privacy preserving smart grid infrastructure have been suggested by [17] and based on overall Privacy by Design (PbD) principles [18]. However, privacy concerns of consumers which are closely related to security have not been in the focus of research in the fields of smart grids, yet. Therefore we collected consumer requirements with regard to smart grid privacy and security concerns. We conducted a multi-perspective and multi-methodological requirements analysis with different smart grid user groups (private and professional energy consumers). We sent out an online survey and conducted two focus groups for gathering results on who consumers perceive to be a trusted authority for energy data storage, what is considered to pose a privacy threat for consumers and how consumers feel energy data should be handled by a smart grid ICT infrastructure. In detail the answers of (n=240) online survey respondents and (n=30) focus group participants indicate that consumers consider the energy utility as a trusted authority to store their personal energy data. Closely connected are the results indicating that access control mechanisms are important to address consumers concerns. Furthermore, our consumer-driven results highlight the demand for information transparency on what data is collected and how is it used. Consumers require secure data transmission and highlighted the importance of decentralized data storage.

3.2. Technical requirements

Merging the security and privacy requirements, we derived the following technical requirements that need to be addressed by a Smart Grid SOA:

- Decentralized data storage Data need to be kept as distributed as possible and under the control of the data owner in order to avoid any abuse that is possible with large centralized databases.
- Platform independent interfaces Platform independency based on open standards is required to make the decentralized data stores available to applications or stakeholders that need to access them.
- Authentication Strong authentication mechanisms are required in order to identify entities in the Smart Grid SOA. This concerns system actors, like servers or embedded devices, but also human actors that interact with applications.
- Authorization Each data access or action needs to be authorized on a fine grained level. For instance, providing aggregated smart meter values on a daily basis imposes less privacy concerns than a readout of minute based values, which is clearly more sensitive.
- Data access policy It is required to provide the data owner with a way to see i) where the data is stored, ii) who has access to the data, iii) for what purpose the data is used, and iv) to offer the possibility to grant or revoke access to the data for certain stakeholders.

4. A Service-oriented Architecture for the Smart Grid

The general goal of a Smart Grid SOA is to provide easy access to a variety of information and functionality spread over different stakeholders that need to interact with each other. Furthermore, it should be easy to deploy applications in the context of Smart Grids that can take advantage of these services. The rationale behind this is to enable the reuse of existing infrastructure and to gain added value by additional applications realized on top of a Smart Grid service infrastructure. As an example, think about an energy utility that wants to provide APIs for third party application providers to benefit from a large developer community or to make it easy to build new applications. To realize a SOA for the

Smart Grid, the service-oriented computing principles have to be followed. A SOA is an architectural style of software design and guides how to design services. For the realization of a Smart Grid SOA, a variety of technologies are available. It is possible to realize a SOA with communication middlewares like CORBA, .NET remoting, DCOM, or Java RMI to name a few examples. However, the state-of-the-art is to use Web services, either based on SOAP, or following a REST approach using HTTP and XML. Information and functionality that need to be provided through services range from smart meter readings, offering the load management capability of an energy consumer, energy market information and tariffs, customer data and information regarding electric vehicles and charging stations. As the examples let imagine, the ICT infrastructure of the required systems and the involved stakeholders are heterogeneous and the realization of a cross-enterprise SOA for the Smart Grid is a challenging topic. The following subsections describe our approach to realize a Smart Grid SOA.

4.1. Architecture overview

Figure 1 gives an overview on the different stakeholders within a Smart Grid SOA and the services provided based on the interaction overview of Smart Grid actors defined by the NIST Framework and Roadmap for Smart Grid Interoperability Standards [7]. The fundamental building block of the Smart Grid SOA are the services offered by the different stakeholders. Beside these interfaces several auxiliary functionalities are required. Due to the central position in a Web service based SOA we call it Smart Web Grid Core. The features bundled in this core component are considered to be our main contribution in the context of Smart Grid research.

The Smart Web Grid Core needs to act as i) application repository, ii) service repository, iii) identity provider, and iv) authorization policy decision point.

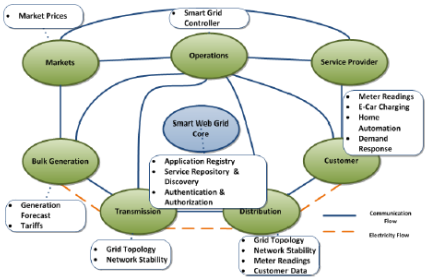

Figure 1. Concept overview

4.2. Architectural alternatives

For the realization of a Smart Grid SOA that takes into account security and privacy from the early beginning of the design phase several architectural decisions and technology alternatives are available. Table 1 provides an overview of these alternative architectural components and visualizes on which building blocks our proposed SOA is based.

