DOI: 10.1504/IJIPSI.2012.046133

Minimizing disclosure of client information in credential-based interactions

Claudio A. Ardagna, Sabrina De Capitani di Vimercati, Sara Foresti

DTI - Università degli Studi di Milano,
26013 Crema - Italy
E-mail: *firstname.lastname*@unimi.it

Stefano Paraboschi

DIIMM - Università degli Studi di Bergamo,
24044 Dalmine - Italy
E-mail: parabosc@unibg.it

Pierangela Samarati

DTI - Università degli Studi di Milano,
26013 Crema - Italy
E-mail: pierangela.samarati@unimi.it

Abstract The advancements in ICT allow people to use and access resources and services on the Web anywhere and anytime. Servers offering resources typically require users to release information about them, which is then used to enforce possible access policies on the offered services. Effective access to such resources requires the development of approaches for enabling the user to organize and manage all her credentials and regulate their release when interacting with other parties over the Web.

In this paper, we provide a means for the user to specify how much she values the release of different properties, credentials, or combinations thereof as well as additional constraints that she might impose on information disclosure. Exploiting a graph modeling of the problem, the user can determine the credentials and properties to disclose to satisfy a server request while minimizing the sensitivity of the information disclosed. We develop a heuristic approach that shows execution times compatible with the requirements of interactive access to Web resources.

Keywords: privacy, credential, minimal disclosure, portfolio management.

Biographical notes:
Claudio A. Ardagna is an assistant professor at the Information Technology Department, Università degli Studi di Milano, Italy. He received the laurea and PhD degrees, both in computer science, from the Università degli Studi di Milano in 2003 and 2008, respectively. His research interests are in the area of information security, privacy, access control, mobile networks, and open source. He is the recipient of the ERCIM STM WG 2009 Award for the Best Ph.D. Thesis on Security and Trust Management. He has been program co-chair for WISTP 2011. The URL for his web page is http://www.dti.unimi.it/ardagna.

Sabrina De Capitani di Vimercati is a professor at the Information Technology Department, Università degli Studi di Milano, Italy. She received the Laurea and PhD degrees both in Computer Science from the Università degli Studi di Milano, Italy, in 1996 and 2001, respectively. Her research interests are in the area of information security, databases, and information systems. On these topics she has published more than 100 refereed technical papers in international journals and conferences. She has been an international fellow in the Computer Science Laboratory at SRI, CA (USA). She is member of the Steering Committees of the European Symposium on Research in Computer Security (ESORICS) and of the ACM Workshop on Privacy in the Electronic Society (WPES). She is vicechair of the IFIP WG 11.3 on Data and Application Security and Privacy. She is co-recipient of the ACM-PODS'99 Best Newcomer Paper Award. The URL for her web page is http://www.dti.unimi.it/decapita.

Sara Foresti is a post doc researcher at the Information Technology Department, Università degli Studi di Milano, Italy. She received the PhD in Computer Science from the Università degli Studi di Milano in April 2009. Her PhD thesis received the ERCIM STM WG 2010 award for the best PhD thesis on security and trust management in a European University. Her research interests are in the area of data security and privacy, with particular consideration of access control and information protection in the data outsourcing scenario. She has been/will be program co-chair for DBSec 2010 and ESORICS 2012. The URL for her web page is http://www.dti.unimi.it/foresti.

Stefano Paraboschi obtained the Laurea degree in Electronic Engineering in 1990 and a PhD in Computer Science in 1994, both from Politecnico di Milano, Italy. In 1996 he became an assistant professor and in 1998 an associate professor, both positions at Politecnico di Milano, Italy. Since 2002 he is a full professor at the School of Engineering of the Università di Bergamo, where he is the chair of the Computer Engineering Program. In 2003 he became a founding member and deputy-chair of the Dipartimento di Ingegneria Gestionale e dell'Informazione of the Università di Bergamo (now Dipartimento di Ingegneria dell'Informazione e Metodi Matematici). He spent research periods at the Department of Computer Science of Stanford University (host: Hector Garcia Molina), at the IBM Almaden Research Center in San Jose (host: Jennifer Widom), and at the Center for Secure Information Systems of George Mason University (host: Sushil Jajodia). His research has focused on active databases, multidimensional databases, workflow management systems, Web technologies, computer, and information security. The URL for his web page is http://cs.unibg.it/parabosc.

