Extension for Information Card Systems to Achieve User-Controlled Automated Identity Delegation

Thorsten Hoellrigl, Holger Kuehner, Jochen Dinger, Hannes Hartenstein
Steinbuch Centre for Computing (SCC) and Institute of Telematics
Karlsruhe Institute of Technology (KIT), Germany
Email: {Hoellrigl|Holger.Kuehner|Dinger|Hartenstein}@kit.edu

***Abstract*—The growing number of IT services in distributed systems is directly related to the security and privacy of personal data. User-centric federated identity management (FIM) attends to the privacy issue by enabling users to approve each data dissemination between the providers of identity-related information, so-called identity providers (IdPs), and the consumers of this information, the service providers (SPs). Furthermore, user-centric FIM tries to improve security and usability by providing users with a consistent digital-identity experience using so-called information cards (InfoCards). The InfoCard-based approach can help to improve usability, privacy and security, however, the approach is limited to front-channel communication and requires that each data exchange is manually approved by the user. A back-channel communication might be required in scenarios, in which an IdP wants to notify SPs about e.g. a deactivation of a user. In [3] we proposed an approach, named User-Controlled Automated Identity Delegation, that allows a back-channel communication by automating user approval based on delegation. In this paper we demonstrate the practicality of the approach in a real-world scenario by providing a performance evaluation conducted on a prototypical implementation.**

I. Introduction

With the growing number of IT services in distributed systems, privacy and security of personal data is gaining in importance. In addition, usability is becoming a crucial asset as users have to be able to keep track of which personal data is retained by which service. The user-centric federated identity management (FIM) strives to improve *privacy* by "empowering human beings to control their identities" [4]. A major characteristic of this "user empowerment" is that a user is enabled to use multiple identity providers (IdP), each managing a part of her identity-related information, namely attributes. The users may even be allowed to manage some attributes by themselves at so-called self-hosted IdPs. To ensure *usability*, user-centric FIM introduces so called information cards (InfoCards). InfoCards are representations for a collection of user attributes issued by the IdPs providing users with a consistent digital user experience. In real-world analogy these cards are used via wallet-like graphical user interfaces called identity selectors. Concerning *security*, InfoCard-based systems, for example, effectively counter phishing attacks.

As demonstrated in [3] user attributes are typically stored redundantly in the InfoCard-based approach, therefore, raising consistency issues [2]. However, in consideration of consistency, the InfoCard-based approach suffers from certain drawbacks. First of all, current InfoCard-based systems only support front-channel communication, i.e., all communication between an IdP and SPs has to pass the user client as this component enables the user to approve the data exchange [6]. A back-channel communication, i.e., a direct communication between IdP and SPs without involving the user client is not considered, as this carries with it the danger of an unauthorized attribute dissemination [6]. In other words, if the user is offline, data exchange is not possible. However, scenarios exist in which an IdP might want to inform SPs about certain changes of user attributes, in particular, when these changes are directly related to service authorization, such as a deactivation of a user (cp. [5]). If these changes cannot be communicated, this might lead to an unauthorized and incorrect service delivery. Another limitation of current InfoCard-based systems is that a user is not able to select multiple cards in a single transaction, e.g., to combine access control relevant attributes.

In summary, we argue that the InfoCard-based approach can help to improve security, privacy and usability in distributed IT systems, but needs to be enhanced. Therefore we introduced an approach called *User-Controlled Automated Identity Delegation (UCAID)* in [3]. To demonstrate the practicality of the approach in a real-world IdM system, we provide a qualitative and quantitative evaluation in this paper. The performance evaluation conducted on a prototypical implementation based on Windows CardSpace shows acceptable response times. The paper is structured as follows: In Section II we provide a brief summary of UCAID. A qualitative evaluation is presented in Section III, followed by a performance evaluation in Section IV. Section V concludes the paper.

