Machine-Readable Privacy Certificates for Services

Marco Anisetti[1], Claudio A. Ardagna[1], Michele Bezzi[2], Ernesto Damiani[1], and Antonino Sabetta[2]

[1] Università degli Studi di Milano, Dipartimento di Informatica, Italy
[2] SAP Product Security Research, Sophia-Antipolis, France

Abstract. Privacy-aware processing of personal data on the web of services requires managing a number of issues arising both from the technical and the legal domain. Several approaches have been proposed to matching privacy requirements (on the clients side) and privacy guarantees (on the service provider side). Still, the assurance of effective data protection (when possible) relies on substantial human effort and exposes organizations to significant (non-)compliance risks. In this paper we put forward the idea that a privacy certification scheme producing and managing machine-readable artifacts in the form of privacy certificates can play an important role towards the solution of this problem. Digital privacy certificates represent the reasons why a privacy property holds for a service and describe the privacy measures supporting it. Also, privacy certificates can be used to automatically select services whose certificates match the client policies (privacy requirements).

Our proposal relies on an evolution of the conceptual model developed in the Assert4Soa project and on a certificate format specifically tailored to represent privacy properties. To validate our approach, we present a worked-out instance showing how privacy property *Retention-based unlinkability* can be certified for a banking financial service.

Keywords: privacy, certification, testing.

1 Introduction

The success of the Web as a platform for the provisioning of services and the huge amount of personal information disseminated, collected, and managed through the network comes with important concerns for the privacy of users' data. The growing awareness of users, on one hand, and the increasing regulatory pressure coming from governments, on the other, are imposing new requirements and constraints on businesses that need to handle sensitive data to carry out their services.

As part of the effort of ensuring compliance to the new data protection laws and regulations, several solutions have been proposed defining different privacy-aware languages that help users in defining which of their data can be used, by whom, when, and for which purposes [1,2,3,4,5,6]. These solutions constitute an

R. Meersman et al. (Eds.): OTM 2013, LNCS 8185, pp. 434–450, 2013.

important step towards increasing the availability of practical data protection technology, leveraging automated processing of privacy policies. To this end, by capturing the data protection requirements as explicit policies, they do address a key facet of the problem. However, their applicability remains limited unless the data protection *guarantees* offered by providers are expressed similarly in a format that can be processed automatically. Initiatives such as *EuroPriSe* [7] and *Trust-E* [8] represent an initial move in the direction of explicitly representing the data protection measures put in place by a service (or by a software product in general), but they have significant limitations. Firstly, they take an *all-or-nothing* approach to compliance (a product is either compliant with their certification schema or not), which does not allow reasoning about privacy assurance based on richer, finer-grained client requirements. Secondly, these schemes rely on a format that, although structured, is essentially based on a natural language description meant for human consumption, but that is not suitable for automated processing. Finally, the evaluation process followed to assess the correctness and adequacy of the privacy protection measures declared in the certificates is not described in detail, which makes the evaluation itself somewhat opaque to the client.

Unfortunately, even the adoption of such first-generation privacy certification schemes is quite an exception; most frequently, service providers release just a text document (typically a web page in their website) describing what data they handle and what protection measures they put in place to protect those data. Such a description, expressed in natural language, requires an understanding of the data protection problem and solution spaces that users cannot be expected to have. This means that it is extremely difficult for users to determine if the protection measures offered by a service adequately cover their needs. Furthermore, the privacy statements associated to services are usually self-declarations by the service provider, which are difficult (if not impossible) for the client to check.

To address these problems, we believe that *i)* the privacy protection statements should be expressed in an explicit, machine-readable format so that the matching of privacy measures (offered by candidate services) with the corresponding client policies (privacy requirements) can be automated; and *ii)* the statements by service providers should be checked and endorsed by a third party that users trust (e.g., a recognized certification entity). Addressing these two aspects would unlock new scenarios that are not possible today: users could discover services based on their data protection guarantees, determine whether a service fulfills a particular privacy policy, and compare similar services based on the extent to which each of them targets a specific data protection goal.

Machine-readable certification of *privacy* statements serves the interests of both clients and providers. As for clients, explicit privacy certificates mean improved *transparency*. Companies (especially SMEs) may have not the adequate resources to assess the "quality" of offered services, especially from the point of view of security an privacy. The opportunity of having security and privacy features described in a structured and machine-readable artifact, as in a digital security and privacy certificate, can support users to make meaningful comparisons, which may also be

(partly) automated and supported by tools, similarly to what is described in [9]. On the other hand, privacy certificates are an effective means for service providers to demonstrate compliance with data protection regulations and customer requirements. Although legal compliance with privacy an data protection regulations are mandatory nowadays, organizations often struggle to deal with the large diversity of regulation across geographies and sectors. For example, EU data protection directives often differ from US privacy regulatory framework, not to mention that the EU directive can be differently implemented in the 27 EU member states or sector specific regulations (e.g., HIPAA). Privacy certifications can provide a "stamp of approval" of a trusted, expert third-party attesting the adherence to specific legal and privacy frameworks, ultimately supporting the users to adopt service-based solutions that are provably compliant to national and sector specific regulations.

Over the last three years, the ASSERT4SOA project [10] has investigated ways of realizing a novel, light-weight approach to security certification of services, according to which finer-grained security properties of applications and services are evaluated by independent third parties and can be expressed in machine-readable artifacts (called ASSERTs). Among other results, the project defined a conceptual model and a certificate representation, which provide a concrete structure to represent security certification artifacts. Also, a reference architecture has been defined to support the processing of certificates, the discovery of services based on their security properties, and the automated matching of client requirements with services having corresponding certified properties.

