

Enhance Data Privacy In Service Compositions Through A Privacy Proxy

Zhendong Ma*, Jürgen Mangler†, Christian Wagner*, Thomas Bleier*

*Austrian Institute of Technology, Austria
{zhendong.ma, christian.wagner, thomas.bleier}@ait.ac.at

†Secure Business Austria, Austria
jmangler@sba-research.org

Abstract—Web services are loosely coupled Web-enabled applications that can be dynamically invoked to facilitate business interactions through well-defined interfaces over the Internet. However, since personal data will be exchanged between nested Web services, the question how to preserve a user’s data privacy becomes a challenging issue. In this paper we aim to minimize personal data disclosure in service composition that consists several nested Web services. To do so, we propose a practical, scalable and light-weight privacy-enhanced design that uses a privacy proxy to achieve data privacy. We furthermore show that by utilizing the privacy proxy in combination with advertising its capabilities and requirements as service level agreements (SLA’s), it is possible to enhance data privacy in existing service infrastructure in a minimal invasive manner.

Index Terms—data privacy, privacy proxy, Web services, service composition

I. INTRODUCTION

Web services are platform-independent, loosely coupled Web-enabled applications that can use Extensible Markup Language (XML) and protocols such as HTTP and SOAP to engage other Web services to complete a concrete task or conduct a business transaction [1]. As one of the enabling technologies for Service Oriented Architecture (SOA) as well as Cloud Computing, Web services use well-defined interfaces and message-based interaction to expose business services and applications over the existing Web infrastructure, hence enable agile, robust and cost-efficient business interactions among individuals and organizations. Meanwhile, however, the nested usage of Web services (called Web service composition) also raise security and privacy concerns [2], [3], [4], [5], [6], because the control over which and how data is shared between nested services is lost.

To illustrate the privacy issue in Web services, consider a “typical” use case of “travel reservation” adapted from [7] (illustrated in Fig. 1). In this use case, a customer books a complete travel package from a composition of loosely-coupled Web services. The customer finds a travel agent service on the Internet and provides the travel agent service with the travel destination and preferred travel dates. The travel agent service searches and contacts several airline and hotel services to obtain information on the flights and hotel rooms according to the customer’s specification. Then the

travel agent service assembles a list of travel options and presents them to the customer. The customer makes his/her choices and provides the travel agent service with personal information for booking. The travel agent service contacts the credit card service to request a payment authorization. The credit card service checks the request and responses with a payment authorization. With the payment authorization, the travel agent service books the flight and hotel room according to the customer’s choice and finishes the deal by charging a fee from the custom.

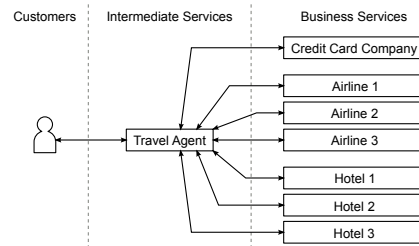


Fig. 1. Use Case: Travel Reservation

During the travel package booking, personal data such as name and credit card number are exchanged and processed by the services. It might be argued that privacy would not be an issue, if all Web services can correctly implement security measures such as XML-Encryption [8] and WS-Security [9] that provide secure data transportation and access control, and the service providers are legally bound to privacy laws or contractual agreement. However, one might also ask whether current approaches are adequate to preserve data privacy in Web services? How can we ensure that a user’s personal data is handled in accordance with the privacy policies and used only for the stated purpose? How can we prevent accidental data leaks resulting from human errors? As Web services are becoming the preferred choice of (big and small) businesses for exposing and providing services on the Internet, more and more personal data will be exchanged and processed by Web services. We therefore conclude, that in order to minimize the risk of misuses and accidental leaks of personal in a SOA architecture, the following two design principles: (1) **minimal disclosure**, implement tight control over how many

times personal data is accessed, (2) **direct disclosure**, allow for a means to determine by whom personal data is accessed, have to be employed, such that privacy principles such as data minimization (i.e., *limit the collection of personal information to what is directly relevant and necessary to accomplish a specified purpose* [10]) can be really implemented in practice.

