Mathematical Modelling of Identity, Identity Management and Other Related Topics

Md. Sadek Ferdous, Gethin Norman, and Ron Poet
School of Computing Science, University of Glasgow
Glasgow, G12 8QQ, Scotland
m.ferdous.1@research.gla.ac.uk, gethin.norman@glasgow.ac.uk,
ron.poet@glasgow.ac.uk

ABSTRACT

There exist disparate sets of definitions with different semantics on different topics of Identity Management which often lead to misunderstanding. A few efforts can be found compiling several related vocabularies into a single place to build up a set of definitions based on a common semantic. However, these efforts are not comprehensive and are only textual in nature. In essence, a mathematical model of identity and identity management covering all its aspects is still missing. In this paper we build up a mathematical model of different core topics covering a wide range of vocabularies related to Identity Management. At first we build up a mathematical model of Digital Identity. Then we use the model to analyse different aspects of Identity Management. Finally, we discuss three applications to illustrate the applicability of our approach. Being based on mathematical foundations, the approach can be used to build up a solid understanding on different topics of Identity Management.

Keywords

Identity, Identity Management, Mathematical Modelling.

1 Introduction

With the tremendous expansion of the Internet during the last twenty years or so, more and more identities and credentials have been issued, making their management challenging, both for service providers and users. Identity Management (denoted IdM thereafter) was introduced initially by industry to facilitate online management of user identities. There are currently a number of different incompatible IdM solutions that resulted from separate isolated projects. The lack of cohesion and co-operation among these efforts lead to different notions and semantics for many central topics of IdM, and some fundamental concepts of IdM such as Identifiers, (Digital) Identity and Partial Identity have different definitions in [10, 19, 3, 17, 22]. For example, Identity is defined as "the equivalent to the physical presence of that entity" in [10], whereas [3] considers Identity as "the set of permanent or long-lived temporal attributes associated with an entity". The same concept of Identity is defined as the "the dynamic collection of all of the entity's attributes" in [19]. The first definition is a mere abstraction, whereas the second and third have significantly different means. Such differences in these core topics introduce inconsistencies and such inconsistencies in turn introduce confusion, making it difficult to gain an understanding of IdM.

To rectify the situation, an effort has been undertaken to develop an extensive list of vocabularies [22]. The result of this work is an evolving document that aims to build up a common language to remove inconsistencies. Their focus is somewhat narrow as it only deals with those definitions that are related to privacy aspects of IdM. The domain of IdM is much broader, and thus [22] is not comprehensive in nature. A more general effort with the similar goal is [19], this work provides definitions for different concepts related to IdM, however the relationships between the definitions is not considered. In addition, a textual form for the definitions is used which can lead to misinterpretation. A mathematical model of IdM can be found in [29]. Unfortunately, similarly to [22], this work focuses only on the privacy aspects of IdM. Therefore, as noted in [1], a mathematical model of identity and IdM covering all its aspects is still missing. In this paper, we aim to fill in that gap by building up a mathematical model of the central topics of Digital Identity and IdM, and show how our model can be used to analyse different issues related to IdM. Moreover, the model provides the foundations for formal analysis of IdM and for an *Identity Calculus* which can then be used to study the dynamics of identities in different systems. The main contributions of this paper are listed below.

- A formal model of digital identity by mathematically defining atomic concepts for Entity, Context, Attribute, Identifier, Partial Identifier, Partial Identity and Data.
- We show how the formal model can be used to characterise different aspects of IdM.
- We give three applications to illustrate how our model can be used to analyse different scenarios and to characterise two popular IdM models mathematically.

The paper is structured as follows. In Section 2, we build up our model of digital identity. In Section 3 we use our devised model of digital identity to characterise the behaviour of an IdM System and Section 4 provides three applications to illustrate the applicability of our proposed model. We conclude in Section 5 with a discussion of future work.

SIN '14, September 09 - 11 2014, Glasgow, Scotland Uk

http://dx.doi.org/10.1145/2659651.2659729.

2 Modelling Digital Identity

In this section we develop our Digital Identity model from atomic concepts.

Entity. An entity is a physical or logical object which has a separate distinctive existence either in a physical or logical sense [7]. Different disciplines interpret the term Entity in different ways and with different meaning. We focus on digital identities, and therefore assume for the remaining of this paper that entities are digital and use the symbol E to denote the set of digital entities.

Context. Like entity, the term context is used with different meanings in different disciplines. In the scope of IdM, a context is the environment under which a (digital) entity exists and operates. It can be regarded as the application domain or namespace in which an entity is represented and identified uniquely. We let $CONTEXT$ denote the set of contexts and use E_c to denote the set of entities in context $c \in CONTEXT$.

As a running example, consider two contexts (or systems): one for blogging (BLOG) and one for emailing (EMAIL). Assume that each system has two entities (or users): *JOHN* and *RAHIM* in the BLOG, and *RAHIM* and *ALICE* in the EMAIL, then we have: $E_{\text{BLOG}} = \{JOHN, RAHIM\}$ and $E_{\text{EMAIL}} = \{ALICE, RAHIM\}$.

Attributes. An attribute is a distinct, measurable *named* property belonging to an entity in a context whose *value* can be used to identify the entity (not necessarily uniquely) within the context [19]. Here, the term "identify" is used with its literal meaning. We will define this term more precisely later. Accordingly, each attribute has a name and value. The name (or simply the attribute) alone cannot identify an entity, the value is also needed. In a context, the value of an attribute is provided either by each entity or a third party. Another important aspect of an attribute is its data type which defines the values an attribute can take. For brevity, we will not consider the data type of attributes and instead focus on attributes (names) and values. In common terminologies, values are also known as *data*. Let A_c denote the set of attributes and AV_c attribute values in context $c \in CONTEXT$. The attributes in different contexts may overlap as attributes can exist in different contexts.