Pierangela Samarati is a professor at the Information Technology Department, Università degli Studi di Milano, Italy. Her main research interests are in data protection, access control models, and information privacy and security. She has published more than 170 papers in international journals and conferences. She has been a computer scientist at SRI International, CA (USA) and a visiting researcher at Stanford University, CA (USA), and George Mason University, VA (USA). She is the chair of the IEEE Technical Committee on Security and Privacy in Complex Information Systems (TCSPCIS) and of the Steering Committees of the European Symposium on Research in Computer Security, and of the ACM Workshop on Privacy in the Electronic Society. She is the Coordinator of the Working Group on Security of the Italian Association for Information Processing (AICA), the Italian representative in the IFIP (International Federation for Information Processing) Technical Committee 11 (TC-11) on "EDP Security". She is a member of the Steering Committee of: ACM Symposium on InformAtion, Computer and Communications Security (ASIACCS), International Conference on Information Systems Security (ICISS), and International Conference on Information and Communications Security (ICICS). In 2009, she has been named ACM Distinguished Scientist. She has served as General Chair, Program Chair, and program committee member of several international conferences. The URL for her web page is http://www.dti.unimi.it/samarati.

1 Introduction

The development in the past years of the Internet and associated Web technology has produced a large impact on society. Still, both 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., [5, 12, 22]) 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 stored in a client portfolio.

The support for client-side solutions to regulate credential 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 [8, 11, 20] and domain of such information. In this paper, we respond to this need and propose an approach for empowering the user with the ability to specify privacy preferences on her private information. Our organization of the client portfolio accommodates emerging credential technologies (e.g., anonymous credentials), supporting selective release of properties within certificates. Our model allows a user to assign sensitivity labels and context restrictions (expressing both the domain and the transactions for which they can be used) to properties, credentials, or combinations thereof in her portfolio. Basically, sensitivity labels provide a quantitative estimation of the privacy value of the different elements of the portfolio, as perceived by the user. Expressing sensitivity labels, our approach allows for minimizing information release. Context restrictions limit the disclosure of properties and credentials within a set of interactions. We characterize the disclosure of information in a portfolio in terms of a 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. Finally, we describe our heuristics, exploiting the graphical representation of the model components, for computing a solution to the problem and present experimental results that confirm that the approach is applicable to interactive scenarios. In [2] we presented an early version of our proposal that here has been extended to possibly consider dependencies among properties and credentials that reduce the sensitivity label of combinations of portfolio elements, and disclosure constraints representing sets of properties and credentials whose disclosure should be prohibited. Also, the model has been enriched with the definition of context restrictions complementing sensitivity labels, which allow the user to restrict the release of information to those elements in the portfolio considered relevant for a given interaction. The graph modeling of the portfolio and of a disclosure has been refined to include both context restrictions and disclosure constraints. We further extended our solution enabling the client to consider previous releases within a generic time window in the computation of a minimal disclosure. Finally, the heuristic algorithm originally proposed in [2], which determines a minimal satisfying disclosure for a given request, has been changed to consider context restrictions and disclosure constraints.