Our contributions in this paper are as follows. First, we introduce a privacy-enhanced architecture design which applies the concept of a privacy proxy to Web services. Privacy is enforced in the form of a proxy service that assists a user to **tightly control and thus minimize its personal data exposure** in service compositions. Our design is transparent, such that there will be no need to modify existing services and the underlying infrastructures. Second, we propose the utilization of service level agreements (SLA's) to (1) advertise the disclosure capabilities to compliant clients, (2) allow clients to decide upon the potential consequences of a particular service invocation. Third, to evaluate the practical applicability of our design, we conduct an empirical study which shows that the adoption of our design will cause only a negligible impact on the overall performance.

The remainder of the paper is organized as follows: Sec. II reviews the related work, Sec. III describes our system and threat assumptions, Sec. IV presents our design details, Sec. V evaluates the privacy protection and practicability of our proposed design, followed by conclusion in Sec. VI.

II. RELATED WORK

One direction to enhance privacy in Web-based environment is the use of machine-readable, platform-independent privacy policies and the mechanisms to exchange and interpret them among the participants. Example policy languages include Platform for Privacy Preferences (P3P) [11] from World Wide Web Consortium (W3C) and privacy policy profile of eXtensible Access Control Markup Language (XACML) [12]. The similar but more SOA-specific Service Level Agreements (SLA's) allow a user to agree to privacy terms, and monitor their adherence at the service provider. Benbernoui et al. [13] differentiate between (1) data rights, operations services are allowed to perform on user data, and (2) data obligations, operations expected from services to be performed on user data (i.e., deleting user data after usage, notifying clients about usage). They then introduce these concepts into WS-Agreement (WSA), by means of custom terms and a language to describe a process model connecting data rights and data obligations. This approach requires changes to service interfaces, and does not consider means to enforce privacy. In [14] they further this approach by introducing a privacy monitor, assuming that (1) services communicate their use of user data via events, (2) all exchange of user-data between services can be observed from the outside. While this may be true for intra-organizational use of services, the inter-organizational use of services follows different rules. Khazankin et al. [15] go into the same direction by introducing PROVIDENCE, a framework to check security policy violations by means of

content inspection. This approach again assumes the availability of data (and logging information) in an intra-organizational environment. Ul-Haq et al. [16] provide a possible solution for deeply nested (untransparent) hierarchies of services, by proposing a methodology to aggregate and merge SLA's, so that the all security terms hidden in the hierarchy become transparent to a client. We assume this approach to be a possible supplement to our approach as described in Section IV. Carminati et al. [17] propose to do secure composition of services by putting Security Assertions Markup Language (SAML) annotated services together. A matchmaker is intended to compare security requirements, i.e. capabilities and constraints similar to the concepts of Benbernoui et al., to a set of existing services. We again assume that this approach is useful in our setting, as described in Section IV. The effect of privacy policies and SLAs depend on the technical competence and willingness of a Web service to enforce and correctly implement. Our approach in this paper aims at introducing additional means of technical enforcement of privacy.

Another attractive approach in recent years is the development of privacy-enhanced identity management systems and anonymous access schemes. For example, the EU project Privacy and Identity Management for Europe (PRIME) has developed a comprehensive architecture and components for privacy-enhanced identity management and access control. A set of guidelines to develop privacy-aware identity management for different systems is given in [18]. Liberty alliance (now Kantara Initiative) has developed a privacy-enhanced federated identity management system for Web services [19]. Although anonymous identities and credentials enable secure and anonymous access to Web service resources, their effect to enforce content privacy in composite Web services are limited. To give out personal data required by the services. This paper addresses the data privacy problem that cannot be fully solved by anonymous identity management systems.