The developed SOA and the access control concept is dedicated to SOAP-based Web services. Several already existing standards and technologies in the Smart Grid context, like oBIX, OPC UA, OpenADR, BACnet/WS or Energy Interoperation are either using SOAP for message exchange or XML for message serialization. Hence, they can be incorporated into SOAP services conveniently. For other technologies, it is feasible to provide a SOAP based gateway. Within our SOA, we used an integration approach based on oBIX to integrate smart grid appliances and data sources like smart meters and home and building automation systems.

4.3. Layered functional overview

To illustrate our proposed architecture in more detail, a layered functional overview is provided in Figure 2. On the system layer existing Smart Grid infrastructure like smart meters, car charging stations, energy utility enterprise systems like meter data management systems, billing and CRM systems reside. All these existing systems are consolidated using a service layer based on SOAP Web services available over the Internet. New SOAP services and XML schemas are defined where no existing protocols or standards are available, existing SOAP based protocols are integrated where possible. The SOA core infrastructure provides components like an application registry, a certification authority, a service repository and facilities to allow to discover services, to authenticate service consumers and providers, and to be the basis for a fine grained authorization for Web service access.

Table 1. Architectural Alternatives

Architectural Decision	Alternatives			
Data storage	*Centralized*	Decentralized		
Web Service Technology	*SOAP*	REST		
SoA Integration Technology	*oBIX*	OPC UA		
Service Repository	*UDDI*	ebXML	DNS-SD	Customized
Target Application Platform	*Web application*	Rich Client Platform	Mobile App	
System Authentication	*X.509 Certificate*	Username & Passphrase	Transport level	Token based
User Authentication	*X.509 Certificate*	Username & Passphrase	Transport level	Token based
Distribution of Authentication Logic	*Centralized*	Federated	Distributed	
Authorization Technology	*XACML*	XrML	EPAL	Customized
Authorization Distribution	*Centralized*	Decentralized		
Authorization Policy Administration	*Static administration*	On-demand authorization		

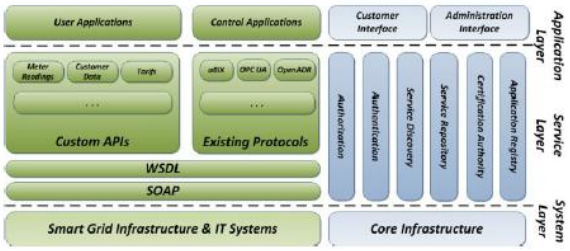

Figure 2. Layered functional overview

4.4. Open building information exchange (oBIX)

For the integration of building automation systems, sensor networks or more general machine-to-machine (M2M) information into a service-oriented architecture based on Web services, oBIX is the approach that we used within our smart grid SOA. It provides a standard XML syntax for representing information provided by appliances using enterprise friendly technologies like XML, HTTP and URIs. A further goal is to provide a standardized representation for common M2M features like data points, histories and alarms and extensibility for custom enhancements. oBIX can be used on embedded devices or on gateway devices that provide access to building automation systems or smart meter infrastructures. For building automation systems like KNX, BACnet, EnOcean, ZigBee and smart meters using Wireless M-Bus we developed such an oBIX gateway as presented in [19]. It is also available as open source and allows to interface all technologies using the SOAP protocol binding of oBIX. oBIX can also be used within the enterprise bus of a company's IT system.

oBIX object model: A core element of the oBIX specification is a generic and simple meta-model for information modeling represented through the oBIX object model. It defines 17 base object types, including 10 value types, and further comes with the concept of contracts. Contracts allow to define a classification of certain types of oBIX objects. They consist of a template model and allow the generation of platform specific types (e.g. Java or .NET classes). Contracts by themselves are also expressed as oBIX objects and allow to convey semantics for human developers with default values for objects adhering to these contracts. In this way, flexible type inheritance is guaranteed for oBIX objects. Objects are based on a set of standard attributes but can also be extended with custom attributes (facets) and further can contain other objects as children (aggregation) or reference other objects (composition). Objects can also expose functionality as oBIX operations which can be invoked by a client. The oBIX core library comes with a set of standard contracts including a watch service, histories and alarming.

In oBIX, each appliance is represented as object. Listing 1 provides a simple example for a smart meter. It provides two basic children objects for the current power and the accumulated consumed energy. According unit information is provided as attribute. Further, there are reference elements to standard history objects that can be used to query the power history. oBIX provides here also a standard query interface to retrieve, for example, a 15 minute rollup for the power values of a smart meter for a defined time interval.

