Security Assurance of Services through Digital Security Certificates

Samuel Paul Kaluvuri
Applied Research Security&Trust
SAP Labs France
samuel.paul.kaluvuri@sap.com

Hristo Koshutanski
Computer Science Dept.
University of Malaga, Spain
hristo@lcc.uma.es

Francesco Di Cerbo
Applied Research Security&Trust
SAP Labs France
francesco.di.cerbo@sap.com

Antonio Maña
Computer Science Dept.
University of Malaga, Spain
amg@lcc.uma.es

***Abstract*—Service Oriented Computing (SOC) has facilitated a paradigm shift in software provisioning models: software gets consumed as a "service" providing enormous benefits, however lack of security assurance of third-party services is hampering their wider adoption in business- and security-critical domains. Security certification typically provides the required assurance, however applying it as is to SOC is infeasible, given that the natural language representation of resulting certificates is a major obstacle for typical SOC scenarios like service discovery, service composition and so on. To overcome the limitations of existing security certificates we present the concept of a digital security certificate for services. It is realized by a language which enables the representation of a security certificate in a structured, machine processable manner that would enable automated reasoning to be performed on them and thus make it feasible for certified security features to be part of typical SOC scenarios.**

I. Introduction and Motivation

Service Oriented Computing (SOC) has facilitated a paradigm shift in software provisioning models, such as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). These new provisioning models relieve consumers from the complexity of procuring and maintaining large-scale IT infrastructures, and provide significant economic benefits [1]. However, in such provisioning models, the consumer no longer has full control over the software used, nor its operational environment.

This lack of control, especially in critical domains such as finance, defence and healthcare, raises concerns about the security of these services [1]. Pre-established trust relations between consumers and providers, such as Service-Level Agreements, traditionally represent a mitigation to this situation. But they can be hardly established in a dynamic and scalable manner, that would fit in the service environment as new and cost-attractive offerings, from different service providers, are frequently launched. In traditional software provisioning models, security certification of software by trusted third party entities is used to provide security assurance to consumers. Certification schemes such as Common Criteria [2] are well established and quite successful in providing the required security assurance to consumers in a scalable manner. However, current certification schemes result in certificates that are represented in natural language, which do not cope well with the dynamic service environment.

For service consumers, the possibility to compare the (certified) security features of a service with their security requirements is a relevant aspect in the service selection process. However, security certificates represented in natural language prevent any sort of automated reasoning to be performed on them, and consequently neglect an adequate scalability of the selection process.

In order to make security certification beneficial in Service Oriented Computing (SOC), several modifications to the current state of the art are necessary [3]. Among them, this paper focuses on those related to the *security certificate representation*. In this regard, we have identified the following key requirements that need to be addressed in order to facilitate security certificate adoption in SOC.
Requirement **R1**: the security certificates must be machine processable in order to allow automated reasoning to be performed on them.
R2: security certificates should contain enough information about the certified entity so that they can cater to consumers with varying levels of security knowledge, such as regular users with limited security understanding to security experts of organizations. In other words, it is necessary that the certificates are *descriptive* [4], meaning, that they describe with sufficient details the security features of their services, together with supporting evidences.
R3: Mechanisms must exist in order to bind a service and its security certificate, given that a service implementation can change, while maintaining the same external interface or API. Consumers would need to have trustworthy and dynamic means to verify whether a service implementation they are using is the certified one.

Paper Contribution: We present our proposal for a digital security certificate concept (from now on referred as $\mathcal{CRT}$), that in our view addresses the mentioned security certification issues in SOC. In fact, $\mathcal{CRT}$s are designed to be machine readable (and thus addressing **R1**). They are designed to cope with existing web service interaction models and standards, thus to ease their adoption and integration in SOC. Moreover, $\mathcal{CRT}$s are designed to be completely *descriptive* and thus allowing automated reasoning upon them (to cope with **R2**). In particular, $\mathcal{CRT}$s can provide evidence of the presence and implementation of a service's security features, that are collected through a service evaluation phase; this permits further customer analysis, with respect to specific security requirements (**R2**). The binding between a service implementation and its security certificate (**R3**), even though it is a significant aspect, is not in the scope of this paper, even if the current form of $\mathcal{CRT}$ foresees a specific element for addressing

this issue.

The paper is structured as follows: Sect. II describes a SOC scenario for the application of the $\mathcal{CRT}$ concept, while Section III presents the state of the art in security certification and digital certificates. Sections IV and V depict different aspects of the $\mathcal{CRT}$, respectively its conceptual model and its technical representation. The $\mathcal{CRT}$ contribution to achieve security assurance is discussed in Sect. VI, and the following Sect. VII concludes the paper.

II. Scenario

Digital Security Certificates can impact on many SOC aspects. One typical application scenario is represented by an assessment of a service's security features during a service discovery process. Let us consider the example of a hypothetical cloud storage service, *Titanium Box*. We assume that the service runs on the Apache CXF framework, and uses Amazon S3 service in the back-end to store the user files after encrypting them. This architecture is similar to the popular cloud storage service, Dropbox [5].

When such a service undergoes security certification process, similar to the Common Criteria, it results in a security certificate that is captured in human readable form. Clearly, such a certificate does not allow automated reasoning to be performed, thus not supporting assessments and comparisons among alternatives which is essential in scenarios such as service discovery. In the next sections, we present the concept of a *Digital Security Certificate* , and we will use Titanium Box description to materialize its relevant parts.