III. SYSTEM AND PRIVACY ANALYSIS

Web services are autonomous pieces of functionalities with well defined interfaces that use HTTP/HTTPS and SOAP for transport and message exchange. Web services provide abstract APIs which are typically (1) used by other Web services to form a service composition, and (2) are relied on by high level software (e.g. web portal, process aware information systems) which also expose a user interface (UI). We assume that for message exchange between a Web service and a client, common security mechanisms will be implemented at the transport level as well as at the message level (i.e., confidentiality, integrity, identification, and authentication [2] are guaranteed). Furthermore, we assume the existence of a public key infrastructure (PKI) in order to ensure the identity of partners during business interactions. However, security is a necessary but not sufficient precondition for privacy. Privacy breaches can still happen at a secured Web service. The example in Fig. 1 shows that, as a legitimate business provider, a travel agent can archive its customers' travel records. Such information can be exploited to derive individual

travel demands for targeted advertisements or sold to interested parties for profits. A customer will have no means to control and trace its personal data once the data is given to services that often distribute over different business and organization domains. Therefore, we assume that all Web services that collect, process, and store a customer’s personal data pose potential privacy threats to the customer’s data privacy. In this paper, personal data is referred to personally identifiable data, i.e., data that can be uniquely linked to a person.

Usually a customer establishes a business relationship with an *Intermediate Service*, a Web service provider that provides additional value to a customer by wrapping multiple other Web services, furthermore called *Business Services*. For example, a *Travel Agent* is an intermediate service, and the *Hotel* and *Airline* are business services (see Fig. 1). For the relationship between these services, we identify the following requirements for the above mentioned stakeholders.

Minimal Disclosure: It is in a customer’s interest to provide personal data only when necessary.

Direct Disclosure: It is in a customer’s interest to hand (clear text) personal data directly to business partners (no personal data exchange between intermediate and business services).

Hidden Business Logic: It is in an intermediate service’s interest to hide business interactions with business services from the customer to protect its business value.

IV. PRIVACY PROXY DESIGN

To enhance customer data privacy and to fulfill requirements of various stakeholders in service compositions, we propose a Privacy Proxy Service (PPS) design. In our architecture, a PPS is established as a trusted third party in the interaction between a customer and composite services. The main function of a PPS is to temporarily store a customer’s personal data items to minimize the customer’s data disclosure to intermediate services, as well as to control and trace the access to such data. As a third party, the PPS is trusted by:

- the customers to delegate their personal data items (PDI’s).
- the intermediate and business services who require un-tampered customer data, and do not want to expose their business logics,

Personal data items stored in such a PPS may include:

Name	Religion	Billing Address
Address	Phone Number	Passport Number
Birth Date	Email Address	Passport Expiry Date
Sex	Credit Card Type	Passport Issue Date
Nationality	Credit Card Info	

To boost trust and assurance, a PPS can be endorsed and supervised by a trusted organization in the real world, e.g., a national data protection agency, and announced to the public. In order to avoid any issues that would arise from a central storage of personal data such as single point of failure, we designed PPS to really only act as a proxy. It is designed to guarantee the following properties:

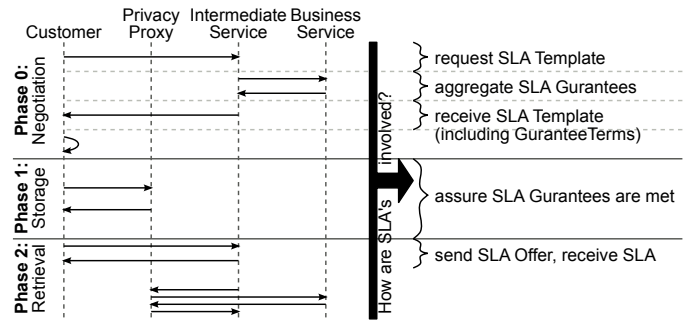


Fig. 2. Privacy Proxy Service Architecture Overview

- Each PDI is stored separately
- Each PDI is stored only for a limited amount of time
- Each PDI is identified by a unique key, further referred to as ticket.
- Each PDI is only accessible once (deleted from PPS after access).
- Tickets are not linkable.