The remainder of this paper is organized as follows. Section 2 introduces basic concepts related to the modeling of the client portfolio, which also includes emerging paradigms for producing and presenting credentials. Section 3 illustrates a graph-based modeling of the client portfolio. Section 4 defines server requests and its graph-based modeling. Sections 5 and 6 illustrate how users can specify privacy preferences and additional requirements on the different elements of their portfolio. Section 7 extends the graph modeling of the portfolio to include the concepts described in Sections 5 and 6. Section 8 characterizes the disclosure of information in a portfolio in terms of our graph-based model. Also, the problem of determining a disclosure satisfying a server request while minimizing the sensitivity of released information has been modeled as the problem of determining a minimum isomorphic graph matching. Section 9 discusses how our solution can accommodate communications characterized by a sequence of requests by a server. Section 10 describes our heuristic algorithm for computing a minimal disclosure satisfying a given request and presents experimental results proving its efficiency and effectiveness. Section 11 discusses related work. Finally, Section 12 presents our conclusions.

2 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)

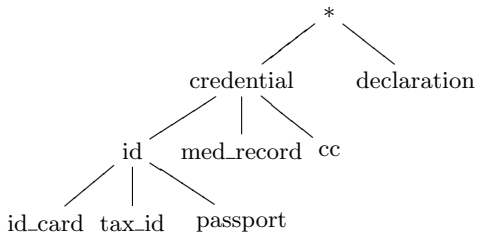


Figure 1 An example of hierarchy of credential types

properties that the client can utter. Like in the literature [5], 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 $\mathcal{H}_T$ of credential types is a pair $(\mathcal{T},\succeq_T)$, where $\mathcal{T}$ is the set of all types, and $\succeq_T$ is a partial order relationship over $\mathcal{T}$. Given two types t_i and t_j in $\mathcal{T}$, $t_i \succeq_T t_j$ if t_i is an abstraction of t_j. For instance, *id* is an abstraction of credential types *id_card*, *tax_id*, and *passport* (i.e., $id \succeq_T id\_card$, $id \succeq_T tax\_id$, and $id \succeq_T passport$). Figure 1 illustrates an example of hierarchy of credential types. Each credential is then 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 instance, 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., *id_card* 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 of type *declaration*, whose identifier is *decl*, containing all the properties of the portfolio.

3 Client portfolio

We enrich the client portfolio with novel concepts, enabling 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 (*cc*) will all refer to their specific credit card number, and therefore to a different piece of information. Credential-dependent properties might have different occurrences, one for each credential including them.

- *Atomic vs non-atomic credentials.* X.509 is today the most common kind of credentials used in distributed systems. 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 a portion of the credential content. Instead, modern credential technology (e.g., UProve and Idemix [6, 7]) supports the release of individual properties extracted from the credential. Our model includes this aspect of modern credential technology and classifies 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 properties, credentials, and associations among them, to the aim of minimizing the 'amount' of information released for acquiring a service. We discuss portfolio sensitivity in Section 5.

- *Context restrictions*. Information in the client portfolio is naturally characterized by a domain (e.g., medical, financial). Each interaction between a client and a server is characterized by a context, representing the domain of the server and the

transaction. A user may want to restrict the release of credentials and properties in her portfolio based on the server domain and on the transaction they are performing. As an example, the credit card should only be released to financial servers when the user is paying for a service or a product. We provide a way for the user to specify for each credential, property, or combination thereof the set of domains and transactions where they can be released. We discuss context restrictions in Section 6.

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

Definition 3.1 (Portfolio Graph) *Let $\mathcal{C}$ and $\mathcal{P}$ be a set of credentials and properties, respectively, in a client portfolio. The* portfolio graph *$G(V_C \cup V_P, E_C)$ is a bipartite graph having a vertex for each credential in $\mathcal{C}$ 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 by one vertex (connected to all credentials where it is contained). Each credential-dependent property is instead represented with several vertices, one for each credential where it appears, that is, one for each of its instantiations.

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

Example 3.1 *Figure 2 illustrates an example of a portfolio graph, including credentials* myId (*of type* id_card), myTaxId (*of type* tax_id), myPassport (*of type* passport), myDialysis (*of type* med_record), *and* myVISA *and* myAMEX (*both of type* cc). *Credentials* myPassport *and* myDialysis *are the only non-atomic credentials in the portfolio. Properties* `Name`, `DoB`, `SSN`, `Country`, `BloodType`, `Phone`, *and* `NickName` *are credential-independent, while* `CCNum` *is credential-dependent* (*having a different occurrence for each credit card*).