III. Related Work

1) Security Certification Schemes: A survey of the current security certification schemes reveals that there are quite a few established and successful schemes such as Common Criteria for Information Security (CC), Commercial Product Assurance (CPA) and so on. Security certification schemes can be broadly classified based on the domains that they are applicable in, the recognition of the certification schemes, the descriptive or normative character of the issued certificates and so on. Among the existing schemes, CC is a widely recognized [6], used [7], multi-domain [8], partially descriptive certification scheme [9]. The CC scheme is very generic, as it aims to evaluate security of products that range from software, firmware to hardware. It avoids an all or nothing benchmark, by providing security assurance at varying levels, called Evaluation Assurance Levels (EAL), this provides flexibility for product vendors to get their product certified at lower assurance levels and improve the EAL over time. The CC scheme is primarily "claims" based, where the vendor makes claims about the security functionalities in the product in a document called "Security Target" (CC-ST) [10]. However, consumers can specify their requirements in a document called "Protection Profile" (CC-PP), and vendors can build products that conform to a Protection Profile (and claim conformance in the CC-ST). The CC-ST is the descriptive part of the CC scheme, containing the Target of Evaluation (CC-TOE) and the standardized Security Functional Requirements (SFRs) [11] and Security Assurance Requirements (SARs) [12] that are met by the product. The standardized SFRs and SARs are the "common" part of the CC scheme allowing, in theory, comparison of security features of certified products.

However, in practice, the comparison of products which have different "claims" can be very hard. This is due to the representation of the CC related documents (CC-PP, CC-ST) in natural language, which is often filled with *legalese and heavy security jargon* making it rather complex to understand for non-security experts. Hence it becomes extremely difficult to determine if a particular product satisfies a consumer's security requirements and to compare different products against their requirements. It was observed [3] natural language representation of certificates is not a scalable solution when we consider service marketplaces such as Google Apps for business, Salesforce etc. There, the analysis of thousands of application/service offerings and their human readable security certificates would represent an unsustainable burden for customers; though the availability of security certificates could represent a means to gain assurance on offerings, it cannot facilitate any sort of automated reasoning such as compare and/or contrast the security properties of different services. It also prevents consumers to search for services based on not only their functional but also security requirements.

2) Lack of digital representation of Security Certificates: The resulting security certificates from current security certification schemes are not represented in a digital format. Although there are a few "digital security seals" such as the TRUSTe privacy seal [13], McAfee SECURE seal [14] and so on, these seals are purely normative statements regarding the security feature of an entity, which can be seen as a step towards digital security certificates, but cannot provide any meaningful assurance to consumers as they do not contain any information regarding the certified entity.

3) Container for Digital Security Certificates: In order to facilitate an easier and faster adoption of $\mathcal{CRT}$s in the *SOC*, we choose to use the existing standards as *Containers* for the $\mathcal{CRT}$. In this regard, we have considered the *digital certificate* standards, that are primarily used for identity and authorization management, as possible candidates given their widespread usage and acceptance. Among the existing standards, X.509 [15], SAML [16] are the most widely used in practice. Both standards support public-key (identity) certificates and attribute certificates for purposes of user authentication and authorization. The *attribute* certificates of X.509 and SAML standards support extensibility of the attribute part of the certificate to accommodate domain-specific data. This aspect makes both standards suitable to provide a PKI-compliant container for encapsulating the content of $\mathcal{CRT}$s.

IV. Conceptual Model of a Security Certificate

The conceptual model for the $\mathcal{CRT}$, is designed to capture information emanating from security certification processes. In particular, we have considered the CC scheme, as it is the most widely used scheme currently. The CC-ST, which is the descriptive part of the CC scheme, serves as a foundation for our $\mathcal{CRT}$. However, we have extended this significantly, in order to make it machine processable (**R1**) and suitable for service-specific needs. In contrast to CC, and other existing certification schemes, the digital security certificate is designed to be completely descriptive [4] (**R2**), and hence it contains the

description of the certified entity, the security properties of the certified entity, the evaluation details that prove the certified properties.

Definition 1. *The digital security certificate is a tuple* $\mathcal{CRT} = \langle \mathcal{SD}, \mathcal{SPS}, \mathcal{ESD}, \mathcal{UDE} \rangle$ *where,* $\mathcal{SD}$ *is the service description,* $\mathcal{SPS}$ *is the security property specification,* $\mathcal{ESD}$ *is the evaluation specific description and* $\mathcal{UDE}$ *is User defined extensions.*

The $\mathcal{SD}$ provides details about the service and its underlying architecture, thereby mitigating the concerns of the consumers on the lack of transparency of services since services just expose an interface and the internal dynamics of the service and its architecture are hidden from the consumer. The $\mathcal{SPS}$ provides details about the security properties of the service at varying levels of abstraction. The $\mathcal{ESD}$ provides details regarding the evaluation process and its results, such as the test suites that were executed or the formal models and proofs used to verify and validate the security properties of the service. While the *User Defined Extensions* ($\mathcal{UDE}$) can be either used by the service providers to disclose any additional information and/or by the certification authorities to state any further criteria. These four elements serve different purposes and together contribute in providing assurance on the security of the service.

A. Service Description

In (CC-ST), the assets are described in natural language and no identifiers are provided for them; therefore, an explicit link cannot be made between the security properties and the assets that they secure. In order to overcome this we adopt a asset-centric approach with explicit references between the assets and the different elements in the certificate.

Definition 2. *An Asset, a, is an entity that is of some value to the consumer or the provider. Assets can be data, applications, the IT equipment on which the service operates or even users of the Information System.*

The CC-ST contains the Target of Evaluation (TOE) that describes the system that is being certified and the the boundaries of the evaluation are indicated, albeit in an ad-hoc manner. However for a machine readable certificate there should be a clear distinction between the system that is being certified and the aspects of the system that are subject to evaluation. It is of utmost importance in service based systems, due to the fact that services can be easily composed of external services and this information should be a part of the service description but clearly marked as outside the scope of evaluation.