II. Background

UCAID provides the basis for a privacy-preserving back-channel communication in InfoCard-based systems. The approach introduces an additional party called *Identity Delegate* that acts on behalf of the user when services want to retrieve attributes, in particular, when the user is offline. Thereby, the delegate ensures two major aspects of *user centricity*: first, the delegate acts on behalf of the user in consideration of the user approval process by applying user-defined policies when a service wants to retrieve information, and second, the delegate acts in place of the user client in the attribute flow, i.e., instead of passing through the user client, the attribute flow passes through the delegate. UCAID does not require manual user interaction for the approval, instead, user approval is achieved

by enabling the user to define policies, based on which the delegate is able to *automate* user approval, therefore, achieving a privacy-preserving back-channel communication.

The user trusts in the delegate to retrieve and disseminate attributes according to her policies[1]. However, according to the principle of least privilege, the delegate is only assigned restricted permissions. Therefore, an arbitrary access to user attributes is avoided, so the required amount of trust in the delegate is reduced. In particular, the concept assumes that IdPs are trusted to be configured to issue only attributes to the delegate that have already been encrypted and signed for requesting SPs. As FIM in general requires users and SPs to trust in IdPs, this is not increasing required trust. Therefore, the delegate acts as "identity relay" [1] that only gathers and forwards attributes and does not store attributes locally. Hence, an attacker taking control of the delegate is not able to read, store or alter user attributes.

Using the delegate involves different possibly recurring steps. In the *preparation step*, the delegate, designed as a service provided by a third party, has to be prepared by the operator to be usable for the user. The next step, named *registration step*, enables the user to introduce her IdPs to the delegate. In the *authorization step*, the user authorizes an SP to retrieve attributes through the delegate by specifying SP-specific attribute retrieval policies. More SPs can be authorized by repeating this step. Essential for achieving back-channel communication is the last step, named *update step*, where the SP requests attributes from the delegate without requiring any manual user interaction.

III. Qualitative Evaluation

In the following, we qualitatively evaluate how the central ideas of user centricity – usability and privacy – have been addressed by UCAID.

A. Usability

The user is able to delegate the task of controlling disclosure of attributes in her absence to the Identity Delegate, thereby instructing the delegate to automatically control each data dissemination. This procedure saves the user a manual dissemination of attribute changes. An automatic control of attribute dissemination is achieved through dissemination policies, enabling the user to control which provider is able to retrieve which attributes from which IdPs. Based on the dissemination policies, the Identity Delegate provides a central point for the user to keep track of the dissemination of her attributes. In addition, both the authorization of an SP and federating identities are performed by reusing the underlying mechanisms of the information card system, thus, the user authorizes services and federates identities by using the identity selector. Therefore, UCAID provide a consistent and uniform user experience.

[1]As a basis of a qualitative privacy evaluation, we provide a complete trust model in Section III.

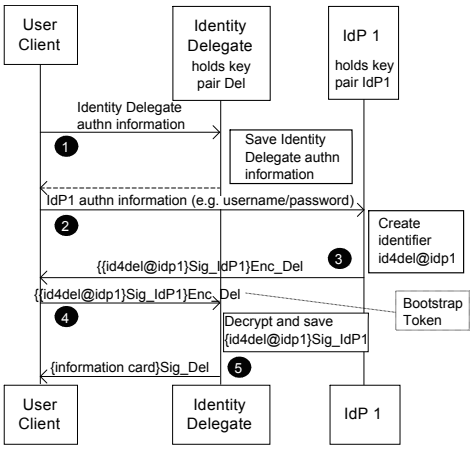

Fig. 1. Initial registration step for one IdP. Payload P signed with private key of participant X is represented by $\{P\}Sig\_X$, whereas payload P encrypted with public key of participant Y is represented by $\{P\}Enc\_Y$.

B. Privacy

UCAID requires the user to trust in SPs and IdPs to keep her attributes confidential, and it requires SPs to trust in IdPs to authenticate the user and provide attributes correctly. These trust assumptions hold true for FIM systems in general. In addition, UCAID depends on the following trust assumptions:

- The Identity Delegate is trusted to enforce the dissemination policy exactly as specified by the user.
- The Identity Delegate is trusted to perform the lookup of linking information correctly, i.e., all tokens sent to an SP refer to the user for which the SP requested attributes.