4 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 server 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 a 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 $type \succeq_T type(c)$ (i.e., the type of credential *c* is a specialization of the type in the request). Different terms must be certified by different credentials.

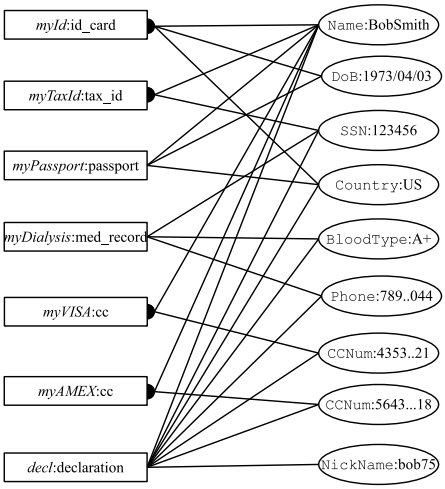


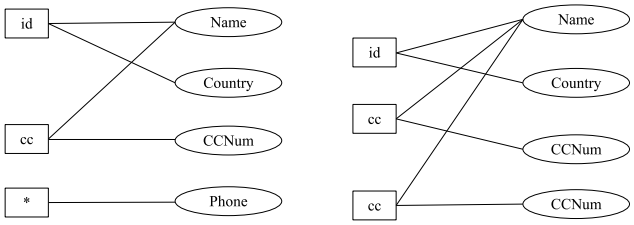
Figure 2 An example of portfolio graph

Example 4.1 *A request* $\mathcal{R} = r_1 \vee r_2$, *with* $r_1 = (\text{id.}\{\texttt{Name},\texttt{Country}\} \wedge \text{cc.}\{\texttt{Name},\texttt{CCNum}\} \wedge *.\{\texttt{Phone}\})$ *and* $r_2 = (\text{id.}\{\texttt{Name},\texttt{Country}\} \wedge \text{cc.}\{\texttt{Name},\texttt{CCNum}\} \wedge \text{cc.}\{\texttt{Name},\texttt{CCNum}\})$ *can be satisfied in two different ways. The first possibility discloses:* i) *properties* `Name` *and* `Country` *from a credential of type* id*;* ii) *a credit card; and* iii) *property* `Phone` *from any credential. The second possibility discloses:* i) *properties* `Name` *and* `Country` *from a credential of type* id*; and* ii) *properties* `Name` *and* `CCNum` *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 vertex *type* with the vertices representing $p_1,\ldots,p_m$. A request graph is formally defined as follows.

Definition 4.1 (Request Graph) *Let r be a simple request in a request $\mathcal{R}$. The* request graph *$G^r(V^r_T \cup V^r_P, E^r_T)$ of r is a bipartite graph having a vertex for each term and each property in it, and an edge connecting each term to the corresponding properties.*

In the following, when clear from the context, we will call request either a request $\mathcal{R}$ or a simple request *r* in



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

Figure 3 An example of request graphs

$\mathcal{R}$. Figure 3 illustrates the request graphs for requests r_1 and r_2 described in Example 4.1.

5 Portfolio sensitivity

The motivation of our work is to provide the client with an intuitive and easily manageable approach for regulating the disclosure of her portfolio. As a matter of fact, when the server offers choices on the properties or credentials to be provided, the client may prefer to disclose some over others. For instance, a user may prefer to release her email over her address and either of the two instead of her phone number. Intuitively and naturally, users perceive a value associated with their information that reflects the sensitivity of this information. Different properties and different credentials enjoy therefore different values in this respect.

5.1 Sensitivity labels

We express information privacy preferences as *sensitivity labels* that the user can associate with the different elements (or combinations thereof) in her portfolio and reflect how much the user values their disclosure.