In this paper we present an evolution of the ASSERT4SOA conceptual model and certificate format that is specifically tailored to represent privacy properties.

The remainder of this paper is organized as follows. Section 2 illustrates the motivation of our work and a reference scenario. Section 3 presents our certification model for privacy, and Section 4 illustrates its application using concrete examples of certificates and focusing on privacy property retention-based unlinkability. Section 5 discusses related work and, finally, Section 6 concludes the paper.

2 Motivating Example

We consider a scenario in which a client is searching for a privacy-aware IFX-based[1] financial service that addresses its privacy policies (e.g., any personal information provided to the service stays confidential or is deleted after a given period of time). This scenario involves *i)* a client accessing the IFX-based financial service with a set of privacy requirements, *ii)* a service provider implementing

[1] Interactive Financial eXchange (IFX) Standard (`http://www.ifxforum.org/standards/`), a financial messaging protocol initially defined in the 1997 by financial industry and technology leaders. IFX aims to exchange data electronically to accomplish a variety of transactions between (unknown and) distributed entities. The IFX standard supports many financial and security functionalities, and integrates them in a service-oriented architecture.

an IFX-based service exposed as a SOAP-based service on the Web, and *iii)* a certification authority certifying the privacy properties of web services.

In particular, we consider an IFX-based service implementing a *Deposit and Withdrawal* service that enables clients to make deposits (operation *CreditAdd*), possibly via cheque, and withdrawals (operation *DebitAdd*) in/from their bank account, using a reverse ATM. This service puts strong requirements on security and privacy of the clients, such as, confidentiality of the messaging exchange, integrity of data, authenticity of the involved parties, privacy of data in the cheques, and introduces the need of a security- and privacy-oriented certification scheme.

In this paper, we focus on the certification of privacy property *retention-based unlinkability*, meaning that the service is certified to maintain the client's personal data following the client's requirements specified through a retention-based privacy policy. Let us consider the scenario in which a client deposit a cheque using the reverse ATM connected to our *Deposit and Withdrawal* service. The reverse ATM scans the cheque and allows the client to specify a retention period for the cheque scan when stored at the *Deposit and Withdrawal* service storage.[2] We note that if the retention period is not specified a default one will be used by the service provider. The cheque scan is sent as a parameter of the request to operation *CreditAdd* of the *Deposit and Withdrawal* service, via the SOAP with attachment standard (`http://www.w3.org/TR/soap12-af/`), and the retention period specified by the client is associated as an annotation to it. Additional parameters of operation *CreditAdd* (e.g., amount and identity token) are associated with a default retention period possibly dictated by the legislation of the country in which the reverse ATM resides.

3 Privacy-Assert

We propose the concept of digital privacy certificate for services (P-ASSERT). P-ASSERTs are machine-readable, signed statements, bound to services, that certify the privacy properties guaranteed by a service. As in current certification schemes, the assessment of the property is performed by an independent third party (e.g., a certification authority), who issues (and signs) the P-ASSERT. The certification is based on an evaluation of the service characteristics (e.g., using formal methods or testing), which can be represented in the P-ASSERT. Differently from existing schemes (e.g., EuroPriSe [7] or Common Criteria [11]), P-ASSERTs are represented as (signed) XML documents, a format suitable for automated reasoning and processing.

In the following, we present the structure and main features of P-ASSERTs, which extend the digital security certificates (ASSERT) introduced in [12]. As in the original ASSERT, each P-ASSERT includes three main parts (see Figure 1).

[2] The cheque scan can provide additional information of interest like the cheque transfers and bounces, signatures, dates, which may be sensitive from a privacy point of view.

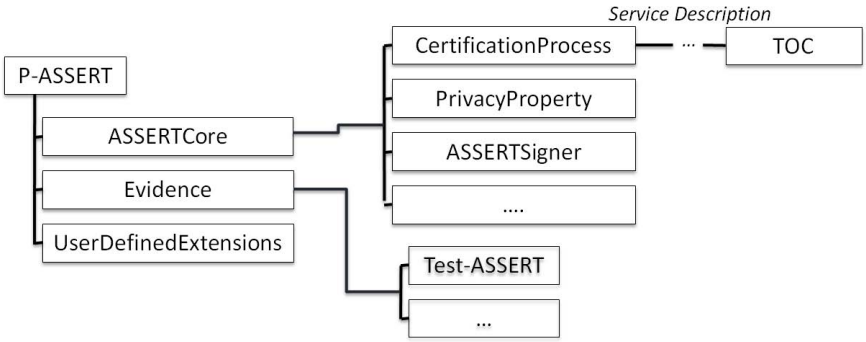

Fig. 1. P-ASSERT: high-level structure

- A *Core* part that provides information about the certified entity (service description), the specification of the privacy property of the certified entity, and additional contextual information such as the certification authority, signature, and certification process.
- An *Evidence* part that describes the details of the evaluation performed by the certification authority (see Section 3.2) and supports the certification claims, such as a structured description of the test suites executed as part of the evaluation process (see [13] for details).
- An additional part reserved for extensions, that can be used, e.g., to cover domain-specific concerns or to provide additional information on top of the content of the *Core* and/or the *Evidence*.

More in detail, in the core part, the service description contains the Target of Certification (TOC) describing the service being certified, and the Target of Evaluation (TOE) describing the part of the Target of Certification that is evaluated and the rationale for protecting the assets that are identified. More sophisticated models can be present in the *evidence* section of the P-ASSERT to describe the evaluation performed, e.g., Symbolic Transition System model for the generation of test cases (see Section 3.2).