Returning to our example, assume the context *BLOG* has three different attributes: username, age and postcode of residence. Furthermore, for entity *JOHN* the values of these attributes are *john*, *32* and *G3* respectively, while for *RAHIM* the values are *rahim*, *21* and *G3* respectively, then:

$$\begin{aligned} A_{BLOG} &= \{username, age, postcode\} \\ AV_{BLOG} &= \{john, rahim, 21, 32, G3\} \end{aligned}$$

Now we relate attributes and entities in a context.

Definition 1. *Let $atEntToVal_c : A_c \times E_c \to AV_c$ be the (partial) function that for an entity and attribute returns the corresponding value of the attribute in context c.*

The function is partial as not all entities have a value for each attribute. We use this definition to introduce the function $atToValSet_c : A_c \to \mathcal{P}(AV_c)$ that, for a given attribute, returns the set of values associated with at least one entity. Formally, for $a \in A_c$, we have $atToValSet_c(a)$ equals

$$\{atEntToVal_c(a, e) \mid e \in E_c \text{ and } atEntToVal_c(a, e) \text{ is defined}\}$$

According to our example:

$$\begin{aligned} atEntToVal_{BLOG}((username, RAHIM)) &= rahim \\ atEntToVal_{BLOG}((postcode, JOHN)) &= G3 \\ atToValSet_{BLOG}(username) &= \{john, rahim\} \end{aligned}$$

The following functions map attributes and values to the corresponding entities.

Definition 2. *Let $atValToEnt_c : A_c \times AV_c \to \mathcal{P}(E_c)$ be the function that maps an attribute-value pair to the set of entities which have the value for the attribute in context c. Formally, for $a \in A_c$ and $v \in AV_c$:*

$$atValToEnt_c(a, v) = \{e \in E_c \mid atEntToVal_c(a, e) = v\}.$$

Returning to our example:

$$atValToEnt_{BLOG}((postcode, G3)) = \{JOHN, RAHIM\}$$

We next introduce three classes of attributes. The first uniquely identify an entity, the second can identify at least one entity and the third no entities.

Identifier. An identifier is an attribute whose value can be used to uniquely identify an entity within a context [13]. There may be many attributes in a context that can uniquely identify an entity at a certain point in time. However, when more entities are added into the context, it may happen that the attribute no longer uniquely identifies an entity. To avoid unnecessary complications, each context considers one attribute as the identifier and ensures that its value can always uniquely identify an entity.

Definition 3. *The function ident : $CONTEXT \to A_c$ returns the* identifier *for a context such that if $i = ident(c)$, then*

- *$atEntToVal_c(i, e)$ is defined for all $e \in E_c$;*
- *$atEntToVal_c(i, e_1) \neq atEntToVal_c(i, e_2)$ for all distinct $e_1, e_2 \in E_c$.*

for all contexts $c \in CONTEXT$.

The above conditions correspond to the following (required) properties of an identifier in a context: each entity of the context must have a value for the identifier and the value of the identifier uniquely identifies an entity in the context. These conditions can be ensured during the registration process (an essential step of IdM which we will explore later).

Partial Identifier. When the value of an attribute identifies at least one entity within a specific context, the attribute is defined as a Partial Identifier [13]. An example of a partial identifier is *Surname*. Many people may share a surname, and therefore it can identify more than one person.

Definition 4. *The set of* partial identifiers *in context c is given by $PI_c \subseteq A_c \backslash \{identifier(c)\}$ such that $pi \in PI_c$ if and only if $atEntToVal(pi, e)$ is defined for some $e \in E_c$.*

Null Identifier. A Null Identifier are those attributes which do not have a value for any entities. Formally we have the following definition.

Definition 5. *The set of* null identifiers *in context c is given by $NI_c \subseteq A_c \backslash \{ident(c)\}$ such that $ni \in NI_c$ if and only if $atEntToVal_c(ni, e)$ is undefined for all $e \in E_c$.*

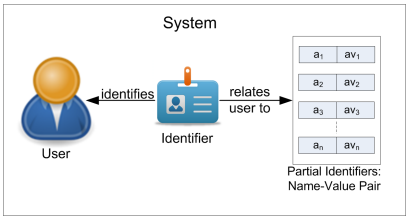

Figure 1: Relation between user, identifier and partial identifiers.

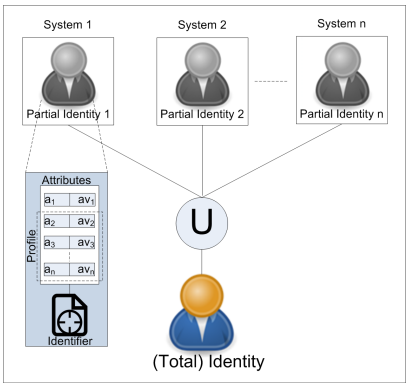

Figure 2: The (total) identity of an entity.

From the above definitions, it follows that $A_c = \{ident(c)\} \cup PI_c \cup NI_c$ and these sets are disjoint.

Returning to our example, let *username* be the attribute that has been selected as the Identifier for the *BLOG* system. That is $ident(BLOG)$=*username*. All other attributes are partial identifiers, i.e. $PI_{BLOG} = \{age, postcode\}$. There are no null identifiers in our example.