The TOE in a CC-ST also contains the system architecture, the different components that compose the system among other information such as configuration in which the system is evaluated, the underlying IT architecture etc., and this is represented in natural language accompanied by architecture diagrams. This poses another issue in representing the TOE in a machine processable manner. So in order to address these two issues, we introduced an element called Target of Certification ($\mathcal{TOC}$) that describes the service being certified, in addition the TOE which describes the part of the Target of Certification that is evaluated.

Definition 3. *A Target of Certification is a tuple* $\mathcal{TOC} = \langle \mathcal{ACI}, \mathcal{DM}, \mathcal{TT} \rangle$*, where* $\mathcal{ACI}$ *is Asset-Component Identification,* $\mathcal{DM}$ *is the Deployment and Implementation Model and* $\mathcal{TT}$ *is the TOC Type.*

Definition 4. *An Asset-Component Identification is a tuple* $\mathcal{ACI} = \langle \mathcal{A}, \mathcal{C}, \alpha \rangle$*, where* $\mathcal{A}$ *is a set of all the assets identified for the TOC,* $\mathcal{C}$ *is a set of all the components in the TOC and* $\alpha \subseteq \mathcal{A} \times 2^{\mathcal{C}}$ *maps each Asset with a set of Components.*

Definition 5. *The* $\mathcal{TOE}$ *is a subset of the Asset-Component Identification.* $\mathcal{TOE} \subseteq \mathcal{ACI}$

The TOC *Components* are an integral part of the $\mathcal{TOC}$ as they allow the $\mathcal{TOC}$ to be expressed in a modular and structured manner. It comprises an abstract model of the Component, the *Component Model*: it can be as simple as just containing the interfaces of the component, or a more detailed specification of the internal dynamics of the component as deemed sufficient by the *Certification Authorities*. It must also contain technical specifications of the Component, again at the level of abstraction as deemed sufficient.

Definition 6. *A Component is a tuple* $\mathcal{C} = \langle \mathcal{CM}, \mathcal{TM} \rangle$*, where* $\mathcal{CM}$ *is the component model and* $\mathcal{TM}$ *is the technical model.*

We do not provide a formal definition for $\mathcal{CM}$ and $\mathcal{TM}$ as such concepts can be considered atomic for the sake of our work.

A *Security Problem Definition* (spd) is essential in a security certificate as it provides the rationale for securing the assets. The rationale for securing the assets can stem from the threats that are identified for the assets by the service provider or from the service provider's security policy (which in turn could be due to compliance to regulations etc.).

Definition 7. *The security problem definition is a tuple* $spd = \langle \hat{\mathcal{A}}, spr \rangle$*, where* $\hat{\mathcal{A}} \subseteq \mathcal{A}$ *is a set of assets that need to be secured and* spr *is a security problem rationale for securing the assets.*

Definition 8. *The security problem rationale (*spr*) is a union of threats* $\mathcal{T}$ *and service provider's security policy* $\mathcal{SSP}$ *.* $spr = \mathcal{T} \cup \mathcal{SSP}$

The service description must contain the description of the certified system, the part of the system that is evaluated and the rationale for protecting the assets that are identified.

Definition 9. *The Service Description is a tuple* $\mathcal{SD} = \langle \mathcal{TOC}, \mathcal{TOE}, \mathcal{SPD} \rangle$ *where,* $\mathcal{SPD}$ *is the set of security problem definitions(*spd*).*

B. Security Property Specification

The CC-ST contains a vast amount of information but is often presented in heavy-jargon; this rarely allows a consumer (a non security expert) to get a high level perspective of the security features provided by the software/service. Hence we introduced a new element in the $\mathcal{CRT}$ model called as "security property specification" which enables a fine grained description of the security property that originates from a multi-layered model. It comprises of different elements, from abstract security properties to concrete security mechanisms.

Definition 10. *An* Abstract Security Property $\hat{p}$ *is an atomic security attribute for an asset.*

For example, abstract security properties can be confidentiality, integrity, availability, authenticity, non-repudiation, utility, privacy and so on.

Since abstract security properties by themselves do not convey any information on how the property is applied, there is a need for contextual information. Hence we define *Contextual Security Property*.

Definition 11. *A Contextual Security Property is an abstract security property realized in a certain context.* $\hat{p_c} = \langle \hat{p}, c \rangle$ *where* c *is a context.*

Contexts depend on the abstract security property. Abstract security properties that are data centric such as the *CIA triad* can have contexts such as transit, rest and usage. Such as *Confidentiality in rest* and *Integrity in transit*. However these properties still a subject, i.e., no indication of "what" is being secured. This is addressed by the certified security property.

Definition 12. *A certified security property,* p*, is a contextual security property (* $\hat{p_c}$*) applied on a set of assets (*$\hat{A}$*).* $p = \hat{p_c} \times \hat{\mathcal{A}}$

The (certified) security property provides a high level overview of how an asset is secured. But this does not provide any information on how the $\mathcal{SPD}$ are addressed. This is overcome by using the concept of "Security Objectives" similar to the CC scheme. A security objective, so, counters, mitigates or detects a spd that is identified for the $\mathcal{TOE}$ and contributes to the realization of a security property p for the TOE.

