A Consistency Model for Identity Information in Distributed Systems

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

Abstract—In distributed IT systems, replication of information is commonly used to strengthen the fault tolerance on a technical level or the autonomy of an organization on a business level. In particular, information related to the identity of a user, which is used to authorize service access, is often replicated for these reasons. To ensure correct authorization decisions, replicas have to be kept consistent. However, an appropriate definition of “consistency” is required that takes into account the need for the following aspects: (i) semantic and causal relations between identity information, and (ii) temporal aspects with respect to an acceptable duration of the dissemination of occurring attribute changes. Both identity-information specifics and temporal aspects are not addressed sufficiently by existing consistency models.

In this paper we introduce a consistency model for identity information in distributed systems named ID-consistency. ID-consistency is based on a formalization of identity information and considers semantic and causal relations as well as a so-called inconsistency window that denotes the time period between a change to information and the moment when the change is fully disseminated. Therefore, the model reveals the fundamental structure of an IdM system and helps in the design and analysis of corresponding dissemination middleware in distributed systems. We exemplarily show how to make use of the concept of ID-consistency to analyze and improve a real-world IdM system using CardSpace for demonstration purposes.

I. INTRODUCTION

A key concern in distributed and ubiquitous IT systems is the way a user is provided with a seamless and secure access to services. Single sign-on (SSO) that allows a user to authenticate to one service and then access another one without reauthentication can improve usability. To enable SSO, identity-related information – namely attributes – has to be exchanged. For instance, the service that authenticates a user has to send an authentication status to subsequent services. In addition, attributes such as an affiliation might be exchanged for authorization purposes. For the exchange of attributes typically an identity management (IdM) system is used.

The federated identity management (FIM) model, a promising IdM approach in distributed systems, is intended to exchange identity-related attributes between different autonomous organizations [8]. The major idea behind the FIM model is that service providers (SPs) are able to delegate the

authentication and user management to so-called identity providers (IdPs). This delegation opportunity avoids redundancy and information inconsistencies respectively as identity-related information is gathered directly at the authoritative sources. However, there are various technical as well as business-driven reasons to replicate information between different organizations. For instance, replicating information strengthens the fault tolerance on a technical level as well as the autonomy of an organization on a business level. Another reason are legacy systems, e.g., systems that cannot gather identity-related information from external repositories. This information replication in distributed, autonomous IT systems has certain consequences: (i) each user can have multiple independent identities as many SPs and IdPs require to create a new one. (ii) Same information might have heterogeneous representation of the because typically the autonomous systems do not coordinate their schema. (iii) Changes to information have to be disseminated to all replicas and semantically related heterogeneous information to avoid inconsistencies.

An analysis of current FIM software and protocols [6] showed that they cope with multiple user identities by enabling to link them with each other via so-called identity federation [13]. However, they do not consider the consistency issue sufficiently [6]. Neither the dissemination of occurring changes nor the heterogeneous character of information is addressed adequate. Therefore, the question arises what could be an appropriate consistency model for identity information in distributed environments. In other research areas, e.g., in the field of shared databases, data consistency is well analyzed and specified via various consistency models. Thus, the following questions have to be addressed: (i) are existing consistency models adequate to define identity-information consistency in distributed system? (ii) Are identity-information specifics such as semantic and causal relations addressed? (iii) Are there any measures for consistency, e.g., considering time constraints?

“In an ideal world there would be only one consistency model: when an update is made all observers would see that update” [17]. Synchronous transaction models approximate such an ideal consistency model. But, as we will show later, synchronous transaction models might not be optimally suited for identity-information consistency in distributed environments, mainly due to their impact on performance and availability of the overall systems (see e.g. [21]). More relaxed consistency models do not have such a strong impact on the...
performance and availability, whereas these models assume that inconsistency can be tolerated to a certain degree. Our thesis is that identity-information consistency in distributed environments can be loosened to a certain degree in the favor of performance and availability. A reason therefore is that identity-related information typically has a very low alteration rate and concurrent access to local copies is uncritical in distributed IdM systems. However, identity information has characteristics that influence consistency, but are not considered by current consistency models. For instance, the heterogeneous character of identity information is not addressed. In addition, current relaxed consistency models do not limit the time period between a change to information and the moment when the change is fully disseminated, which would help to clearly state when information is consistent.

The contributions of this paper are as follows:

(i) We use existing information consistency concepts and extend them by identity-information specifics, e.g., semantic and causal relations, resulting in a formal model that allows to define so-called ID-consistency for identity information in ubiquitous, distributed systems. The model considers a limitation of the time duration between an occurring change and the moment when the change is fully disseminated via a so-called inconsistency window.

(ii) We exemplarily demonstrate how the model helps when coping with consistency in real-world IdM systems using the example of CardSpace [19]. Furthermore, we indicate how to determine the inconsistency window via a performance evaluation of an exemplary implementation that fulfills ID-consistency in CardSpace.

The paper is organized as follows: We discuss background and related work in Section II. In Section III we formulate a consistency model for identity information in distributed systems. Using CardSpace as an example, we demonstrate the application of the consistency model in Section IV. Furthermore, we cover the experience made with an implementation to cope with ID-consistency in CardSpace. Section V concludes the paper and discusses future work.

II. BACKGROUND AND RELATED WORK

In the following, we survey the consequences of identity-information replication. Typically replication in distributed, autonomous systems results in information heterogeneity. Therefore, we briefly discuss approaches that try to cope with data heterogeneities. Furthermore, we analyze the adequacy of existing consistency models and introduce other identity management related models.