Note that, traditional security (as Common Criteria) or privacy (as EuroPrise) certification schemes do not make a clear distinction between the system that is being certified and the aspects of the system that are subject to evaluation, limiting the description to the TOE in natural language. However, this distinction becomes more relevant whenever we want to use the certificates in service-based systems, because services can be easily composed of multiple, external services, and it should be clear which part is evaluated. Similarly, to allow for machine-readability, the service description also provides a list of assets, which will be explicitly referred in the different parts of the certificates. These assets replace the natural language description of assets-to-be protected in today certification schemes.

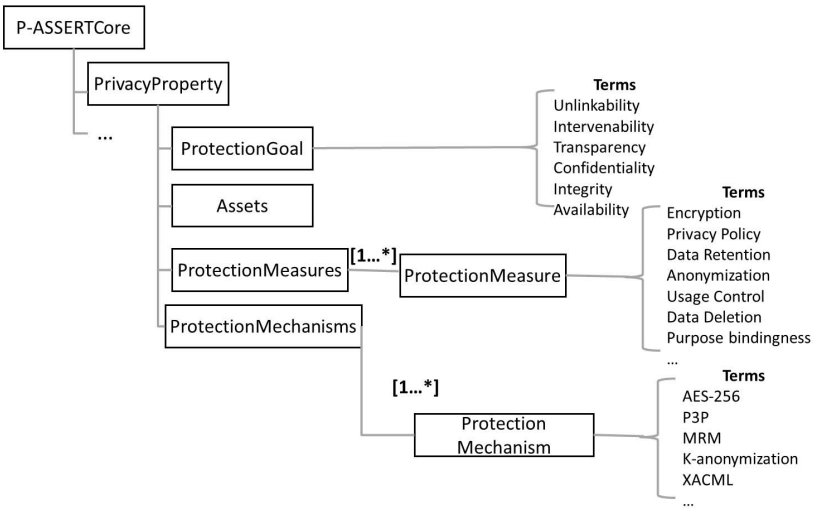

Fig. 2. *PrivacyProperty*: main elements

The privacy property specification element contains a multi-level description of the privacy property at different abstraction levels. We discuss this element in detail in Section 3.1. The evaluation specific portion of the certificate defines the representation of the details and results of the service evaluation process needed to support the certified property, describing, for example, the models used to generate the test cases and the tests performed on the system. We will describe this part in Section 3.2.

The proposed structure of the P-ASSERT is based on the the ASSERT model (which targets security properties), the main difference is in the description of the property. In the next sub-section, we will present the privacy property specification element, we refer the reader to [12,13], for a complete analysis of the remaining part of the ASSERT.

3.1 Privacy Property

Privacy properties need to be specified at different levels of abstraction, from more abstract concepts to fine-grained representations. The advantage is two-fold: first, it allows end-users with different levels of expertise to understand the privacy features of the service, increasing transparency; second, it permits users (being machines or human beings) to search for services that match their privacy requirements at different levels of complexity. Accordingly, we propose different elements, from abstract concepts to the technical implementation mechanisms, to describe the property, as illustrated in Figure 2. A privacy property for a certain asset (*Assets* element to the service description) is described with a three layer structure, as in [14,15], in terms of *Protection Goal*, *Protection Measures* and

Protection Mechanisms. All the other elements are common with the previously introduced ASSERT representation for security certificate (see [13] for details).

Regarding the most abstract layer, Privacy by Design principles [16] constitute a natural starting point to express privacy principles. On the other hand, they mix regulative and engineering criteria, making them unsuitable for describing technical features only. Recently, in analogy with the "classical" data security protection goals of *confidentiality*, *integrity*, and *availability*, three additional *data protection goals* (and corresponding protection measures) have been proposed [14,15].

Transparency means that "the collection and processing operations of data and its use can be planned, reproduced, checked and evaluated with reasonable efforts." It can be supported by measures such a clear privacy policy, breach notification, and so on.

Unlinkability ensures that personal data cannot be linked across domains or used for a different purpose than originally intended [15]. It can be supported by measures like: limited retention period, data erasure, anonymization, data minimization, separation of contexts by different identifiers.

Intervenability is the "ability to intervene" for data subjects, operators and supervisory data protection authorities to apply corrective measures if necessary. For example, it includes the right to rectification and deletion of data for the data subject. It can be supported by measures such as mechanisms for handling data subject's correction requests, *break-the-glass* policies, and the like.

These protection goals provide a high-level description of the privacy properties, but they do not give any information how these goals can be achieved. The P-ASSERT contains a list of *protection measures*, which are linked to a protection goal, indicating the necessary measures to reach the goal. For example, the *unlinkability* protection goal can be supported by *anonymization* and *data retention* measures. Measures are realized by specific protection mechanisms, describing the techniques or procedures used to realize a specific protection measure. For example, *anonymization* protection measure can be implemented by specific *k-anonymity* algorithms, with a set value of k.

More formally, a privacy property p is a pair $(\hat{p},A)$, where $p.\hat{p}$ is an protection goal and $p.A$ is a set of class attributes referring to specific characteristics of the privacy function implemented by the service (i.e., detailed description of protection measures and mechanisms). For instance, property p=(*confidentiality*,{`measure`=encryption,`algo`=DES,`key`=112bit,`ctx`=in transit}) describes a privacy property whose *protection goal* is confidentiality in transit, *protection measure* is encryption, and *protection mechanism* is DES encryption algorithm with key length of 112bits.

