Security for Open Distributed Geospatial Information Systems

19

Andreas Matheus

Contents

19.1 **Introduction** . . . 567
19.1.1 SOA Implementation Options . . . 568
19.2 **Security Requirements** . . . 568
19.2.1 Thinking About the Threats – What Is the Enemy . . . 569
19.2.2 What Is the Web Browser Same Origin Policy? . . . 569
19.2.3 Which Requirements Are Geo-Specific? . . . 570
19.3 **Standards for Interoperable Implementation of Security Functions** . . . 570
19.3.1 Standards for Implementing Confidentiality and Integrity . . . 572
19.3.2 Standards for Implementing Authentication . . . 573
19.3.3 Standards for Implementing Access Delegation and User Claims . . . 574
19.3.4 Standards for Implementing Access Control . . . 574
19.4 **Summary** . . . 576
References . . . 577

Abstract

This chapter gives a brief introduction to relevant security requirements and how they can be implemented based on standards for a Web Services and Web-API based approach. It is not the intention to provide individual solutions, as an adequate solution typically depends on many more factors than can be taken under consideration in this chapter. Instead, we like to see this as a starting point from where the reader can follow references to applicable standards for further reading.

Keywords

security · SOA · API · OAuth2 · OpenID Connect · XACML · GeoXACML

A. Matheus (✉)
Secure Dimensions GmbH
Munich, Germany
e-mail: a.m@secure-dimensions.de

19.1 Introduction

Security for geographic information systems (GIS) has gained in importance since Service Oriented Architecture (SOA) enabled the implementation of large, open distributed systems for the creation, processing, viewing, and maintenance of geographic information. Its main characteristic, as specified in [1], is that SOA is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different operators and ownership domains. SOA as an architectural pattern for designing application programming interface(s) (API) that focuses on the separation of responsibility and independence.

As such, SOA APIs cause challenges when implementing effective security functions that take into consideration not only the traditional requirements for installing a GIS in one's own local area network with known and trusted users, but also communication with insecure network segments such as the Internet without knowing which computers and users have access to that network. The traditional paradigm of *we are secure because we have a firewall* no longer holds, as API execution can intrude into an internal system over firewall port 80 via HTTP or port 443 via HTTPS (HTTP over Transport Layer Security/Secure Sockets Layer). However, because there has to be an open port as an essential requirement for participating in a distributed processing system, the question exists of how to properly make one's own system secure and protect it from unauthorized access and prevent attacks that might come in via that open port. Even though it is important to think of all attack vectors when designing an API, it is not the intention of this chapter to elaborate a holistic security approach that encompasses all existing requirements and evaluate all possible options to determine the best solution. Rather, we will address common aspects to provide better understanding of what security means in the context of SOA and APIs and which standards exist to make a geosystem secure for participation in a larger open distributed system.

W. Kresse, D. Danko (Eds.), *Springer Handbook of Geographic Information*, Springer Handbooks,
https://doi.org/10.1007/978-3-030-53125-6_19

When it comes to the decision that *we* intend to participate in a distributed geospatial information system, many questions arise related to security: Are we going to use the traditional Web services or the modern Web API approach? What do we need to do to prevent unauthorized access to the geospatial information and services that we are going to provide? Which potential attacks are we facing, hence which threat models do we need to consider, and can we mitigate or prevent attacks? Can we build a solution based on standards, and which standards are that?

19.1.1 SOA Implementation Options

In service-oriented architectures, any service (component) defines an application programming interface that can be used by other components to communicate/execute the service. Many different implementation approaches for APIs exist. Which one to use highly depends on the overall aspects of the computing environment like scalability, accessibility, availability, as well as robustness and other criteria. For the purpose of this chapter focusing on security, we would like to focus on two common implementation options: (i) Web services and (ii) Web APIs.

Web Services-Based Implementation

When leveraging the traditional Web services approach, the API is a service that can be executed via HTTP. For each service function, a particular input and output message structure encoded in XML must be defined. The entire API can be described using the Web Services Description Language (WSDL) as described in [2]. Therefore, the actual implementation can be based on some HTTP methods like GET and POST and a set of XML messages that define the input and output.

From a security perspective, it is important to note that requirements can be implemented on the actual XML messages independent from the underlying transport protocol. Introducing SOAP [3] and applying WS-Security [4] to the input and output messages, it is possible to implement end-to-end security regarding integrity and confidentiality. Moreover, because the XML processing is part of the application itself, each security feature implemented requires interoperability. This can be achieved by describing the security requirements for the XML (SOAP) messages using the WS-SecurityPolicy standard [5].

Web API-Based Implementation

When leveraging the modern approach to use Web-APIs, one has deliberately tied the implementation to the capabilities of the underlying communication protocol: HTTP. An API endpoint is a URL that allows in combination with a particular HTTP method another component to execute the associated behavior.

From a security perspective, the choice of Web API by itself does not prevent the use of XML messages and, therefore, the option to apply security via WS-Security. However, the modern approach to Web API also implies the use of a more lightweight message structure and the deliberate choice to be more tightly coupled with the capabilities of HTTP [6]. So, instead of sending heavy XML including digital signatures and encrypted SOAP messages, the choice is to use JSON [7] and to attach security with HTTP requests.

The choice of using JSON plus security in HTTP headers is a common approach these days to apply security with Web-APIs and to have the API be accessed by a web application. Even this approach is very tangible for securing one's own system; it introduces additional security challenges and requirements triggered, for example, by the web browser security policy named same origin [8].

19.2 Security Requirements

Before we begin, it is essential to define what we mean by security in the context of this chapter: what it is and is not concerned with. *Security* is described as the characteristic of a system (whether distributed or not) that prevents unwanted, hence unauthorized, actions to be executed on the system itself with potential side effects on information that is accessible via the system. The Trusted Computer System Evaluation Criteria, also known as *Orange Book* states [9]:

> In general, secure systems will control, through use of specific security features, access to information such that only properly authorized individuals, or processes operating on their behalf, will have access to read, write, create, or delete information.

Extending this definition for a single system to a distributed system, which consists of multiple autonomous computers that communicate through a computer network, the communication will not have any influence. This means that the capability of the system to prevent unauthorized access to the information needs to include the communication between the distributed systems.

The typical requirements that exist when securing a distributed system are described in ISO 10181 consisting of

- ISO 10181-1 *Overview* [10]
- ISO 10181-2 *Authentication* [11]
- ISO 10181-3 *Access control* [12]
- ISO 10181-4 *Non-repudiation* [13]
- ISO 10181-5 *Confidentiality* [14]
- ISO 10181-6 *Integrity* [15]
- ISO 10181-7 *Security audit and alarms* [16].

