DOI:10.1109/SocialCom.2010.115

Minimizing Disclosure of Private Information in Credential-Based Interactions: A Graph-Based Approach

Claudio A. Ardagna*, Sabrina De Capitani di Vimercati*, Sara Foresti*, Stefano Paraboschi†, Pierangela Samarati*
*DTI - Università degli Studi di Milano, 26013 Crema - Italy
†DIIMM - Università degli Studi di Bergamo, 24044 Dalmine - Italy
Email: firstname.lastname@unimi.it, parabosc@unibg.it

***Abstract*—We address the problem of enabling clients to regulate disclosure of their credentials and properties when interacting with servers in open scenarios. We provide a means for clients to specify the sensitivity of information in their portfolio at a fine-grain level and to determine the credentials and properties to disclose to satisfy a server request while minimizing the sensitivity of the information disclosed. Exploiting a graph modeling of the problem, we develop a heuristic approach for determining a disclosure minimizing released information, that offers execution times compatible with the requirements of interactive access to Web resources.**

***Keywords*-privacy, portfolio management, credentials.**

I. Introduction

The development in the past years of the Internet and associated Web technology has produced a large impact on society. Still, technology experts and final users perceive crucial open problems in the area of security and privacy. A specific goal receiving considerable attention is the design of a Web infrastructure offering protection against adversaries interested in improperly acquiring user access privileges. At the same time, users want to easily access all the resources available to them, without the need to remember passwords or manage a specific account for each of the systems they access. Cryptographic credentials offer a great potential for the satisfaction of the above requirements. Using credentials it is possible to verify that the counterpart on the other side of the Internet communication channel exhibits given properties, proved in a robust way using a simple challenge/response interaction. Today, the most significant use of credentials is represented by X.509 certificates exhibited by servers to prove that they are the legitimate owners of a given domain.

The use of credentials to regulate interactions in open systems has received considerable attention in the last ten years. A large number of approaches (e.g., [3], [11], [15]) have been developed proposing novel policy languages and engines to specify and enforce access control regulations in the presence of requests coming from clients not known a priori and to communicate them the requirements they need to satisfy. Most proposals have typically focused on the server side of the problem of supporting interactions, typically assuming that at the client side a symmetric approach could be applied for specifying possible regulations on the release of information in the client portfolio.

The support for client side solutions to regulate credentials release is crucial for a wide-scale deployment of credentials. However, access control-like specifications do not completely fit the possible protection requirements at the client side, where users may need a way to specify preferences on the information to disclose based on the sensitivity of such information [6], [10], [14]. In this paper, we focus on this aspect and provide a way for users to specify privacy settings on their portfolio for regulating information release.

The paper contribution is multifold. First, after a recollection of basic concepts (Section II), we introduce novel concepts in the modeling of the client portfolio, which also consider emerging paradigms for producing and presenting credentials, and illustrate a graph-based modeling of such concepts (Section III) and of the server request (Section IV). Second, we provide a way for the client to specify at a fine-grain level the sensitivity of all the components in its portfolio. Our modeling of sensitivity is very general and can accommodate different ways in which sensitivity could be specified, also capturing sensitive associations of client's properties (Section V). Third, we characterize the disclosure of information in a portfolio in terms of the graphical modeling and represent the problem of determining a disclosure satisfying the request while minimizing information as the problem of determining a minimum isomorphic graph matching (Section VI). Finally, we describe our heuristic, exploiting the graphical representation of the model components, for computing a solution to the problem and present experimental results that confirm the approach is applicable to interactive scenarios (Section VII).

This work was supported in part by the EU within the 7FP project "PrimeLife" under grant agreement 216483 and by the Italian Ministry of Research within the PRIN 2008 project "PEPPER" (2008SY2PH4).

II. Basic concepts

The information that a client can provide to acquire services forms a *portfolio* that includes properties in certificates, signed by third parties, as well as (uncertified) properties that the client can utter. Like in the literature [3], we refer to certificates as *credentials* and to uncertified information as *declarations*. Credentials are organized by type, where the type of a credential identifies the properties that the credential certifies. Abstractions can be defined over the credential types, possibly introducing a hierarchy of types. Formally, a hierarchy $\mathcal{H}$ of credential types is a pair $(\mathcal{T},\preceq_{isa})$, where $\mathcal{T}$ is the set of all types, and $\preceq_{isa}$ is a partial order relationship over $\mathcal{T}$. Given two types t_i and t_j in $\mathcal{T}$, $t_i \preceq_{isa} t_j$ if t_j is an abstraction of t_i. For instance, *id* is an abstraction of credential types *photo-id* and *student-id* (i.e., *photo-id* $\preceq_{isa}$ *id* and *student-id* $\preceq_{isa}$ *id*).

Each credential certifies a set of properties and is characterized by its type, unique identifier, and issuer.

Our modeling of the portfolio includes all the concepts described above, and distinguishes types from instances, and credentials from declarations.