Definition 13. *A security objective is a tuple* $so = \langle \mathcal{O}, \mathcal{OT}, \widehat{\mathcal{SPD}} \rangle$*, where* $\mathcal{O}$ *is the objective,* $\mathcal{OT}$ *is the objective type,* $\widehat{\mathcal{SPD}} \subseteq \mathcal{SPD}$ *is a set of security problem definitions.*

All the security objectives are necessary and sufficient conditions to realize the security property. In other words, a TOE can have a security property p if and only if all the security objectives for the TOE are satisfied. Security Objectives are realized by security mechanisms that should be implemented in the TOE.

A Security Mechanism, sm, is an action, device, procedure, or a technique that meets or opposes (counters) a threat or an attack by eliminating or preventing it, by minimizing the harm it can cause or by discovering and reporting it so that corrective action can be taken. Security mechanisms refer to the security objectives that they satisfy, and they can be mapped to specific functional criteria of a particular certification schemes.

Definition 14. *A security mechanism is a tuple* $sm = \langle \mathcal{M}, \mathcal{SFC}, \widehat{\mathcal{SO}} \rangle$ *where,* $\mathcal{M}$ *is the mechanism that is implemented,* $\mathcal{SFC}$ *is the security functional criteria of a certification scheme and* $\widehat{\mathcal{SO}} \subseteq \mathcal{SO}$ *is a set of security objectives that the mechanism realizes.*

Definition 15. *A Security Property Specification is a tuple* $SPS = \langle \mathcal{P}, \mathcal{SO}, \mathcal{SM}, \gamma, \eta \rangle$ *where* $\mathcal{P}$ *is a set of certified security properties,* SO *is a set of Security Objectives,* SM *is a set of Security Mechanisms ,* $\gamma \subseteq \mathcal{P} \times 2^{\mathcal{SO}}$ *maps each security property to a set of security objectives and* $\eta \subseteq \mathcal{SO} \times 2^{\mathcal{SM}}$ *maps each security objective to a set of security mechanisms.*

This fine grained representation has two major advantages: allows consumers with varying security understanding to gain understanding of the security features provided by the service (security properties to security mechanisms); allows the certified security property to be machine processable that enables consumers to easily search for services that match their security requirements.

C. Evaluation Specific Details

The $\mathcal{ESD}$ defines the representation of the details and results of the service evaluation process needed to support the certified security property. The details of the evaluation specific portion in the certificate fall beyond the scope of this paper, however, we identified these three different categories for evaluation of services: Evaluation through testing [17], [18], Evaluation through formal analysis [19], and Evaluation through ontology-based analysis [20].

V. Realization of the Conceptual Model

In order to realize the conceptual model of the digital certificate $\mathcal{CRT}$, we have developed an XML-based language that enables the representation of the certificate in a machine processable form, which from henceforth we refer to as an `ASSERT`. A detailed version of the schema can be found in [21] and in this section we will explain its most relevant elements using the example introduced in Section II.

A. SAML as Container of ASSERT

The management and exchange of the `ASSERTS` is an important consideration for a successful implementation of a certification ecosystem life-cycle, i.e., production, maintenance, consumption of certificates. In this context, the container of the `ASSERTS` assumes significant importance as it is needed to encapsulate the certificate data into an interoperable format that can be used with existing web service standards and technologies. We have chosen the SAML standard [16] as a container because it is widely used in decentralized systems for its support for request and exchange of "SAML Assertions", be that for authentication or authorization of entities, or any attributes of an entity. The SAML standard has support for several standard profiles for usage of SAML tokens in specifications such as WS-Security [22], WS-SecurityPolicy [23], WS-Trust [24], etc. These aspects make SAML a good choice to be a container for exchanging `ASSERTS` in service environment. We use the SAML Assertion tokens to encapsulate `ASSERT`-specific data.

Figure 1 shows the main elements of the SAML assertion token structure where the `<Statement>` element defines an abstract statement of an assertion. We extended this element[1] to provide a statement about a service's description, its security property along with the corresponding evidence. The standard field *Issuer* in the SAML token is used as a means to capture the `ASSERT` Issuer's identity (the certification authority

[1] similar to how SAML authentication and authorization decision statements extend the abstract <Statement> element.

```
<ns1:Assertion ID="..." IssueInstant="..." Version="2.0"
 xmlns:ns1="urn:oasis:names:tc:SAML:2.0:assertion"
 xmlns:ns2="urn:assert4soa:assert:2.0">
 <ns1:Issuer/>
 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#"/>
 <ns1:Subject/>
 <ns1:Conditions NotBefore="..." NotOnOrAfter="..."/>
 <ns1:Statement SerialNumber="..." Version="2.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="ns2:ASSERTSAMLAssertionStatementType">
   <ASSERTCore>
     <CertificationProcess>
       <TargetOfCertification/>
       <SecurityProblemDefinition/>
       <CertificationCriteria/>
       <PerformedBy/>
       <EvaluationDate/>
     </CertificationProcess>
     <SecurityProperty/>
     <ServiceBinding/>
     <ASSERT4Humans/>
     <ASSERTSigner/>
   </ASSERTCore>
   <ASSERTTypeSpecific>
     <Property/>
     <ServiceModel/>
     <Results/>
   </ASSERTTypeSpecific>
   <UserDefinedExtensions/>                    ASSERT
 </ns1:Statement>
</ns1:Assertion>                                  SAML
```

Fig. 1. SAML Assertion Token as Container of ASSERT Data

issuing the ASSERT). The *Subject* field represents the identify of the certificate requester, which in most cases will be the service provider. And the validity conditions and the signature data are inherent to all security tokens.