In principle the set of sensitivity labels could be any set of values, provided the existence of a (partial) order relationship over them and a composition operator $\oplus$ that determines the label resulting from the combination of two labels. 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) 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 this paper, we assume the set of labels to be the set $\mathbb{Z}$ of (positive and negative) integer values, the dominance relationship to be the $\geq$ total order relationship, and the composition operator $\oplus$ to be the sum $+$ of values.

We now illustrate how the user can specify the sensitivity of the elements of her portfolio via a labeling function $\lambda : 2^{\mathcal{C} \cup \mathcal{P}} \rightarrow \mathbb{Z}$, associating a sensitivity label with individual elements (properties or credentials) as well as combinations of them. Also, we model further constraints that the user might want to specify on the disclosure of information in her portfolio.

5.2 Sensitivity of properties and credentials

The first step for the user to specify how much she values information in her portfolio is to associate a sensitivity label with each property p and credential c.

- $\lambda(p)$: 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. The more sensitive the information, the greater the sensitivity value associated with it. For instance, if property `SSN` is considered more sensitive than `Name`, then $\lambda(\texttt{SSN}) > \lambda(\texttt{Name})$.
- $\lambda(c)$: defines the sensitivity of the *existence* of a credential. This is the additional information carried by the credential itself, regardless of the information contained in it. For instance, consider a dialysis certificate including properties `SSN`, `BloodType`, and `Phone`. The certificate sensitivity is greater than the composition of the labels of the properties in it. In fact, the existence of the certificate itself has a sensitivity that goes beyond its properties. Note that, for non-atomic credentials, $\lambda(c)$ reflects the sensitivity assigned to the existence of the credential regardless of the release of the properties within it.

5.3 Sensitivity of associations

The release of a set of elements entails a sensitivity corresponding to the combination of the sensitivity of all the elements involved. There are however cases where releasing together some elements might produce an information release whose sensitivity does not precisely correspond to the composition of the labels of the individual elements. In such cases, we allow the user to specify an additional (positive or negative) sensitivity label that should be considered in computing the sensitivity of the set of elements jointly released. Let A be any set of properties and/or credentials. The user can specify $\lambda(A)$ as the *additional*, positive or negative, sensitivity to take into account in computing the sensitivity label of an association among the portfolio elements in A, with the following semantics.

- *Sensitive views* ($\lambda(A) > 0$). They reflect the fact that a set of portfolio elements jointly released carries *more* information than the composition (i.e.,

the sum) of the labels of the individual elements. For instance, the association between the `Name` of a user and her certificate *myDialysis* can be considered more sensitive than the composition of the sensitivity labels of the two. In fact, not only it discloses the user name and the existence of a dialysis certificate, but also the fact that they are in association (i.e., the name of the user suffering from dialysis problems). $\lambda(\{\texttt{Name}, myDialysis\})$ expresses the *additional* sensitivity of the information when joined.

- *Dependencies* ($\lambda(A) < 0$). They reflect the fact that a set of portfolio elements jointly released carries *less* information than the composition (i.e., the sum) of the labels of the individual elements. For instance, the association between `Phone` and `Country` can be considered less sensitive than the sum of the sensitivity labels of the two. As a matter of fact, the information carried by the phone number permits to determine the country where the user lives. Hence, releasing the phone number together with the country does not release additional information. $\lambda(\{\texttt{Phone}, \texttt{Country}\})$ expresses the sensitivity to be *removed* when the two properties are joined. In this specific example, there is essentially a functional dependency between the two elements and therefore $\lambda(\{\texttt{Phone}, \texttt{Country}\}) = -\lambda(\texttt{Country})$. In general, $\lambda(A)$ can take any negative value, provided that its absolute value be at most equal to the sensitivity label of the most sensitive element in A. Formally, $|\lambda(A)| \leq max\{\lambda(el) : el \in A\}$.