- *Credential types vs instances*. Our model allows referring to credentials at the granularity of instance or type. For example, while a client can refer to credential types or their specific instances, which it knows (e.g., a specific identity card), the requests by the server will typically be expressed in terms of credential types (e.g., *photo-id* or *id*). Note that a portfolio may contain different credential instances of the same type. Also, while directly belonging to a single type, a credential indirectly belongs to all the abstractions of such a type.
- *Credentials vs declarations*. We explicitly model declarations allowing the inclusion in the client portfolio of properties that do not belong to any credential. In addition, we assume that any property appearing in credentials can be uttered in an uncertified way by the client, and therefore can be stated as a declaration. In our modeling, we conveniently represent declarations as a self-signed credential, whose identifier is *decl*, containing all the properties of the portfolio.

III. Client portfolio

We enrich the client portfolio with novel concepts, aiming at providing ability to the client to organize and manage its portfolio at a fine-grain level for regulating disclosure of credentials and properties.

- *Credential-dependent vs credential-independent properties.* Credentials certify some properties of the client. We distinguish between properties associated uniquely with the client, regardless of the credentials that certify them (*credential-independent properties*), and properties associated with a specific credential of the client (*credential-dependent properties*). For instance, date of birth is a property of the user, and possible occurrences of the property in different credentials refer all to the same piece of information. In other words, the value of credential-independent properties depends only on the credential's owner, and not on the specific credential certifying the value. By contrast, a property such as credit card number is specific of some given credentials of the user. Different instances of the credit card credential type will all refer to their specific credit card number, and therefore to a different piece of information. Credential-dependent properties might have different occurrences, depending on the existence of different credentials including them.
- *Atomic vs non-atomic credentials.* The most common kind of credentials used today in distributed systems is represented by X.509 certificates. One of the limitations of X.509 certificates is their rigidity: the signature is computed on the hash of the content of the credential, and the use of the credential requires to access its complete representation. In other words, it is not possible to selectively disclose only part of the credential content. Instead, modern credential technology (e.g., UProve and Idemix [4], [5]) supports the release of individual properties extracted from the credential. Our model includes this aspect of modern credential technology and classify credentials as atomic or non-atomic. Atomic credentials can only be released as a whole, that is, their release entails the disclosure of all the properties they certify. Properties in non-atomic credentials can instead be selectively released. The self-signed credential *decl* is clearly non-atomic.
- *Information sensitivity*. Previous works have put forward the idea of a preference relationship among credentials/properties defining that release of some information is to be preferred over release of other information. We provide a way for the user to specify the sensitivity of her credentials, properties, and associations among properties, to the aim of minimizing the 'amount' of information released for acquiring a service. We discuss portfolio sensitivity in Section V.

We model the client portfolio as a *portfolio graph*, defined as follows.

Definition III.1 (Portfolio Graph)**.** *Let $\mathcal{C}$ and $\mathcal{P}$ be a set of credentials and properties, respectively, in a client portfolio. The* portfolio graph *$G(V_C \cup V_P, E_{CP})$ is a bipartite graph having a vertex for each credential in*

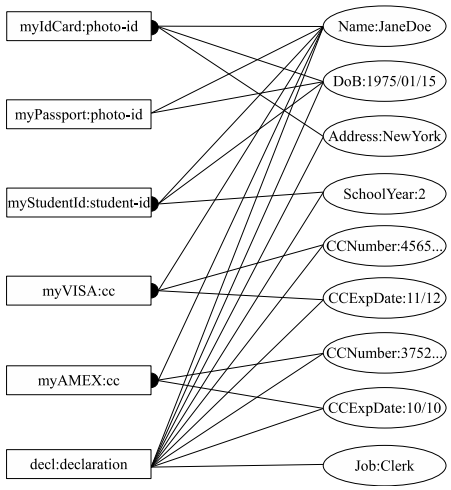


Figure 1. An example of a portfolio graph

$\mathcal{C}$ and each property in $\mathcal{P}$, and an edge connecting each credential to the properties contained in it.

The label of a vertex representing a property is of the form p:*value*, where p is the property name and *value* its value. The label of a vertex representing a credential is of the form c:*type*, where c is the credential identifier and *type* its type. For simplicity, in the following, we will use c (p, resp.) to refer to either the credential (property, resp.) or the label c:*type* (p:*value*, resp.) of the corresponding vertex. We will also denote with *type*(c) the type of credential c. Note that, in the portfolio graph, each credential-independent property is represented as a single vertex (connected to all credentials in which it is contained). Each credential-dependent property is instead represented with several vertices (one for each credential where it appears, i.e., each of its instantiations).

In the graphical representation, credential vertices are represented as rectangles and property vertices are represented as ovals. Also, we distinguish atomic from non-atomic credentials by attaching all the edges incident to an atomic credential to a black semicircle.

Example III.1. *Figure 1 illustrates an example of a portfolio graph, including credentials* myIdCard *and* myPassport (*both of type* photo-id), myStudentId (*of type* student-id), *and* myVISA *and* myAMEX (*both of type* cc). *The only non-atomic credential is* myPassport. *Properties* `Name`, `DoB`, `Address`, `SchoolYear`, *and* `Job` *are credential-independent, while* `CCNumber` *and* `CCExpDate` *are credential-dependent* (*having a different occurrence for each credit card*).