B. ASSERT Structure

Figure 1 shows main elements of the ASSERT structure. It has three major elements: *ASSERTCore*, *ASSERTTypeSpecific* and *UserDefinedExtensions*. The *ASSERTCore* part contains elements that are independent of the evaluation of a service, i.e. the $\mathcal{SD}$ and the $\mathcal{SPS}$ elements in the conceptual model. The evaluation information in the conceptual model, i.e. $\mathcal{ESD}$, is contained in the *ASSERTTypeSpecific* element, while the $\mathcal{UDE}$ is captured in its namesake element *UserDefinedExtensions*.

1) ASSERTCore: The *ASSERTCore* element contains, in addition to the $\mathcal{SD}$ and $\mathcal{SPS}$, elements such as *ServiceBinding* that provides a robust link between the service and its ASSERT, *CertificationProcess* that provides information related to the certification process of a given service, and a textual description of the certificate in the *Assert4Humans* element where the certified service and the certified property are explained in natural language for end-user comprehension. The *AssertSigner* element identifies the entity that signs the ASSERT, while the *PerformedBy* element in the *CertificationProcess* identifies the entity who performed the service evaluation. Since multiple entities can be involved in a certification process, for example the ASSERT issuing process and service evaluation process may be undertaken by different entities, we provide this feasibility so as to increase the accountability during the production of certificates. In order to better illustrate the ASSERT language we provide code excerpts from the ASSERT of the example we provided in Section II.

```
<TargetOfCertification
 Type="http://assert4soa.eu/ontology/a4s-language#Software-as-a-service">
 <Description>TitaniumBox is a secure file storage service…</Description>
 <TOCComponents>
  <TOCComponent ComponentServiceRef="SN.userdatastorage"
   ID="TOC.service_operation_upload"
   InTargetOfEvaluation="true">
   <Description>Defines an operation of a SOAP based web service of
    TitaniumBox to upload file data.</Description>
   <TechnicalModel>
    <Description>Depends on the Apache CXF service framework.
     Web service implementation is developed in Java. </Description>
   </TechnicalModel>
  </TOCComponent>
  <TOCComponent ID="TOC.amazon_s3_service"
   InTargetOfEvaluation="false">
   <Description>Defines a storage service offered by a third-party, Amazon,
    used internally by the TitaniumBox for storage of user data.</Description>
  </TOCComponent>
 </TOCComponents>
 <Assets>
  <Asset ID="A.user_file"
   Type="http://assert4soa.eu/ontology/a4s-language#InputParameter">
   <Name>fileData</Name>
   <APIBinding>/*/SOAP-ENV:Body//tns:fileData</APIBinding>
   <Description>File uploaded by user to his TitaniumBox account.</Description>
   <TOCComponents>
    <TOCComponent TOCComponentRef="TOC.service_operation_upload"/>
   </TOCComponents>
  </Asset>
  <Asset ID="A.user_file_name"
   Type="http://assert4soa.eu/ontology/a4s-language#InputParameter">
   <Name>path</Name>
   <APIBinding>/*/SOAP-ENV:Body//tns:path</APIBinding>
   <Description>File name defined by the user.</Description>
   <TOCComponents>
    <TOCComponent TOCComponentRef="TOC.service_operation_upload"/>
   </TOCComponents>
  </Asset>
 </Assets>
 <DeploymentAndImplementationModel>
  <Description>The user file data is encrypted by the TitaniumBox service
   before the data is stored using the Amazon S3 service.</Description>
 </DeploymentAndImplementationModel>
</TargetOfCertification>
```

Fig. 2. ASSERT Snippet: Target of Certification

Service Description in ASSERT Core: The $\mathcal{SD}$ in the conceptual model is mapped to the *CertificationProcess* element in the ASSERT language. It contains the elements such as *TargetOfCertification* and *SecurityProblemDefinition* which map to the $\mathcal{TOC}$ and $\mathcal{SPD}$ respectively in the conceptual model. In addition, we have incorporated an element called *CertificationCriteria* used to represent any specific criteria followed during the service certification process (e.g., compliance to regulations).

The *TargetOfCertification* element is depicted in Figure 2. The elements in the $\mathcal{ACI}$ are represented directly in the *TargetOfCertification* element i.e., the *Assets*, *TOCComponents*. It also contains the *Type*, *DeploymentAndImplementaionModel* and *Description* providing textual description of the *TargetOfCertification* for end-user comprehension. We enforce the explicit identification of both the *Assets* and *TOCComponents* by making the use of the *ID* element mandatory. The set that maps assets with components, α, in the $\mathcal{ACI}$ is realized within the asset definition by mapping each asset to specific components using the *TOCComponentRef* (which is of type *IDRef*) to provide a binding between the assets and components.

```
<SecurityProperty
 PropertyAbstractCategory="http://assert4soa.eu/ontology/security#Confidentiality"
 PropertyContext="http://assert4soa.eu/ontology/a4s-language#InStorage">
 <Description>Confidentiality of user file data in storage.</Description>
 <NameID>confidentiality_in_storage_TitaniumBox_service</NameID>
 <Assets>
  <Asset AssetRef="A.user_file"/>
  <Asset AssetRef="A.user_file_name"/>
 </Assets>
 <SecurityObjectives>
  <SecurityObjective ID="O.user_data_protection"
   Type="http://assert4soa.eu/ontology/a4s-language#DataProtection"
   Scope="http://www.assert4soa.eu/ontology/a4s-language#TOE">
    <Name>User data protection in storage</Name>
    <SecurityProblemDefinitionRef ProblemDefinitionRef="SPD.data_access"/>
    <Description>The TitaniumBox storage service must provide means of
     protecting user file data from disclosure to any third-party...</Description>
  </SecurityObjective>
 </SecurityObjectives>
 <SecurityMechanisms>
  <SecurityMechanism
   Type="http://assert4soa.eu/ontology/usdl-sec#Cryptography">
   <Name>AES-256</Name>
   <Description>High-grade symmetric encryption... </Description>
   <SecurityObjective SecurityObjectiveRef="O.user_data_protection"/>
   <SecurityFunctionalCriteria Scheme="CC">
       FCS_COP.1.1</SecurityFunctionalCriteria>
  </SecurityMechanism>
 </SecurityMechanisms>
</SecurityProperty>
```

Fig. 3. ASSERT Snippet: Security Property Specification

The TOE is not represented as an explicit part of the service description in the *ASSERTCore*, but we use the flag *InTargetOfEvaluation* in the *TOCComponent* element that indicates whether the component is a part of the TOE, and avoids a duplicate representation of the components in both the TOE and TOC to have an optimized ASSERT.