ISO 10181-1 describes the organization of security frameworks, defines relevant security concepts, and describes relationships of the services of the frameworks. To do this it uses security architecture definitions from ISO/IEC 7498-2 [17], such as access control, availability, denial of service, digital signature, and encryption. It also provides other relevant definitions, such as security information, security domain, security policy, trust entities, trust, and trusted third parties, and for the security information, it defines security labels, cryptographic check values, security certificates, and security tokens. In addition, it defines denial of service and availability in such a sense that denial of service cannot always be prevented. In these cases, other security services can be used to detect the lack of availability and allow the application of corrective measures. Annex A of 10181-1 provides an example of protection measures for security certificates and defines the key management framework, as its functions are applicable to any information technology environment where digital signatures and encryption are used.

ISO 10181-2 defines all aspects of authentication in open systems and the relationship with other security functions, such as access control.

ISO 10181-3 defines all aspects of access control in open systems, as it applies to the interactions of user with processes, user with data, process with process, and process with data. It also defines the relationships to other security functionality, such as authentication and audit.

ISO 10181-4 introduces all aspects of nonrepudiation and extends the concepts defined in ISO/IEC 7498-2.

ISO 10181-5 defines confidentiality as a service *to protect information from unauthorized disclosure* in retrieval, transfer, or management.

ISO 10181-6 defines integrity as a property that *data has not been altered or destroyed in an unauthorized manner.* This applies to data in retrieval, transfer or management.

ISO 10181-7 defines the basic concepts of a general model for and identifies relationships between services for security audit and alarms.

When it comes to classified information, and in the geospatial domain, you can find examples for classified information quite easily; additional requirements exist that extend the typical access control requirements where rights are associated to users either directly or by role to ensure the confidentiality of the information and its integrity, including security labels.

Information flow control models such as the Bell–La Padula [18] and Biba models [19] are relevant, as outlined in RFC 1457 [20].

To guarantee the confidentiality of classified information, *The Orange Book* names the Bell–La Padula (information flow control) model [9] that defines secure state, modes of access, and rules that grant/deny access. It ensures that classified information is not flowing from higher classification to lower classification. Therefore, the model is also known for its main purpose: *no read up – no write down*.

The Biba model addresses integrity of information by defining conditions to ensure: *no read down – no write up*.

19.2.1 Thinking About the Threats – What Is the Enemy?

Before thinking of a particular implementation of security aspects, it is essential to think about the relevant, and hence applicable, security requirements. Perhaps it is not always relevant to implement them all. To determine this, the question of which threats potentially exist must be asked. There is a big difference if you consider the *Internet threat model* and/or the *browser threat model* as a relevant cause for any attacks to your system.

With the Internet threat model, it is considered that the communicating end systems can be trusted, but that the communication is unsafe. As defined more precisely in RFC 3552 [21], the attacker has control of the communications channel over which the end systems communicate, and the attacker can read any protocol data on the network and undetectably remove, change, or inject forged information.

In addition to the defined Internet threat model, other threats exist that relate to browsing the Internet that are sometimes listed under the umbrella of the browser threat model. This model considers that the client, the browser application running on an end system, for example, its users are vulnerable to attacks such as phishing, identity theft, etc.

In addition to these general threat models, specific attacks like cross-site request forgery (CSRF or XSRF), also called one-click attack, mainly leverage HTML image tags or JavaScript XMLHttpRequest elements to execute otherwise unauthorized commands as the current user without the user's knowledge. Another form of attack, which is particularly relevant with Web applications implemented in JavaScript, is cross-site scripting (XSS). Here, the aim is to inject malicious JavaScript code as trusted by the application to lever out the Web browser's Same Origin policy.

Without elaborating on this in more detail, it is important to understand which of the listed requirements are important and which standards are applicable to build the solution.

19.2.2 What Is the Web Browser Same Origin Policy?

The Web browser processes content loaded from different Web servers in security sand boxes to prevent malicious website operators to interfere with trusted websites. Different sand boxes exist for different types of content. The Same Origin policy is linked to a sand box for safeguarding the

execution of JavaScript initiated network requests. The policy does allow that JavaScript code, contained in a Web page loaded from Web server 'A' to load content from the same a Web Server 'A' or any other Web server, so long these Web servers are considered "same origin". The concept of a Web origin is defined in [8], and the detailed protocol for JavaScript initiated network requests to allow cross-origin resource sharing is defined in [22].

19.2.3 Which Requirements Are Geo-Specific?

Requirements stated in ISO 10181-4 are not specific to Geospatial Information Systems (GIS). But geo-specific access conditions must be considered. This has to do with the characteristic of the information: attributes of the information objects as well as the fact that the user can hold geometry information that represents the location, extent, etc. of the object or user. For geospatial data and services or APIs, use cases exist that require the declaration and enforcement of access rights based on the

- Location of the subject
- Geometry of the object (resource)
- Location of the subject and the geometry of the resource
- Topological relations between geometries
- Results of complex processing on geometries.

19.3 Standards for Interoperable Implementation of Security Functions

When it comes to the implementation of security functions, it is a particularly good idea to review existing standards to determine whether there is not (at least) one that can be used. Why? Because many experts have found a keen and practical solution to a problem, and typically software exists, in the form of either libraries or even larger software packages, that has implemented the standard (Chap. 15).

Figure 19.1 provides a first overview of security-related standards that are applicable to secure a distributed geospatial information system based on Web services supporting implementation of the listed requirements. It is worth mentioning that actually one geo-specific specification from the Open Geospatial Consortium (OGC) exists: GeoXACML (Geospatial Extensible Access Control Markup Language). We will elaborate more on GeoXACML in Sect. 19.3.4.

Figure 19.1 is structured such that it categorizes the standards and stacks the layers in a similar way to the Open Systems Interconnection (OSI) model [23].

The Internet Engineering Task Force (IETF) RFCs (request for comments) IPSec (Internet Protocol Security) [24] and TLS/SSL [24] are applicable to actual OSI (Open Systems Interconnection) network layers: IPSec falls into the OSI network layer, and TLS/SSL falls into the transport layer.

The IETF HTTP RFC [6] falls into the OSI application layer, as does SOAP (Simple Object Access Protocol) [3].

As SOAP enables communication using Extensible Markup Language (XML) notation, the next layer above are the XML security standards that contain the W3C recommendations XML digital signature [25] and XML encryption [26].

The next category, message security, is concerned with enabling integrity and confidentiality in XML messages exchanged via SOAP messages. Here the most dominant standard is the OASIS (Organization for the Advancement of Structured Information Standards) WS-Security [4]. As a supplement, one can see the relevance for expressing the requirements that a Web service places on a client to establish communication. WSDL (Web Services Description Language) [2], WS-Policy [27], and WS-SecurityPolicy [5] provide these capabilities.

The next category, concerned with authorization, contains the OASIS XACML [28–30] and the OGC GeoXACML [31–33] standards. An extension to authorization is licensing, which is the next category up. It contains the ISO standard (Mpeg)REL (Rights Expression Language) [34], OMA's (Outlook Mobile Access) ODRL (Open Digital Rights Language) [35] and content guards XrML (Extensible Rights Markup Language) [36].