Listing 1. oBIX object for smart meter

```
<obj href="/simMeter" is="iot:SmartMeter">
  <real name="power" href="power" val="125.0"
        unit="obix:units/watt"/>
  <real name="energy" href="energy" val="3215.0"
        unit="obix:units/kilowatthours"/>
  <ref name="power history" href="power/history"
       is="obix:History"/>
  <ref name="energy history" href="energy/history"
       is="obix:History"/>
</obj>
```

Networking and protocol binding: For networking, oBIX defines the client-server communication model and defines three request types: i) read for accessing the current state of an object, ii) write for updating the current state, and iii) invoke for execution of an operation. For the SOAP binding of oBIX these request types are directly mapped to according SOAP operations.

5. Smart Web Grid Core

This section provides a closer view on components that we suggest to use for the Smart Web Grid Core.

5.1. Data owner

A central entity in the Smart Web Grid Core concept is the data owner. The data owner is a natural person or juristic entity to which certain information (e.g. smart meter readings) or interfaces (e.g. home automation) are linked. Web service endpoints are registered for a data owner. Applications (service consumers) can look up the service endpoint for a certain data owner at runtime. Data owners can specify access policies that define which application is allowed to access data related to the data owner. The data owner concept is important in the scenario where third-party applications are used by a customer. Another scenario within the SOA is the machine-to-machine communication, e.g. based on oBIX communication.

5.2. Identity provider & policy administration point

The identity provider allows the various users of the SOA to authenticate them. There are several user roles ranging from the residential customer acting as data owner, to system and grid operators over to SOA administrators. The identity provider is based on an identity store (e.g. LDAP or relational database) that holds the user credentials. The SOA we present in this paper allows username and password authentication as well as user authentication based on X.509 certificates, likely to be equipped on a smart card. The identity provider can be used to establish a single sign-on (SSO) context between the Smart Web Grid Core and third-party application providers. In that way, a customer only needs one set of credentials for authentication in the Smart Grid context. Since the user authentication happens at the Smart Web Grid Core it is secure to let the data owner administrate access control polices. For this reason, the policy administration point provides a Web-based user interface.

5.3. Service Repository

A central component within a SOA is the service repository. It acts as registry and allows a service provider to register a Web service. Service consumers may use the service registry to discover services and service descriptions at design time of a system and furthermore use the registry to look up service endpoints (e.g. URLs) at runtime.

For the realization of a service registry, several alternatives are available like UDDI, ebXML or DNS-SD. Service discovery might also be realized in a local way using WS-Discovery.

The service repository in the presented architecture is a custom implementation and acts as a lightweight variant of UDDI taking some concepts of WS-Discovery as input. The reason is to align the registry to the specific needs introduced with the data owner concept.

5.4. Application registry

The application registry is used to register applications, which works in a similar way to the registration of service consumers. Each application uses a private/public key pair and for each application a X.509 certificate is signed by the Smart Web Grid Core certification authority (CA). These certificates are used for authentication at the system level. The private key is used to sign messages and to securely exchange key material for encrypting messages based on symmetric encryption algorithms.

5.5. Certification authority

The certification authority is the core of the Smart Web Grid PKI and provides certificates to users and systems for the purpose of authentication.

6. Access control in the Smart Grid SoA

This section contains a detailed view on how access control is handled by the Smart Grid SOA. To illustrate the interactions and the details of the involved components the handover of meter readings to a third-party energy advisor application is taken as use case. The access control concept applies SAML and XACML as state of the art technology, but extends these technologies for the specific requirements of this SOA.

6.1. Component overview

Figure 3 provides an overview of the components involved in the access control mechanism and at which node they are located. In this generalized overview only a single service provider and service consumer are presented as placeholder for multiple concrete data interfaces and applications. The service provider in this case can be the meter data management system of an energy utility offering an interface to an energy consultant application provided by a third party.

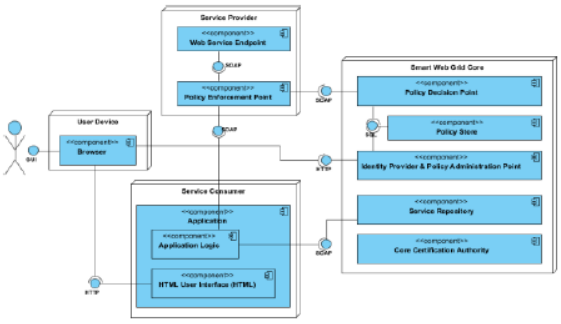

Figure 3 Access control components

The Service Provider (SP) offers a SOAP based Web service to access data or an interface to some capabilities. The interface does not need to be adapted for the access control concept. Instead a generic Policy Enforcement Point (PEP) is used as SOAP intermediary between the Service Consumer (SC) and the SP following the XACML data flow model. The task of this component is to interpret the incoming SOAP request and send an authorization decision request to the Policy Decision Point (PDP). The PDP offers an XACML SAML protocol binding based on SOAP for such decision requests. In the backend it uses a policy store that holds the XACML policies used to evaluate the access control request. The policies are maintained by the data owner using