IV. Server request

The request of the server is modeled as a boolean formula $\mathcal{R}$, which describes the set of properties (and the way in which they should be certified) that the client needs to disclose to acquire a given service. For simplicity and without loss of generality, we assume the request $\mathcal{R}$ to be expressed as the disjunction of simple requests, that is, $\mathcal{R}=r_1 \vee \ldots \vee r_i$. Each simple request r is the conjunction of terms of the form *type*.$\{p_1,\ldots,p_m\}$, where each term prescribes the disclosure of the set $\{p_1,\ldots,p_m\}$ of properties from a single credential c in the client portfolio, such that *type*$(c) \preceq_{isa}$ *type*. Different terms must be certified by different credentials.

Example IV.1. *A request* $\mathcal{R} = r_1 \vee r_2$, *with* $r_1=$(id.{`Name`,`DoB`} $\wedge$ cc.{`Name`,`CCNumber`,`CCExpDate`} $\wedge$ $*$.{`SchoolYear`,`Job`}) *and* $r_2 =$ (id.{`Name`,`DoB`,`Address`} $\wedge$ cc.{`Name`,`CCNumber`} $\wedge$ cc.{`Name`,`CCNumber`}) *can be satisfied by two different ways. The first possibility discloses:* i) *properties* `Name` *and* `DoB` *from a credential of type* id*;* ii) *a credit card; and* iii) *properties* `SchoolYear` *and* `Job` *from any credential* (*denoted with* $*$). *The second possibility discloses:* i) *properties* `Name`, `DoB`, *and* `Address` *from a credential of type* id*; and* ii) *properties* `Name` *and* `CCNumber` *from two different credit cards.*

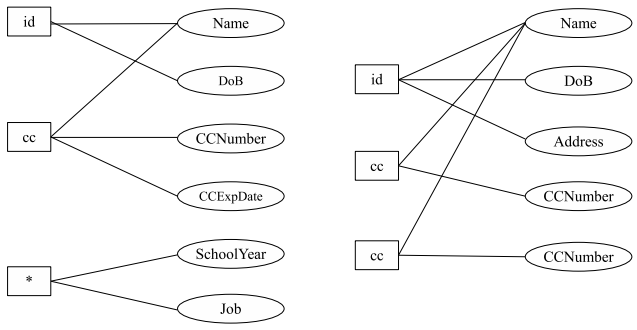
A request $\mathcal{R}$ can be graphically represented as a set of *request graphs*. Each request graph models a simple request r, where, for each term *type*.$\{p_1,\ldots,p_m\}$, there is a vertex with label *type*, a vertex for each property, whose label is the property name, and an edge connecting each term with the properties it requires. A request graph is formally defined as follows.

Definition IV.1 (Request Graph)**.** *Let r be a simple request in a request $\mathcal{R}$. The* request graph $G^r(V^r_T \cup V^r_P, E^r_{TP})$ *of r is a bipartite graph having a vertex for each term and each property, and an edge connecting each term to the properties contained in it.*

In the following, when clear from the context, we will call request either a request $\mathcal{R}$ or a simple request r in $\mathcal{R}$. Figure 2 illustrates the request graphs for the request $\mathcal{R}$ described in Example IV.1.

V. Portfolio sensitivity

The major motivation of our work is to allow the client to automatically select which properties/credentials to release for acquiring access to a given service, while minimizing disclosure of sensitive information. In fact, when the server allows choices on the properties or credentials to present, the user may prefer to disclose some over others. For instance, one may prefer to release her address instead of her school year, and either of the two instead of her credit card number.



(a) G^{r_1} (b) G^{r_2}

Figure 2. An example of request graphs

With a very general setting, we assume privacy requirements on a portfolio to be specified in terms of the *sensitivity labels* of the portfolio components.[1] The only assumptions we make on such labels are the existence of a (partial) order relationship over them and the definition of a composition operator that determines the label resulting from the combination of two labels.

Sensitivity labels are then defined as follows.

Definition V.1 (Sensitivity Labels)**.** *Sensitivity labels* Λ *are a set of values in a partial order relationship* $\succeq$ *and over which an operator* $\oplus$ *is defined such that, for any two labels* λ_i *and* λ_j *in* Λ, $\lambda_i \oplus \lambda_j \succeq \lambda_i$ *and* $\lambda_i \oplus \lambda_j \succeq \lambda_j$.

The condition on operator $\oplus$ in the definition requires the operator to be monotonic; intuitively, combining two sensitivity labels (i.e., merging information) cannot decrease sensitivity.