As depicted in our example (Fig. 1), the customer interacts with a *travel agent* in order to establish a business contract. Part of this business contract are (1) the object of agreement (e.g. the organization of a trip) as well as (2) privacy regarding the customers personal data.

A. Interaction Between Privacy Stakeholders

Based on the properties we defined for PPS, we derived the architecture depicted in Fig. 2, including the interactions among different stakeholders.

The interactions are divided into three phases.

Negotiation: After discovering a suitable travel agent, the customer has to provide information about the planned trip in order to receive (1) a set of trips to select from (2) a set of PDIs necessary to book the trip.

Storage: The customer stores each PDI in the PPS. For each PDI a ticket is returned. If a PDI (e.g. the customers name) is needed more than once, it has to be stored more than once.

Retrieval: The travel agency redirects business service calls to the PPS. E.g.,

```
bookFlight (date, name, nationality, passport number)
```

becomes

```
bookFlight (date, ticketa, ticketb, ticketc).
```

The PPS replaces the tickets with actual data, calls the business service and returns the result to the intermediate service.

Note that during the retrieval phase the Business services are communicating solely with the PPS, not with the intermediate service, and not with the customer. The rationale behind this design is as before mentioned (1) disclosing personal data items only directly to business services that need them to fulfill their task, (2) hiding the interaction between intermediate service and business services, allowing for service abstraction from the customers point of view, while simultaneously protecting the business value of the intermediate services.

Furthermore, by letting the PPS call the business services it is possible to use currently existing services without modifying their interface or the underlying infrastructure. This non-invasive privacy enhancement design thus allows to smoothly integrate with current Web service based infrastructures.

B. Service Level Agreements

Service level agreements (SLAs) represent contracts between service providers and customers. They are used to determine a set of common terms, duties and how to monitor them. The currently prevalent machine readable language to describe SLAs is WS-Agreement [20], [21]. The message flow between SLA provider and customer is described in Fig. 2: (1) a customer asks one or more intermediate services for SLA templates, (2) the intermediate service identifies a set of suitable business services, aggregates PDIs and create and SLA Template. On the basis of this template (3) the customer ensures that all preconditions (guarantees) are fulfilled, (4) creates an SLA offer from the template and (5) sends it to the SLA providers, which in turn agrees by sending a SLA.

List. 1. SLA TEMPLATE INCLUDING REQUIRED PDIs

```

1 <ag:Template
2   xmlns:ag="http://schemas.ggf.org/graap/2006/07/ws-agreement"
3   xmlns:wsa="http://www.w3.org/2005/08/addressing"
4   xmlns:pri="http://privacyproxy.org/ns/privacy">
5 <ag:Name>PrivacyTemplate</ag:Name>
6 <ag:Context>
7 <ag:AgreementInitiator>http://travelagent.org</ag:AgreementInitiator>
8 <ag:AgreementResponder>http://privacyproxy.org</ag:AgreementResponder>
9 <ag:ServiceProvider>AgreementResponder</ag:ServiceProvider>
10 </ag:Context>
11 <ag:Terms>
12 <ag:All>
13 <ag:ServiceReference ag:Name="PrivacyProxy" ag:ServiceName="PPS">
14 <wsa:EndpointReference>
15 <wsa:Address>http://privacyproxy.org</wsa:Address>
16 <wsa:ServiceName>PrivacyProxyService</wsa:ServiceName>
17 </wsa:EndpointReference>
18 </ag:ServiceReference>
19 <ag:GuaranteeTerm ag:Name="Privacy">
20 <ag:ServiceScope><ag:ServiceName>PPS</ag:ServiceName></ag:ServiceScope>
21 <ag:QualifyingCondition>
22 <pri:pdi required="true" quantity="4">Name</pri:pdi>
23 <pri:pdi required="true" quantity="1">CreditCardType</pri:pdi>
24 <pri:pdi required="true" quantity="1">CreditCardInfo</pri:pdi>
25 <pri:pdi required="true" quantity="1">PassportNumber</pri:pdi>
26 <pri:pdi required="true" quantity="1">Nationality</pri:pdi>
27 </ag:QualifyingCondition>
28 <ag:ServiceLevelObjective>Privacy</ag:ServiceLevelObjective>
29 </ag:GuaranteeTerm>
30 </ag:All>
31 </ag:Terms>
32 </ag:Template>