In some cases, a partial order can be defined over privacy properties based on attribute values, inducing a hierarchy $\mathcal{H}_P$ of properties as a pair $(\mathcal{P},\preceq_P)$, where $\mathcal{P}$ is the set of properties and $\preceq_P$ the partial order. Given two properties p_i and p_j, we write $p_i \preceq_P p_j$, if p_i is weaker than p_j (see [17] for a more detailed discussion on partial ordering of security properties). The hierarchy of privacy

properties is fundamental for comparing different services from a privacy point of view, which is one of the most prominent functionality for a privacy-aware SOA infrastructure.

3.2 Evidence Representation in P-ASSERT

In this section, we describe a test-based certification scheme, that is, a process producing evidence-based proofs that a (white- and/or black-box) test carried out on the software has given a certain result, which in turn shows that a given high-level security property holds for that software [18]. The evidence in P-ASSERT contains test-based artifacts and details on how these artifacts support privacy property p defined in the core part of P-ASSERT. More in detail, it is divided into two main sections as follows.

Service model m: A Symbolic Transition System (STS) [19] that specifies service behavior and interactions as a finite state automaton. It is used for automatic generation of test cases. The service model specifies a label *Model* that describes its level of detail and assumes values in: *i) WSDL* when m models the Web Service Definition Language (WSDL) interface only, *ii) WSCL* when m models the client-server conversation in the Web Services Conversation Language (WSCL) document, and *iii) implementation* when m models the implementation of service operations. We note that the service model only describes those operations, called Most Important Operations (MIOs), that are needed to certify privacy property p and to maintain the correctness of the service model. A set of quantitative indexes v^m (e.g., number of states) is defined to calculate a quality measure for the model (the complete set of indexes is provided in [17]). As an example, a WSCL-based model for the *Deposit and Withdrawal* service described in Section 2 is depicted in Figure 3.

Figure 3 shows the WSCL conversation that allow the client to access operations *CreditAdd* and *DebitAdd* of the service. First the client has to log into the system. Operation *Signon* returns the variable *result*='ok' with a *token* in case of successful authentication, *result*='failure' otherwise. After a successful *Signon*, the client can call either operation *CreditAdd* or *DebitAdd*. *CreditAdd* takes as input variables *amount* (the amount of money to be deposited), *scan* (an optional parameter with a scan of the cheque used to transfer money and passed as an attachment to the SOAP message of the request), *token* (an authentication token returned by operation *Signon*), and *rp* (the retention policy attached to variable scan). *DebitAdd* takes as input variables *amount* (the amount of money to be withdrawn) and *token* only. Both operations return the *result* of the execution as output.

Test Evidence e: The testing artifacts proving a property for the certified service. Test Evidence e is composed of the set of test cases executed on the service, their category (i.e., functionality, robustness, penetration) and type (e.g., random input, equivalence partitioning), and a set of test attributes (e.g., the cardinality of the test set). It also specifies a set of test coverage metrics (e.g., branch coverage,

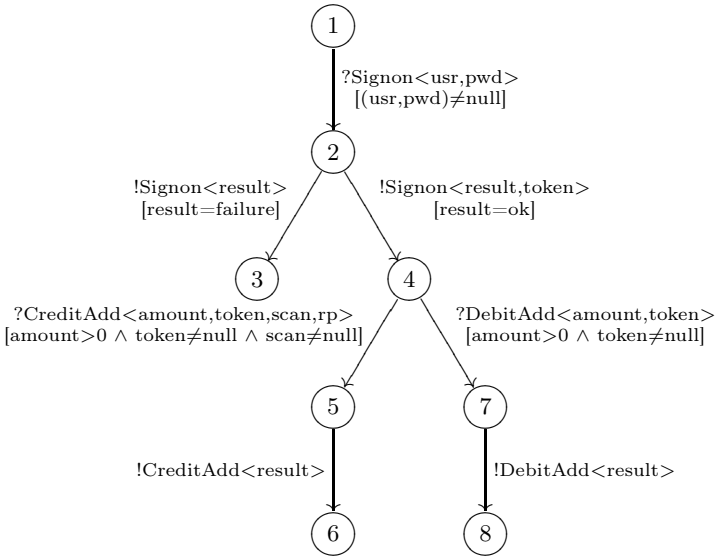

Fig. 3. WSCL-based model for deposit and withdrawal service

path coverage) measuring how much the test set covers the service model, and is complete and exhaustive. More formally e={$cat(e)$,$type(e)$,$ta(e)$,$tc(e)$,$tr(e)$,v^e} where $cat(e)$ is the test category, $type(e)$ is the test type and $ta(e)$ are the related attributes; $tc(e)$ are the test cases while $tr(e)$ are the results of their execution. v^e_j is the set of test coverage metrics. The complete set of metrics is provided in [17] and can be used to calculate an aggregated quality metric.

Test evidence can support a variety of privacy-related properties. For instance a certification authority can award a privacy certificate C(p,m,e) to a service s with privacy property C.p=(*Confidentiality*,{`measure`=encryption,`algo`=*DES*, `key`=*112*,`ctx`=*in transit*}), service model C.m={*WSCL*,∗}, and evidence C.e= (*Functionality*,*Input Partitioning.Equivalence Partitioning*,{`card`=130},∗,∗,∗), where C.e.`card` is the cardinality of the test set and ∗ means any value. Certificate C provides the evidence proving that s holds the privacy property with the protection goal of confidentiality at communication (*in transit*) level, using a 112-bit *DES* algorithm. In this example, evidence is produced using a WSCL-based model and a set of 130 test cases with test category *Functionality* and test type *Input Partitioning.Equivalence Partitioning*.