a third party application. The third party application has to be registered at the Smart Web Grid Core and receives an X.509 certificate that can be used by the application to authenticate and encrypt (using the related private key) at the system level. For authorizing third party service consumers access, the data owner is redirected to the Identity Provider (IDP) that authenticates at the user level. The IDP also acts as Policy Administration Point (PAP) allowing a user to administrate the access control policies. The IDP provides a single sign-on (SSO) interface for the third party application provider using the SAML Web Browser SSO Profile with the HTTP redirect protocol binding for the exchange of SAML assertions. The communication interfaces between the various components are either based on HTTP if a human actor is involved. For all other machine to machine communications, SOAP is used. The SOAP interface offered by the service provider is domain specific and does not need to be altered since a generic access control mechanism is employed.

6.2. Generic access control concept

XACML policy language: The XACML policy language as outlined in Figure 4 provides a powerful way for specifying policies. An XACML policy structure consists of one or many policy sets that can be recursively encapsulated and contain one or multiple policies. Within a policy set, the policy combining algorithm defines the final result of the evaluation. A policy set or a single policy may address a certain target which can be identified through specifying the resource, the subject, the action or the environment. Within a policy rules can be specified. A rule consists of a target, a condition and an effect. The condition element allows refining the applicability of the rule based on XACML built-in functions. The effect is simple: either permit or deny.

Policy structure: To have a generic access control mechanism the policies are based on the information that is available within a SOAP request without referring any domain or protocol specific vocabulary. Our policy model is structured as follows. The various data sources in the Smart Grid for each data owner represent the resource. The resource identifier has therefore two attributes: firstly the data owner identifier which is a random generated UUID pseudo-identifier; secondly the service name which is based on a standard terminology used within the Smart Web Grid core. The service name could be represented through a URI that identifies a certain SOAP port type.

A subject is identified through the available public key information which is represented through a hash identifier based on the used public key. In this way, the access control concept can be kept generic and independent of which public key encryption algorithm is used for signing messages.

The action is represented through the according SOAP action that a service consumer wants to call on the Web service. For each data owner identified a policy set is used that targets the certain data owner identifier. This policy set contains multiple children

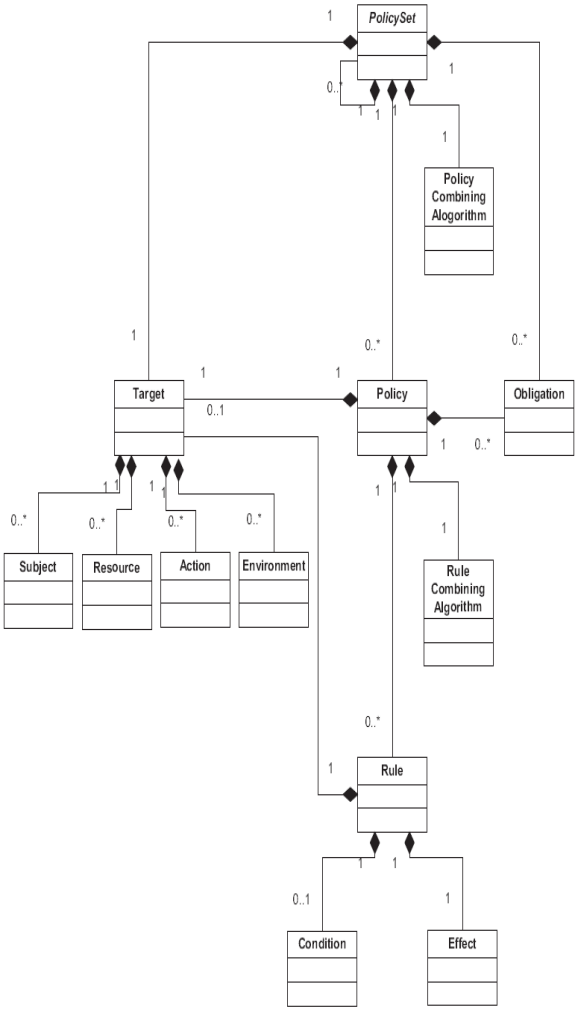

Figure 4 XACML policy language [6]

policy sets, where each policy set targets the available data sources represented through the service name. Then for each application represented through the hash identifier of the public key a policy file resides that contains a rule specifying which SOAP operation is allowed to be used. In this way, a fine grained and generic authorization mechanism based on XACML is possible. For SOAP Web services that are rather generic like for example the oBIX read and write operation the policy structure can be extended to also cover the URI of a certain oBIX object, e.g. the smart meter object or the power history object.