```

The structure of an SLA template, as shown in List. 1 is quite flexible, which consists of a name, context to identify contract stakeholders and a set of terms. For each SLA it is possible to define different kinds of terms and to specify which combination of terms are allowed.

The elements included in the *GuaranteeTerm* in List. 1 have the following properties:

- They can be given by both, the contract provider or customer.
- They include a *Service Scope*, identifying a service (part) for which the guarantee applies.
- They include *Qualifying Conditions* to describe the circumstances under which a guarantee is valid.
- They include a guaranteed *Service Level Objective*.
- They may include a set of *Business Values* (rewards and penalties).

Both *Qualifying Conditions* and *Service Level Objective* may be described using a suitable custom expression language.

In our example, the guarantee “*Privacy*” is given by the travel agent, identifying the PPS (<http://privacyproxy.org>) as the service scope. The important part for the PPS is the set of *Qualifying Conditions*, which are in our case a list of personal data items (PDIs), that are (1) required or optional, (2) are required in a certain quantity.

In order to describe this *Qualifying Conditions* we created a simple XML based expression language as shown in List. 2. The language is described in XML Schema and includes all possible PDIs as well as a simple syntax to describe the required conditions.

List. 2. PRIVATE DATA ITEMS (PDIs)

```

1 <xsd:schema
2   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3   xmlns:pri="http://privacyproxy.org/ns/privacy"
4   targetNamespace="http://privacyproxy.org/ns/privacy">
5 <xsd:simpleType name="TPersonalDataItem">
6 <restriction base="xsd:string">
7 <enumeration value="Name"/>
8 <enumeration value="Address"/>
9 <enumeration value="CreditCardType"/>
10 <enumeration value="CreditCardInfo"/>
11 <enumeration value="PassportNumber"/>
12 ...
13 </restriction>
14 </xsd:simpleType>
15 <xsd:complexType name="pdiType">
16 <xsd:simpleContent>
17 <xsd:extension base="pri:TPersonalDataItem">
18 <xsd:attribute name="required" type="xsd:boolean"/>
19 <xsd:attribute name="quantity" type="xsd:int"/>
20 </xsd:extension>
21 </xsd:simpleContent>
22 </xsd:complexType>
23 <element name="pdi" minOccurs="0" maxOccurs="unbounded" type="pri:pdiType"/>
24 </xsd:schema>

```

From a PPS’s perspective, the SLA’s are used to communicate to the customer (1) that privacy protection is available, (2) the location of a preferred PPS, and (3) the necessary tickets that have to be supplied in order to do business with an intermediate service.

C. Detailed example

The sequence diagram in Fig. 3 gives an detailed example of data exchange between the services in the travel reservation use case with our PPS design. The annotation on a straight arrow indicates the data items exchanged in a service request/response call. The annotation on a curved arrow indicates a local action at a service. As a service provider as well as an intermediate service, the travel agent service is responsible for negotiating a SLA over the personal data usage with the user agent and directing service calls to the PPS first. With a privacy proxy in place, a user gains more control over who and how its personal information can be accessed in service compositions.

D. Alternatives

One might ask whether a PPS is necessary to achieve minimal disclosure in composite Web services? One alternative is to use **direct communication**, i.e., to bypass all intermediate services and let individual business services directly communicate with the user to obtain personal data. Logically, this can be regarded as to move the functionality of the PPS to the customer side. However, direct communication will be against the requirement on hidden business logic (cf. Sec. III).