Authentication is a cross-layer topic that mainly consists of the IETF RFC for X.509 [37] and OASIS SAML V2 (Security Assertion Markup Language) standard [38–40]. Also, Kerberos [41] and LDAP (Lightweight Directory Access Protocol) [42, 43] for X.500 fall into this category.

Figure 19.2 is structured such that it categorizes the standards and stacks the layers in a similar way to the open systems interconnection (OSI) model [23]. Up to and including the OSI transport layer, the security standards are identical to the ones introduced before. For the OSI application layer, the instance digests in HTTP [44] allow to transmit a checksum for the information send with the HTTP request or response. CORS [22] defines a particular protocol to use HTTP Request and Response headers to overcome the Web browser's same origin policy.

Even though CORS is a non geo-specific security requirement, it applies to any distributed open system implemented as SOA like a spatial data infrastructure, but in particular using Web APIs and Web applications. One typical example is a Web mapping application that is loaded from Web Server one. Once loaded, the JavaScript executes the mapping application that intends to load maps or perhaps geographic features from other Web servers that are not of the same origin as Web server one. This behavior triggers the Web browser's same origin policy and it enforces the protocol defined in W3C CORS [22]. Because the Web mapping application will

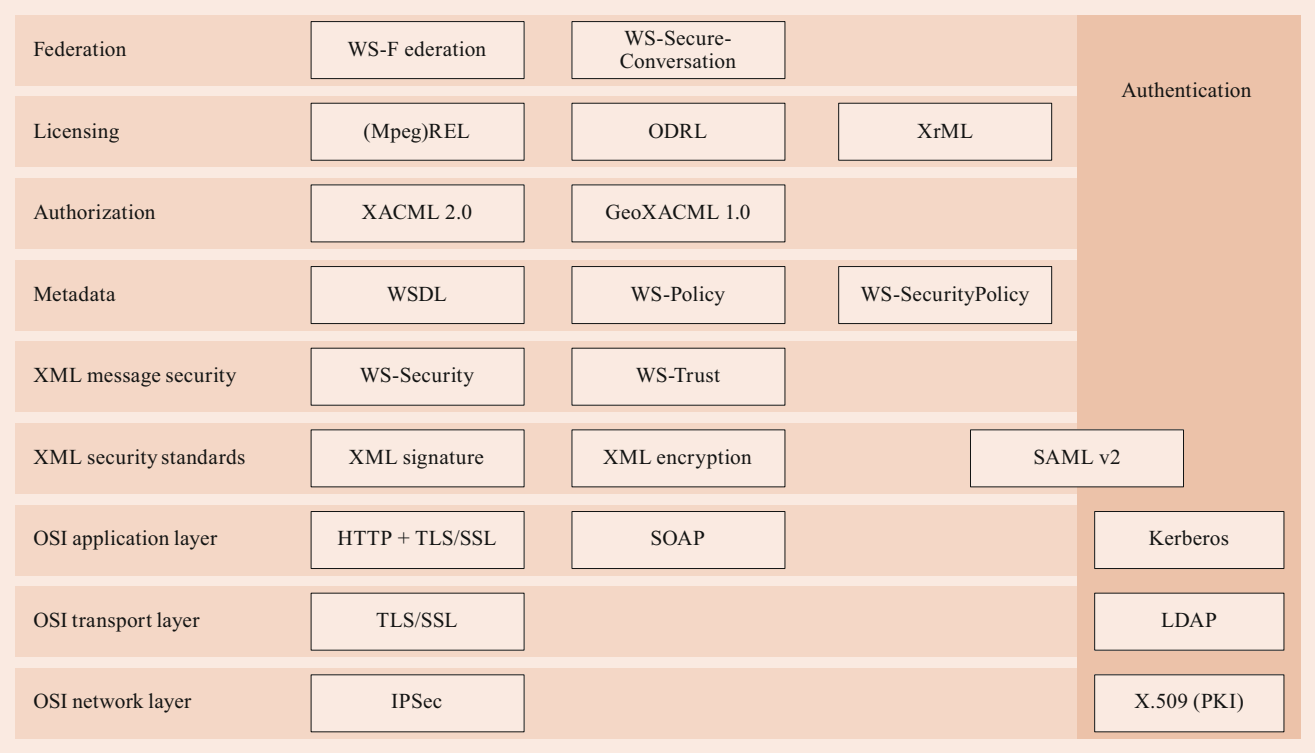

Fig. 19.1 Security standards overview in the context of Web services (subset)

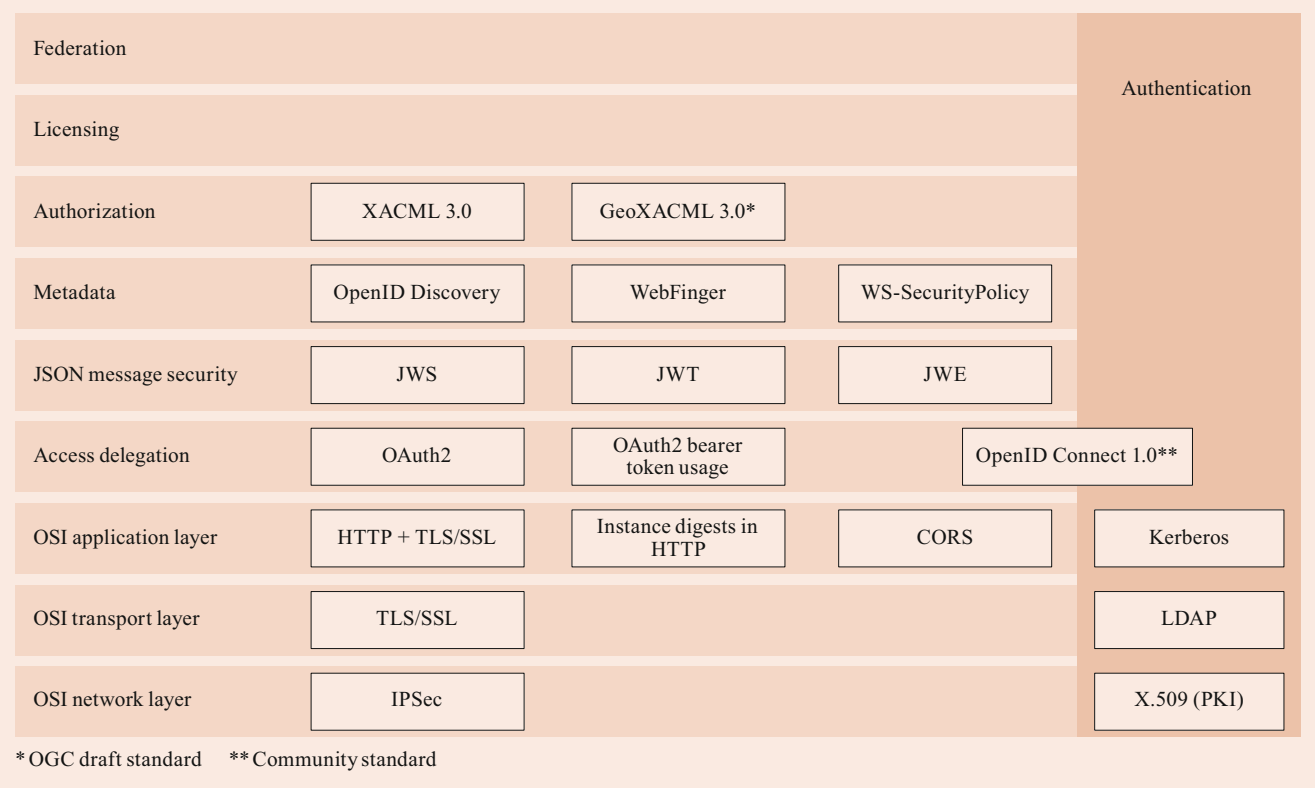

Fig. 19.2 Security standards overview in the context of Web APIs (subset)