4 Certification of Retention-Based Unlinkability

In the previous section, we briefly presented an example of p-Assert certificate for Confidentiality of data in transit supported by functionality test. In this section, we present a complete worked-out example showing how privacy property

Retention-based unlinkability can be certified. We assume that a simplified language is available permitting to specify the *retention period* of data contained in a service request.[3] After the retention period expires, the service provider must delete any reference to the data, assuring users that their data are no longer available for access.

To show the process of certifying a service for property *Retention-based unlinkability*, we consider the *Deposit and Withdrawal* service in Section 2 implementing a privacy retention mechanism similar to the one adopted by Microsoft Exchange Server 2010, which supports the definition and enforcement of retention tags and policies [20], but at filesystem level. We assume that each request performed by a client specifies a *sticky policy* [1] for each data item with the retention period. Sticky policies are data handling policies attached to the personal data they protect and regulate how personal data will be handled at the receiving parties (i.e., data controllers and processors). Users specify these policies to define restrictions on the retention period of their data, thus keeping a level of control on their information also after its release. Clearly, the retention period specified in the request by a user must comply with the requirements for retention defined by the service (see Section 2); if not specified a default retention period applies for the user request.

In the following, we consider the certification of a cheque-based *CreditAdd* only and assume that the retention period for a cheque scan attached to the request in a cheque-based deposit is directly specified by the client using the reverse ATM (see parameters *scan* and *rp* in Figure 3). We note that a retention period can be also specified for parameters *amount* and *token*, though not discussed in our scenario.

Suppose that the service supports a retention mechanism with frequency of control 1 second, minimum retention period 1 day, and maximum retention period 1 year. To this aim, the service implements a mechanism that periodically checks (every 1s) the retention period of each request and deletes all data for which the retention period is expired. In particular, the process implemented by the retention mechanism is composed of the following steps:

1. the client sends a *CreditAdd* request to a service annotated with a retention period *rp* for the cheque scan. The retention period is defined in seconds and automatically transformed in a precise date and time at the service side;
2. the service checks if the retention period complies with its requirements (e.g., minimum retention for cheque scan). If yes, the cheque scan is stored in the filesystem with the retention period; if not, the user request fails;
3. the service periodically controls the cheques' storage and flags all cheque scan for which the retention period is expired (i.e., the date and time in the retention policy are before the current date and time);
4. flagged cheques are deleted.

[3] This assumption is not restrictive since our solution can be easily adapted for working with any privacy policy language supporting retention, including classic P3P [6].

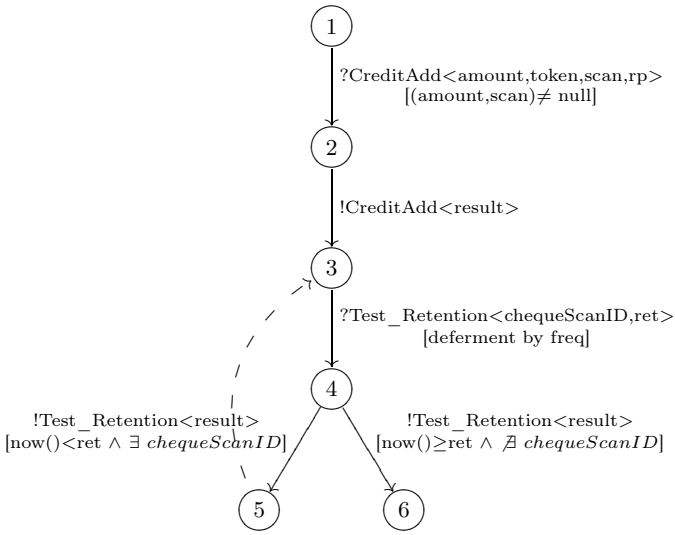

Fig. 4. WSCL-based Test model for Deposit and Withdrawal example

As soon as the service provider wants to certify this service for privacy property p=(*Unlinkability*,{`measure`=retention,`frequency`=1s, `min_retention`=1d, `max_retention`=1y}), the certification authority (CA) verifies the correct working of the above described retention mechanism by means of *deferred testing execution*. *Deferred testing execution* is a largely used test execution approach in which the executions of two consecutive test cases are deferred by a specific temporal delay. The delay of the deferment is usually due to the fact that the components that a test case exercises may not be ready for inspection by the time the test runs (e.g., due to instantiation of classes declared with deferred initialization stages). In our approach, execution deferment is used to test properties, like retention, that should become true (or false) after a given period of time. Our testing strategy relies on the model m of the service (see Figure 4), on test category *cat*, and test type *type* to produce the test model used for test case generation [17]. The test model allows deferred test execution via the definition of specific timing conditions on the STS-based service model (see Figure 4). In this specific case, to support the testing activities, the WSDL of the service is extended with a new operation *Test_ Retention*(*chequeScanID*,*ret*). This operation, which is used at certification time only and then removed, provides the test code for evaluating the retention mechanism. It takes the name of the file representing the cheque scan (*chequeScanID*) in the service storage and the retention period *ret* (date/time) derived from the retention policy *rp* as input, and returns the result of the testing activity as output.