Next, we relate identifiers, partial identifiers and entities.

Definition 6. *Let $idValToEnt_c : AV_c \to E_c$ be the (injective) function that maps a value of an identifier to the respective (unique) entity in context c. That is, for any $v \in atToValSet_c(ident(c))$ we have $idValToEnt_c(v)$=e where e is the unique entity with $atEntToVal_c(ident(c), e)$=$v$.*

Returning to our example, we have:

$$idValToEnt_{BLOG}(john) = JOHN$$
$$idValToEnt_{BLOG}(rahim) = RAHIM$$

We next relate identifiers to partial identifiers.

Definition 7. *Let $idValToPi_c : AV_c \to \mathcal{P}(PI_c \times AV_c)$ map a value of the identifier in context c to the corresponding set of partial identifiers and values. Formally, for $v \in atToValSet_c(ident(c))$, if $idValToEnt_c(v)$=e, then*

$$idValToPi_c(v)=\{(pi, v) \mid pi \in PI_c \text{ and } atEntToVal_c(pi, e)=v\}.$$

According to our running example:

$$idValToPi_{BLOG}(john)=\{(age, 31), (postcode, G3)\}$$

Figure 1 provides a graphical analogy of the relation between the user (entity), the identifier and the partial identifiers.

Credentials. We will consider a credential as an attribute that accompanies an identifier to attest the authority of an entity over the supplied identifier (known as the authentication process). The simplest and weakest form of a credential is a password. Attribute certificates, biometrics such as fingerprints, voice recognition, retina scan or secure hardware tokens such as the OTP (One Time Password) are examples of more secure credentials. Let, $cred_c$ be the attribute that holds the credential for each entity in context c. Then we can define the following function that checks the identifiers and credentials of entities.

Definition 8. *For a context c, the function*

$$checkCred_c : (ident(c) \times AV_c) \times (cred_c \times AV_c) \to \mathbb{B}$$

returns `true` *if the supplied identifier and credential values match and* `false` *otherwise.*

Modelling credentials at this abstract, high level means that we do not need to be specific regarding the type of credentials. For example, the credential attribute (*cred*) could be represented using passwords in one system and biometrics in another system.

Partial Identity, Identity and Profile. The partial identity of an entity in a context is the set of identifier and partial identifiers along with their values within that context [22]. Formally, we have the following definition.

Definition 9. *For a context c, the* partial identity *of an entity $e \in E_c$ within context c, denoted $parIdent_c^e$, is given by the set:*

$$\{(a, v) \mid a \in A_c, \ atEntToVal_c(a, e) \text{ is defined and equals } v\}.$$

For our running example, for example $parIdent_{BLOG}^{JOHN} = \{(username, john), (age, 31), (postcode, G3)\}$.

The (total) identity of an entity is the union of all its partial identities in all contexts (Figure 2) [22].

Definition 10. *For an entity $e \in E$, the* identity *of e is given by the set:*

$$ident^e = \cup\{(c, parIdent_c^e) \mid c \in CONTEXT \text{ such that } e \in E_c\}.$$

Note that such a combined view of identity only makes sense from the first person perspective of a user. From an organisational point of view, such a combined view would allow any organisation to gain unlimited control over the user data and hence is very lucrative, but potentially privacy invasive as far as the user is concerned. A more privacy-friendly approach is to share a limited view of a user's data across organisational boundaries whenever needed. We define such a limited view below (see Definition 11). Sometimes, it may happen that two different contexts are combined (maybe due to the merging of organisations) into a single context, it may seem that two entities may end up with the same set of attributes and therefore the same identity. However, it is important to remember that, by definition of an identifier in a context, two entities cannot have the same identifier value. This ensures that the partial identities of two entities in a single context are different as they will have at least one different attribute value. Therefore during such merging of contexts, the system may need to update their sets of identifiers to ensure consistency.

Definition 11. *For a context c, the profile of an entity $e \in E_c$ is given by $PROFILE_c^e = \{(ident(c), v)\} \cup \mathcal{A}$ where v=$atEntToVal_c(i, e)$ and $\mathcal{A} \subseteq idValToPi_c(v)$.*

An example of the profile of the user *JOHN* in *BLOG* using our example is:

$$PROFILE^{JOHN}_{BLOG} = \{(username, john), (age, 31)\}$$

If a profile is passed between different contexts (which is often the case in traditional IdM Systems) and the Identifier is permanent in nature it raises the question of privacy invasion since the user can be tracked or profiled in another context with this permanent identifier. In such cases, an Anonym (a non-identifiable attribute, that cannot be associated with an entity) or a Pseudonym (an arbitrary one-time identifier of an entity [19]) can be used instead of a permanent identifier. Here in *PROFILE*, the Identifier is treated at an abstract high-level (without specifying if it is a permanent, anonymous or pseudonymous identifier) to allow the modelling of any implementation of an Identifier.

Digital Identity Model. Having formalised the required sets and functions to represent a digital identity, we can combine them into a unified model.