Authorization decision request: The policy enforcement point that resides locally at the Web service endpoint is responsible to create an according

XACML authorization decision request and constructs it in the following way. Take as example the SOAP request provided in Listing 2. The request queries the smart meter object as given in Listing 1 using the oBIX SOAP Web service. For simplicity and illustration, the example request is not encrypted.

Listing 2. Smart meter SOAP request

```
<s:Envelope
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sec="http://schemas.xmlsoap.org/soap/security/2000-
12">
 <s:Header>
  <sec:Signature>
   <ds:Signature
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
     <ds:SignedInfo>...</ds:SignedInfo>
     <ds:SignatureValue>...</ds:SignatureValue>
     <ds:KeyInfo>
       <ds:KeyValue>...</ds:KeyValue>
     </ds:KeyInfo>
   </ds:Signature>
  </sec:Signature>
 </s:Header>
 <s:Body>
 <read xmlns="http://obix.org/ns/wsdl/1.1"
       href="http://obixgateway/smartmeter" />
</s:Body>
</s:Envelope>
```

The request provides all information to decide if a request is allowed or denied. The subject identifier is represented through generating a hash value of the public key which can be found in the ds:KeyInfo field of the WS-Security header. The signature is used to ensure the integrity of the request and that it was created by the owner of the according application that owns the public and private key pair.

The resource is identified through the SOAP port type which is in this case http://obix.org/ns/wsdl/1.1 and the data owner identifier which in this case has to be configured in the oBIX gateway. If a Web service offers data for multiple data owners, it needs to be extracted either out of the SOAP header or of an argument within the called SOAP operation.

The resulting XACML decision request is provided in Listing 3 and illustrates how the information is used to generically create a XACML decision request. This request is sent to the policy decision point using the SOAP binding. This request is also secured through a signature and by encrypting the payload.

Listing 3. XACML decision request

```
<samlp:RequestAbstract
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:xacml-
samlp="urn:oasis:xacml:2.0:saml:protocol:schema:os"
  xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
ID="ID_a9bd16e8-bfa2-450d-b131-2d953adae395"
  Version="2.0"
  IssueInstant="2012-04-23T16:36:50.764+02:00"
  xacml-samlp:InputContextOnly="true"
  xacml-samlp:ReturnContext="true"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xsi:type="xacml-samlp:XACMLAuthzDecisionQueryType">
  <Request
    xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
 xmlns:ns2="urn:oasis:names:tc:xacml:2.0:policy:schema:os">
    <Subject
xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-
category:access-subject">
      <Attribute
xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-
id"      DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="smartwebgrid">
        <AttributeValue
xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os">PUBL
IC_KEY_HASH</AttributeValue>
      </Attribute>
    </Subject>
    <Resource
xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os">
      <Attribute
xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource
-id"     DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="smartwebgrid">
        <AttributeValue
xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os">http
://obix.org/ns/wsdl/1.1</AttributeValue>
      </Attribute>
      <Attribute
xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
AttributeId="urn:tuwien:auto:smartwebgrid:resource:resource
-object"
DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="smartwebgrid">
        <AttributeValue
xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os">/sma
rtmeter</AttributeValue>
      </Attribute>
    </Resource>
    <Action
xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os">
      <Attribute
xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
        DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="smartwebgrid">
        <AttributeValue
xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os">read
</AttributeValue>
      </Attribute>
    </Action>
  </Request>
</samlp:RequestAbstract>
```

6.3. Interaction

Figure 5 illustrates the interaction between the various stakeholders in the Smart Grid SOA in respect to the access control concept. The first step happens when a user wants to access a Web-based application. The browser of the customer is redirected to the central identity provider where the user authenticates him using either a username and password or mutual authentication using a X.509 certificate. On the first usage of an application the user is redirected to the policy administration point where the access rights of the application on the user data sources can be administrated and transformed from a domain specific terminology into the generic XACML access policy structure. After the user is authenticated a SAML assertion based on the SAML