Another approach is to use **public key encryption**, i.e., to encrypt personal data using the end service’s public key. The

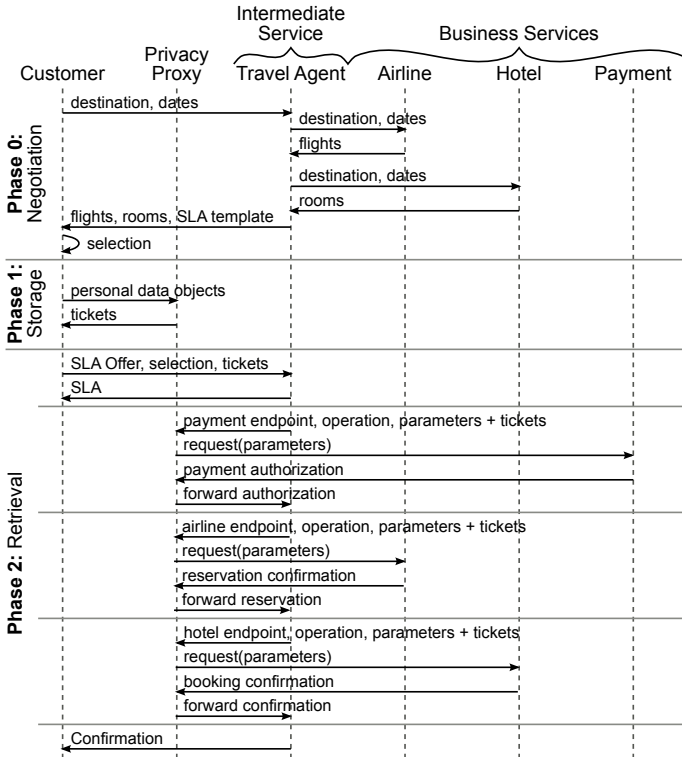


Fig. 3. Data exchange in travel reservation with PPS

prerequisite is that an intermediate service needs to forward the public keys of the final target services to the user. However, this alternative also breaks hidden business logic and threatens the intermediate service's business value.

V. EVALUATION

This section evaluates our design to show that it is effective in data privacy enhancement, scalable, and practical.

A. Data-flow analysis

Data-flow analysis was used to gather information in a computer program for debugging and optimization of the program [22], or to reason about the sensitive information exposed by a program during its execution [23]. The same principle can also be applied to trace personal data communicated and exchanged in a service composition.

The data items collected at each of the Web services in the use case (cf. Fig. 3) are summarized in Tab. I. Items in bold form indicate personal data. Since no personal data is collected at the travel agent service, the rest of the data items cannot be linked to an identifiable person. As a result, a user can book a complete trip without revealing any personal data to the travel agent service. Other services in the table have collected personal data. However, for the business services, the data items are the minimal required data to complete a business transaction. The PPS has collected the most personal data. If it can enforce all measures for privacy protection, the PPS will pose no privacy threats to the users.

TABLE I
DATA ITEMS AT EACH SERVICE

Service	Data
Privacy Proxy	name, credit card type, credit card info passport number, nationality
Travel Agent	destination, travel dates, flights, rooms, tickets
Airline	name, passport number, nationality, flight
Hotel	name, room
Payment	name, credit card type, credit card info

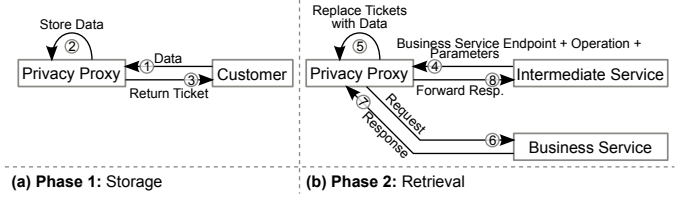


Fig. 4. Performance Evaluation

B. Performance evaluation

One of the often used arguments against a privacy-enhanced design is the performance issue in terms of computation and communication overhead. Admittedly, a PPS will introduce additional complexity and latency in Web service transactions. However, if privacy is arguably a service that should not be given for free, then the overhead is acceptable as long as it is minimized. In the following, we evaluate our design by analytical as well as empirical analysis.