It is easy to see that our generic definition of sensitivity labels permits to capture different ways of expressing preferences, including the kinds of preferences put forward in other works, as a very specific case. For instance, sensitivity labels could be classical multilevel security classifications (e.g., Top Secret, Secret, Confidential, Unclassified, possibly with associated categories) with the $\oplus$ operator corresponding to the *least upper bound*. Also, they could be positive integer values, where the $\oplus$ operator can be either the *sum* (i.e., $\lambda_i \oplus \lambda_j = \lambda_i + \lambda_j$) and therefore reflect an *additive property*, or the *maximum* (i.e., $\lambda_i \oplus \lambda_j = max(\lambda_i, \lambda_j)$). These examples are just two specific instantiations of sensitivity labels, and our modeling accommodates a variety of ways in which preferences over the credentials and properties to release can be specified and composed.

In our model, sensitivity labels can be specified at different granularities: properties, credentials, and sets of properties. Let us discuss the semantics of the labeling function λ that associates a sensitivity label with a property p, a credential c, or a set of properties a.

- $\lambda(p)$. Defines the sensitivity of property p individually taken. It reflects how much the user considers the property sensitive and therefore how much she values its release.
- $\lambda(c)$. Defines the sensitivity of the existence of credential c. The specification of sensitivity labels for all the properties of a credential is not sufficient to express the sensitivity of a credential. In fact, the existence of a credential itself may bear some information that the client considers sensitive. For instance, a dialysis certificate may include only properties `Name` and `Address`, but the existence of the certificate itself has an additional sensitivity that goes beyond the demographic information of the user. Also, in the case of a non-atomic credential, $\lambda(c)$ reflects the sensitivity assigned to the existence of the credential regardless of the release of the properties within it.
- $\lambda(a)$. Defines the sensitivity of an association a among a set of properties $\{p_1,\ldots,p_n\}$, whose joint release carries more information than the individual release of each property [7]. $\lambda(a)$ is the additional sensitivity over the combination of the labels of the properties composing it. The sensitivity of the disclosure of $\{p_1,\ldots,p_n\}$ is therefore $\lambda(a) \oplus \lambda(p_1) \oplus \ldots \oplus \lambda(p_n)$. For instance, the association between a name and the last four digits of a social security number can be considered more sensitive than the $\oplus$ composition of the sensitivity labels of the two. This additional sensitivity is expressed in our model by defining the association between the properties and expressing a sensitivity label for the association.

In our model, we explicitly represent sensitive associations by extending the portfolio graph to be a tripartite graph, where the third set of vertices is represented by associations (a vertex for each sensitive association) with edges connecting each association with the involved properties.

Definition V.2 (Portfolio Graph – Extended)**.** *Let* $G(V_C \cup V_P, E_{CP})$ *be a portfolio graph and* $\mathcal{A}$ *be a set of sensitive associations over* $\mathcal{P}$. *A* portfolio graph *extended by sensitive associations* $G(V_C \cup V_P \cup V_A, E_{CP} \cup E_{AP}, \lambda)$ *is a tripartite labeled graph, having an additional vertex for each sensitive association in* $\mathcal{A}$*, and an additional edge connecting each sensitive association to the properties contained in it. The labeling function* λ *assigns a label*

[1] A client may want to specify different preferences for different servers. Such a situation can be modeled in our approach by considering different labels depending on the server requesting the release; the approach would then work on the specific instance of the labels determined by the server with whom the client is interacting.

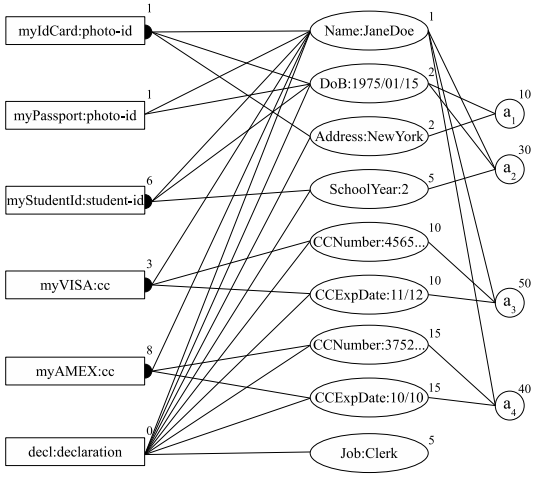


Figure 3. An example of a portfolio graph, extended with sensitive associations and sensitivity labels

$\lambda(v)$ to each vertex v, corresponding to the sensitivity of the information represented by the vertex.

Example V.1. *Figure 3 illustrates an example of a portfolio graph, obtained by extending the portfolio in Figure 1. For concreteness and simplicity, sensitivity labels are integer values and are composed using the sum operator. They are indicated next to the vertices. The portfolio specifies four sensitive associations: a_1) date of birth and address (since they could work as quasi-identifier [13], their association is more sensitive than the simple combination of their sensitivity); a_2) name, date of birth, and school year (as the association might disclose that the client is a mature student); a_3) name, credit card number, and expiration date of* myVISA (*since they could be used for unauthorized payments); a_4) similar to a_3 but with less sensitivity, since for credential* myAMEX *the risk of unauthorized charges is considered lower.*