We note that the user could explicitly define the sensitivity label of an association (specifying the *additional* sensitivity given by the joint release of the involved elements), or could have it derived based on the difference between the overall sensitivity she perceives for the association and the sensitivity of the individual properties and/or credentials involved. For instance, with reference to the portfolio in Figure 2, assume that $\lambda(\texttt{Name})=1$ and $\lambda(myDialysis)=15$. If the sensitivity associated with the combined release of `Name` and *myDialysis* is equal to 50, the sensitivity label of the association $\lambda(\{\texttt{Name}, myDialysis\})$ is therefore $50-\lambda(\texttt{Name})-\lambda(myDialysis) = 34$.

5.4 Disclosure constraints

In addition to specifying the sensitivity labels of credentials, properties, and associations in the portfolio, the user may want to specify additional constraints that cannot be simply expressed with a sensitivity label. Some associations of the portfolio elements in fact might be not only much more sensitive than the combination of the labels of the elements, but should be definitely prohibited by the user, meaning that the user never wants to release some information in association. We accommodate this requirement by allowing the specification of

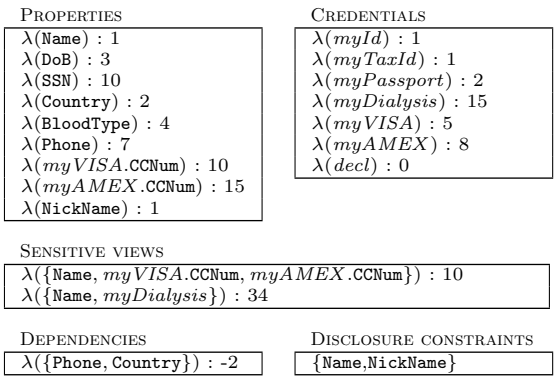


Figure 4 Sensitivity specification for the portfolio in Figure 2

disclosure constraints as a set of properties and/or credentials that should never be released together. For instance, a user might have in her portfolio both a `Name` and a `NickName`, each one with a sensitivity label (to be considered when the element is released), but their association should never be disclosed.

Example 5.1 *Figure 4 illustrates an example of a labeling function λ for the portfolio illustrated in Figure 2. The figure also reports two sensitive views* {`Name`,myVISA.`CCNum`,myAMEX.`CCNum`}*, since the combined release of the specified elements could permit a server to link different credit cards to the same user, and* {`Name`, myDialysis}*, since the combined release of the specified elements could result in unauthorized inference on sensitive medical information; one dependency* {`Phone`, `Country`}*, since the knowledge of the phone number also discloses the country; and one disclosure constraint* {`Name`,`NickName`}*, since the user does not want to release both her name and her nickname.*

6 Context restrictions

A user may want to specify which properties/credentials in the portfolio can be disclosed, on the basis of the domain of both the server and the information in the portfolio. For instance, a user may want to release her dialysis certificate to a hospital, while forbidding its disclosure to a pharmaceutical company. The user should then define as many views on her portfolio as the number of servers she interacts with. However, this practice is complex and difficult to manage for a user, who can easily lose control, since the number of servers in the system could be huge. Intuitively, users may classify servers in categories (e.g., government, medical) that reflect their functional and/or competence areas. All the servers in a category enjoy the same view on the client portfolio. Even when interacting with servers in the same category, users may want to further restrict access to their data depending on the transaction executed (e.g., payment, reservation, diagnosis consult). As an example, the user may restrict

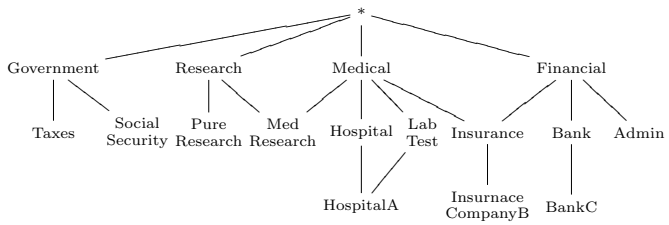


Figure 5 An example of hierarchy of categories

the release of her credit card to financial servers in the context of payment transactions only.