The *Test_ Retention* operation checks the storage for cheque *chequeScanID*. The retention mechanism is correctly working and the *Test_ Retention* returns true if: *i)* the cheque is in the storage and the retention period is not expired

Privacy Property: Unlinkability
Class Attributes: *measure*=retention, *frequency*=1s, *min_retention*=1d, *max_retention*=1y

$$TC1 = \begin{cases} I_1: & CreditAdd(amount, token, scan, rp) \in valid\ partitions \\ EO_1: & result = ok \\ PR_2: & deferment\ by\ freq \\ I_2: & Test\_Retention(scan.chequeScanID, ret)\ with\ now() < ret \\ EO_2: & EO_2: result = True\ \wedge\ scan.chequeScanID\ is\ found \end{cases}$$

$$TC2 = \begin{cases} I_1: & CreditAdd(amount, token, scan, rp) \in valid\ partitions \\ EO_1: & result = ok \\ PR_2: & deferment\ by\ freq \\ I_2: & Test\_Retention(scan.chequeScanID, ret)\ with\ now() \geq ret \\ EO_2: & result = True\ \wedge\ scan.chequeScanID\ is\ not\ found \end{cases}$$

$$TC3 = \begin{cases} I_1: & CreditAdd(amount, token, scan, rp)\ with\ amount, token, rp \in valid\ partitions \wedge \\ & \quad \wedge\ scan \in invalid\ partition \\ EO_1: & result = err \\ PR_2: & deferment\ by\ freq \\ I_2: & Test\_Retention(scan.chequeScanID, ret)\ with\ anytime \\ EO_2: & result = True\ \wedge\ scan.chequeScanID\ is\ not\ found \end{cases}$$

Fig. 5. Test cases for retention for *Deposit and Withdrawal* service

(while being less than one year) or is expired by less than 1s (frequency with which the retention is evaluated), *ii)* the cheque is not in the storage and the retention period is expired. It returns false in the other cases.

To test the retention-based privacy property, we first remove operation *Signon* because it is not a MIO and thus does not contribute to the generation of the test cases (see Figure 4). Also, for simplicity, we removed operation *DebitAdd* and focused on the certification of *CreditAdd*. As a consequence, the test model only involves operations *CreditAdd* and *Test_Retention*. Since the retention test is a deferred testing we add deferring time conditions to the STS-based test model in such a way that the *Test_Retention* can be executed before the retention time is expired ($now() < ret$) and after the retention time is expired ($now() \geq ret$). The test model in Figure 4 generates multiple calls to operation *Test_Retention* with different deferment times (i.e., *deferment by freq*) for proving the correctness of the retention mechanism implemented by the service. Our model includes a cycle which iterates until the cheque scan is no longer stored by the service, that is, the retention control mechanism is proved to be correctly implemented for that specific scan. We note that operation *Test_Retention* is iteratively executed according to the frequency used by the retention mechanism.

Some examples of test cases generated by the test model in Figure 4 are shown in Figure 5. TC1 and TC2 belong to the functionality test category and consider valid test types (all parameters are in their valid partitions). In general, the test model in Figure 4 will generate a set of TC1-like test cases, until the test time (indicated using *now()*) is greater or equal to the retention time *ret* (in the form of date/time) derived from the user's privacy policy. TC3 is a

PRIVACY PROPERTY: Unlinkability
CLASS ATTRIBUTES: *measure*=retention, *frequency*=1s, *min_retention*=1d, *max_retention*=1y

$$TC4 = \begin{cases} I: & request = CreditAdd \ \wedge \ (1d < rp < 1y) \\ EO: & result = ok \end{cases}$$

$$TC5 = \begin{cases} I: & request = CreditAdd \ \wedge \ rp < 1d \\ EO: & result = error \end{cases}$$

$$TC6 = \begin{cases} I: & request = CreditAdd \ \wedge \ rp > 1y \\ EO: & result = error \end{cases}$$

Fig. 6. Test cases for verifying retention period boundary values

robustness test case that verifies whether the cheque scan is invalid (e.g. not correctly scanned), while the other parameters of the *CreditAdd* are valid. In this case the result of *CreditAdd* is an error and the cheque scan must be not saved or deleted immediately from the cheque scan storage; the operation may be maintained in the system log with the other parameters of the function call, depending on the legislation of the country in which the reverse ATM resides. The execution of *Test_Retention* with a wrong cheque scan must return a *scan not found* independently by the precise time in which it is executed.

The certification authority can also verify the correct support for minimum and maximum retention periods, using the additional test cases showed in Figure 6.

5 Related Work

Different languages and format for machine-readable privacy policy for web applications and services have been proposed. The XML-based P3P (Platform for Privacy Preferences Project) language and APPEL (A P3P Preference Exchange Language) [6] are used for describing privacy policies and privacy negotiations between a web site and users. The Enterprise Privacy Authorization Language (EPAL) [21] is based on the same concepts of the eXtensible Access Control Markup Language (XACML) [5], but it implements more privacy-specific conditions, such as purpose-based access control. More recently, the PrimeLife Privacy Language (PPL) [22] was defined as an extension of the XACML authorization language. PPL allows to handle complex privacy policies, including specifying secondary usage of the data (e.g., when an external data processor is involved) and privacy obligation handling, relying on XACML for access control conditions.

These privacy policy languages allow for a description and processing of privacy policies, but they do not provide the elements for specifying the protection measures used nor for detailing the evidences supporting their correct

implementation. As mentioned in Section 4, p-Assert can rely on policy languages for expressing the privacy conditions, and complement them with a more granular description of the protection measures and evidences.