A. Consequences of identity-information replication in distributed systems

Fig. 1 shows a typical scenario of identity-information replication in distributed systems. In the depicted scenario identity information that was initially registered at IdP1 was replicated to SP1 and SP2. In the following, we briefly discuss the main consequences of information replication referring to the example in Fig. 1:

(i) Each user can have multiple identities that are initially unrelated to each other. Thus, IdP1 that identifies a user with the Id J.Doe@IdP1 is not able to refer to an identity with the Id JD@SP1 of the same user at SP1.

(ii) Identity information is not only replicated but rather transformed. In the example, the attribute Role with the value Student at IdP1 was replicated to SP2 that transforms and stores this information in the attribute Studierender. Thus, in another language. Consequently, the attributes are named differently and have differing values, but they are semantically identical. Transformations can range from name alterations, over concatenations of multiple attributes to complex transformations [9], e.g., DoB with the value 1987-12-15 is transformed to IsOfFullage with the value yes.

(iii) When the information located at IdP1 changes, SP1 and SP2 do not see this change, which potentially results in inconsistencies. In addition, when the Fullname at IdP1 changes it is necessary to adapt SN at SP1 as well as Anzeigenname at SP2 to keep these heterogeneous representation of the same information consistent.

B. Data Heterogeneity

As identity-information heterogeneity crucially influences information consistency, we briefly survey approaches that try to cope with data heterogeneities.

In [7] the authors discuss data heterogeneities in shared database systems. The authors categorize data heterogeneities in syntactic and semantic heterogeneity. A formal approach is introduced that addresses semantic similarities of information, i.e., semantic equivalence and semantic relationship of two objects, initially introduced by [14]. Semantic equivalence of two objects is given if there is a bijective mapping between the domain and values of one object to the domain and values of the other object. Semantic similarities also include generalizations, i.e., the value of one object is mapped to a more general value of another object.

Another approach that addresses semantic heterogeneity of information is by the use of ontologies [10]. An ontology is a formal description that uses explicit definitions of a particular conceptualization [16] and allows to separate the meaning of information from its syntactical representation. Ontologies provide a reasonable approach to cope with the heterogeneous character of identity information. The use of
ontologies can help to determine semantic relations between identity information, which is the basis for the consistency model introduced in Section III. Therefore, ontologies can be seen complementary to our formal model.

Paci et al. [11] investigate the problem of naming heterogeneity in identity verification processes. They address naming heterogeneities of information caused by the use of different vocabularies. To tackle these heterogeneities, the authors propose an identity attribute name matching protocol. The authors only discuss naming heterogeneity. It stays unclear how they address datatype and -format heterogeneities or more complex transformations.

C. Inadequacy of Current Consistency Models

An introduction to consistency in distributed systems can be found in the context of read and write operations on shared data in [15]. The notion of consistency is provided by consistency models, which substantially are contracts consisting of a set of rules that have to be obeyed by all processes operating on data stores to achieve consistency. Consistency models can be divided in two categories: data-centric and client-centric consistency models [15]. The data-centric consistency models are “strict” consistency models that aim at providing a system-wide consistent view to processes that simultaneously work on data. Client-centric consistency models are “weaker” and try to hide inconsistencies. Such relaxed consistency models require that changes to information are reflected in all replicas eventually [17]. For instance, a consistency model relevant for this work is monotonic-read consistency, which guarantees that a process reading a value of a data item \( x \) at time \( t \) will never see an older version of \( x \) at a later time.

Regarding the adequacy of data-centric models, performance and availability of the overall system should be considered. The key idea to implement data-centric consistency is to perform a change as a single atomic operation or transaction. Consequently, all involved systems have to be coordinated. The effort and time required to coordinate systems influences the performance and availability of the overall system [21].

A key idea behind the FIM model is that systems do have a high degree of autonomy. Thus, one system should be independent from the performance and availability of another system. Therefore, data-centric consistency models are especially inadequate in FIM systems.

Client-centric consistency models can be implemented via asynchronous mechanisms that allow to decouple concerned systems w.r.t. their availability, which in turn has no negative impact on the performance of the overall system [6]. As the access-to-update ratio of a great fraction of identity-related information is typically high and concurrent access to local copies of information are uncritical in distributed IDM systems, inconsistency can be tolerated for a specified period of time for the benefit of an improved performance and availability.

Our thesis is that client-centric consistency models do provide a basis for an appropriate consistency model for identity information in distributed systems, but adaptations can provide further improvements. In particular, the following restrictions of current client-centric consistency models do motivate further adaptations:

(i) The time duration between a change to information and the moment when the change is fully disseminated to all replicas is not limited.

(ii) Identity-information specifics such as semantic and causal relations are not considered.

(iii) Transformations of data are not considered.

D. Identity Management Related Models

The authors in [12] address the issue of attribute consistency in federated systems. They mention the dissemination models applied in the area of access control systems first stated by the authors of [4]. To implement these dissemination models they suggest to use attribute synchronization [1], or alternatively a delegation approach. For the attribute synchronization the authors of [1] suggested to use a meta-directory or virtual directory approach [5]. Attribute synchronization by meta- or virtual directory approaches might not be appropriate for distributed environments because a strong dependency on a central directory is created, e.g., considering schema changes, resulting in a decreased autonomy. The second approach they suggest is to delegate the user management to IdPs, which might not be applicable to every scenario due to the mentioned reasons such as legacy systems or the autonomy of an organization. The authors do not provide a formal model or a precise definition of information consistency.