In the following, we shortly summarize the security mechanisms leveraged by UCAID. The confidentiality and integrity of each data exchange in UCAID is protected by transport security, e.g., by TLS. Fig. 1 illustrates the registration step. In this step, the user first authenticates against the delegate via web browser, e.g., using username/password or client certificate authentication. Thereafter, she authenticates against one or more IdPs by means of the Identity Selector (2), so phishing attacks are mitigated. Furthermore, the unidirectional identifier created by the IdP is encrypted and signed (3), thus, neither user nor delegate are able to tamper with the identifier.

To prevent phishing attacks in the authorization step, the user authenticates to the delegate via the Identity Selector. The unidirectional identifier generated by the delegate for the SP is protected by asymmetric encryption to preserve confidentiality.

Fig. 2 depicts the update step. In this step, SPs and the delegate typically authenticate using client certificates (1 and 2). The IdPs sign user attributes and encyrpt them with the public key of the SP before sending them to the delegate (3), so the delegate is neither able to read nor to alter the attributes.

Acting on the trust assumptions and security mechanisms, threats to the confidentiality, integrity and unlinkability of user identites can be identified.

First, IdPs or SPs could maliciously try to correlate the local identities of their users by means of the user identifier issued by the delegate. This allows them to get a more comprehensive

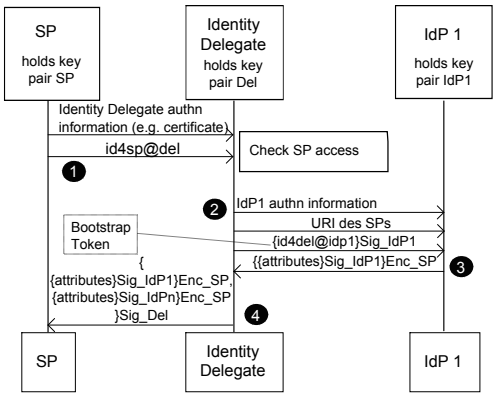

Fig. 2. Back-channel attribute retrieval from one IdP. Payload P signed with private key of participant X is represented by $\{P\}Sig\_X$, whereas payload P encrypted with public key of participant Y is represented by $\{P\}Enc\_Y$.

profile of their users by, e.g., compiling service usage histories. As the delegate issues unidirectional identifiers, i.e., different IdPs and SPs hold different identifiers to refer to the same delegate user, a correlation of local user identities is not easier for IdPs or SPs than in any other FIM system.

Second, an attacker could get in possession of all the data the delegate holds. In this case, the attacker only gets the username, possibly some information related to user authentication at the delegate, e.g., a password hash, and a set of identifiers for each delegate user. These identifiers are unidirectional and opaque, so the attacker does not get the username of the user on any IdP or SP. Furthermore, he does not get any user attributes like the full name or the address, as these attributes are not saved by the delegate. Only if the attacker collaborates with at least two IdPs or SPs, a correlation of local user identities on these IdPs or SPs would be possible by using the linking information gained from the delegate.

Third, an attacker could control the delegate and gain access to its private key. In this case, he may get access to the data the delegate holds. The attacker is not able to modify user attributes before they are sent to an SP, as attributes are signed by the IdPs, and is not able to read attributes by himself, because attributes are encrypted for the respective SP (step (3) in Fig. 2). However, the attacker is able to forward user attributes to a collaborating SP, no matter if this SP were authorized by the user.

In summary, an attack compromising the Identity Delegate will not lead to major privacy leaks unless the attacker manages to collaborate with at least one IdP or SP.

IV. Performance Evaluation

In the following, we present performance tests conducted on the prototype to gain an indication of its practicality in a real-world scenario.