The *SecurityProblemDefinition* element in the *ASSERTCore* contains a list of *ProblemDefinition*. Each *ProblemDefinition* is mapped to the *spd* in the conceptual model.

Security Property Specification in ASSERT Core: The *SecurityProperty* element maps to the p element in the conceptual model. However, on the representation (language) level we have defined a single property certified in ASSERT. Such "separation" of certified properties allows us to have practical implications on management of ASSERTs throughout their life-cycle, such as issuance, consumption (reasoning), and revocation of ASSERTs. For example, if an ASSERT certifies two properties, say "confidentiality in transit" and "confidentiality in storage", and during the ASSERT lifetime the given service does not anymore comply/provide "confidentiality in transit" due to some technical reasons, the certification authority has to revoke the ASSERT although the second property may still hold.

Figure 3 shows the *SecurityProperty* element structure consisting of an abstract security property realized in a context and on a set of assets. The *SecurityProperty* contains a *NameID* that defines a name identifier of the described property. The *NameID* allows reference to external ontologies to describe the certified security property. The *PropertyAbstractCategory* defines the abstract category of the security property. The *PropertyContext* element defines a context in which the abstract security property is realized. The *Assets* defines a set of *Asset* elements on which the security property applies. Each *Asset* element is a reference to an *Asset* definition in the *TargetOfCertification* section.

The *SecurityObjectives* defines a set of *SecurityObjective* elements of the security property. The main elements of the *SecurityObjective* are: *a)* an identifier of the described security objective; *b)* a set of *SecurityProblemDefinitionRef* each referring to a *ProblemDefinition* defined in the *SecurityProblemDefinition* section; *c) Name* that contains the name of the security objective; *d) Description* which describes the security objective. It is an implicit assumption that all *SecurityObjective*s together contribute to the realization of the *SecurityProperty*. A *SecurityObjective* can refer to one or more *ProblemDefinition*s.

The *SecurityMechanisms* defines a set of *SecurityMechanism* elements. Each element consists of an *ID* that identifies the security mechanism, the *Type* of the security mechanism (or the family of the security mechanism), a set of *SecurityObjectiveRef* elements each referring to a security objective that the seurity mechanism corresponds to. A *SecurityMechanism* can refer to one or more *SecurityObjective*s.

2) Evaluation Specific Details in an ASSERT: The three different categories of evaluation are referred as *ASSERT-E* - for test based evaluation, *ASSERT-M* - for formal analysis and *ASSERT-O* - for ontology based evaluation. We identified three abstract elements that are common to the different types of evaluation: TypeSpecific-Property specification, ServiceModel specification, and Results of evaluation. These elements facilitate advanced reasoning to be performed on the certificates, by comparing and contrasting services based on the evaluation details such as the cardinality of the test suites, the number of tests executed and so on [17]. These elements depend on the processes and the results of each evaluation type and require different syntactic structures. However, we consider such details to be outside the scope of this paper as it would involve discussing the current evaluation methodologies and practices. The ASSERT language, at this point, supports a choice between the three evaluation types, thus restricting an ASSERT to have one type of evidence. This is needed as the evaluation processes and the results from the three different categories are heterogeneous in nature and having multiple types of evidences in a single ASSERT would complicate the processing of the ASSERT especially in certificate comparison.

C. Ontology Integration for Enhanced Processability

In the conceptual model we have presented the elements that need to be captured in a digital security certificate and we have presented a language through which we realize this conceptual model in a machine processable manner using SAML Assertions. However, the ASSERT language provides a data structure to represent the certificates but it does not provide nor prescribe the data that should be contained in the data structures. This is an intentional choice in order to have a clear separation between the conceptual model, the realization and the actual content of the certificates, that for instance would ease the adoption of $\mathcal{CRT}$ with different certification schemes and evaluations.

Therefore, the ASSERT language elements *should* make use of vocabularies, that could be defined by different certification authorities for their respective schemes based on the

certification/evaluation processes and the types of products that are certified.

Vocabularies can make of the existing security ontologies to describe different elements in the ASSERT language, thus permitting reasoning on them, also taking benefit from the *Linked Open Data* paradigm, with respect to establish a link to other ontologies. An example of this flexibility is represented by the use of an ontology, called USDL-SEC [2], for expressing the security mechanisms in the ASSERT language, while specific vocabularies are also foreseen for the expression of other ASSERT elements, like for security properties.

VI. Security Assurance through Asserts

The main objective of a security certificate of a service is to provide security assurance to potential consumers. The existing security certification schemes certify products at varying levels of assurance. The number of levels and the type of assurance depends on the certification scheme. Each certification scheme *fixes* these levels based on carefully designed and selected criteria. In the $\mathcal{CRT}$ there is no explicit element defined for these "levels" of assurance, which is an intentional choice as the $\mathcal{CRT}$ is designed to be certification scheme agnostic. However, the $\mathcal{UDE}$ element can be used to represent these certification scheme-specific information.