III. FORMALIZATION OF CONSISTENCY IN DISTRIBUTED IDM SYSTEMS

In the following, we specify a formal model that highlights the structure of IdM systems and “uncovers” them from marketing philosophies or protocol and technology terminology by focusing on information consistency.

Table. I gives an overview of the symbols and functions of the formal model.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>Set of primary attributes</td>
</tr>
<tr>
<td>( X )</td>
<td>Set of transformed attributes</td>
</tr>
<tr>
<td>( M )</td>
<td>Set of transformation functions</td>
</tr>
<tr>
<td>( C )</td>
<td>Set of causality relations</td>
</tr>
<tr>
<td>( H )</td>
<td>Set of holders of identity information</td>
</tr>
<tr>
<td>( P )</td>
<td>Set of authoritative information sources</td>
</tr>
<tr>
<td>( e_i )</td>
<td>( i )th (digital) identity of a user</td>
</tr>
<tr>
<td>( a_i )</td>
<td>Datatype of an attribute ( a_j )</td>
</tr>
<tr>
<td>( v_j )</td>
<td>Value of an attribute ( a_j )</td>
</tr>
<tr>
<td>( f )</td>
<td>Federated identity of a user</td>
</tr>
<tr>
<td>( T )</td>
<td>Inconsistency window</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m_k(v_1, \ldots, v_l) )</td>
<td>( k )th transformation function</td>
</tr>
</tbody>
</table>

TABLE I
OVERVIEW OF THE SYMBOLS AND FUNCTIONS OF THE FORMAL MODEL
(i) Introducing primary and transformed attributes allows to formalize identity-related information: in particular, semantic and causal relations can be described.

(ii) The determination of transformation functions enables to compare values of primary and transformed attributes.

(iii) The specification of an inconsistency window that denotes the time period between a change to information and the moment when the change is fully disseminated facilitates to state when an attribute is consistent.

(iv) Together these aspects lead to a consistency model for identity information in distributed environments.

A. Formalization of Attributes

Depending on the context, a real-world user is represented by a digital identity, in short identity. In this model, the context does not refer to different locations, times, or roles of a user, but rather to a system context, i.e., an SP or IdP. Consequently, a user can have multiple identities, dependent on the number of SPs and IdPs the user is using services at. To describe an identity, user-related information is used that is expressed via attributes. Besides descriptive information, a digital identity needs to guarantee the uniqueness and distinctiveness of a user in a specific context. Therefore, a subset of the set of attributes composing an identity must describe an identity uniquely in a certain context. This subset is called identifier of an identity.

In the model attributes are characterized as follows:

(i) All acquired or assigned descriptive information including internal generated identifiers, any user preferences, and all characteristics appropriate to describe an identity are expressed via attributes.

(ii) An attribute is described by a name, a datatype, and a value. Within the life cycle of an attribute only the attribute value is time-variant. The datatype of an attribute \( a_i \) is denoted by \( y_i \) and the value is denoted by \( v_i \).

(iii) Each attribute is unique independent of its datatype and value. For instance, the attribute lastname of a user located at her bank institute is not the same as the attribute lastname at her employee.

(iv) An attribute can be semantically equivalent to other attributes, i.e., attributes might have different names and different types, but they do have the same meaning.

For example, many different identities of a user do have an attribute that represents the firstname of a user. Although these attributes might have differing datatypes or even values, e.g., due to differing encodings, they do have the same semantic. In addition, an attribute can be a generalization of one or more other attributes. As mentioned in Section II-B a generalization of an attribute means that the values of one or more attributes are mapped to a more general value of another attribute. An example of a generalization could be the attribute initials that is a generalization of the attribute fullname. Other possible combinations could be concatenations or semantically equivalent attributes with differing formats (see Section II-A).

When coping with information consistency the name of an attribute is irrelevant, which is why we focus on the datatypes and values of attributes. The relationship between datatype and value of an attribute \( a_i \), with \( i = 1, \ldots, n \) and \( n \) denoting the total number of user attributes at a certain point in time \( t \), can be described via a function that maps a time \( t \) in the time interval \([t_0, t_d]\), which is defined by the life cycle of an attribute where \( t_0 \) is the moment in time when \( a_i \) is created and \( t_d \) is the moment in time when \( a_i \) is deleted, to a value \( v_i \in Y_i \). Therefore, an attribute \( a_i \) is denoted by:

\[
a_i : [t_0, t_d] \to Y_i \quad t \mapsto v_i
\]

B. Semantic Relations between Attributes

An attribute can be semantically related to one or more other attributes. All attributes that are semantically equivalent to each other or where one attribute is a generalization of other attributes compose a set of semantically related attributes.

The semantic relationship has the following properties:

(i) An attribute is semantically equivalent to itself. Thus, each attribute \( a_i \) of a user, with \( i = 1, \ldots, n \) is in at least one set of semantically related attributes.

(ii) Each set of semantically related attributes includes a number of attributes that represent the current value and therefore the real-world state for all attributes within the set of semantically related attributes. These attributes are called primary attributes. The set of all primary attributes of a user is denoted by \( A \).

(iii) Attributes that are not primary attributes are called transformed attributes denoted by \( X \), where \( A \cap X = \emptyset \).