only be able to load content from those Web servers that honor the W3C CORS protocol, it is required that each Web API participating in the distributed open system, aka a spatial data infrastructure, must be compliant with W3C CORS.

New from the standards overview as illustrated in Fig. 19.1 is the access delegation layer. The main standards here are OAuth2 [45], the OAuth2 Bearer Token Usage [46], and OAuth2 Token Introspection [47] (not illustrated) RFCs. Also in the layer of Access Delegation is OpenID Connect 1.0 [48] as it is an extension to OAuth2 that bridges to Authentication. It extends the OAuth2 framework with the ability that the Authorization Server can release so called ID tokens that contain user information. It is also possible for Resource Servers to use the specified UserInfo interface to fetch user information associated with an access token.

The JSON Message Security is naturally different from the corresponding layer in Fig. 19.1 as the target is not messages in XML encoding but JSON. The JSON Web Signature (JWS) [49] standard is concerned with the ability to apply a digital signature to arbitrary JSON data. One specialization of JWS is JSON Web Token (JWT) as standardized in [50] that defines how to apply a digital signature to JSON encoded claims. JSON Web Encryption (JWE) as standardized in [51] allows to encrypt the JSON claim. Other standards not illustrated are JSON Web Key (JWK) [52] that is concerned with describing a key in JSON format and JSON Web Algorithm (JWA) [53] that standardizes the registration of cryptographic algorithms with IANA to become usable in JWS and JWE.

The metadata layer contains the standards that are concerned with the description and discovery of endpoints to facilitate the verification of identities of users based on authentication by OAuth2 Authorization Servers as defined in OpenID Discovery [54]. For the purpose of discovery, the OpenID Connect specification leverages the WebFinger [55] standard. The description of Web APIs can be done using OpenAPI [56].

Compared to Fig. 19.1, the authorization layer contains the same standards but with newer versions. In particular, the XACML 3.0 standard [57] enables that authorization request and responses can be encoded in JSON to facilitate better uptake in Web applications. As GeoXACML 3.0 [58], which currently is a draft standard at OGC, is an extension to XACML 3.0, it inherits the JSON encoding for authorization request and responses.

19.3.1 Standards for Implementing Confidentiality and Integrity

Protecting the conversation between two entities can be implemented by leveraging functions from different layers of the OSI reference model; for example, IPSec as a secure extension to Internet Protocol (IP) that resides in layer 4 (the network layer) can be used to encrypt the entire communication between communication end systems. Here, the application itself cannot control how the encryption is done, which is good on the one side, as it takes away the burden from the application programmer to incorporate security functions. A kind of hybrid solution that partially involves the application but still encrypts the entire communication between end systems is TLS/SSL, which can be located in the OSI transport layer. For use cases that require more flexible control over the protection of XML structured communication messages or end-to-end protection, only functions that can be directly controlled by the application and applied to the XML message are feasible.

It is important to note that, for the chaining of Web services, where integrity and, confidentiality span multiple intermediary services, end-to-end protection is required, and therefore, WS-Security-based protection should be applied. Point-to-point protection, as provided by the transport layer, is not sufficient, as information is available in the clear on the intermediary services (Fig. 19.3).

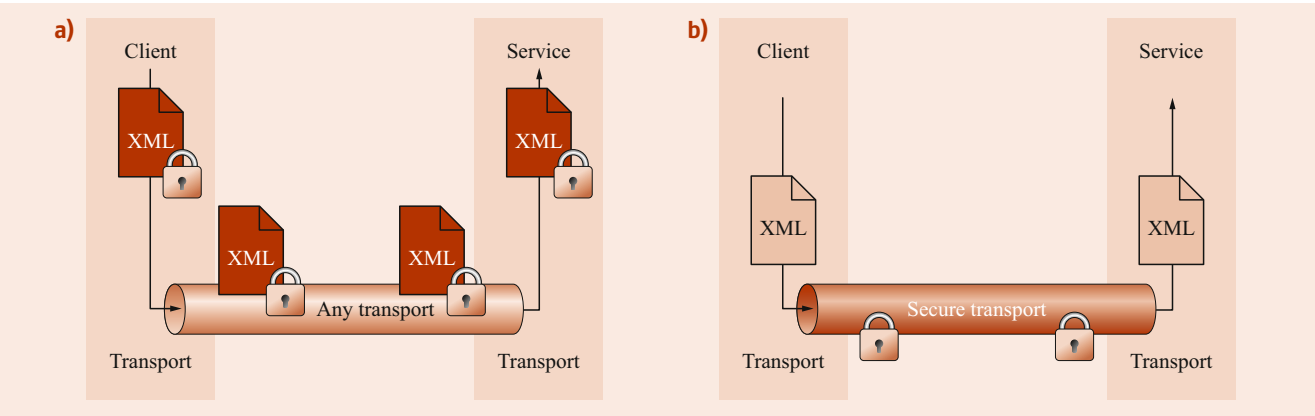

Fig. 19.3 Transport layer **(a)** versus application layer **(b)** integrity/confidentiality

WS-Security is a standard by OASIS that can be associated with the application layer of the OSI reference model. It defines how to use XML digital signature and XML encryption on SOAP messages to ensure confidentiality and/or integrity. Because how and which parts of the message are protected can be controlled by the application in a very flexible manner, WS-Security comes into play, as it defines exact patterns for applying a digital signature to an XML document (or parts of it) and how to encrypt parts of the document and create the relevant metadata for the receiver in XML to undo the encryption or use signed hash values to check integrity. As a full introduction to WS-Security and the related standards would exceed the size of this chapter; the interested reader is encouraged to follow the links given in the References.

With Web-APIs, the information exchange is not based on XML but on the much simpler JSON encoding. Even though HTTPS (HTTP over TLS) is typically used to ensure confidentiality and integrity of transmitted information, it is sometimes inevitable to submit information as part of the URL. In general, but in particular in these cases, the confidentiality and integrity of JSON-encoded information can be established by applying particular IETF specifications, as outlined in Sect. 19.3 and Fig. 19.2.