The levels of assurance provided by the current schemes distinguish certified products based on security features (such as FIPS-140) or the rigour of evaluation (such as CC, CPA). However, "security assurance" is a multi-dimensional property that amalgamates assurance gained from security features of the product, evaluation of the product, etc. Hence, the levels of assurance provided by the existing schemes are a part of the overall security assurance that a consumer gains from the certificate. We have identified some of the aspects that can impact the security assurance provided to a consumer: *i)* the rigour of the evaluation; *ii)* the trust that the consumer has on the certificate issuer and evaluators; *iii)* the extent of information that is provided in the $\mathcal{CRT}$.

***(i)* Assurance from Evaluation:** The consumer can know the rigour of evaluation either from the "level" of evaluation that is provided by the certification authorities using the $\mathcal{UDE}$ or from the content in the $\mathcal{ESD}$ in case the certification authorities and the product owners are willing to disclose the evaluation information which would make the certificate completely *descriptive*.

***(ii)* Assurance from Certification Entities:** Since the security certification process involves multiple entities, e.g., certificate issuers, evaluators, the strength of the assurance gained depends on the identity of these entities. That is, the assurance gained from certificates depends on the trust the consumer has on the certificate issuers and evaluators. Since the ASSERT language is certification scheme agnostic, it enables product vendors to issue *self-signed* ASSERTS. This is important in order to allow the usage of ASSERTS in huge ecosystems, where the service provider would want to provide security assurance of their services to consumers but they cannot afford to go through a security certification process (which might be the case for small developers) or in cases where they cannot afford the time to go through the certification process (in cases, where time to market can be crucial) or in cases where the recognized certification authorities are not available in their countries. Self signed ASSERTS, clearly, will provide a lower assurance since it is the service provider himself issuing these ASSERTS.

***(iii)* Assurance from Disclosure:** In the service environment, the consumer does not have any information regarding the service internals or its architectures and so on. This lack of transparency affects the overall assurance that that can be gained by the consumer, despite having a certificate. Since the ASSERT allows service providers to disclose the service internals and its architectures at a level that is suitable for them by using the $\mathcal{SD}$. The $\mathcal{SPS}$ which is tied to an asset, provides consumers information on how their assets are secured. The $\mathcal{ESD}$ provides more assurance to consumers by providing information on how the service has been evaluated and the results of the evaluation process.

Overall Security Assurance: The overall security assurance that a consumer can gain from the $\mathcal{CRT}$ is a function of consumer criteria over: Evaluation, Certificate Issuer and Evaluator and Information Disclosed in the certificate.

Consumer criteria can depend on several factors, for example, in very sensitive domains such as healthcare, financial, defence etc., self signed certificates might not be considered as providing any assurance or in the case of governmental organizations, they only consider certificates issued by a few certification authorities.

VII. Conclusions and Future Work

In this paper, we discussed the need for security assurance of services and how current security certificates do not scale well to service environments, and we identified a number of requirements for the use of security certificates in SOC. To address them, we presented an approach for a structured and machine processable representation of the security certificates of services. For its implementation, we have proposed an XML based language that also aims at facilitating the adoption of security certificates in service oriented provisioning and consumption (requisite **R1** defined in Sect. I), by taking advantage of SAML Assertions. We also showcased the possibility of using different ontologies to create the digital certificates as an initial step towards full reasoning on certificate contents (requisite **R2**). Finally, we have discussed how assurance can be gained from these digital certificates and the different dimensions to be taken into account while computing the assurance.

There are still a number of open issues that we are currently working on: first of all, we are conducting a validation of our approach with domain experts, especially focusing on the CC scheme. Secondly, we are investigating on the establishment of a robust binding between services and their corresponding certificates (requisite **R3**) – solutions to which can be either purely legal solutions, purely technical solutions by using TPMs[3] for remote attestations or could be a hybrid solution. We are also validating the TOC representation to ensure that

[2] http://linked-usdl.org/usdl-sec

[3] Trusted Platform Module - that resides on the platform on which the service is run on

it can cope with the different service architectures (PaaS, SaaS and so on).

Another direction of future work is to bring the concept of an ASSERT *profile* [25] in the SOC domain. It is used to facilitate: *a)* easier comparison among ASSERTS; *b)* production of ASSERTS compliant to certification authorities' criteria; *c)* consumers to specify their security requirements similar to the CC-PP. When services comply to such ASSERT profiles, it eases the decision making process for the consumers as the compliance to a profile implies that their requirements are met by the service.

Last but not least, there are several ongoing efforts [26], [27] for making use of digital security certificates. They specially focus on the development of tools with automated reasoning capabilities on the certified security features, and also including their preliminary validation: a tool with such reasoning capabilities can facilitate, for example, analysis and comparison in service discovery operations by partial ordering of certified security properties of different services.

To conclude, we claim that the ASSERT adoption can represent significant benefits for an uptake of third-party service offerings in business-critical domains such as financial, defence and healthcare [28]. The digital security certificates provide security assurance of services that would allay the security concerns of potential customers, which is one of the most relevant obstacles nowadays [1]. Moreover, their scalability would permit a large scale usage in contexts like service marketplaces; for instance, it would permit customers to use their security requirements for service browsing, discovery and selection processes, taking advantage of support tools based on automated reasoning on the certified security features of different offerings.

ACKNOWLEDGMENT

This work was partly supported by the EU-funded project ASSERT4SOA (grant no. 257361).

REFERENCES

[1] Gartner, "Forecast overview: Public cloud services," report G00234817, 2012.