(iv) Two different primary attributes cannot be semantically equivalent to each other or one primary attribute cannot be a generalization of another primary attribute.

We use the following notation to denote that \( a_i \) is semantically equivalent to \( a_j \), with \( i, j \in \{1, \ldots, n\} \): \( a_i \Leftrightarrow a_j \). If an attribute \( a_j \) is a generalization of \( a_i \), with \( i \neq j \), i.e.,

\[
\text{the values of } a_i \text{ are mapped to a more general value of } a_j \text{ w.r.t. its informational content, we denote it by: } a_i \rightarrow a_j.
\]

Fig. 2 exemplarily depicts semantically related attribute sets. Primary attributes are emphasized in the figure. For instance, the primary attribute firstname is semantically equivalent to the transformed attribute givenname. In addition, the example shows attributes like sn that are in multiple sets of semantically related attributes. Furthermore, an attribute fullname is depicted that is semantically equivalent to a combination of two other attributes\(^2\), e.g., firstname and sn, whereas firstname and sn are not semantically related to each other.

The semantical relationship between primary and transformed attributes can be described via transformation functions that map the values of one or more primary attributes to the value of a transformed attribute. Let \( M \) be the set of transformation functions that preserve or generalize the semantic of the domain of definition. Transformation functions \( m_k \in M \) that map values \( v_1, \ldots, v_k \) of the attributes \( a_1, \ldots, a_k \in A \), with \( k \in \{1, \ldots, |A|\} \) and codomains \( y_1, \ldots, y_k \) to a value \( v_x \in y_x \) of attribute \( a_x \in X \) are denoted by:

\[\text{...}\]

\(^2\)To simplify matters, we only depicted a subset of the attributes that are semantically related to the attribute fullname.
C. Causal Relations between Attributes

Besides a semantic relationship, identity information can also be causally related, i.e., a change to one attribute might cause a change in another attribute. For example, the attributes street and street number are causally related, i.e., when a user moved and her street changed, it is also necessary to adapt street number. We denote by $a_1 \leftrightarrow a_2$ that $a_2$ is causally related to $a_1$. The set of causality relations is denoted by:

$$C := \{(a_i, a_j) \in A \times (A \cup X) : a_i \leftrightarrow a_j\}$$

Causality relations do have the following characteristics:

(i) The definition of primary and transformed attributes implies that a transformed attribute is causally related to each primary attribute it is semantically related to, as a transformed attribute can be derived via a transformation function with semantically related primary attributes as input. Thus, if a primary attribute taken as input changes, the value of the transformed attribute might also have to be changed to preserve the semantical relation.

(ii) Causality relations are directed, i.e., $a_1 \leftrightarrow a_2$ does not imply that $a_2 \rightarrow a_1$. For example, a transformed attribute is causally related to at least one primary attribute, but a primary attribute is not causally related to a transformed attribute. However, it is possible that two primary attributes are bilaterally causally related to each other, e.g., the primary attributes city and postal code are bilaterally causally related as a change to the city of a user causes a change to the postal code and vice versa.

(iii) Causality relations are independent of the value of an attribute. Hence, causality relations are time-invariant.

Due to these characteristics, the model assumes that primary attributes that are causally related to each other are not spatially distributed. This assumption reduces the time duration required to disseminate occurring changes to causally related primary attributes as they are not spatially distributed.

A key benefit of the stated causality relations is that it helps IdPs and SPs to structure distributed IdM systems: in particular, primary attributes are clearly specified. For instance, the primary attributes postal code, city, street and street number should be located at the same IdP or SP.

D. Formalization of the Federated Identity

In the following, we formalize the federated identity of a user, based on the notion of primary and transformed attributes. As mentioned before, a user might have multiple identities, whereas we assume that a user has exactly one identity per IdP or SP. Thus, an IdP or SP is able to identify each user via exactly one identity, whereas this identity only requires to be unique in the context of that IdP or SP. The set of all IdPs and SPs of a user is denoted by $H$. IdPs or SPs that manage primary attributes for a user are called authoritative information sources. The set of authoritative information sources of a user within an FIM system is denoted by $P$, where $P \subseteq H$. An identity $e_i$, where $i = 1, \ldots, h$ with $h = |H|$ is denoted as follows:

$$e_i := \{a_1, \ldots, a_m\} \cup \{a_{m+1}, \ldots, a_n\}$$

where $\{a_1, \ldots, a_m\} \subseteq A$, $\{a_{m+1}, \ldots, a_n\} \subseteq X$.

The federated identity $f$ of a user, i.e., the identity composed of all identities of a user is denoted by:

$$f := \bigcup_{i=1}^{h} e_i,$$ where $h = |H|$.

E. Definition of Identity-Information Consistency in Distributed Systems

As stated in Section II-C, consistency defined by current client-centric models does not consider the temporal aspect of the duration a change needs to be fully disseminated to all “replicas” (including transformed or causally related attributes). In other words, without relaxing the notion of consistency w.r.t. the temporal aspect, systems would never be in consistent state. However, a specified limit for the duration the dissemination process needs should be considered. Consequently, we introduce a so called inconsistency window (denoted by $T$) to quantify the duration we allow for the dissemination process to update causally or transformed attributes. Before we define the concept of ID-consistency, we like to note that in case of primary attributes $a_i, a_j \in A$ and $(a_i, a_j) \in C$ these attributes are located at the same authoritative information source by definition. Thus, we assume that in this case there is a local transaction that ensures “classical consistency”. For the case of arbitrary transformed attributes $a_i \in X$ (this includes the case of causally related transformed attributes), we define ID-consistency as follows:

At any time $t_j$ within the life cycle of $a_i$ exists a transformation function $m \in M$ and a set of primary attributes $\{a_1, \ldots, a_k\} \subseteq A$ and it exists a moment in time $t_p \in [t_j - T, t_j]$ so that the following holds: $a_i[t_j] = m(a_1[t_p], \ldots, a_k[t_p])$.