DEFINITION 12. *A* Digital Identity Model (DIM) *consists of the following basic components:*

- E *is the set of entities;*
- $CONTEXT$ *is the set of contexts;*
- A *is the set of attributes and* AV *attribute values;*
- $ident : CONTEXT \to A$ *returns the identifier for each context;*
- $E_c \subseteq E$, $A_c \subseteq A$ *and* $AV_c \subseteq AV$ *are the sets of entities, attributes and attribute values in context* $c \in CONTEXT$*;*
- $cred_c \in A_c$ *holds the credential for each entity in the context* $c \in CONTEXT$*;*
- $atEntToVal_c : A_c \times E_c \to AV_c$ *returns the value of an attribute of an entity in context* $c \in CONTEXT$*;*
- $checkCred_c : (ident(c) \times AV_c) \times (cred_c \times AV_c) \to \mathbb{B}$ *checks if an identifier and credential with values match in context* $c \in CONTEXT$*.*

Using these basic components the following additional components can be defined:

- $PI_c, NI_c \subseteq A_c$ *are the sets of partial and null identifiers respectively in context* $c \in CONTEXT$*;*
- $parIdent^e_c \subseteq A_c \times AV_c$ *is the partial identity of the entity* $e \in E$ *in context* $c \in CONTEXT$*;*
- $PROFILE^e_c \subseteq A_c \times AV_c$ *is the profile of the entity* $e \in E$ *in context* $c \in CONTEXT$*;*
- $ident^e \subseteq \{(c, \mathcal{P}(A_c \times AV_c)) \mid c \in CONTEXT\}$ *is the total identity of the entity* $e \in E$ *in all contexts;*
- $atValToEnt_c : A_c \times AV_c \to \mathcal{P}(E_c)$ *maps attribute-value pairs to the set of Entities which have the value for the attribute in context* $c \in CONTEXT$*;*
- $idValToEnt_c : AV_c \to E_c$ *maps a value of the identifier to the respective (unique) entity in the context* $c \in CONTEXT$*;*
- $idValToPi_c : AV_c \to \mathcal{P}(PI_c \times AV_c)$ *maps a value of the identifier to the corresponding partial identifiers and values in context* $c \in CONTEXT$*.*

Other functions that have been discussed can be constructed using the basic components of Definition 12.

3 Modelling Identity Management

The ever increasing number of both online services and users that access each service leads to an even faster increase in digital identifiers. Therefore the issues of managing these identifiers and corresponding credentials is becoming a harder task. IdM was proposed to facilitate management of online identities [7]. Formally, IdM consists of technologies and policies for representing and recognising entities with their digital identities [14]. A system that is used for IdM is called an IdM System (IMS, in short). Each IMS involves the following parties:

Client/User. A client/user receives services from a service provider (see below). Any entity can be a client, however we assume that each client is a user or person.

Service Provider. A service provider (SP, in short) is an organisation that provides services to the clients or to other SP. It is also known as the Relying Party. We will use the notation *SP* to denote the set of service providers.

Identity Provider. An identity provider (IdP, in short) is an organisation that provides digital identities to allow clients to receive services from a SP. We will use the notation *IDP* for the set of identity providers.

3.1 Steps in Identity Management Systems

Each IMS, generally, utilises the following set of steps, known as the life-cycle of an IMS, to allow any user to manage her identity in that specific context and to access services from the SP. The contexts of an IMS consist of the sets of IdPs and SPs, i.e. $CONTEXT = IDP \cup SP$. The subscript in the following operations specifies the context (IdP or SP) in which each operation is valid.

Registration. Registration is the initial step in which a user, who wants to access services, registers herself at the respective IdP by providing personal information. At this step, the user either chooses a unique value for the identifier of the service and a value for the corresponding credential or they are created automatically by the IdP. The user may also provide values (data) for partial identifiers and then the IdP updates its respective attributes with the user supplied data. The process can be modelled in the following way:

$$\begin{aligned} E'_{idp} &= \{e'\} \cup E_{idp} \\ AV'_{idp} &= \mathcal{AV}' \cup AV_{idp} \end{aligned}$$

Here, e' represents the newly joined entity and $\mathcal{AV}'$ represents the newly created data for that entity including the compulsory unique identifier value. E'_{idp} and AV'_{idp} represent the updated sets of entities and attribute values respectively for the context idp.

After registration, the components of Definition 12 for context idp whose domain or range contain E_{idp} or AV_{idp} are updated to E'_{idp} and AV'_{idp} respectively. For example:

- if the entity e' supplied value $av' \in \mathcal{AV}'$ for attribute $a \in A_{idp}$, then $atEntToVal_{idp}(a, e') = av'$;
- if the entity e' supplied values $av'_1, av'_2 \in \mathcal{AV}'$ for $ident_{idp}$ and $cred_{idp}$ respectively, then:

  $checkCred_{idp}((ident(idp), av_1), (cred_{idp}, av_2)) = \texttt{true}$

  if $av_1 = av'_1$ and $av_2 = av'_2$ and `false` otherwise.

Many systems allow a user to create more than one account using the same procedures described above. The user must choose a unique value for the identifier and can add one or more attribute values. It is important to understand that the system will treat each registered account as belonging to a separate digital user even if they are created by the same physical user. This particular scenario is easy to capture in our framework. Using our running example, suppose user *JOHN* wants to register himself in the *Blog* System as a new user with values *jack, 26, G11* for the attributes *username, age, postcode* respectively. Let us denote this entity as $JOHN'$. Even though the system treats *JOHN* and $JOHN'$ as separate digital entities, in reality they are the same physical entity. As these two accounts belong to two separate entities in the system, all previously discussed DIM components are applicable to them as well.

Identification & Authentication. Before any service can be accessed, the user needs to be identified and authenticated. Identification is the process of finding an association between an identifier (or in general an attribute) value and the entity and authentication is the process of proving the association. We can use the DIM component *idValToEnt* (see Definition 6) for identification. Similarly, the component *checkCred* (see Definition 8) can be used for authentication. In many systems, the process of identification and authentication are combined together in a single step.