19.3.2 Standards for Implementing Authentication

The Security Assertion Markup Language V2 is an OASIS standard that first of all specifies a markup language for describing assertions about a subject. SAML2 distinguishes between three different types of assertions:

1. *Authentication assertion*, which provides information about the asserted subject regarding the means by which a subject was authenticated, by whom, and at which time.
2. *Attribute assertion*, which provides information about the characteristics of the asserted subject.
3. *Authorization assertion*, which states that access to a particular resource is permitted or denied for the asserted subject.

SAML2 is one ideal standard to implement authentication in distributed systems, where the user (principal) is known by the identity provider (asserting party) and the protected services are hosted by the service provider (relying party). These two are typically separate entities. To establish secure exchange of assertions concerning the identity of and additional information regarding the user between these parties, SAML2 specifies profiles and bindings. XML digital signatures and XML encryption or both can be applied to guarantee the integrity and or confidentiality of the assertions. The most important profiles are (not ordered)

- *Assertion query and request protocol*, which defines the processing rules for how existing assertions can be queried and the structure of the messages.
- *Authentication request protocol*, which enables the relying party to request assertion statements about the means by which a subject was authenticated.
- *Artifact Resolution Protocol*, which defines how SAML2 artifact references can be exchanged instead of the assertions itself.
- *Name Identifier Management Protocol*, which defines how an asserting party can change the name of an identifier that was previously established and is being used by relying parties.
- *Single Logout Protocol*, which defines a sequence of message exchange with the goal of terminating all existing sessions of the subject with other relying parties in close to real time. However, there is no confirmation message because the logout with all relying parties cannot be guaranteed.
- *Web Browser SSO Profile*, which defines how a Single-Sign-On (SSO) can be established using a (regular) Web browser as the client.
- *Enhanced Client or Proxy (ECP) Profile*, which defines the exchange of request/response messages for a client (not a Web browser) that knows which asserting party to contact.
- *Identity Provider Discovery Profile*, which defines mechanisms by which a relying party can discover which asserting parties a principal uses for the *Web Browser SSO Profile*.

The actual use of one or more of these profiles depends on the deployment environment for the services. To accommodate different characteristics, SAML2 defines multiple bindings for the profiles listed above:

- *SAML2 SOAP binding*, which defines how SAML2 assertions are to be exchanged using SOAP messages and how SOAP header elements are to be used to do so.
- *Reverse SOAP (PAOS) binding*, which describes a mechanism where the client is able to act as a SOAP relay relevant for implementing the ECP profile.
- *HTTP Redirect binding*, which enables the exchange of SAML2 messages as Uniform Resource Locator (URL) parameters. To ensure the length limit of a URL is not exceeded, message encryption is used. This binding is relevant where HTTP user agents of restricted capabilities are involved in the message exchange.
- *HTTP POST binding*, which defines how SAML2 messages can be sent inside an (X)HTML form using base64 encoding.
- *HTTP artifact binding*, which defines how SAML2 request and response messages are exchanged using a ref-

erence – the artifact. This binding is essential for implementing the *artifact resolution profile*.

It is worth mentioning that the applicability of a binding depends on the identified threat model: the Internet threat model allows leveraging of any profile, whereas the browser threat model mandates the artifact profile. The artifact profile relies on a secure *back-channel* between the service and the identity provider to exchange the actual assertion(s). The client just gets hold of the artifact, which is a protected, Internet-wide unique reference to associated assertion(s). However, because the client is missing the keys to set up a trusted back-channel with the identity provider, this profile is safe even if the attacker has prepared the client to intercept and wire-tape the communication. With the browser POST profile, for example, the user assertion(s) is (are) pushed from the identity provider to the service provider through the client. A manipulated client could fetch the assertions and potentially use them to carry out attacks. To prevent this, encrypted and digitally signed SAML2 assertions can be exchanged via the client application.

An alternative approach using a Secure Token Service (STS) is defined in WS-trust [59]. Web Services Trust (WS-trust) is an OASIS standard that defines extensions to WS-Security for managing (issuing, renewing, canceling, validating) security tokens for the purpose of establishing brokered trust relations between Web services of communication partners through the exchange of secured SOAP messages. To support brokered trust, this standard introduces the concept of a STS. To use the STS in an interoperable way, XML message formats are defined. It is important to note that this specification does not define any security token types. It specifies how to deal with them to establish trust between Web services and or clients as not directly trusted communication partners.

19.3.3 Standards for Implementing Access Delegation and User Claims

When a user interacts with services via applications, it might be relevant to allow the application or the service to access protected data owned by the user. In this case, the immediate question becomes, how the user can provide credentials to the application or the service so that access to the user's resources becomes possible without the user disclosing the master credentials, i.e. username and password. The solution is access delegation, which allows the user to delegate a controlled set of rights to the application or service. The standard to achieve this is OAuth2 [45], which is a particular realization of the STS concept [60] adopted for HTTP that standardizes different protocols how access tokens are delivered. The OAuth2 framework defines how the Resource Owner can authorize an Authorization Server to release access tokens to applications so that they can access the user's protected resources hosted at the Resource Server. Any application must be registered with the Authorization Server before it can obtain an access token. This registration process is typically a manual interaction of the application developer and the Authorization Server's registration page. For the simplification of the registration process, additional RFCs [61] and [62] can be leveraged.

As OAuth2 is just concerned with access delegation, applications and services have no information about the acting user; the user acts so to say anonymously. In modern Web applications this is insufficient as personalization and proper salutation is not possible at all.

OpenID Connect [48], which must be considered a community standard because it was not released from a major standardization body, is the extension to the OAuth2 framework that allows the application (or the service) to obtain user information. To better control which pieces of user information can be obtained, OpenID Connect specifies the concept of scopes. A scope like profile, email, or address represent a particular set of user attributes, called claims in OpenID Connect. If compared to SAML2, OpenID Connect defines a simplified version of the Attribute Authority concept. To manage trust with applications and which OpenID Connect scopes an application can request, the application must be registered with the Authorization Server with the scopes it want to use. When executing the application, the user must approve the application to access the user information possible via the authorized scopes. This concept does not exist in SAML2 Attribute Authority.

In the OAuth2 specification, the Resource Server is the passive component that accepts and processes access tokens. The OAuth2 Bearer Token Usage RFC [46] guarantees interoperability and ensures proper processing of access tokens submitted by an application. The OAuth2 Token Introspection [47] defines the interface of an Authorization Server that can be leveraged by the Resource Server to validate access and refresh tokens and to obtain additional metadata for a token.