Another major source of related work for this paper resides in the area of software testing. Similarly to this paper, several works (e.g., [23,24,25]) focused on testing non-functional requirements of software systems. As far as web service testing is concerned, the line of research closest to the one in this paper considers the problem of testing a web service to assess its correct functioning and to automatically generate test cases used in the verification process [26,27]. Heckel and Lohmann [28] propose a solution for testing web services that uses Design by Contract and adds behavioral information to the web service specifications. More recently, Bentakouk et al. [29] propose a solution using STS-based testing and STM solver to check the conformance of the composite service implementation with respect to its specifications and/or client requirements. Endo and Simao [30] present a model-based testing process for service-oriented applications. Existing approaches have mainly focused on static or dynamic testing, while they have not focused on certification. More in detail, these approaches test services with the scope of verifying their security mechanisms (i.e., policy enforcement) ex-post. The p-Assert approach elaborates on the approach presented in [17] and concentrates on container-level certification, which implies security policy testing and certification. In [31], a security testing method for stateful Web Services is proposed. It defines specific (i.e., for each security property to be tested) security rules, eventually derived from policy, using Nomad language with the scope of generating test cases. This rule set allows to test different properties such as availability, authorization, and authentication by means of malicious requests based on random parameters, or on SQL and XML injections. The rules are applied to the operation set (obtained from service specifications like WSDL) of the service under test, to generate test requirements (modeled as STSs), which are then synchronized with the specifications to produce the test set. The goal is to perform a black-box testing of web services exploiting a rule-based approach for the generation of an ad hoc test set. The test set is aimed at discovering if the service under test violates or not the security rules. Other approaches focusing on general testing or on authentication and authorization policies, construct abstract test cases directly from models describing policies [32,33,34]. Le Traon et al. [32] proposed test generation techniques to cover security rules modeled with OrBAC. They identified rules from the policy specification and generated abstract test cases to validate some of them via mutation. In [33], the authors developed an approach for random test generation from XACML policies. The policy is analyzed to generate test cases by randomly selecting requests from the set of all possible requests. In [34], the authors proposed a model-driven approach for testing security policies in Java applications. The policy is modeled with a control language such as OrBAC and translated into XACML. In our case we describe an approach for testing privacy-specific policies for web services and how is it possible to generate a certification scheme for them.

6 Conclusions

We proposed a representation for digital privacy certificates (p-Assert), which describes the outcome of a privacy certification process for web services in a machine-readable format. These structured and machine-readable certificates enable the service consumer to: *i)* know the details about the privacy features of the service (*transparency*), *ii)* access detailed information on the evidence supporting the claim of the certificate (*assurance*) *iii)* automatically reason about privacy properties (e.g., allowing service discovery based on privacy requirements). Our proposal extends the security certification framework developed in the context of the European project Assert4Soa. Following the same approach, we described a digital privacy certificated supported by a model-based testing process, which allows to automatically produce evidence that a given privacy property holds for the service. The corresponding machine-readable certificate contains the certified property, the model of the service used for the automatic generation of the test cases, and the evidence produced by their execution. We also provided a worked-out example of the application of our scheme to the certification of privacy property *retention-based unlinkability*. In this context, we introduced the concept of *deferred testing* as a testing activity that specifies the time intervals between consecutive test cases. Our example focused on *offline* deferred testing, meaning that the test cases on retention policies are executed in an accredited Lab in the framework of a certification process and the retention periods are randomly generated to maximize the coverage of their domain. We note however that our solution can support *online* deferred testing, verifying the correctness of the retention mechanism implemented by the service on real client requests and client-defined retention periods. We plan to further analyze this post-deployment testing scenario in our future work. Our future work will also evaluate the efficiency of our approach, analyzing the overhead required for automatic test generation at the increasing of policy complexity.

Acknowledgements. This work was partly supported by the EU-funded projects Assert4Soa (contract n. FP7-257351) and Cumulus (contract n. FP7-318580). We thank Kirsten Bock and Marit Hansen for fruitful discussions.

References

1. Ardagna, C., Camenisch, J., Kohlweiss, M., Leenes, R., Neven, G., Priem, B., Samarati, P., Sommer, D., Verdicchio, M.: Exploiting cryptography for privacy-enhanced access control: A result of the prime project. Journal of Computer Security (JCS) 18(1), 123–160 (2010)
2. Ardagna, C., Cremonini, M., De Capitani di Vimercati, S., Samarati, P.: A privacy-aware access control system. Journal of Computer Security (JCS) 16(4), 369–392 (2008)
3. Ardagna, C., De Capitani di Vimercati, S., Foresti, S., Paraboschi, S., Samarati, P.: Minimizing disclosure of private information in credential-based interactions: A graph-based approach. In: Proc. of the 2nd IEEE International Conference on