In our analytical analysis, we calculate the latency from introducing a PPS with respect to the system without the PPS. Let t_c be the latency of one service call, and it is reasonable to assume that network latency is uniform for all service calls between any two services over the Internet. Based on Fig. 4, we can calculate the latency of additional service calls in our design to be $2t_c + 2nt_c$. The first term $2t_c$ (1)(3) is the latency in the storage phase, and the second term $2nt_c$ (4)(8) is the latency in the retrieval phase, in which n is the number of business services that need to get personal data from the PPS. Business service calls (6)(7) can be neglected, as without a PPS it would also be necessary for the intermediate service to call business services, and it is reasonable to assume that the latency between PPS and business services is similar to the latency between intermediate and business services. Furthermore, let t_p (2)(5) be the latency at the PPS service, i.e., the time lapse of receiving a service call and responding to it. Consequently, the total latency due to the processing time at the PPS is $t_p + nt_p$, in which n is again the number of business services that need to get personal data from the PPS. Hence, we have the total additional latency t_{add} as

$$t_{add} = 2t_c t_p (1 + n) \quad (1)$$

Since t_c and t_p can be assume to be constant, n are natural numbers incremented by 1, the increase of latency caused by the PPS will be linear and gradual.

The above analysis considers only one user. In our design, the PPS must handle multiple users at the same time. In order

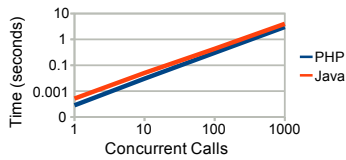


Fig. 5. SOAP Performance of Privacy Nodes

to gain further insight into the practicability of our design, we created two prototype implementations of the PPS, in JAVA and PHP, respectively. The PPS itself is a simple key/value store with a SOAP interface. For the retrieval phase it is able to receive arbitrary SOAP calls, replace tickets, forward the calls to business services and return the result to the intermediate service. The prototype is implemented on a PC with Intel(R) Xeon(TM) CPU, 4 cores, 3.00GHz per core, 2048 KiB cache per core. During operation the PPS prototypes showed the following performance characteristics:

TABLE II
WS PERFORMANCE OF CONCURRENT CALLS IN SECONDS

# of concurrent calls	PHP	JAVA
1	0.005197	0.0094684
10	0.0559051	0.0925024
100	0.5565541	0.7970778
1000	5.5153792	7.3207632

The numbers in Tab. II are the average of 10 subsequent runs ($5 \times$ store, $5 \times$ retrieve), the PPS is capable of handling loads of about 190 concurrent calls per seconds. As can be seen in Fig. 5 the increase in latency is indeed linear and gradual, which is in accordance with our derivation in (1).

VI. CONCLUSION AND FUTURE WORK

Web services are becoming a popular technology to implement SOA and Cloud computing to provide services over the Internet. As more and more personal data will be exchanged and processed by Web services, data privacy becomes an issue. In this paper, we proposed a privacy proxy service (PPS)-based architecture to enhance user data privacy in service compositions with nested Web services. Our privacy-enhanced design helps the achievement of minimal disclosure of personal data and tighter control over the access to such data.

We will extend our work into several directions in the next step. Our current design does not prevent services from colluding. Hence preserving privacy in spite of colluding services while keeping the overhead at minimal will be a challenging and interesting problem to solve. Besides, current SLA negotiation approaches are still quite static. Therefore, dynamic SLA negotiation for privacy purpose will be addressed at both the theoretic and practical level. We envision that our PPS design will have practical value for the upcoming boom of Cloud computing. As a result, designing load balancing and other mechanisms to make the PPS more robust in Cloud computing is another direction of our future work.

ACKNOWLEDGEMENTS

The research was partially funded by COMET K1, FFG - Austrian Research Promotion Agency.