For authentication, we define an abstract high-level algorithm (see Algorithm 1) that takes as input the user's supplied identifier and attribute values and either returns a successful result if the entity (the user) can be identified and returns an unsuccessful result otherwise. We let $AUTHN_{idp}$ denote the set of authenticated entities.

Algorithm 1 $IdAuthentication(e, i, v_1, v_2, cred_{idp})$. Authenticates an entity through identifier and attribute values.

Input: $e \in E_{idp}$, $i{=}ident(idp), v_1 \in atToValSet_{idp}(i)$, $v_2 \in atToValSet_{idp}(cred_{idp})$
if $(e{=}idValToEnt_{idp}(v_1)$ **and** $checkCred((i, v_1), (cred_{idp}, v_2)))$ **then**
  return `true`
else
  return `false`

Authorisation. Authorisation is the process to decide if an entity can perform a certain action on a specific resource in a specific context based on an identifier value or partial identifier values [19]. The authorisation usually takes place in the SP. We let ACT_{sp} and RES_{sp} denote the sets of actions and resources respectively in the corresponding SP.

Authorisation only takes place if an entity is authenticated in the IdP, meaning the entity is in the set $AUTHN_{idp}$. The SP requires the identifier and/or partial identifiers and their values for the entity, as well as the relevant action and resource. A subset of the user profile containing this information together with the requested action and resource are transferred to the SP using an Identity Protocol (see below). Upon receipt, the SP utilises an access control mechanism to authorise the entity. The mechanism may include an Access Control List (ACL) [28] or a Role Based Access Control (RBAC) feature [8]. At an abstract high-level this is just a set that lists which entity having which particular identifier or partial identifiers can perform which actions on what resources. We let ACL_{sp} denote the set of tuples at the SP that lists which entities and profiles can perform what actions on which resources. Formally we have $ACL_{sp} \subseteq \cup_{e \in AUTHN_{idp}} \mathcal{P}(PROFILE^{e}_{idp}) \times ACT_{sp} \times RES_{sp}$.

We can now define another abstract high-level algorithm (see Algorithm 2), that takes the requested inputs and either returns a successful result if the tuple can be found in ACL and an unsuccessful result otherwise.

Algorithm 2 $CheckAuthorisation(ACL_{sp}, e, p, act, res)$. Checks if an authenticated entity can perform an action on a resource.

Input: ACL_{sp}, $e \in AUTHN_{idp}$, $a \in ACT_{sp}$, $r \in RES_{sp}$, $p \subseteq PROFILE^{e}_{idp}$
if $((p, a, r) \in ACL)$ **then**
  return `true`
else
  return `false`

Service Provisioning. Once the user is identified, authenticated and authorised to access a particular service, they can access the service. In terms of IdM, accessing a service is known as service provisioning. An example service is that of an entity e being able to retrieve and update attribute values. Formally, when authorised to access this service in context e, an entity e will have partial control over the basic DIM component $atEntToVal_c$ (see Definition 1), in being able to view and update entries $atEntToVal_c(a, e)$ for all $a \in AV_c$. On the other hand, if the entity is not authenticated it will not able to either view or update its attribute values, meaning formally the entity has neither access nor control over the component $atEntToVal_c$.

De-registration. The final step is the de-registration process which allows users to de-register from an IdP. This process usually removes the entity and the association between the entity and both the identifier and attributes from a specific context (IdP). It is the reverse process of registration and is modelled in the following way. For $idp \in IDP$:

$$\begin{aligned} AV'_{idp} &= AV_{idp} \setminus \mathcal{AV}' \\ E'_{idp} &= E_{idp} \setminus \{e'\} \end{aligned}$$

where E'_{idp} and AV'_{idp} represent the updated set of entities and attribute values. Similarly to registration, the components of Definition 12 for context idp are updated if their domains/ranges change.

From the discussion above, the operations of registration, identification, authentication and de-registration take place at the IdP while authorisation and service provisioning take place at the SP. We will use the notation $IdmOP$ to denote the combined set of operations.

3.2 Identity Protocol

Careful readers will have noticed a discrepancy in the Authorisation step where inputs from different contexts are used in Algorithm 2. In particular, the authenticated entity and profile information is from the context of the IdP, while the remaining inputs relate to the SP. In reality, the SP receives the relevant information from the IdP during the authorisation phase using a secure transport mechanism called the *Identity Protocol* and is part of every IMS. Different IMSs have different Identity Protocols. Examples of widely used identity protocols include SAML (Security Assertion Markup Language) [27], OAuth [20] and OpenID [21].

4 Applications

In this section, we will explore three possible scenarios where our model can be applied. In the first scenario, we will analyse the effect of multiple partial identities of a user. In the second scenario, we analyse the cause of identity theft. In the last scenario, we will show how our models can be used to characterise the behaviour of two popular IdM models.