19.3.4 Standards for Implementing Access Control

The major concern of access control is to prevent unauthorized use or disclosure of protected information. The typical solution is to assign identity rights on objects for particular actions that can be invoked on the object. This is a very challenging task already and becomes even more complicated for a distributed system because harmonization of access rights

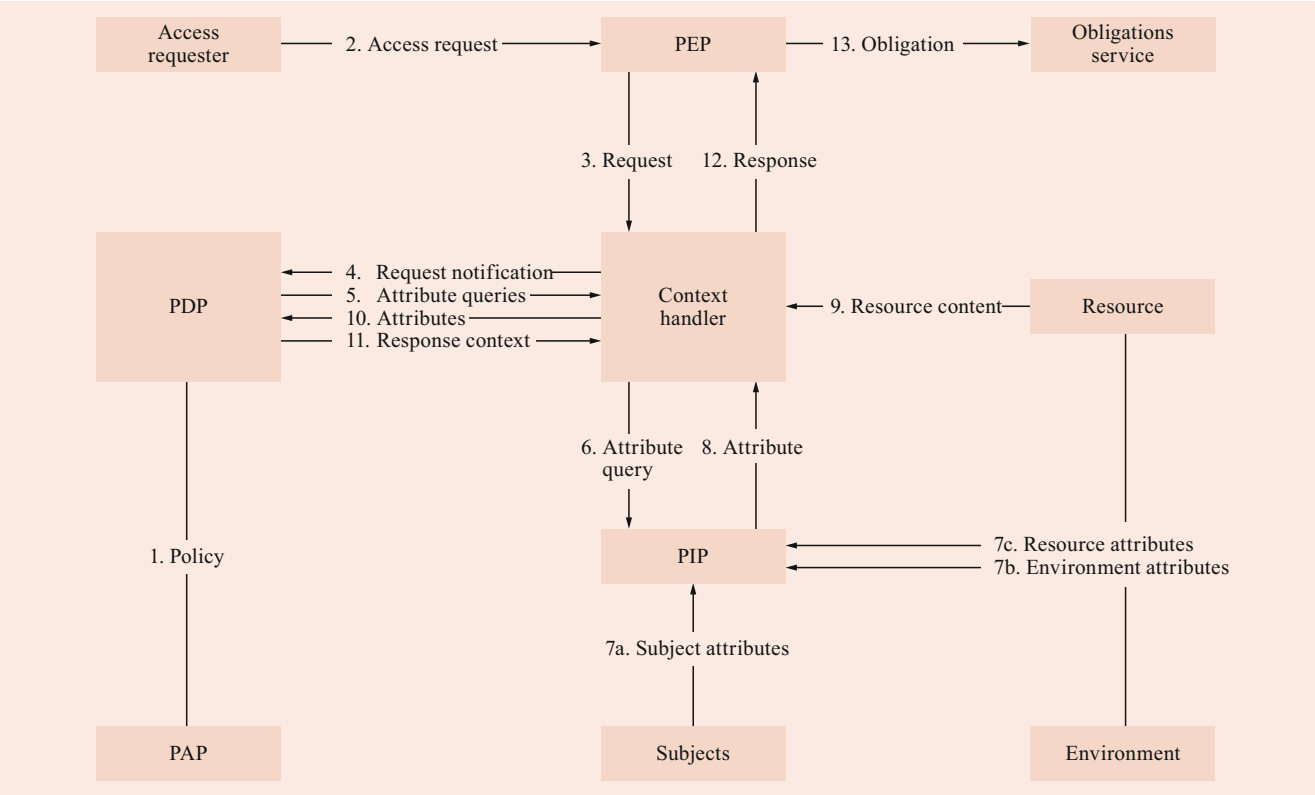

Fig. 19.4 XACML v2 information flow

across jurisdictions requires a language, so that rights declared by one party can be interpreted unambiguously by another involved party.

The Extensible Access Control Markup Language (XACML) by OASIS defines such a language to support the declaration of access rights in XML. It is also possible (of course) to derive authorization decisions based on the rights declared in the policy and an authorization decision request. As the service that derives the decisions (a so-called PDP, Policy Decision Point) can be deployed as an autonomous service, XACML defines the interface and the message format for the XACML authorization decision request and the XACML authorization decision response. XACML V2 [28] mandates the use of XML encoded authorization decision requests and responses, but XACML V3 [57] also supports JSON encoded authorization decision requests and responses [63].

Based on the version of XACML, different profiles exist. The Role-Based Access Control (RBAC) profile defines how to model RBAC0 (pure RBAC) and RBAC1 (role inheritance) [64] in an XACML2 policy. It is important to note that XACML also supports the Bell–La Padula and Biba models to ensure valid information flow control. Through the use of obligations, it is possible to create events for security audit and alarms.

The request to a protected resource is intercepted by the policy enforcement point (PEP). Before the protected resource can be accessed, the PEP involves the context handler to obtain all information relevant to construct a XACML authorization decision request to the Policy Decision Point (PDP). This can involve fetching resource information, and information on the user and the environment through a Policy Information Point (PIP). The PDP, on receiving the authorization decision request, derives an authorization decision based on available policy(ies). The decision is sent back to the PEP, which permits or denies the intercepted request. A decision received from the PDP can optionally contain an obligation, which is to be executed when permitting or denying the request. The Policy Administration Point (PAP) is not involved in runtime processing, as it provides an administrative interface for the creation and maintenance of policies.

As the declaration and enforcement of geo-specific access rights is not supported by XACML, the OGC has drafted a geo-specific extension to XACML 3 [58] and released a standard as the geo-specific extension to XACML 2.0 called Geospatial eXtensible Access Control Markup Language (GeoXACML) 1.0, which builds on top of XACML by using the available extension points. It extends XACML 2.0 by defining the data type *Geometry* and geo-specific functions based on ISO 19125-1 *Geographic information – Sim-*