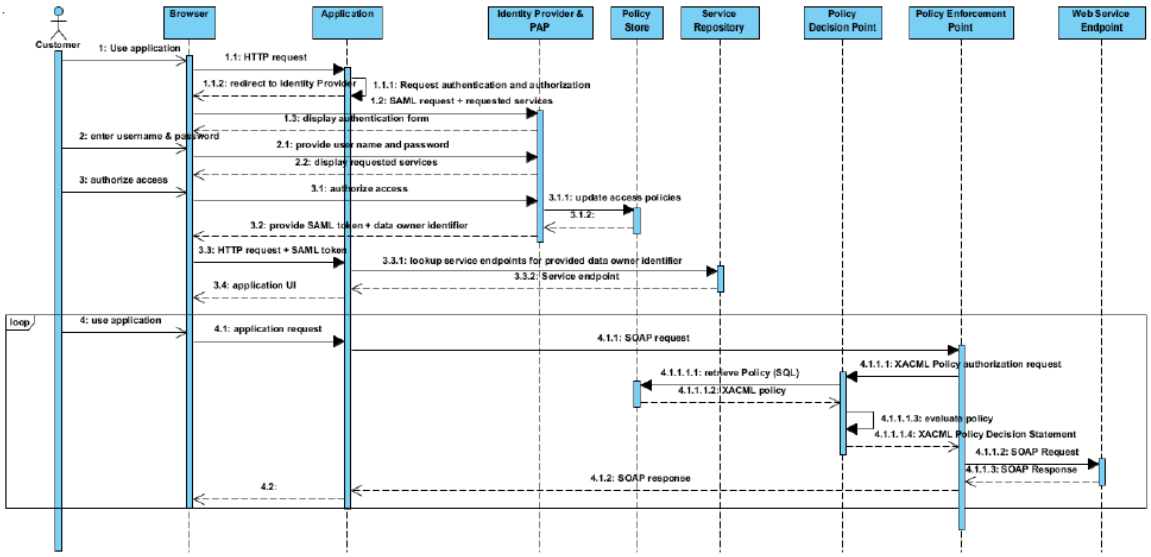

Figure 5 Access control interaction

Web single sign-on profile is returned to the browser of the customer including the data owner identifier as identity attribute.

The application server receives this SAML assertion and creates an authentication context for the customer. Based on the data owner identifier it queries the service repository for the required Web service endpoints. The application server then issues the according SOAP requests, e.g. a query to the smart meter object offered by the oBIX interface. The SOAP request is encrypted and signed by the application. The PEP acts as a SOAP intermediary in front of the Web service endpoint that provides the interface to the data source. It extracts all information to create an XACML decision request and sends it to the central PDP. The PDP evaluates the policies and decides whether access is permitted or denied and returns the decision. Based on the decision the original SOAP request is forwarded by the PEP and processed. Otherwise a SOAP fault message is returned to the application server.

7. Proof of concept implementation

The presented generic access control concept for a Smart Grid SOA has been implemented to show the feasibility and to analyze the performance and scalability.

7.1. Core platform

The Smart Web Grid Core infrastructure is implemented within a test bed using Java 6 and JBoss 7 as platform. The IDP and PDP are based on the open source JBoss Picketlink framework, but customized for the specific needs of the Smart Web Grid Core. The components of the core platform are implemented using the Java enterprise EJB framework in order to provide a component-oriented software development framework.

7.2. Identity Provider

The identity provider acts as central single sign-on component for all Web applications in the Smart Web Grid ecosystem. Based on PicketLink, it offers a standardized SAML2 Web SSO profile that can be used by third parties to authenticate users. As identity store a relational database (MySQL) that contains the user credentials and the assigned data owner identifier. Within the authentication procedure the data owner identifier is added as attribute to the SAML SSO token and passed to the third-party application.

7.3. Policy administration point

The policy administration point is a simple Web application that also resides on the core infrastructure and is responsible for the creation of the XACML policies. It therefore has to use a domain or application specific terminology to let a human user administrate the policies. For example, it has to display access rights like smart meter readings realtime or smart meter history with a defined resolution and map these to the according XACML access policy for the data owner and the application.

7.4. Policy decision point

The PDP is a central component of the proposed architecture. For the time being, PicketLink only supports file based policies that are loaded at startup by default. The size of the policies and the huge number that can be anticipated for a full Smart Grid deployment require to store the policies in a

database. Therefore, we extended PicketLink with a custom DatabasePolicyLoader that is able to load and store policies during runtime based on the data owner identifier. The PDP provides a SOAP endpoint according XACML 2.0 SAML profile that can be used by the PEP to submit an authorization decision request. For securing the communication between the PDP and the PEP, the core infrastructure uses a public/private key pair, where the public key is trusted by the distributed PEPs.

7.5. Policy enforcement point

For the implementation of the PEP the membrane router is used as SOAP intermediary. It listens on a configured port and is placed between the service provider and the service consumer. The PEP is transparent to the service consumer and its application logic. In this way it is possible to place the PEP in front of any SOAP based Web service endpoint. This allows to integrate the proposed access control concept easily into existing Smart Grid technologies like OPC UA, oBIX or Open ADR.