4.1 Effect of Multiple Partial Identities

The typical use of online services requires a user to register with multiple identity (or service) providers by providing different attribute values during the registration process as explained above. This means the user ends up with multiple partial identities scattered across multiple providers. If $n_{e,c}$ denotes the number of attribute values provided by the entity (user) e in the context (provider) c, then $n_{e,c} = |parIdent^e_c|$. Now, supposing $c_1, \ldots, c_m$ are all the contexts (providers) in which e has partial identities, the total number of attributes for the user e is given by:

$$N_e = \sum_{i=1}^{m} n_{e,c_i} .$$

The value N_e signifies the number of attributes that need to be managed by the user e. Note that some attributes and their values may overlap in different contexts. Nevertheless, they must all be accounted for when calculating the total number of attributes as they need to be managed separately in different contexts. Ideally, N_e will have a small value allowing the user to manage its attributes in a convenient way. Unfortunately, with the proliferation of novel online services requiring users to register to access those services, the value of N_e keeps increasing. One of the central focuses in IdM research is to reduce N_e for every entity e. There are two ways it can be reduced: i) by lowering $n_{e,c}$ in each context c, i.e. only storing smaller number of attributes at each provider and ii) by lowering m so that attributes are stored by a small number of providers. The second option is more suitable for two reasons. Firstly, it may not be always possible to know beforehand which attributes might be required/requested by the SP later on, hence the IdP might prefer to store as many attributes as possible. Secondly, when attributes are scattered across many IdP, it becomes increasingly difficult for the user to effectively manage attributes stored in those IdP. Minimising m would enable the user to manage attributes efficiently and hence is the focus of existing identity management systems.

Ideally, m=1 would be the most suitable choice as far as the user is concerned. It means that there is only one IdP storing all attributes of the user and providing them to the SP. With this goal, Microsoft introduced the *passport system* as the IdP of the Internet [4]. However, the attempt failed and the reason behind this failure was that the passport was included in each interaction between a user and the SP was not properly justified and users were not very confident and comfortable about a third party holding all their attributes [2]. Since then, it has been predicted and envisioned that there will exist a number of IdPs each with their specific purpose. For example, a bank IdP can be used for financial activities and the governmental IdP for accessing governmental services. This probably means that the value of m will always be more than 1. The optimal value of m that will enable users to manage their attributes in the most efficient manner is yet to be found and might vary from user to user.

4.2 Analysing Identity Theft

Identity theft is one of the major online fraudulent activities. It has been a source of huge financial losses in recent years, for example, in 2013 the financial loss due to the identity theft reached nearly $21 billion in the USA [11]. Identity theft has been a difficult issue to tackle and many attempts have been proved fruitless so far. Before we use our model to analyse the root cause of identity theft, let us look at a suitable textual definition of identity theft.

The term *identity theft* has been defined in many ways in the literature [12, 18]. Consider Koops et. al. [16] definition: "*Identity 'theft' is fraud or another unlawful activity where the identity of an existing person is used as a target or principal tool without that person's consent*". The person whose identity is being abused is the *victim* and the person who abuses the victim's identity is the *attacker*. This definition is too literal, there is no mention if the term *identity* in the definition means the total identity or the partial identity in a particular context.

For the time being, let us assume that it refers to the partial identity of an entity e in a particular context c. Now, to steal the entity's partial identity in a context, an attacker needs to get hold of all the attribute values provided by the victim in the context c which is represented by set $parIdent^e_c$. This set can be generated if one has access to the basic DIM component $atEntToVal_c$ (see Definition 1). As discussed in the *Service Provisioning* step of Section 2, an attacker will have sufficient access to this component to obtain e's profile, if the attacker is identified, authenticated and authorised as e in context c. Furthermore, as explained in the *Identification & Authentication* step of Section 2, achieving this is dependent on the attacker getting hold of the values of both the identifier (e.g. a username) and credential of e for context c. There are numerous ways an attacker can get hold of these two pieces of information such as using simple or complex social engineering techniques, having access to a piece of paper containing such information in written format, deploying advanced phishing techniques, by guessing or brute forcing a password.

There have been many attempts to tackle this problem, mostly focus on hardening the credential (e.g. strong passwords which are difficult to guess or brute force, biometrics which are difficult to forge or hardware tokens to generate one time password), using secure hardware (e.g. smartcard) or using multi-factor authentication (e.g. using a fixed password at first and then providing a secondary password in the mobile phone). Nonetheless, except for biometrics, it does not matter how strong the credential is, if the attacker gets hold of it, there are always chances of identity theft. It seems that the root cause might be something else. To analyse that, let us introduce the following function.

Definition 13. *Let the following function return the entity who has supplied the value of an identifier and credential pair in the context c:*

$$suppliedBy_c : (ident(c) \times AV_c) \times (cred_c \times AV_c) \rightarrow E$$

Notice, *suppliedBy* can return an entity who may not exist in that context.

It is assumed in the setting of current online services that:

$$idValToEnt_c(v) = suppliedBy_c((ident(c), v), (cred_c, v'))$$

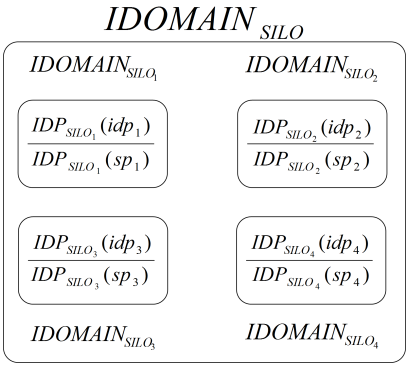

Figure 3: SILO Model.

where $v \in atToValSet_c(ident(c))$ and $v' \in atToValSet_c(cred_c)$. However, in case of an identity theft, we have:

$$idValToEnt_c(v) \neq suppliedBy_c((ident(c), v), (cred_c, v'))\,.$$

Therefore to tackle identity theft we just need a function with the capability of Definition 13. Unfortunately, developing and deploying such a function is extremely difficult, if not impossible, with the current structure of the Internet where anybody can be anything.