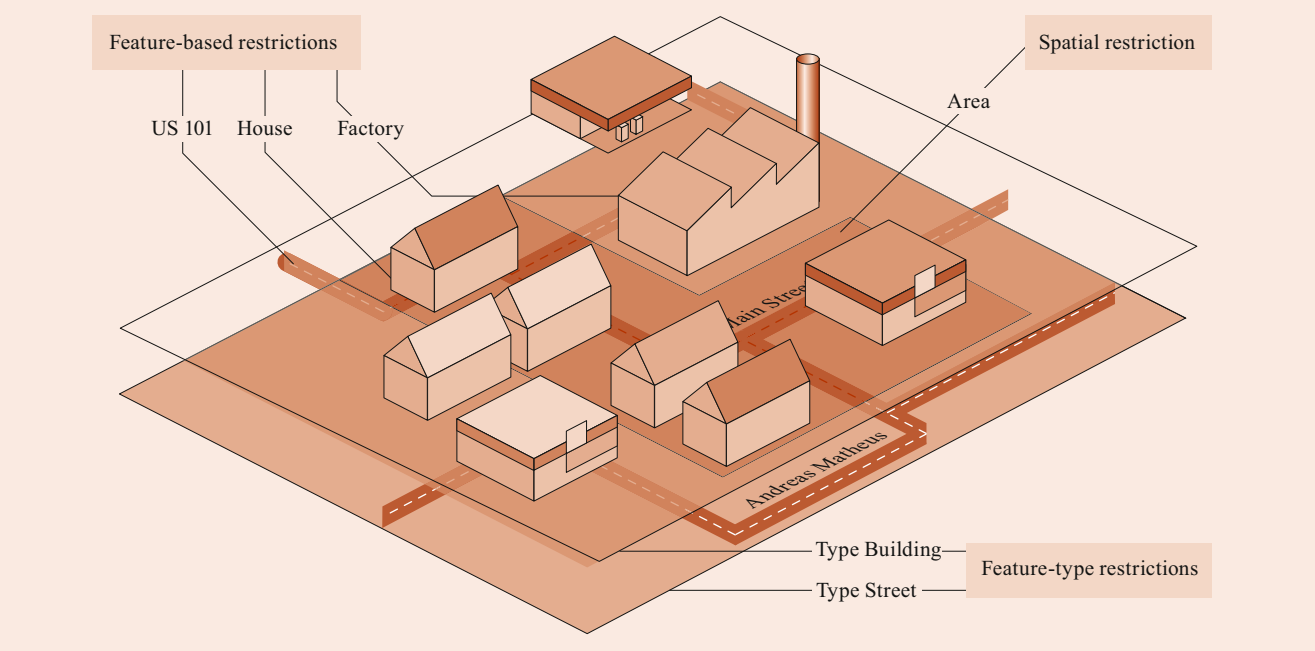

Fig. 19.5 GeoXACML access right example

ple feature access – Part 1: Common architecture, which is identical to OGC document #06-103r3 [65]. The functions allow testing and processing of geometries involved in the process of deriving an authorization decision.

Topological functions allow testing of the topological relation between two geometries; *bag and set functions* allow construction of results or test conditions based on a collection of geometries. Note that the XACML standard defines a bag as an unordered collection of elements with possible duplicates, whereas a set is considered free of duplicates. *Geometric functions* contains constructive and scalar functions for processing new geometries or to request characteristics of a geometry. Finally *conversion functions* (not from ISO 19125-1) support the conversion of length and area values to meters, the mandatory unit of measure.

GeoXACML defines two conformance classes that apply to an implementation of the Policy Decision Point (PDP) as it is a part of the XACML standard informative component diagram (Fig. 19.4). The conformance class BASIC requires a PDP implementation to support the functions listed as topological, bag/set, and conversion functions. The STANDARD implementation of a PDP requires implementation of all functions mandatory for the BASIC conformance class plus the functions listed as geometric functions. In addition, a BASIC or STANDARD implementation must also implement at least one extension (or perhaps all). Currently, the GeoXACML 1.0 core specification is accompanied by two extensions that support the OGC standards GML2 [66] and GML3 [67] encoding of geometries. Because GeoXACML defines an extension to XACML, all of its profiles can be used with GeoXACML too.

Figure 19.5 summarizes the typical capabilities of GeoXACML to control access to a geographic feature.

Rights can be associated with feature types, a particular area, or individual features, as illustrated in Fig. 19.5. As these different types of rights can be combined in any way, one can create very flexible and relevant access policies.

Leveraging the temporal capability from XACML, one can declare and enforce spatial-temporal access restrictions leveraging the Attribute Based Access Control model.

19.4 Summary

Securing a distributed geospatial system mainly involves non geo-specific standards – it requires knowledge of mainstream information technology (IT) to leverage existing standards and implementations in an appropriate way. In this chapter, we introduced an important set of standards covering this subject and associated them with different implementations of SOA: Web Services and Web APIs. The only identified requirement that is geo-specific is access control. Here, an existing standard from the OGC supports the declaration and enforcement of spatio-temporal access rights for geographic information.

References

1. OASIS: Reference Model for Service Oriented Architecture 1.0, OASIS Standard (2006). http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf
2. WSDL: Web Services Description Language (WSDL) 1.1, W3C Note (2001). http://www.w3.org/TR/wsdl
3. SOAP: Simple Object Access Protocol (SOAP), W3C Recommendation, 2nd ed. (2007). http://www.w3.org/TR/soap/
4. Web Services Security: SOAP Message Security 1.1 (WS-Security 2004) – OASIS Standard Specification (2006). http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
5. WS-SecurityPolicy: WS-SecurityPolicy 1.2, OASIS Standard (2007). http://docs.oasis-open.org/ws-sx/WS-Securitypolicy/200702/WS-Securitypolicy-1.2-spec-os.pdf
6. HTTP 1.1: IETF RFC 2616 – Hypertext Transfer Protocol – HTTP/1.1 (1999)
7. Bray, T.: The JavaScript Object Notation (JSON) Data Interchange Format. IETF 8259. (2017)
8. Barth, A.: The Web Origin Concept. IETF 6454. (2011)
9. United States Government Department of the Defense: Trusted Computer System Evaluation Criteria (1985)
10. ISO/IEC 10181-1:1996: Information Technology – Open Systems Interconnection – Security Frameworks for Open Systems: Overview (1996). http://www.iso
11. ISO/IEC 10181-2:1996: Information Technology – Open Systems Interconnection – Security Frameworks for Open Systems: Authentication Framework (1996). http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=18198
12. ISO/IEC 10181-3:1996: Information Technology – Open Systems Interconnection – Security Frameworks for Open Systems: Access Control Framework (1996). http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=18199
13. ISO/IEC 10181-4:1996: Information Technology – Open Systems Interconnection – Security Frameworks for Open Systems: Non-Repudiation Framework (1996). http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=23615
14. ISO/IEC 10181-5:1996: Information Technology – Open Systems Interconnection – Security Frameworks for Open Systems: Confidentiality Framework (1996). http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=24329
15. ISO/IEC 10181-6:1996: Information Technology – Open Systems Interconnection – Security Frameworks for Open Systems: Integrity Framework (1996). http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=24330
16. ISO/IEC 10181-7:1996: Information Technology – Open Systems Interconnection – Security Frameworks for Open Systems: Security Audit and Alarms Framework (1996). http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=18200
17. ISO 7498-2:1989: Information Processing Systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture (1989)
18. Bell, D.E., LaPadula, L.J.: Secure Computer Systems: Unified Exposition and Multics Interpretation. MTR-2997 Rev. 1. MITRE, Bedford (1976)
19. Biba, K.J.: Integrity Considerations for Secure Computer Systems. MTR-3153. MITRE, Bedford (1977)
20. IETF RFC 1457: Security Label Framework for the Internet (1993). http://tools.ietf.org/pdf/rfc1457
21. IETF RFC 3552: Guidelines for Writing RFC Text on Security Considerations (2003). http://tools.ietf.org/pdf/rfc3552
22. van Kesteren, A.: Cross-Origin Resource Sharing. W3C Recommendation. (2014)
23. ITU-T: X.200: Information Technology – Open Systems Interconnection – Basic Reference Model: The Basic Model (1994). https://www.itu.int/rec/T-REC-X.200-199407-I
24. IPSec: IP Security – IETF RFC 4301 (2005). http://tools.ietf.org/html/rfc4301. obsoletes RFC 2401 from 1998
25. XML Digital Signature: XML-Signature Syntax and Processing – W3C Recommendation (2002). http://www.w3.org/TR/xmldsig-core/
26. XML Encryption: XML Encryption Syntax and Processing – W3C Recommendation (2002). http://www.w3.org/TR/xmlenc-core/
27. WS-Policy: Web Services Policy 1.5 – Framework, W3C Recommendation (2007). http://www.w3.org/TR/2007/REC-WS-Policy-20070904/
28. XACML 2.0: eXtensible Access Control Markup Language (XACML) Version 2.0, OASIS Standard (2005). http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf
29. XACML RBAC Profile: Core and Hierarchical Role Based Access Control (RBAC) Profile of XACML v2.0, OASIS Standard (2005). http://docs.oasis-open.org/
30. XACML SAML Profile: SAML 2.0 Profile of XACML v2.0, OASIS Standard (2005). http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf
31. Open Geospatial Consortium: GeoXACML 1.0: Geospatial eXtensible Access Control Markup Language (GeoXACML) v1.0 (2008). http://portal.opengeospatial.org/files/?artifact_id=25218
32. Open Geospatial Consortium: GeoXACML Extension A: Geospatial eXtensible Access Control Markup Language (GeoXACML) Extension A – GML2 Encoding Version 1.0 (2007). http://portal.opengeospatial.org/files/?artifact_id=25219
33. Open Geospatial Consortium: GeoXACML Extension B: Geospatial eXtensible Access Control Markup Language (GeoXACML) Extension B – GML3 Encoding Version 1.0 (2007). http://portal.opengeospatial.org/files/?artifact_id=25220
34. ISO/IEC 21000-5:2004: Information Technology – Multimedia Framework (MPEG-21) – Part 5: Rights Expression Language (2004). https://www.iso.org/standard/36095.html
35. ODRL: Open Digital Rights Language (ODRL) Version 1.1, W3C Note (2002). http://www.w3.org/TR/odrl/
36. XrML: XrML – eXtensible Rights Markup Language, ContentGuard (2010). http://www.xrml.org/
37. X.509/PKI: IETF, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (2008). http://tools.ietf.org/html/rfc5280
38. SAML: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard (2005). http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf
39. SAML-Bindings: Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard (2005). http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf
40. SAML-Profiles: Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard (2005). http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf
41. Kerberos: Kerberos: The Kerberos Network Authentication Service (V5) (1993). http://tools.ietf.org/html/rfc4120. IETF RFC 4120 (2005) obsoletes 1510
42. LDAP: Lightweight Directory Access Protocol (LDAP): The Protocol – IETF RFC 4511 (2006). http://tools.ietf.org/html/rfc4511
43. IETF: The X.500 String Representation of Standard Attribute Syntaxes: IETF RFC (1993). http://tools.ietf.org/html/rfc1488
44. Mogul, J.: Instance Digests in HTTP. IETF RFC 3230. (2002)
45. Hardt, D.: The OAuth 2.0 Authorization Framework. IETF RFC 6749. (2012)
46. Jones, M.: The OAuth 2.0 Authorization Framework: Bearer Token Usage. IETF RFC 6750. (2012)
47. Richer, J.: OAuth 2.0 Token Introspection. IETF 7662. (2015)