Information Privacy, Security, Risk and Trust (PASSAT), Minneapolis, Minnesota, USA (August 2010)
4. Ashley, P., Hada, S., Karjoth, G., Schunter, M.: E-P3P privacy policies and privacy authorization. In: Proc. of the ACM workshop on Privacy in the Electronic Society (WPES), Washington, DC, USA (November 2002)
5. Moses, T.: eXtensible Access Control Markup Language (XACML) Version 2.0 (February 2005), http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf
6. W3C: Platform for privacy preferences (P3P) project (April 2002), http://www.w3c.org/TR/P3P/
7. Bock, K.: Europrise trust certification. Datenschutz und Datensicherheit - DuD 32(9), 610–614 (2008)
8. Trust-E: website, http://www.truste.com
9. Ali, M., Sabetta, A., Bezzi, M.: A marketplace for business software with certified security properties. In: Proc. of Cyber Security and Privacy EU Forum (2013)
10. Assert4Soa consortium: Assert4Soa project website, http://www.assert4soa.eu
11. Herrmann, D.: Using the Common Criteria for IT security evaluation. Auerbach Publications (2002)
12. Bezzi, M., Sabetta, A., Spanoudakis, G.: An architecture for certification-aware service discovery. In: Proc. of the 1st International Workshop on Securing Services on the Cloud (IWSSC), pp. 14–21. IEEE (2011)
13. Kaluvuri, S.P., Koshutanski, H., Di Cerbo, F., Maña, A.: Security assurance of services through digital security certificates. In: Proc. of the 20th IEEE International Conference on Web Services (ICWS), pp. 539–546. IEEE (2013)
14. Rost, M., Bock, K.: Privacy by Design and the Protection Goals - English translation of Privacy By Design und die Neuen Schutzziele - Grundsätze, Ziele und Anforderungen. DuD 35(1), 30–35 (2011), https://www.european-privacy-seal.eu/results/articles/BockRost-PbD-DPG-en.pdf (2010)
15. Hansen, M.: Top 10 mistakes in system design from a privacy perspective and privacy protection goals. In: Camenisch, J., Crispo, B., Fischer-Hübner, S., Leenes, R., Russello, G. (eds.) Privacy and Identity Management for Life. IFIP AICT, vol. 375, pp. 14–31. Springer, Heidelberg (2012)
16. Cavoukian, A.: Privacy by design. IEEE Technology and Society Magazine 31(4), 18–19 (2012)
17. Anisetti, M., Ardagna, C.A., Damiani, E., Saonara, F.: A test-based security certification scheme for web services. ACM Trans. Web 7(2), 5:1–5:41 (2013)
18. Damiani, E., Ardagna, C., Ioini, N.E.: Open source systems security certification. Springer, New York (2009)
19. Frantzen, L., Tretmans, J., Willemse, T.A.C.: A symbolic framework for model-based testing. In: Havelund, K., Núñez, M., Roşu, G., Wolff, B. (eds.) FATES 2006 and RV 2006. LNCS, vol. 4262, pp. 40–54. Springer, Heidelberg (2006)
20. Microsoft: Understanding Retention Tags and Retention Policies (December 2012), http://technet.microsoft.com/en-us/library/dd297955%28v=exchg.141%29.aspx
21. IBM: IBM, Enterprise Privacy Authorization Language (EPAL (1.2) (November 2003), http://www.w3.org/Submission/2003/SUBM-EPAL-20031110
22. Ardagna, C., Bussard, L., di Vimercati, S.D.C., Neven, G., Pedrini, E., Paraboschi, S., Preiss, F., Samarati, P., Trabelsi, S., Verdicchio, M.: Primelife policy language. In: Proc. of the W3C Workshop on Access Control Application Scenarios, W3C (2009)

23. Chandramouli, R., Blackburn, M.: Automated testing of security functions using a combined model and interface-driven approach. In: Proc. of the 37th Annual Hawaii International Conference on System Sciences (HICSS), Big Island, HI, USA (January 2004)
24. Jürjens, J.: Model-based security testing using UMLsec: A case study. Electronic Notes in Theoretical Computer Science 220(1), 93–104 (2008)
25. Zulkernine, M., Raihan, M.F., Uddin, M.G.: Towards model-based automatic testing of attack scenarios. In: Buth, B., Rabe, G., Seyfarth, T. (eds.) SAFECOMP 2009. LNCS, vol. 5775, pp. 229–242. Springer, Heidelberg (2009)
26. Bozkurt, M., Harman, M., Hassoun, Y.: Testing web services: A survey. Technical Report TR-10-01. Department of Computer Science, King's College London (January 2010)
27. Canfora, G., di Penta, M.: Service-oriented architectures testing: A survey. In: De Lucia, A., Ferrucci, F. (eds.) ISSSE 2006-2008. LNCS, vol. 5413, pp. 78–105. Springer, Heidelberg (2009)
28. Heckel, R., Lohmann, M.: Towards contract-based testing of web services. In: Proc. of the International Workshop on Test and Analysis of Component Based Systems (TACoS), Barcelona, Spain (March 2004)
29. Bentakouk, L., Poizat, P., Zaïdi, F.: Checking the behavioral conformance of web services with symbolic testing and an SMT solver. In: Gogolla, M., Wolff, B. (eds.) TAP 2011. LNCS, vol. 6706, pp. 33–50. Springer, Heidelberg (2011)
30. Endo, A., Simao, A.: Model-based testing of service-oriented applications via state models. In: Proc. of the 8th IEEE International Conference of Service Computing (SCC), Washington, DC, USA (July 2011)
31. Salva, S., Laurencot, P., Rabhi, I.: An approach dedicated for web service security testing. In: Proc. of the 2010 Fifth International Conference on Software Engineering Advances, ICSEA 2010, pp. 494–500. IEEE Computer Society, Washington, DC (2010)
32. Le Traon, Y., Mouelhi, T., Baudry, B.: Testing security policies: going beyond functional testing. In: Proc. of the International Symposium on Software Reliability Engineering, ISSRE, Sweden (2007)
33. Martin, E.: Automated test generation for access control policies. In: Companion to the 21st ACM SIGPLAN Symposium on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA 2006, pp. 752–753 (2006)
34. Mouelhi, T., Fleurey, F., Baudry, B., Le Traon, Y.: A model-based framework for security policy specification, deployment and testing. In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., Völter, M. (eds.) MoDELS 2008. LNCS, vol. 5301, pp. 537–552. Springer, Heidelberg (2008)