4.3 Modelling Identity Management Models

Finally, we show how our IdM Model can be extended to characterise the behaviour of two popular IdM Models: the Isolated User Identity (SILO) Model and the Federated User Identity (FED) Model. For this we need to define an *Identity Domain*. An Identity Domain is the set of application domains in which an Identifier is unique [15]. Different models utilise different types of identity domains which act as the context for those models. We will use the notation $IDDom_{m_i}$ to denote a single identity domain i of an IdM Model m. Additionally, the notation IDP_m and SP_m will be used to denote the sets of all IdPs and SPs respectively in a model m. In addition, we require a function that returns the set of SPs shared by an IdP in a specific IdM Model.

Definition 14. *Let $sharedBy_m : IDP_m \to \mathcal{P}(SP_m)$ be the function that maps an IdP to the set of SP that utilise the service of the IdP for operations in IdM model m.*

4.3.1 Isolated User Identity (SILO) Model.

The Isolated User Identity (SILO) Model represents the most common and the simplest IdM model [15]. There are only two parties involved: a SP (combined with its own IdP) and the clients. As the SP and IdP are the same entity, they will be used interchangeably. Each SP maintains its own identity domain and the identity operations performed in one domain are not valid in any other. Formally, we have $CONTEXT = \{SILO_1, \ldots, SILO_n\}$ for some $n \in \mathbb{N}$ where:

- $IDP_{SILO_i} = \{idp_i\} = \{sp_i\} = SP_{SILO_i}$ for $1 \leq i \leq n$;
- $IDDom_{SILO_i} = IDP_{SILO_i}$ for $1 \leq i \leq n$.

The requirements above signify that each identity domain in the SILO Model, denoted $SILO_i$, consists of an IdP idp_i which can also be regarded as the SP since they are the same entity. Figure 3 illustrates the SILO Model for n=4. Moreover, for any $1 \leq i \neq j \leq n$:

- $|sharedBy_{SILO_i}(idp_i)| = 1$ for $idp_i \in IDP_{SILO_i}$;

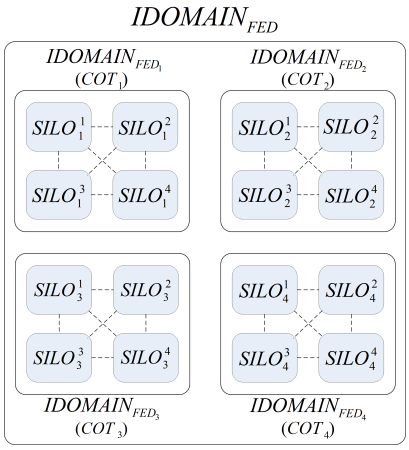

Figure 4: Federated Model.

- $sharedBy_{SILO_i}(idp_i) \cap sharedBy_{SILO_j}(idp_j) = \emptyset$ for $idp_i \in IDP_{SILO_i}$ and $idp_j \in IDP_{SILO_j}$.

That is, each IdP is used by precisely one SP and two IdP belonging to different identity domains use different SPs. IdM operations in one domain (inside one IdP or SP) is completely separate from the operations in other domains. More formally, we have $IdmOP_{IDDom_{SILO_i}} \cap IdmOP_{IDDom_{SILO_j}} = \emptyset$ for $1 \leq i \neq j \leq n$.

4.3.2 Federated User Identity (FED) Model.

In the FED Model each identity domain is shared between a number of previously agreed SPs [15]. The unified identity domain is commonly known as the Federated Domain and the SPs and together with their IdP are said to be a part of the so-called Circle of Trust (CoT). The client of one SP can authenticate herself with her corresponding IdP and then enjoy the services of other SPs that are inside the same CoT. Shibboleth is an example of a system based on this model [25].

In this model (see Figure 4) we have

$$CONTEXT = \{SILO_i^j \mid 1 \leq i \leq n \wedge 1 \leq j \leq n_i\}$$

for some $n \in \mathbb{N}$ and $n_i \in \mathbb{N}$ where:

- $IDDom_{FED_i} = COT_i = \bigcup_{j=1}^{n_i} IDDom_{SILO_i^j}$ for $1 \leq i \leq n$;
- $|sharedBy_{FED_i}(idp)| = n_i$ for $idp \in COT_i$ and $1 \leq i \leq n$.

The first property signifies that each federated domain or each CoT (COT_i), in essence, consists of several SILO identity domains (denoted as $SILO_i$). However, unlike the SILO model, this model offers interoperability among different SILO domains, meaning that IdM operations such as identification and authentication that take place inside an IdP are shared among several SP inside the same CoT and is indicated by the second property above. To ensure interoperability, an agreement is signed among participating organisations to create a CoT. The combined federated identity domain consists of several CoTs.

In the FED model, IdM operations are different in different CoTs. That is, for any $1 \leq i \neq j \leq n$:

$$IdmOP_{COT_i} \cap IdmOP_{COT_j} = \emptyset\,.$$

However, once the user is authenticated in one IdP, she will also be considered authenticated at all IdP belonging to the same CoT. So, for any $1 \leq i \leq n$ and $idp_i, idp_i' \in COT_i$ we have $AUTHN_{idp_i} = AUTHN_{idp_i'}$.