48. OpenID Connect: OpenID Connect Core 1.0 Incorporating Errata Set 1 (2014). http://openid.net/specs/openid-connect-core-1_0.html
49. Jones, M.: JSON Web Signature (JWS). IETF RFC 7515. (2015)
50. Jones, M.: JSON Web Token (JWT). IETF 7519. (2015)
51. Jones, M.: JSON Web Encryption (JWE). IETF 7516. (2015)
52. Jones, M.: JSON Web Key (JWK). IETF 7517. (2015)
53. Jones, M.: JSON Web Algorithms (JWA). IETF 7518. (2015)
54. OpenID Discovery: OpenID Connect Discovery 1.0 Incorporating Errata Set 1 (2014). https://openid.net/specs/openid-connect-discovery-1_0.html
55. Jones, P.: WebFinger. IETF RFC 7033. (2013)
56. OpenAPI: The OpenAPI Specification (2021). https://github.com/OAI/OpenAPI-Specification
57. XACML 3.0: eXtensible Access Control Markup Language (XACML) Version 3.0, OASIS Standard (2013). http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf
58. Open Geospatial Consortium: GeoXACML 3.0 DRAFT Standard: Geospatial eXtensible Access Control Markup Language (GeoXACML) v3.0 (2013). https://portal.opengeospatial.org/files/?artifact_id=55232
59. WS-Trust: WS-Trust 1.3, OASIS Standard (2007). http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf
60. WS-Trust: WS-Trust 1.4, OASIS Standard Incorporating Approved Errata 01 (2012). http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/errata01/os/ws-trust-1.4-errata01-os-complete.pdf
61. Richer, J.: OAuth 2.0 Dynamic Client Registration Protocol. IETF 7591. (2015)
62. Richer, J.: OAuth 2.0 Dynamic Client Registration Management Protocol. IETF 7592. (2015)
63. XACML 3.0: JSON Profile of XACML 3.0 Version 1.0 (2017). http://docs.oasis-open.org/xacml/xacml-json-http/v1.0/xacml-json-http-v1.0.html
64. Ferraiolo, D.F., Kuhn, D.R.: Role-based access control. In: 15th Natl. Comput. Secur. Conf. pp. 554–563. (1992). http://csrc.nist.gov/groups/SNS/rbac/documents/ferraiolo-kuhn-92.pdf
65. OGC: OGC: OpenGIS Implementation Specification for Geographic Information – Simple Feature Access – Part 1: Common Architecture (2006). http://portal.opengeospatial.org/files/?artifact_id=18241
66. OGC: OpenGIS Geography Markup Language (GML) Implementation Specification, Version 2.1.2 (2002). http://portal.opengeospatial.org/files/?artifact_id=11339
67. OGC: OpenGIS Geography Markup Language (GML) Encoding Standard, Version 3.2.1 (2007). http://portal.opengeospatial.org/files/?artifact_id=20509
68. HTTP 1.1: IETF RFC 7235 – Hypertext Transfer Protocol (HTTP/1.1): Authentication (2014)
69. Lodderstedt, T.: OAuth 2.0 Threat Model and Security Considerations. IETF RFC 6819. (2013)

Andreas Matheus Andreas Matheus received a Dr rer. nat. in Computer Science from the Technische Universität München and is Founder and Managing Director of Secure Dimensions GmbH, a company that develops interoperable security solutions by applying mainstream IT security standards to the geospatial domain. He is an active member of OGC, Chair of the OGC Security Working Group, and a re-elected member of the OGC Architecture Board since 2006. He is Editor of two OGC standards: GeoXACML 1.0 (2008) and OGC Web Services Security 1.0 (2018) as well as editor of the GeoXACML 3.0 draft standard.