VI. Disclosure modeling

A disclosure represents a subset of the client portfolio, which is communicated to the server for satisfying a request. A disclosure can be modeled as a subgraph of the portfolio graph, called *disclosure graph*. Intuitively, this subgraph includes all the vertices and edges corresponding to credentials, properties, and associations that are exposed by the disclosure. Note that the disclosure to the server of a subset of the properties in the portfolio must also imply the release of a set of credentials certifying them, additional properties included in atomic credentials, and sensitive associations. Therefore, while each disclosure is a subgraph, the vice versa is not necessarily true (i.e., not all subgraphs represent a disclosure). As a matter of fact, a subgraph of the portfolio graph can be considered a disclosure graph only if it correctly represents a possible release of information. In particular, in a disclosure: *1)* each disclosed property must be certified by (at least) a credential, that is, credential existence is also disclosed (*certifiability*); *2)* if a property of an atomic credential is disclosed, all its properties are disclosed (*atomicity*); *3)* if all properties composing a sensitive association are disclosed, the sensitive association must be considered disclosed (*association exposure*). These properties are captured by the following definition of disclosure graph.

Definition VI.1 (Disclosure Graph)**.** *Let $G(V_C \cup V_P \cup V_A, E_{CP} \cup E_{AP}, \lambda)$ be a portfolio graph. A subgraph $G^d(V^d_C \cup V^d_P \cup V^d_A, E^d_{CP} \cup E^d_{AP}, \lambda)$ of G where $V^d_C \subseteq V_C$, $V^d_P \subseteq V_P$, $V^d_A \subseteq V_A$, $E^d_{CP} \subseteq E_{CP}$, and $E^d_{AP} \subseteq E_{AP}$ is a* disclosure graph *iff the following properties hold:*

1) $v_p \in V^d_P \implies \exists\ v_c \in V^d_C$ *s.t.* $(v_c, v_p) \in E^d_{CP}$;
2) $v_c \in V^d_C$ *s.t. credential* v_c *is atomic* $\implies \forall v_p \in V_P$: $(v_c, v_p) \in E_{CP}$, $v_p \in V^d_P$ *and* $(v_c, v_p) \in E^d_{CP}$;
3) $v_a \in V_A$ *s.t.* $\forall (v_a, v_p) \in E_{AP}$, $v_p \in V^d_P \implies v_a \in V^d_A$ *and* $(v_a, v_p) \in E^d_{AP}$.

Condition 1 states that if a property vertex belongs to the disclosure graph, then at least one of its adjacent credential vertices belongs to the graph. Condition 2 states that if a credential vertex representing an atomic credential belongs to the disclosure graph, then all the vertices representing its properties, and the edges modeling the containment relationship of the properties in the atomic credential also belong to the graph. Condition 3 states that if all the vertices representing the properties composing a sensitive association belong to the disclosure graph, then also the vertex representing the association and the edges between the association and the involved properties belong to the graph.

The sensitivity of a disclosure can be computed by composing the sensitivity labels of the vertices in the corresponding disclosure graph. Given a disclosure graph $G^d(V^d_C \cup V^d_P \cup V^d_A, E^d_{CP} \cup E^d_{AP}, \lambda)$ and a monotonic composition operator $\oplus$ for λ, the sensitivity $\lambda(G^d)$ of the disclosure graph is $\lambda(G^d) = \bigoplus_v \lambda(v)$, with v in $V^d_C \cup V^d_P \cup V^d_A$.

Example VI.1. *Figure 4 represents an example of a disclosure graph G^d, which is a subgraph of the portfolio graph in Figure 3. The vertices and edges in the portfolio graph that also belong to the disclosure graph are represented with a bold line in the figure. The sensitivity of the disclosure $\lambda(G^d)$ is computed as the composition of the sensitivity labels of bold vertices, that is,* $\lambda($`Name`:JaneDoe) $\oplus$ $\lambda($`DoB`:1975/01/15) $\oplus$ $\lambda($`Address`:NewYork) $\oplus$ $\lambda($`SchoolYear`:2) $\oplus$ $\lambda($myVISA:cc/`CCNumber`:4565…)

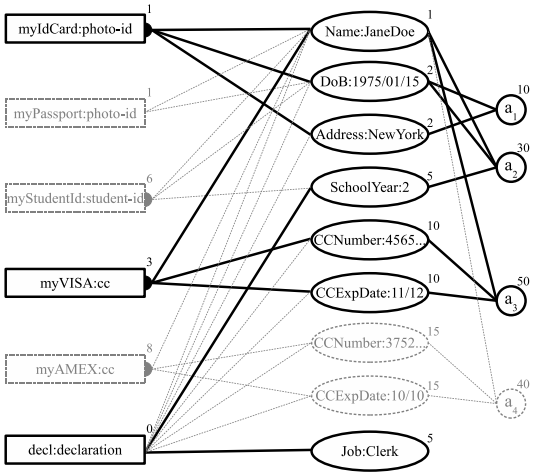


Figure 4. An example of a disclosure graph

$\oplus$ λ(myVISA:cc/`CCExpDate`:11/12) $\oplus$ λ(`Job`:Clerk) $\oplus$ λ(myIdCard:photo-id) $\oplus$ λ(myVISA:cc) $\oplus$ λ(decl:declaration) $\oplus$ $\lambda(a_1)$ $\oplus$ $\lambda(a_2)$ $\oplus$ $\lambda(a_3)$=129.