REFERENCES

- [1] M. P. Papazoglou, *Web Services: Principles and Technology*. Pearson, Prentice Hall, 2008.
- [2] IBM and Microsoft, "Security in a web services world: A proposed architecture and roadmap," <http://msdn.microsoft.com/en-us/library/ms977312.aspx>, 2002, white Paper.
- [3] L. Martino and E. Bertino, "Security for web services: Standards and research issues," *Int. J. of Web Services Research (IJWSR)*, vol. 6, no. 4, pp. 48–74, 2009.
- [4] P. C. K. Hung, E. Ferrari, and B. Carminati, "Towards standardized web services privacy technologies," in *ICWS*, 2004.
- [5] B. Carminati, E. Ferrari, and P. C. Hung, "Exploring privacy issues in web services discovery agencies," *IEEE Security and Privacy*, vol. 3, pp. 14–21, 2005.
- [6] G. O. M. Yee, "Estimating the privacy protection capability of a web service provider," *Int. J. Web Service Res.*, vol. 6, no. 2, pp. 20–41, 2009.
- [7] H. Haas, "Web service use case: Travel reservation," <http://www.w3.org/2002/06/ws-example>, May 2002.
- [8] W3C, "XML Encryption Syntax and Processing," 2002.
- [9] OASIS, "WS-Security 1.1," 2006.
- [10] European Data Protection Supervisor (EDPS), "Glossary of protectoin of personal data," <http://www.edps.europa.eu/EDPSWEB/edps/cache/off/EDPS/Dataprotection/Glossary>.
- [11] W3C, "The Platform for Privacy Preferences (P3P)," <http://www.w3.org/P3P/>.
- [12] OASIS, "Privacy policy profile of XACML v2.0," http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-privacy_profile-spec-os.pdf, 2005.
- [13] S. Benbernou, H. Meziane, Y. H. Li, and M. Hacid, "A privacy agreement model for Web services," in *Services Computing, IEEE International Conference on*. Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 196–203.
- [14] S. Benbernou, H. Meziane, and M. S. Hacid, "Run-Time monitoring for Privacy-Agreement compliance," in *Service-Oriented Computing – ICSC 2007*, 2007.
- [15] R. Khazankin and S. Dustdar, "PROVIDENCE: a framework for private data propagation control in Service-Oriented systems," in *Towards a Service-Based Internet*, 2010.
- [16] I. Haq, A. Huqqani, and E. Schikuta, "Aggregating hierarchical service level agreements in business value networks," in *Business Process Management*, 2009.
- [17] B. Carminati, E. Ferrari, and P. C. K. Hung, "Security conscious web service composition," in *Web Services, IEEE International Conference on*. Los Alamitos, CA, USA: IEEE Computer Society, 2006, pp. 489–496.
- [18] M. Hansen, A. Schwartz, and A. Cooper, "Privacy and identity management," *IEEE Security and Privacy*, vol. 6, pp. 38–45, 2008.
- [19] M. Alsaleh and C. Adams, "Enhancing consumer privacy in the liberty alliance identity federation and web services frameworks," in *Privacy Enhancing Technologies*, 2006, pp. 59–77.
- [20] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, "Web services agreement specification (WS-Agreement), 2007," 2004, [Last access: 02/12/2011]. [Online]. Available: <http://www.gridforum.org/documents/GFD.107.pdf>
- [21] H. Ludwig, "WS-Agreement concepts and Use-Agreement-Based Service-Oriented architectures," *IBM Research Division, Technical Report*, 2006, [Last access: 02/12/2011]. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.476&rep=rep1&type=pdf>
- [22] U. Khedker, A. Sanyal, and B. Karkare, *Data Flow Analysis: Theory and Practice*. Boca Raton, FL, USA: CRC Press, Inc., 2009.
- [23] M. Backes, B. Köpf, and A. Rybalchenko, "Automatic discovery and quantification of information leaks," in *IEEE Symposium on Security and Privacy*, 2009, pp. 141–153.