A. Testbed

Load tests have been conducted in a testbed of multiple *IdPs*, *an SP*, and *an Identity Delegate* based on Windows Server 2008 SP2. All components are hosted as virtual machines in a *VMware ESXi server 4.0* installed on an IBM x3550 with 4x3 GHz Intel Xeon processors 5160 and 32 GB RAM. The IdP is an instance of ADFS 2.0 RC with 2 GB RAM, whereas the SP and the Identity Delegate are hosted on a virtual machine with 4 GB RAM. As all components are hosted in the same physical server, network latency is negligible. As load testing tool, we use Microsoft Visual Studio 2010.

B. Additional Latency Caused by the Delegate

In the following, we evaluate the additional latency caused by UCAID in comparison with the time it would take an SP to retrieve user attributes directly from an IdP. For this purpose, we measure the time it takes an SP in UCAID to retrieve user attributes, i.e., the response time of the Identity Delegate, and compare it to the reponse time of an IdP. All measurements are performed on a prototypical implementation of the Identity Delegate as described in [3] and an ADFS 2.0 IdP.

The following steps that have to be performed by the delegate in addition to direct attribute retrieval at an IdP:

(i) *SP identification* – the Identity Delegate checks the *URL* of the requesting service. (ii) *identifier mapping & policy check* – the delegate maps the identifier of the requested user to the correlated identity on the delegate. In addition, the policy for this SP and user identity is checked. (iii) *determination of authoritative sources* – the Identity Delegate determines the class of semantically related attributes and the authoritative source for each requested attribute and sends the requests to the IdPs. (iv) *transformation determination* – to cope with heterogeneities the delegate then determines the transformation rules required to map the IdP to an intermediary schema and the intermediary to the SP schema. (v) *aggregating, encrypting, and signing token* – in the last step the delegate aggregates retrieved tokens in a new token, adding the transformation rules; the token is encrypted, signed, and sent to the SP.

The duration of these steps, and thereby the additional latency caused by the delegate, may be influenced by the following parameters:

- An increasing number of Identity Delegate users leads to an increase in *user database size*, i.e., the database in which the delegate stores linking information, authoritative sources and other administrative information.
- The *number of attributes* that must be retrieved per SP request can vary.
- An increasing number of IdPs per user leads to an increasing *number of IdPs* from which attributes must be retrieved and aggregated for each SP request.
- Increasing the number of SPs retrieving attributes from the delegate, the number of users an SP retrieves attributes for or the rate at which an SP requests attributes all lead to an increasing *delegate request rate*.

To quantify the influence of these parameters on the additional latency, we first measured response time in a minimal setting, where each parameter was set to a minimum, so only one delegate user exists, and one SP retrieves one user attribute provided by one IdP. Furthermore, the delegate neglects

database access and holds all relevant data in main memory, as we want to abstract from database details in order to avoid influences of the database and the code to access database on the measurements. In further steps, we increase each parameter to gain an indication of the scalability of UCAID.

In the minimal setting, the Identity Delegate response time is about 50 ms, whereas IdP response time is about 20 ms. This results in an additional latency of about 30 ms, which is mainly due to asymmetric encryption and signing.

To measure the influence of an increased delegate *user database size*, we changed the prototype to fetch user-related data from a database hosted by SQL Server 2005 Express Edition. Other data, e.g., transformation rules, were still held in main memory. Increasing the number of delegate users from 1 to 1,000,000 users resulted in the database growing from 3 MB to about 300 MB, but lead to no measurable rise in the additional latency caused by the delegate.

If the *number of attributes* which are requested by an SP is increased, the number of transformation rules which must be sent to the SP grows proportionally. Consequently, the size of the token that is issued by the delegate to the SP is increased. We varied the number of attributes sent to the SP between one and ten typical attributes like name, phone number and so on, and observed no impact on the additional latency.

The influence of an increasing *number of IdPs* from which attributes are retrieved for a single SP request depends if the delegate retrieves attributes from the IdPs sequentially or concurrently. In case of sequential retrieval, the sum of all IdP response times determines the delegate response time, so the additional latency approximately grows proportionally with the number of IdPs. In case of concurrent retrieval, the delegate response time is only influenced by the IdP which is responding most slowly. Therefore, concurrent token retrieval has only minor influence on the additional latency. We like to note that concurrent token retrieval may lead to a large number of resource-consuming threads necessary for an asynchronous processing of IdP responses.