Intuitively, a request $\mathcal{R}$ is satisfied by a disclosure if at least one of the simple requests composing it is satisfied. A simple request r is satisfied by a disclosure that includes, for each term $type.\{p_1,\ldots,p_m\}$ in r, a credential c certifying $\{p_1,\ldots,p_m\}$ such that $type(c) \preceq_{isa} type$. Graphically, a disclosure, represented by a disclosure graph G^d, satisfies a simple request, represented by a request graph G^r, if there exists a subgraph in G^d that is isomorphic to G^r, as formalized by the following definition.

Definition VI.2 (Satisfying Disclosure)**.** *Let G^d be a disclosure graph and $G^r(V^r_T \cup V^r_P, E^r_{TP})$ be a request graph. G^d* satisfies *G^r, denoted $G^d \models G^r$, iff there exists a sub-graph $G^{d'}(V^{d'}_C \cup V^{d'}_P, E^{d'}_{CP})$ of G^d and an isomorphism $f: V^r_T \cup V^r_P \rightarrow V^{d'}_C \cup V^{d'}_P$, such that the following conditions hold:*

1) $\forall v_t \in V^r_T$, $\exists f(v_t) \in V^{d'}_C \wedge type(f(v_t)) \preceq_{isa} v_t$;
2) $\forall v_p \in V^r_P$, $\exists f(v_p) \in V^{d'}_P \wedge v_p = f(v_p)$;
3) $\forall (v_t, v_p) \in E^r_{TP}$, $(f(v_t), f(v_p)) \in E^{d'}_{CP}$.

Condition 1 states that each vertex v_t in the request graph, representing a type in a term of the request, should have a corresponding vertex $f(v_t)$ in the disclosure graph, such that v_t is an abstraction of $type(f(v_t))$. Condition 2 states that each vertex v_p in the request graph, representing a property within a term, should have a corresponding vertex $f(v_p)$ in the disclosure graph. Condition 3 states that each edge in the request graph should have a corresponding edge in the disclosure graph.

Note that the request graph G^r could represent an isomorphic *proper* subgraph of a satisfying disclosure graph G^d. This is due to the presence of atomic credentials and sensitive associations. The additional vertices in G^d represent additional information that is not needed for a successful access, but whose removal from G^d would result in a subgraph of G that does not represent a disclosure (Definition VI.1).

A request $\mathcal{R}$, represented by a set $G^{r_1},\ldots,G^{r_i}$ of request graphs, is satisfied by a disclosure graph G^d, if G^d satisfies a least one of the request graph (i.e., $\exists G^{r_j}$, $j = 1,\ldots,i$, such that $G^d \models G^{r_j}$).

Example VI.2. *Consider the disclosure graph G^d in Figure 4 and the request graphs G^{r_1} and G^{r_2} in Figure 2. It is easy to see that $G^d \models G^{r_1}$, since G^{r_1} is isomorphic to a subgraph of G^d. We have instead that $G^d \not\models G^{r_2}$ since the disclosure of one credential of type* cc *cannot satisfy both the terms with type=*cc *in G^{r_2}.*

Among all disclosure graphs that satisfy the server request, the client is interested in the one that minimizes disclosure of information. To this purpose, it is first necessary to guarantee that the disclosure graph is *minimal* with respect to the request (i.e., removing any of its vertices either does not satisfy the request or violates Definition VI.1). In other words, a disclosure graph G^d is minimal if there does not exist any disclosure graph $G^{d'}$, subgraph of G^d, which satisfies the request.

Definition VI.3 (Minimal Disclosure)**.** *Let G be a portfolio graph and $\mathcal{R}$ be a request. A disclosure graph $G^d(V^d_C \cup V^d_P \cup V^d_A, E^d_{CP} \cup E^d_{AP}, \lambda)$ is a* minimal disclosure *of G w.r.t. $\mathcal{R}$ iff:*

- *G^d satisfies $\mathcal{R}$;*
- *$\nexists$ a disclosure graph $G^{d'}$ of G such that $G^{d'}$ satisfies $\mathcal{R}$ and $V^{d'}_C \cup V^{d'}_P \cup V^{d'}_A \subset V^d_C \cup V^d_P \cup V^d_A$.*

The problem of computing a disclosure graph that satisfies a request and minimizes the sensitivity label can be formally defined as follows.

Problem VI.1 (Min-Disclosure)**.** *Given a portfolio graph G and a request $\mathcal{R}$, find a* minimum disclosure *graph $G^d(V^d_C \cup V^d_P \cup V^d_A, E^d_{CP} \cup E^d_{AP}, \lambda)$ of G w.r.t. $\mathcal{R}$ that satisfies the following requirements:*

- *G^d satisfies $\mathcal{R}$;*
- *$\nexists$ a disclosure graph $G^{d'}$ of G such that $G^{d'}$ satisfies $\mathcal{R}$ and $\lambda(G^{d'}) \prec \lambda(G^d)$.*

We note that any minimum disclosure graph is also a minimal disclosure graph, since the composition operator $\oplus$ defined for λ is monotonic, while the contrary is not true.