As the alteration rate of identity-related information is typically very low, it can be assumed that primary attributes at most change once within the inconsistency window. Hence, the above definition is sufficient to guarantee that any change to a primary attribute is disseminated to semantically related transformed attributes. The definition of ID-consistency does not restrict the way these changes are disseminated, e.g., via a push, pull or brokered approach.
F. Example

To clarify our understanding of consistency, we give examples of performed read and write operations on primary and transformed attributes. In distributed IdM systems an SP or IdP typically does not directly access local data stores of other providers. Thus, it can be assumed that each provider only performs write operations to its own local copy. $W_{p_i}(a_j)[t]$ denotes a write performed by provider $p_i \in P$ to a primary attribute $a_j \in A$ at time $t$. A write operation performed by provider $h_i \in H$ to a transformed attribute $a_x[t] = m(v_1, \ldots, v_k)$ at time $t$ is denoted by $W_{h_i}(a_x[a_1, a_2])[t]$. As it is not possible to perform a write operation to a transformed attribute $a_x \in X$, without knowing the input primary attributes, every write operation to a transformed attribute is preceded by read operations on the attributes that are required to derive the transformed attribute from. $R_{h_i}(a_z)[t]$ denotes a read operation on the attribute $a_z \in A \cup X$ at provider $h_i \in H$ at time $t$.

Fig. 3 shows a situation that shall clarify the time constraint. The depicted scenario shows two IdPs $p_1, p_2 \in P$ and an SP $h_1 \in H$. Fig. 3(1) shows a situation in which ID-consistency is fulfilled. The change performed by write operation $W_{p_1}(a_1)[t_0]$ and write operation $W_{p_2}(a_2)[t_1]$ is successfully read and locally written to $h_1$ by the write operation $W_{h_1}(a_3[a_1, a_2])[t_2]$. The read operation $R_{h_1}(a_3)[t_3]$ fulfills the time constraints as $\Delta t \leq T$. For the model the time $t_0$ is relevant, i.e., when the change to the primary attribute was made, not the time when the change was read. In contrast, Fig. 3(2) is not ID-consistent as $R_{h_1}(a_3)[t_2]$ was performed after $\Delta t > T$ and before $W_{p_1}(a_1)[t_1]$ was successfully disseminated. It shall be mentioned that Fig. 3(2) fulfills monotonic-read consistency (see Section II-C), as $h_1$ reads $a_1$ at time $t_0$ at the IdP $p_1$ and the read operation at time $t_2$ does not return an older version of $a_3$.

3A write operation is any operation that changes an attribute of a user. A read operation does not alter the value of an attribute.

G. General Consistency Model Contributions

The introduced consistency model is particularly suited to IdM as the alteration rate of a great fraction of identity information is typically very low and inconsistency can be tolerated for a specified period of time. By providing clearly stated constraints, the model can help in the design phase to model and structure an IdM system as well as in the operation phase to evaluate an IdM system. The general contributions of the consistency model are:

(i) Modeling attributes and according transformation functions leads to a clearly structured IdM system model, because the way information is exchanged is clarified, i.e., the information flows. In particular, authoritative information sources will be identified. In this respect the model complements other models like UML.

(ii) By specifying an inconsistency window, consistency can be quantified, e.g., for service level agreements.

(iii) Apart from the time, the consistency in the formal model depends on transformation functions. Consistency can be achieved if, and only if, these transformation functions are correctly modeled and implemented, respectively. Incorrect transformation functions can result in logically contradicting identity information.

H. Bounds of the Inconsistency Window

Specifying the inconsistency window is a nontrivial task. The lower bound of the inconsistency window, denoted by $T_L$, can be determined theoretically based on technical factors such as the communication load on the system and existing dissemination models. We will exemplarily show in Section IV-C how the lower bound of the inconsistency window could be determined empirically based on a performance evaluation.

By $T_L(a_i)$ we denote the time duration it takes due to technical constraints to disseminate an occurring change to the attribute $a_i \in X$. Thus, $T \geq \max(T_L(a_1), \ldots, T_L(a_k))$, where $k = |X|$. In addition, $T$ is typically restricted by the length of time an update operation is tolerated in maximum, e.g., due to a certain risk level of an attribute or other security metrics. By $T_U(a_i)$ we denote this non-functional constraint for the attribute $a_i \in X$. Thus, $T \leq \min(T_U(a_1), \ldots, T_U(a_k))$, where $k = |X|$. In summary,

$max(T_L(a_1), \ldots, T_L(a_k)) \leq T \leq \min(T_U(a_1), \ldots, T_U(a_k))$

IV. EXAMPLE: USING THE CONCEPT OF ID-CONSISTENCY WITHIN CARDSPACE

In this section we present a case study in which we make use of the concept of ID-consistency to analyze and improve a real-world IdM system. CardSpace has been chosen as a promising concept for a user-centric FIM system. Applying ID-consistency to CardSpace allows to quantitatively evaluate consistency, in particular, the temporal constraint. We like to note that we do not evaluate CardSpace in general (including all features), but focus only on the consistency issue. Therefore, the following steps are performed:
We show how the building blocks of the model, e.g., primary attributes and according transformation functions, can be identified within an IdM system using CardSpace for demonstration purposes. Based on the building blocks it can be checked if ID-consistency is achievable. If shortcomings are detected, the model indicates missing pieces as it points to constraints that are not fulfilled. In case of CardSpace, the model indicates that the inconsistency window cannot be specified without extensions.