[2] T. C. C. R. Agreement, "Common criteria for information technology security evaluation part 1 : Introduction and general model july 2009 revision 3 final foreword," *NIST*, vol. 49, no. July, p. 93, 2009. [Online]. Available: http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R3.pdf

[3] V. Lotz, S. P. Kaluvuri, F. Di Cerbo, and A. Sabetta, "Towards Security Certification Schemas for the Internet of Services," in *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*, May 2012, pp. 1–5. [Online]. Available: http://dx.doi.org/10.1109/NTMS.2012.6208771

[4] K. Wallnau, *Software component certification: 10 useful distinctions*, ser. Technical note. Carnegie Mellon University, Software Engineering Institute, 2004.

[5] Dropbox inc., "Dropbox security overview," 2012. [Online]. Available: http://www.dropbox.com/dmca#security

[6] Common Criteria, "Common criteria recognition agreement," 2012. [Online]. Available: http://www.commoncriteriaportal.org/ccra/

[7] ——, "Common criteria: Certified products list - statistics," 2012. [Online]. Available: http://www.commoncriteriaportal.org/products/stats/

[8] Lachlan Turner, "How the cc intersects and compares with other security evaluation programs and what this means for the rest of us." 2009. [Online]. Available: http://www.yourcreativesolutions.nl/ICCC10/proceedings/doc/pp/DOMUS.pdf

[9] B. Beckert, D. Bruns, and S. Grebing, "Mind the gap: Formal verification and the Common Criteria," in *6th International Verification Workshop, VERIFY-2010*, M. Aderhold, S. Autexier, and H. Mantel, Eds., Edinburgh, United Kingdom, Jul. 20–21 2010.

[10] Common Criteria, "Common Criteria Part 1: introduction and general model," 2012. [Online]. Available: http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R4.pdf

[11] ——, "Common Criteria Part 2: security functional requirements," 2012. [Online]. Available: http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R4.pdf

[12] ——, "Common Criteria Part 3: security assurance requirements," 2012. [Online]. Available: http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R4.pdf

[13] P. Benassi, "Truste: an online privacy seal program," *Commun. ACM*, vol. 42, no. 2, pp. 56–59, Feb. 1999. [Online]. Available: http://doi.acm.org/10.1145/293411.293461

[14] McAfee, "Mcafee SECURE," 2007. [Online]. Available: http://www.mcafee.com/us/mcafeesecure/index.html

[15] X.509, "The directory: Public-key and attribute certificate frameworks," 2005, ITU-T Recommendation X.509:2005 | ISO/IEC 9594-8:2005.

[16] SAML Specification, "SAML specification," 2012. [Online]. Available: http://saml.xml.org/saml-specifications

[17] M. Anisetti, C. Ardagna, and E. Damiani, "Defining and matching test-based certificates in open SOA," in *Proc. of the Second International Workshop on Security Testing (SECTEST 2011)*, 2011, pp. 520–522.

[18] M. Anisetti, C. A. Ardagna, E. Damiani, C. Pandolfo, and A. Maña, "D4.1 Design and description of evidence-based certificates artifacts for services," ASSERT4SOA Project, Tech. Rep., 2011, available at http://www.assert4soa.eu/deliverable/D4.1.pdf.

[19] A. Fuchs and S. Gürgens, "D5.1 Formal models and model composition," ASSERT4SOA Project, Tech. Rep., 2011, available at http://www.assert4soa.eu/deliverable/D5.1.pdf.

[20] S. D'Agostini, V. Di Giacomo, C. Pandolfo, and D. Presenza, "An ontology for run-time verification of security certificates for SOA," in *Proc. of the 1st International Workshop on Security Ontologies and Taxonomies (SecOnt 2012)*, 2012, pp. 525–533.

[21] H. Koshutanski, A. Maña, R. Harjani, M. Montenegro, S. P. Kaluvuri, F. Di Cerbo, E. Damiani, C. A. Ardagna, M. Anisetti, D. Presenza, S. Gürgens, R. Menicocci, V. Bagini, F. Guida, and A. Riccardi, "ASSERT language v2," ASSERT4SOA Consortium, Project Deliverable D1.2, 2012. [Online]. Available: http://www.assert4soa.eu/deliverable/D1.2.pdf

[22] OASIS, "OASIS WS-Security specification," 2006. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

[23] OASIS, "OASIS WS-SecurityPolicy specification," 2007. [Online]. Available: http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.html

[24] ——, "OASIS WS-Trust specification," 2007. [Online]. Available: http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html

[25] A. Maña, H. Koshutanski, J. Gonzalez, M. Montenegro, R. Menicocci, A. Riccardi, V. Bagini, F. Di Cerbo, and S. P. Kaluvuri, "D1.3 ASSERT profiles," ASSERT4SOA Project, Tech. Rep., 2012, available at http://www.assert4soa.eu/deliverable/D1.3.pdf.

[26] K. Mahbub, L. Pino, G. Spanoudakis, H. Foster, A. Maña, and G. Pujol, "D2.3 ASSERTs aware service based systems adaptation," ASSERT4SOA Project, Tech. Rep., 2012, available at http://assert4soa.eu/deliverable/D2.1.pdf.

[27] M. Bezzi, S. D'Agostini, S. Kaluvuri, A. Maña, C. Pandolfo, G. Pujol, and A. Sabetta, "D6.1 architecture and high-level design," ASSERT4SOA Project, Tech. Rep., 2012, available at http://www.assert4soa.eu/deliverable/D6.1.pdf.

[28] F. Di Cerbo, M. Bezzi, S. Kaluvuri, A. Sabetta, S. Trabelsi, and V. Lotz, "Towards a trustworthy service marketplace for the future internet," in *The Future Internet*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7281, pp. 105–116. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30241-1_10