Example VI.3. *With reference to the request graph G^{r_1} in Figure 2(a), the disclosure graph G^d in Figure 4*

represents a minimal disclosure for G^{r_1}. In fact, the removal of any vertex v in $V_C^d \cup V_P^d \cup V_A^d$ would produce a subgraph $G^{d'}$ that either violates at least a condition in Definition VI.1, or does not satisfy G^{r_1}. However, G^d is not a minimum disclosure. We note that the first term in r_1, that is, id.{Name∧DoB}*, can be satisfied by disclosing either one of:* myIdCard, myPassport, *and* myStudentId. *The release of* myIdCard *discloses a property (i.e.,* Address*) that is not requested and exposes sensitive association a_1. The release of credential* myStudentId *reveals that the client is a student. This information is considered by the client more sensitive than the possession of an identity-card or a passport, as it is visible from the sensitivity labels of the corresponding vertices. The disclosure of (non-atomic) credential* myPassport *overcomes both the above limitations and is therefore preferred.*

The Min-Disclosure problem is NP-hard (the minimum cover problem reduces to it in polynomial time). It is therefore necessary to design heuristic approaches for solving the problem in polynomial time, even for relatively large credential portfolios. We discuss this aspect in the next section.

VII. Computing a minimal disclosure

Our solution models portfolios, requests, and disclosures as graphs. Also, we use graph isomorphisms to check if a disclosure graph G^d satisfies a given request $\mathcal{R}$, by checking if at least one of the request graphs G^r representing the simple requests in $\mathcal{R}$ is isomorphic to a subgraph of the disclosure graph G^d. It seems then natural to consider the problem of computing a minimal disclosure that satisfies a request as a problem of graph matching. However, our model has some peculiarities that cannot be simply handled by off-the-shelf graph matching algorithms. For instance, sensitive associations, which are not defined in the request graph, need to be considered a posteriori when a satisfying disclosure is found. Moreover, we also consider atomic credentials, meaning that a request for a certified property in the request graph can result in a credential disclosure that includes more properties than the ones requested.

We then designed and implemented a heuristic algorithm for computing a minimal disclosure that takes into account all these aspects. Our algorithm takes as input the portfolio graph G and a set of request graphs $G^{r_1},\ldots,G^{r_i}$ representing a request $\mathcal{R}$, and computes a minimal disclosure graph G^d for $\mathcal{R}$. The algorithm computes a minimal disclosure G^{d_j} for each G^{r_j} in $\mathcal{R}$ as follows. It initializes G^{d_j} as G, which corresponds to the disclosure of the whole portfolio. If G^{d_j} satisfies G^{r_j}, the algorithm evaluates the sensitivity label of the disclosure graphs obtained by removing from G^{d_j} either a property of a non-atomic credential or an atomic credential as a whole. Among the graphs obtained, the algorithm selects the one with minimum sensitivity that satisfies G^{r_j}, which becomes the new G^{d_j}. The process of reducing G^{d_j} by removing properties/credentials is repeated until a minimal disclosure is found (i.e., until the removal of any property/credential would result in a disclosure graph that does not satisfy G^{r_j}). Among all the minimal disclosures $G^{d_1}, \ldots, G^{d_i}$, the one with lowest sensitivity is returned.

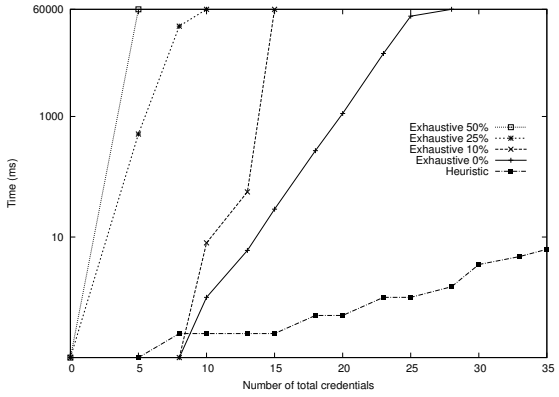


Figure 5. Execution time of the heuristic and the exhaustive algorithms

To assess the efficiency and effectiveness of our heuristic, both in terms of the quality of the computed solution and the execution time required for its computation, the algorithm has been implemented in C++. To compare the solution obtained by the heuristic with the optimum, we also implemented an exhaustive algorithm solving the Min-Disclosure problem (Problem VI.1). Experiments have been run on a PC with two Intel Xeon Quad 2.0GHz L3-4MB, 12GB RAM, four 1-Tbyte disks, and a Linux Ubuntu 9.04 operating system. A large variety of configurations have been tested operating on several parameters: the number of atomic and non-atomic credentials, the number of properties, the structure of the type/abstraction hierarchy, the number of sensitive associations, and the sensitivity of credentials, properties, and associations. Overall, the heuristic algorithm was able to produce the optimum in 98% of the cases, and when the optimum was not identified, the distance from the optimum was on average 13% above the optimum. Figure 5 compares the execution time of our heuristic with the execution time of the exhaustive algorithm, considering an increasing number of credentials (from 0 to 35) and 4 configurations obtained by assuming 50%, 25%, 10%, and 0% of the credentials to be non-atomic. As expected, the heuristic was always able to produce an answer in less than 10ms, whereas the exhaustive algorithm requires exponential time in the size of the

portfolio, with a strong dependence on the number of non-atomic credentials.