(ii) To limit the duration required to disseminate occurring changes within CardSpace, we exemplarily outline an approach that enables SPs and IdPs to request current information via an additional service.

(iii) We exemplarily present how the inconsistency window could be specified based on such a service using an empirical method.

A. Introduction to CardSpace

Windows CardSpace [19] is Microsoft’s implementation of the user client part of an information card based system. CardSpace enables the exchange of identity information between an IdP and relying SPs. Hereby, a user can distribute her identity information over multiple IdPs, whereas IdPs do not synchronize each other. CardSpace represents a reference to an identity on an IdP as a so-called information card. From each of her IdPs, the user obtains and locally stores an information card. When the user connects to an SP supporting CardSpace, the SP provides a policy wherein, for instance, states required and optional user attributes. A client-side component named identity selector evaluates this policy and lists all matching information cards. The user states which information she is willing to give away by choosing a card and by authenticating to the IdP. If the authentication was successful, the IdP returns the attributes in demand to the identity selector on the user client, which finally sends the attributes to the SP. This data exchange is restricted to the period of time when the user is logged in, i.e., when there is an active session between SP and user. Consequently, an SP cannot get the information again after the user signed off in current CardSpace-based systems.

B. Step 1: Identification of the Building Blocks of the Model

Building blocks are identified by the following process:

(i) We analyze CardSpace w.r.t. the way information is replicated and exchanged to identify the sets of primary and transformed attributes.

(ii) We identify and formalize transformation functions to be able to check if the values of primary attributes are correctly mapped to the values of transformed attributes.

(iii) We analyze CardSpace considering the exchange of information to check if a limitation of the inconsistency window is possible.

We first analyze the way information is replicated in CardSpace. Fig. 4 shows an example of a CardSpace system. To simplify matters, we depicted only one SP, two IdPs, and the identity selector as representative of the client-side component. In CardSpace user attributes are replicated across IdPs and SPs. The main reason why information is replicated over multiple IdPs is because both users and service providers may not trust in a central IdP. Furthermore, technical reasons such as a potential single point of failure and bottleneck also motivate to use multiple IdPs [2]. In Fig. 4 information replication applies, e.g., to the tuple (dob, date, 2008-12-15) and (date-of-birth, string, 08-12-15). An SP might store information redundantly because it may comprise legacy components that rely on user attributes and therefore require to store information locally. Thus, both IdPs and SPs are within the set of providers H.

The introduced model requires that changes to primary attributes are disseminated to transformed attributes that are derived from these attributes. Therefore, the way information is exchanged within an IdM system has to be considered when determining primary attributes. In CardSpace the exchange of identity-related information is unidirectional from an IdP to relying SPs. Hence, the set of primary attributes A must be restricted to attributes composing identities at the IdPs, whereas attributes located at SPs must be transformed attributes of a user. Otherwise, occurring changes to primary attributes could not be disseminated to transformed attributes at all. In Fig. 4 we exemplarily marked primary attributes with an asterisk.

As stated in the formal model, two attributes are semantically related if there is a transformation function \( m \in M \) that maps the values of one or more attributes to the values of another attribute while preserving the semantics of the input attributes. Thus, transformations allow to check if the values of primary attributes are correctly mapped to the values of transformed attributes. CardSpace itself provides a so-called claims transformation language (CTL) that allows an IdP to transform its information schema to SP-specific schemas. For example, a transformation rule that maps the attribute (phone, string, 555-123) to the attribute (phone, string, 555123) (see Fig. 4) could be implemented as follows:

\[
\begin{align*}
\text{c:Type} &= \text{“phone_nb”} \\
\text{Value} &= \text{regexreplace(c.Value, “[0-9]”, “”)}
\end{align*}
\]

This transformation function could be adapted to the formal model as follows: let \( v_1 \) denote the value of \( \text{phone_nb} \) and \( v_2 \) the value of \( \text{phone} \) and define

\[
m_1 : [0-9]^* \rightarrow [0-9]^* \\
v_1 \rightarrow v_2, \text{ where } v_1 \text{ is mapped to } v_2 \text{ by removing all occurrences of “”}.
\]
Thus, it can be concluded that the attribute `phone_nb` and the attribute `phone` are semantically related, as there is a transformation function \( m_1 \in M \) that maps these attributes to each other while keeping the semantics. In Fig. 4 the exemplary value for `phone_nb` is 555-123 and \( m_1(555-123) = 555123 \) which is the current value of `phone`. Therefore, `phone` is consistent at that time.