We measured the impact of increasing the *delegate request rate* by a series of load tests. Each load test was run for ten minutes and repeated ten times. Network connection failures have not been considered. Fig. 3 shows the average response time of the Identity Delegate at a constant rate of 40 to 80 requests per second. The response times of an IdP at the same request rates are also plotted. At request rates higher than 80 requests per second, we experienced intensely rising standard deviation and response times which could not be determined reliably. Comparing the response times of delegate and IdP indicates a slow rise of the additional latency, e.g., at a constant rate of 60 requests per second, the additional latency is about 40 ms, whereas at a constant rate of 80 requests per second, the additional latency is about 60 ms.

Thus, assumed that the requests of the providers are uniformly distributed over time at a constant request rate of 80 requests per second, the delegate is able to respond to about 250,000 provider requests per hour with an additional latency of about 60 ms, which is acceptable as network clients typically do not time out before a few seconds. This way, the Identity Delegate can handle up to, e.g., about 250,000 user identities which are updated once an hour. As the alteration rate of a great fraction of identity-related information is typically low, an update duration of one hour is acceptable in many scenarios.

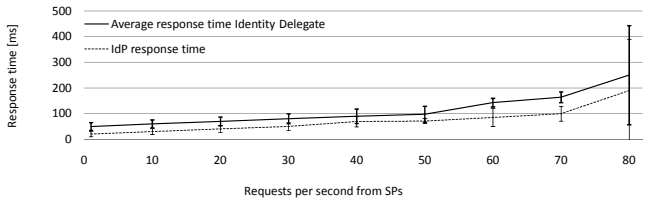

Fig. 3. Response times (the time between sending a request and receiving the response) measured by requesting providers at 1 to 80 requests per second

In summary, the performance tests indicate that the delegate user database size or the number of attributes requested by an SP have only minimal impact on the additional latency caused by the delegate. The number of IdPs can be varied without a major impact on the additional latency if the delegate retrieves attributes from multiple IdPs concurrently. Increasing the delegate request rate leads to a slow rise of the additional latency.

V. Summary and Outlook

In summary, we argued that current InfoCard systems lack mechanisms for back-channel communication to adequately address consistency issues. We briefly summarized *User-Controlled Automated Identity Delegation* that adds back-channel communication and attribute aggregation functionality to InfoCard systems. A qualitative analysis demonstrates how the central ideas of user centricity – usability and privacy – are addressed. Furthermore, a performance evaluation conducted on a prototypical implementation shows the practicality of the approach in real-world scenarios as attribute requests are answered in an acceptable time.

A future task is to enhance the delegate with an update functionality with respect to level of assurance aspects. Thus, the delegate could become an "identity-information control center" enabling users to keep track of which information is retained by which service and allowing them to trigger updates when their identity-related information changes.

References

[1] D. W. Chadwick and G. Inman. Attribute Aggregation in Federated Identity Management. *IEEE Computer*, 42(5):33–40, 2009.

[2] T. Hoellrigl, J. Dinger, and H. Hartenstein. A Consistency Model for Identity Information in Distributed Systems. In *COMPSAC 2010: Proceedings of the 34th IEEE Computer Software and Applications Conference*, pages 252–261. IEEE, 2010.

[3] T. Hoellrigl, T. Kühner, J. Dinger, and H. Hartenstein. User-Controlled Automated Identity Delegation. In *CNSM 2010: Proceedings of the 6th IEEE/IFIP International Conference on Network and Service Management*, pages 230–233. IEEE/IFIP, 2010.

[4] E. Maler and D. Reed. The Venn of Identity. *IEEE Security and Privacy*, 6(2):16–23, 2008.

[5] OASIS SAML V2.0 Change Notify Protocol V1.0 Working Draft, 2010.

[6] OASIS Security Assertion Markup Language V2.0 Glossary, 2005.