7.6. Service registry

The service registry is based on a custom implementation due to the specific needs for querying data owner specific server endpoints at runtime and the fixed set of service types that are under control of the Smart Web Grid Core infrastructure administrator.

7.7. Certification authority

The full implementation of a PKI is left out in the test bed, since it is only required for a real world deployment. For the creation of according public/private-key pairs and X.509 certificates the Java key tool is used.

8. Evaluation

Deploying a central component like the Smart Web Grid Core requires a performant and scalable architecture. In order to evaluate the required computational resources, we carried out a performance analysis based on an analytic queuing

network (QN) model and a validation of this model based on benchmarks within the test bed. For analyzing our architecture, we modeled the system as open queuing network where the system load is provided as input through the request arrival rate. The QN allows to estimate performance metrics of a system, e.g. the average response time, depending on the system load and the available computing capacities.

The following definitions are required to calculate the relevant performance metrics:

- τ: observation period
- λ: average arrival rate
- T: response time
- U_i: utilization of resource i in the observation period
- C_i: total number of service completions from resource i in the observation period
- D_i: total average time spent of a request at the resource i

For applying a QN model to a concrete system the service demands of different request types on the resources need to be identified. Within a preliminary simple benchmark in our test bed we identified the policy decision point as bottleneck in the overall system and further analyzed the different authorization requests that can happen. For a classification of the request types, we measured the service demand of the evaluation requests for policies that result in a permit, deny and not applicable authorization statement.

8.1. Service demands

To identify these service demands the operational analysis methodology presented by [20] is used which is based on the service demand law which states:

$$D_i = \frac{U_i * \tau}{C_i} \qquad (1)$$

This law is used to derive the average CPU demand of the different requests within our test bed by sending uniform distributed requests within an observation period of τ=600 seconds. Table 2 lists the different service demands. The derived service demands show that there are nearly no differences between these request types. Therefore, we reduced the queuing network to consider only a single request type approximated through the service demand of a permit policy evaluation.

Table 2. Service demands of PDP request types

	Permit	Deny	Not Applicable
CPU	0.1070	0.1062	0.1034

Using this service demands it is possible to solve a simple open QN model using the following equations

$$\lambda = \frac{C}{\tau} \quad (2)$$

$$U_i = D_i * \lambda \quad (3)$$

$$T_i = \frac{D_i}{1 - U_i} \quad (4)$$

The calculated response time depends on the average arrival rate λ keeping in mind that Equation 4 is bound to the utilization . The utilization is calculated using the service demand and the arrival rate λ.

8.2. Benchmark

For the benchmark, the Smart Web Grid Core infrastructure is deployed on a virtualized server using a 64-bit Linux 3.0.0 operating system with 4 GB of RAM and a 2 Ghz AMD Opteron 6128 CPU. These settings come close to a medium instance of the Amazon EC21 and allow to quantify the costs for a required infrastructure. For the benchmark, the database is filled with 100.000 XACML policies leading to 1.9 GB of storage. The benchmark uses JMeter for distributed testing scenarios including five JMeter server instances and another client instance, controlling (starting, stopping) the tests. The request type used to evaluate the performance is the same as used to calculate the service demands. The unique data owner identifiers that are extracted to files. Each JMeter server instance has a list of 20000 distinct data owner identifiers. These data owner identifiers are inserted to the XACML decision requests and sent to the PDP. The generation of the jobs is done by a python script, calculating the amount of used threads to generate the desired transactions per second. The average response time is measured during the tests with increasing arrival rate λ starting from 0.5 requests to 20 requests per second with an observation period τ=600 seconds per test and a step size of 0.5. This setting leads to 40 tests each running 10 minutes and a cool down time between the tests of 1 minute.

Figure 6 shows the results of the scalability analysis of the implemented XACML policy decision point Web service, which is identified as the bottleneck within the overall system architecture. It can be seen that the queuing network model provides a good estimation of the expected average response time for a certain workload and at which point the system starts to fail. The results of the benchmark show that at a certain point too many requests are queued and rejected by the application server. Therefore, the average response time stays below the growth of the queuing network model but this is due to the failing requests. The results allow to right-size a real world deployment to the specific needs of the core infrastructure owner (e.g. energy utility) and to adjust the provided computational resources to the expected workload.