In the next step, we determine if CardSpace is able to specify a lower bound for the inconsistency window. As IdPs do not synchronize each other, inconsistencies between different IdPs cannot be solved automatically at all, i.e., the user needs to change all IdP accounts manually. Inconsistencies between an IdP and SPs are only solved when the user logs in to an IdP and then uses a service provided by the SP, as the exchange of identity-related information is restricted to the period of time when the user is logged in. Furthermore, CardSpace does not force a user to login regularly or at certain points in time. Let us assume that \( a_1, a_2 \) are semantically related attributes within a CardSpace system, whereas \( a_1 \in A, a_2 \in X \) and \( m_1 \in M \) is the transformation function that maps the values of \( a_1 \) to \( a_2 \). Furthermore, we assume that \( a_1 \) is changed at time \( t_0 \). As a consequence of the introduced definition of consistency, it has to be guaranteed that \( \exists t_0 \in [t_0, t_0 + T] : a_2(t_0) = m_1(a_1(t_0)) \), with \( m_1 \in M \). From the temporal coupling of the dissemination of identity-related information to the login process, it follows that the user needs to login within \( t \in [t_0, t_0 + T] \), otherwise the change is not disseminated within \( t \in [t_0, t_0 + T] \). Stating a clear limitation for the inconsistency window, might be a desired feature in scenarios like long-lived service delivery [6].

Consequently, the model showed that CardSpace cannot limit the inconsistency window. In the next section we show how CardSpace could be adapted to cope with ID-consistency.

C. Step 2: Adaptions to Support ID-Consistency in CardSpace

To cope with ID-consistency in CardSpace, we introduce an additional component, called identity delegatee. The identity delegatee is a service provided by a trusted third party that allows SPs and IdPs to retrieve information without direct user interaction. Therefore, the user is able to delegate the task of controlling disclosure of information in her absence to the identity delegatee, thereby instructing the delegatee to control each dissemination of her attributes by defining dissemination policies per each IdP or SP. The identity delegatee itself does not retain any user attributes. Furthermore, the transformation rules necessary to map the various information schemas have to be established in advance by the operators of the identity delegatee and the providers, as users cannot be expected to map the various schemas. The implementation is based on Windows CardSpace 2.0 and Active Directory Federation Services 2.0 as part of the Windows Identity Foundation (WIF) [20]. As in the prototype both the SPs and IdPs are configured to trust the identity delegatee, the identity selector does not have to be adapted. The delegatee is an ASP.NET web application with an additional WS-Trust Security Token Service (STS) and information card issuance functionality.

The initial process required to use the identity delegatee is as follows: first, the user registers at the identity delegatee by creating an account. During the registration, the user states the primary attributes per each set of semantically related attributes, and at the same time the user federates her identities\(^4\) on the IdPs with the identity delegatee by authenticating against the IdPs. If the user was successfully authenticated, the respective IdP issues a so-called delegation token, which can be used by the identity delegatee to retrieve user attributes on behalf of the user at a later time. This delegation concept considers privacy as the identity delegatee does not impersonate the user and only has a subset of user rights at the user’s IdPs. Hence, the delegatee is not in possession of any user credentials to authenticate to the IdPs, but authenticates via its own identity and credentials. During the identity federation the user chooses an IdP by selecting an information card via the identity selector provided by CardSpace. Finally, the identity delegatee confirms the registration by sending the user an information card.

To enable information retrieval at any time, we decouple the user interaction from the information retrieval of an SP or IdP by enabling an alternative flow of user attributes, i.e., attributes are no longer required to pass the user client, but may pass the delegatee instead. In this exemplary approach to cope with information consistency in CardSpace, we implement a pull approach, i.e., SPs and IdPs are able to request current user attributes at the delegatee. This implies that each provider that wants to retrieve data at the delegatee has to be enhanced. However, necessary adaptations are only marginal which is the major reasons why we chose a pull approach. The disadvantage of this approach is that attributes have to be retrieved periodically, even if no changes occurred.

Fig. 5 depicts the simplified process to retrieve current information via the identity delegatee. When an SP sends a request to the identity delegatee, the identity delegatee checks if the SP is authorized based on the policies configured by the user. Next, primary attributes and associated authoritative information sources are determined. Then, the delegatee uses the delegation tokens to request the primary attributes at the IdPs. In Fig. 5 we exemplarily depicted two IdPs that are

\(^4\)A concept to link different identities of a user can be found in [3].
requested for current information. The IdPs communicate the requested information via so-called security tokens that are defined as a set of claims, which can be compared to attributes. After the IdPs issued the security tokens, the identity delegatee embeds the received tokens within a new token, which is encrypted and signed for the SP. As the security tokens of the IdPs are encrypted for the SP and signed by the IdPs, the identity delegatee is not able to read or alter the information without detection, thus, it cannot compromise information integrity and privacy is preserved.

The consistency model states that primary attributes reflect the current value of user attributes at any time. Thus, if all information is directly retrieved from authoritative information sources, the lower bound of the inconsistency window is easy to specify. If an IdP or SP could retrieve information by an arbitrary IdP or SP, it would be difficult to state a clear limit of $T_L$. Fig. 6 shows a problematic scenario, in which information retrieval is not restricted in the way an IdP keeps its own information up to date. In the example, pairs of IdPs retrieve attributes bilaterally forming a chain. If the time necessary to achieve consistency between $I_{P_2}$ and $I_{P_{n-1}}$ can be bounded by $\Delta t_2$, $I_{P_{n-1}}$ and $I_{P_{n-2}}$ by $\Delta t_3$ and so on, $T_L$ can be determined by $T_L = \sum_{i=1}^{n} \Delta t_i$ in worst case to establish consistency in the whole chain. Hence, $T_L$ depends on the number of IdPs.