6.1 Categories and transactions

Each property, credential, and association in the client portfolio is labeled with a set of *context restrictions* consisting of a category and a set of transactions, defined as follows.

- *Categories* (Γ): define the functional, or competence, areas of both servers and portfolio elements. Categories implicitly determine the set of servers who can access a given property/credential in the client portfolio (i.e., the servers whose category is more specific than the one of the information). Each user defines her own set of categories, based on her privacy requirements, modeling different groups of servers in the system. We assume, the set Γ of categories defined by the user to be organized in a hierarchy $\mathcal{H}_\Gamma(\Gamma,\succeq_\Gamma)$, where $\succeq_\Gamma$ is a partial order relationship defined over Γ. Given two categories γ_i and γ_j in Γ, we say that γ_i dominates γ_j, denoted $\gamma_i \succeq_\Gamma \gamma_j$, if γ_i represents a category broader than γ_j. We use $*$ to denote the top category in the hierarchy; $*$ represents all the categories defined by the user. For instance, Figure 5 illustrates an example of hierarchy of categories, where category *Financial* is broader than category *Insurance*, *Bank*, and *Admin* (i.e., *Financial*$\succeq_\Gamma$*Insurance*, *Financial*$\succeq_\Gamma$*Bank*, and *Financial*$\succeq_\Gamma$*Admin*). We note that our model can support user-specific categories that refer to one server only (e.g., *HospitalA*, *InsuranceCompanyB*, and *BankC*). Only the servers that belong to a category that is dominated by the one of the property/credential can access it.

- *Transactions* ($\mathcal{K}$): define the set of interactions where a property, credential, or association can be released. For instance, if the client interacts with a server to make a flight reservation, the transaction can be labeled as *reservation* and *payment*. As another example, if the client interacts with the same server to access flight information, the transaction can be labeled as *browsing*. As for categories, each user defines her own unordered set of transactions $\mathcal{K}$. Notation $*$ is used to represent the set of all transactions.

Formally, a context restriction $ctx \in \Gamma \times 2^{\mathcal{K}}$ is defined as a pair $\langle\gamma,\{k_1,\ldots,k_n\}\rangle$, where $\gamma\in\Gamma$ is a category, and $\{k_1,\ldots,k_n\}\subseteq\mathcal{K}$ is a set of transactions. We define a dominance relationship between pairs of sets of context restrictions. A set Ctx_i of context restrictions dominates a set Ctx_j of context restrictions if, for each context restriction ctx_j in Ctx_j, there exists at least a context restriction ctx_i in Ctx_i such that the category in ctx_i dominates the category in ctx_j and the set of transactions in ctx_i is a superset of the one in ctx_j. Formally, this dominance relationship can be defined as follows.

Definition 6.1 (Dominance) *Let Γ be a set of categories with partial order relationship $\succeq_\Gamma$, $\mathcal{K}$ be a set of transactions, Ctx_i and Ctx_j be two sets of context restrictions. Ctx_i* dominates *Ctx_j, denoted $Ctx_i \succeq_{ctx} Ctx_j$, iff $\forall ctx_j=\langle\gamma_j,\{k_{j,1},\ldots,k_{j,n}\}\rangle\in Ctx_j$, $\exists ctx_i=\langle\gamma_i,\{k_{i,1},\ldots,k_{i,m}\}\rangle\in Ctx_i \implies \gamma_i\succeq_\Gamma\gamma_j$ and $\{k_{i,1},\ldots,k_{i,m}\}\supseteq\{k_{j,1},\ldots,k_{j,n}\}$.*

Two sets of context restrictions Ctx_i and Ctx_j are said to be *incomparable* iff neither $Ctx_i\succeq_{ctx}Ctx_j$ nor $Ctx_j\succeq_{ctx}Ctx_i$.