VIII. Related work

Research on credential-based access control (e.g., [2], [3], [9], [11], [12], [15]) primarily focused on solutions for controlling access to resources, for specifying and enforcing policies, and for enabling negotiation strategies, which may be indifferently adopted by the client and the server. Such solutions however do not allow the client to exploit the emerging technology (e.g., SAML [1], OpenID [8], and anonymous credentials [4], [5]) for determining which credentials and/or properties release to minimize the sensitive information communicated to the server. In fact, in the literature only few works have addressed this issue. Chen et al. [6] propose a solution that associates costs with credentials and policies to minimize the cost of a credential release within a trust-negotiation protocol. Similarly, Yao et al. [14] propose a point-based trust management model, where the client labels each credential in its portfolio with a quantitative privacy score, while the server defines a credit for each credential released by the client and a minimum threshold of credits to access a resource. The proposed solution finds an optimal set of client's credentials, such that the total privacy score of disclosed credentials is minimal and the server's access threshold is satisfied. Finally, Kärger et al. [10] propose a logic-based language for the specification of privacy preferences dictating a partial order among the client's attributes. All these solutions provide some treatment of preferences or scores associated with either credentials or properties, but do not address the problem of modeling the client portfolio. By including such a modeling, our work provides a generic setting of the problem and its solution, capturing emerging credential paradigms and the reasoning on the information released, taking into account sensitivity of properties, credentials and their existence, as well as sensitive associations of properties.

IX. Conclusions

An important long-term goal of the evolution of ICT technology is to combine the opportunities for the efficient access, exchange, storage, and dissemination of information, with an adequate level of user control over her own personal information. The approach presented in the paper provides a concrete solution that improves the support for the privacy requirements of the user when interacting in open scenarios. We believe approaches of this kind are going to be implemented in the Internet of the future, leading to the construction of systems allowing users to enjoy the benefits of emerging technology while maintaining awareness and control over their private information.

References

[1] A. Anderson and H. Lockhart. *SAML 2.0 profile of XACML*. OASIS, September 2004.

[2] C.A. Ardagna, S. De Capitani di Vimercati, S. Paraboschi, E. Pedrini, P. Samarati, and M. Verdicchio. Expressive and deployable access control in open Web service applications. *IEEE TSC*, 2010. (to appear).

[3] P. Bonatti and P. Samarati. A uniform framework for regulating service access and information release on the Web. *JCS*, 10(3):241–272, 2002.

[4] S. Brands. Rethinking public key infrastructure and digital certificates – building in privacy. *MIT Press*, 2000.

[5] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proc. of EUROCRYPT 2001*, Innsbruck, Austria, May 2001.

[6] W. Chen, L. Clarke, J. Kurose, and D. Towsley. Optimizing cost-sensitive trust-negotiation protocols. In *Proc. of INFOCOM 2005*, Miami, FL, USA, March 2005.

[7] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Combining fragmentation and encryption to protect privacy in data storage. *ACM TISSEC*, 2010. (to appear).

[8] D. Hardt, J. Bufu, and J. Hoyt. OpenID attribute exchange 1.0, 2007. http://openid.net/developers/specs/.

[9] K. Irwin and T. Yu. Preventing attribute information leakage in automated trust negotiation. In *Proc. of ACM CCS 2005*, Alexandria, VA, USA, November 2005.

[10] P. Kärger, D. Olmedilla, and W.T. Balke. Exploiting preferences for minimal credential disclosure in policy-driven trust negotiations. In *Proc. of SDM 2008*, Auckland, New Zealand, August 2008.

[11] A.J. Lee, M. Winslett, J. Basney, and V. Welch. The Traust authorization service. *ACM TISSEC*, 11(1):1–3, February 2008.

[12] T. Ryutov, L. Zhou, C. Neuman, T. Leithead, and K.E. Seamons. Adaptive trust negotiation and access control. In *Proc. of SACMAT 2005*, Stockholm, Sweden, June 2005.

[13] P. Samarati. Protecting respondents' identities in microdata release. *IEEE TKDE*, 13(6):1010–1027, November/December 2001.

[14] D. Yao, K.B. Frikken, M.J. Atallah, and R. Tamassia. Private information: To reveal or not to reveal. *ACM TISSEC*, 12(1):1–27, October 2008.

[15] T. Yu, M. Winslett, and K.E. Seamons. Supporting structured credentials and sensitive policies trough interoperable strategies for automated trust. *ACM TISSEC*, 6(1):1–42, February 2003.