In summary, based on the assumptions that SPs and IdPs retrieve user attributes periodically at the delegatee and network connection failures do not occur, an upper limit for the inconsistency window can be specified. In the next section, we exemplarily demonstrate how to determine a potential size of the inconsistency window based on a performance evaluation conducted on a prototypical implementation.

**D. Step 3: Determination of the Inconsistency Window**

In a pull approach in which current information is directly fetched from the authoritative sources, the inconsistency window is defined by the data-retrieval interval; in worst case the information are as old as the request interval. Thus, the inconsistency window is mainly influenced by the number of requests the identity delegatee and the authoritative sources are able to answer and consequently by the performance and the scalability of the overall system. Therefore, we focus on the performance tests made in a testbed to gain an indication of a feasible inconsistency window. In other approaches, e.g., in IdM systems with specific dissemination models, i.e., specified time intervals when information is synchronized between systems, $T_L$ is directly influenced by the dissemination model. Therefore, the determination of the inconsistency window would require to analyze existing dissemination models.

For the evaluation we deployed three IdPs, an SP, and an identity delegatee 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 IdPs are Active Directory Federation Servers 2.0 RC each 2 GB RAM, whereas the SP and the identity delegatee are hosted in a virtual machine with 4 GB RAM. As all components are hosted in the same server, network latency is negligible.

For determining $T_L$ the test client measured the response time for a request by an SP or IdP according to the process described in Fig. 5. To achieve that requesting providers are provided with current information, the identity delegatee is not allowed to cache information. Therefore, the performance of an IdP is essential as each request at the delegatee requires that at least one IdP is requested to retrieve the current value of an attribute. In the exemplary test scenario the delegatee fetches ten attributes from three IdPs per request. The first relevant load test that is described briefly has been performed to determine the average response time at the IdPs. Therefore, the load has been varied from 20 requests per second to 60 requests per second (cf. Fig. 7). The depicted tests have been repeated ten times, each for ten minutes with a constant request rate. For example, the average response time of an IdP at a constant load of 60 requests per second is about 85 ms.

The next relevant performance test is to determine the response time of the identity delegatee itself. Fig. 7 shows the average response time of the identity delegatee in the overall process. We assumed that per attribute request of an SP three IdPs have to be requested to retrieve the current value of the primary attributes. At a constant rate of 60 requests per second (see Fig. 8) the delegatee has an average response time of 470 ms. Thus, the delegatee needs in average less than one second to authorize the SP, to determine primary attributes and associated IdPs, to detect correlated identifiers at the IdPs, to retrieve primary attributes at the IdPs, to aggregate the tokens, to encrypt, to sign, and to send the new token to the SP. The lost rate of all depicted load tests has been null, whereas network connection failures have not been considered. At higher rates we experienced an increasing lost rate and
therefore chose the depicted request rates. When using load tests to determine $T_L$, a low lost rate should be achieved, as it has to be accomplished that a request performed by an SP gets answered even if the identity delegatee is under load. In our tests the delegatee needs at a constant rate of 60 requests per second less than 1 s to answer a request. Thus, based on the empirical data $T_L$ could be about 1 s. As the presented approach requires SPs and IdPs to request current information per user in a periodical manner, the specification of $T$ should consider the resulting load. For instance, if $T$ would be one hour the delegatee has to answer a request per user identity per hour to accomplish that changes are disseminated within the inconsistency window. At a constant rate of 60 request per second, the delegatee could keep about 216,000 user identities consistent with $T = 1$ h. Hence, distributed user attributes are in worst case one hour old. Although, the concrete size of $T$ strongly depends on the specific scenario, an inconsistency window of one hour is mostly acceptable, especially, in the favor of performance and availability.

V. CONCLUSIONS

In this paper, we proposed a consistency model named ID-consistency to appropriately define consistency of identity-related information in distributed and ubiquitous IT systems. The model considers identity-information specifics as well as temporal constraints. The loosened notion of consistency provided by ID-consistency is legitimated in many scenarios in distributed environments in the favor of performance and availability. By focusing on the aspect of information consistency, the model provides the “bare bones” structure of IDM systems and “uncovers” them from marketing philosophies or protocol and technology terminology.

Modeling attributes and according transformation functions leads to a clearly structured IdM system model, because the way information is exchanged is clarified. The model provides a structured approach to cope with major challenges typically any IdM system faces in distributed environments, e.g., heterogeneous schemas and multiple identities per user. Therefore, it makes challenges explicit that have to be tackled anyhow, thus, the overhead to apply the model should be acceptable. The effort might even decrease when modeling several similar systems.

We reported on a case study (CardSpace) where we made use of the concept of ID-consistency to analyze and improve a real-world IdM system. Based on the modeling, we showed that ID-consistency is not achieved by default as the inconsistency window cannot be specified. We presented an approach that exemplarily shows how the inconsistency window could be limited. Finally, we empirically determined the duration of the inconsistency window based on a performance evaluation.

Clearly, the proposed formal approach to dissemination of identity-related information can be extended in various directions. For example, the model could be linked to performance models by specifying distributions for dissemination delays or Markov chains for reliability issues. In addition, the consistency model could be combined with identity assurance approaches [8]. Identity assurance is defined as the degree of confidence an IdP or SP can ascribe to the correctness of attributes managed and asserted by another IdP or SP, which would be helpful for the specification of primary attributes.

REFERENCES