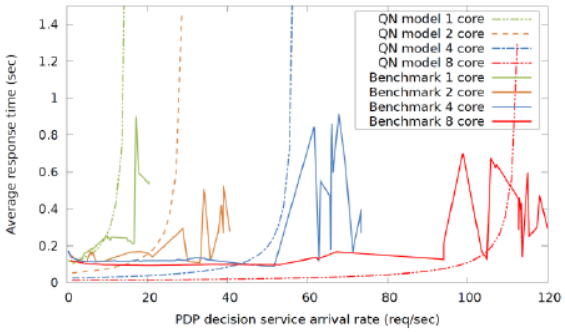

Figure 6. Scalability analysis

8. Conclusion

In this paper, we presented a generic access control mechanism for a Smart Grid SOA. We described a possible instantiation of a Smart Grid SOA which eases the creation of applications and allows third party application providers to conveniently contribute applications in the context of Smart Grids. We showed how the proposed access control mechanism can easily enhance existing protocols and technologies like oBIX, OPC UA or OpenADR that are required to build a Smart Grid SOA.

The presented access control concept allows a data owner to restrict access to her data in a finegrained and technically generic way. Throughout our work we paid special attention to security and privacy related issues and incorporated solutions directly into the architecture of our Smart Grid SOA. Finally, we evaluated the scalability of the system architecture and our results allow predicting the required computational resources for a large scale deployment as well as the costs for operating such an infrastructure.

10. Acknowledgements

This work has been partially funded by the Austrian Climate and Energy Fund within the programme New Energies 2020 and the project Smart Web Grid (P 829 902).

11. References

[1] “Mitigation of CO2 Emissions from the Building Stock. Beyond the EU Directive on Energy Performance of Buildings,” EURIMA, ECOFIS-study, 2004.

[2] S. Darby, “The effectiveness of feedback on energy consumption,” A Review for DEFRA of the Literature on Metering, Billing and direct Displays, vol. 486, 2006.

[3] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin, “Private memoirs of a smart meter,” in Proceedings of the 2nd ACM Workshop on Embedded

Sensing Systems for Energy-Efficiency in Building, 2010, pp. 61–66.

[4] J. Rosenberg and D. Remy, Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption. Pearson Higher Education, 2004.

[5] "Security Assertion Markup Language (SAML) V2.0," OASIS Standard, 2005.

[6] "Extensible Access Control Markup Lanaguage(XACML) Version 2.0," OASIS Standard, 2005.

[7] "NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 2.0," NIST special publication, 2012.

[8] "Energy management system application program interface (EMS-API) - Part 301: Common information model (CIM)," IEC 61970-301, 2007.

[9] M. Postina, S. Rohjans, U. Steffens, and M. Uslar, "Views on service oriented architectures in the context of smart grids," in 2010 First IEEE International Conference on Smart Grid Communications (SmartGrid-Comm), 2010, pp. 25–30.

[10] S. Rohjans, M. Uslar, and H. Appelrath, "Opc ua and cim: Semantics for the smart grid," in Transmission and Distribution Conference and Exposition, 2010 IEEE PES, 2010, pp. 1–8.

[11] A. Claassen, S. Rohjans, and S. Lehnhoff, "Application of the OPC UA for the smart grid," in 2011 2nd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies (ISGT Europe), 2011, pp. 1–8.

[12] C. Ardagna, E. Damiani, S. De Capitani di Vimercati, and P. Samarati, "A Web service architecture for enforcing access control policies," Electronic Notes in Theoretical Computer Science, vol. 142, pp. 47–62, 2006.

[13] M. Menzel, C. Wolter, and C. Meinel, "Access control for crossorganisational web service composition," Journal of Information Assurance and Security, vol. 2, no. 3, pp. 155–160, 2007.

[14] S. Lakshminarayanan, "Authentication and authorization for Smart Grid application interfaces," in Proceedings of the Power Systems Conference and Exposition (PSCE). IEEE, 2011.

[15] T. Baumeister, "Adapting PKI for the smart grid," in Proceedings of the International Conference on Smart Grid Communications, 2011, pp. 249 –254.

[16] T. Chim, S. Yiu, L. Hui, and V. Li, "PASS: Privacy-preserving authentication scheme for smart grid network," in Proceedings of the International Conference on Smart Grid Communications, Oct. 2011, pp. 196 –201.

[17] A. Cavoukian, J. Polonetsky, and C. Wolf, "Smartprivacy for the smart grid: embedding privacy into the design of electricity conservation," Identity in the Information Society, vol. 3, no. 2, pp. 275–294, 2010.

[18] A. Cavoukian, "Privacy by design. the 7 foundational principles." 2011.

[19] M. Jung, J.Weidinger,W. Kastner, and A. Olivieri, "Building automation and smart cities: An integration approach based on a service-oriented architecture," in Proceedings of the 27th IEEE International Conference on Advanced Information Networking and Applications, Barcelona,Spain, Mar. 2013.a

[20] P. J. Denning and J. P. Buzen, "The Operational analysis of queuing network models," ACM Computing Surveys, vol. 10, no. 3, 1978.