5 Conclusion

In this paper, we have built up a mathematical framework to formally model the central aspects of IdM and used the framework to model several aspects of IMS. We have provided three application scenarios showing how our model can be applied. Being based on formal mathematical foundations, the framework can be used to build a deep understanding of the central issues of IdM. This is the first model of its kind and we believe it has the potential for rigorous reasoning and formal analysis of IdM. In particular, it can be used as the framework for an *Identity Calculus* which can be used to study the change of identities that users go through from systems to systems and its effect in those systems. One limitation of our approach is that it does not deal with privacy properties. This is intentional as there are other works, e.g. [22, 29], that have formalised the privacy properties of IdM systems. One next step is to investigate how our model can be integrated with these works. Currently the model is static in nature and is not able to capture the dynamics of practical IMS. In future we plan to extend the model to capture the dynamics of practical systems using the state-based Z language [26] and the event-based Process Algebra CSP [23]. This will help not only to provide a rigorous understanding of these practical systems, but also to formally compare the behaviour of two systems. For example, if the dynamics are captured using CSP, then the tool FDR3 [9] can be used for automated verification (e.g. using refinement checking to compare specifications of IMSs with their implementations). We would also like to extend our framework to accommodate advanced features of IMS such as attribute aggregation [6] and account linking [5] and to give formal models of extensions of IdM such as Mobile IdM [24].

6 References

[1] G. Alpár, J.-H. Hoepman, and J. Siljee. The Identity Crisis. Security, Privacy and Usability Issues in Identity Management. *CoRR*, abs/1101.0427, 2011.

[2] K. Cameron. The Laws of Identity. 14th May, 2005. http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf.

[3] J. Camp. Digital identity. *Technology and Society Magazine, IEEE*, 23(3):34–41, 2004.

[4] D. Chadwick. Federated Identity Management. In *FOSAD'08/09*, volume 5705 of LNCS, pages 96–120, Springer, 2009.

[5] D. Chadwick, G. Inman, K. Siu, and Md. S. Ferdous. Leveraging social networks to gain access to organisational resources. In *DIM'11*, pages 43–52, 2011

[6] D. Chadwick and G. Inman. Attribute aggregation in federated identity management. *Computer*, 42(5):33–40, 2009.

[7] Md. S. Ferdous, Audun Jøsang, K. Singh, and R. Borgaonkar. Security Usability of Petname Systems. In *NordSec'09*, volume 5838 of *LNCS*, pages 44–59, Springer, 2009.

[8] D. Ferraiolo and R. Kuhn. Role-Based Access Control. In *In 15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[9] T. Gibson-Robinson, P. Armstrong, A. Boulgakov, and A. Roscoe. FDR3 - A Modern Refinement Checker for CSP. In *TACAS'14*, volume 8413 of *LNCS*, pages 187–201, Springer, 2014.

[10] U. Glasser and M. Vajihollahi. Identity management architecture. In *ISI'08*, pages 137–144, 2008.

[11] Identity Fraud Report: Data Breaches Becoming a Treasure Trove for Fraudsters. 2013.

[12] Identity Theft and Assumption Deterrence Act of 1998: Title 18 USC 1028. Accessed 1 April, 2014, 1998. http://www.ckfraud.org/title_18.html.

[13] D.-O. Jaquet-Chiffelle, E. Benoist, R. Haenni, F. Wenger, and Harald Zwingelberg. Virtual Persons and Identities. In *FIDIS'09*, pages 75–122, 2009.

[14] A. Jøsang, M. Al, and Z. Suriadi. Usability and privacy in identity management architectures. In *ACSW'07*, pages 143–152, 2007.

[15] A. Jøsang and S. Pope. User Centric Identity Management. In *AusCERT'05*, pages 77–89, 2005.

[16] B.-J. Koops and R. Leenes. Identity theft, identity fraud and/or identity-related crime. *Datenschutz und Datensicherheit-DuD*, 30(9):553–556, 2006.

[17] T. El Maliki and J.-M. Seigneur. User-centric Mobile Identity Management Services. *Management*, pages 33–76, 2008.

[18] N. Mitchison, M. Wilikens, L. Breitbach, R. Urry, and S. Portesi. Identity Theft - A Discussion Paper. Technical report, 2004.

[19] Modinis - Common Terminological Framework for Interoperable Electronic Identity Management. Accessed 28th June, 2011. https://www.cosic.esat.kuleuven.be/modinis-idm/twiki/bin/view.cgi/Main/GlossaryDoc.

[20] OAuth 2.0. http://oauth.net/2.

[21] OpenID Authentication 2.0 - Final. 5 December, 2007. http://openid.net/specs/openid-authentication-2_0.html.

[22] A. Pfitzmann and M. Hansen. A terminology for talking about privacy by data minimization:Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management. V0.34, August 10 2010. http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf.

[23] A. Roscoe. *The theory and practice of concurrency*. Prentice Hall, 1998.

[24] G. Roussos, D. Peterson, and UY. Patel. Mobile Identity Management: An Enacted View. *INT. JOUR. E-COMMERCE, VOL*, 8:81–100, 2003.

[25] Shibboleth. http://shibboleth.internet2.edu/.

[26] J. Spivey and J. Abrial. *The Z notation*. Prentice Hall Hemel Hempstead, 1992.

[27] OASIS Standard. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. 15 March, 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf.

[28] Using Access Control Lists (ACLs). http://www.hp.com/rnd/support/manuals/pdf/release_06628_07110/Bk2_Ch3_ACL.pdf.

[29] M. Veeningen, B. De Weger, and N. Zannone. Modeling identity-related properties and their privacy strength. In *FAST'10*, pages 126–140, 2011.