Example 6.1 *Consider two sets of context restrictions* Ctx_i = {⟨Government,{payment,services}⟩, ⟨Financial,{payment,services}⟩}, Ctx_j={⟨Bank,{payment}⟩}. *Ctx_i dominates Ctx_j since, for the unique context restriction* ⟨Bank,{payment}⟩ *in Ctx_j, there exists a context restriction* ⟨Financial,{payment,services}⟩ *in Ctx_i such that* Financial$\succeq_\Gamma$Bank *and* {payment,services}$\supseteq${payment}.

We introduce a labeling function $\phi : \mathcal{C}\cup\mathcal{P}\cup\mathcal{A} \rightarrow 2^{\Gamma\times 2^{\mathcal{K}}}$ that associates with each property, credential, and association in the client portfolio a set $Ctx=\{ctx_1,\ldots,ctx_n\}$ of context restrictions. As an example, $\phi(myTaxId)$ = {⟨*Government*,{*payment*,*services*}⟩, ⟨*Financial*,{*payment*,*services*}⟩}. Note that the set of context restrictions of atomic credentials and associations should be coherent with the set of context restrictions of their elements. In particular, the set of context restrictions associated with an atomic credential should be dominated by the set of context restrictions associated with each of the properties it certifies. Formally, given an atomic credential c and the set of properties $\{p_1,\ldots,p_n\}$ it certifies, $\forall p_i\in\{p_1,\ldots,p_n\}$, $\phi(p_i)\succeq_{ctx}\phi(c)$, meaning that all the properties certified by c will be relevant at least for the same set of context restrictions as c. On the contrary, since a property can also be declared, its set of context restrictions can also be larger than the one of the atomic credential certifying it. Similarly, the joint release of properties and credentials composing an association is permitted only within the context restrictions defined for the association. Formally, given an association A, $\forall el\in A$, $\phi(el)\succeq_{ctx}\phi(A)$, meaning that all the properties and credentials that belong to A must be labeled with at least the same set of

context restrictions as A. The declaration *decl* is always associated with $\phi(decl)=\{\langle *,*\rangle\}$, which dominates any possible set of context restrictions.

Finally, the client classifies each request from the server with a context restriction $\phi(\mathcal{R})=\{\langle\gamma,\{k\}\rangle\}$, where $\gamma\in\Gamma$ is the category associated with the requesting server and $k\in\mathcal{K}$ is the transaction. We note that each request is associated with exactly one category and one transaction. Also, for each simple request r in $\mathcal{R}$, $\phi(r)=\phi(\mathcal{R})$. A server request $\mathcal{R}$ can be satisfied only by releasing properties and credentials that are relevant for $\mathcal{R}$, that is, whose set of context restrictions dominate $\phi(\mathcal{R})$. Intuitively, the context restriction of a request $\mathcal{R}$ induces a view over the portfolio graph, which includes properties, credentials, and associations that can be possibly released to satisfy $\mathcal{R}$.

7 Extended portfolio graph

Our graph modeling of the client portfolio can easily be extended to represent sensitivity labels, associations, disclosure constraints, and context restrictions. We add a vertex for each association and each disclosure constraint and an edge connecting each association and disclosure constraint with the involved properties and/or credentials. We then define a labeling function ℓ that assigns to each vertex in the portfolio graph representing a property, a credential, or an association both a sensitivity label and a set of context restrictions. The labeling function ℓ is therefore defined as the composition of labeling functions λ and ϕ, which associate a sensitivity label and a set of context restrictions, respectively, with each property, credential, and association in the portfolio. The label ℓ of an element el in the portfolio (i.e., a property p, a credential c, or an association A) is of the form $[\lambda(el),\phi(el)]$, where $\lambda(el)$ is the sensitivity label of the property and $\phi(el)$ is a set of context restrictions. As an example, labeling $\ell(myVISA)=[15,\{\langle Financial,\{payment,reservation\}\rangle\}]$ means that credential *myVISA* has sensitivity label equal to 15 and can be released only in the context of *payment* or *reservation* transactions to servers in the *Financial* category. Formally, the definition of portfolio graph (Definition 3.1) is extended as follows.

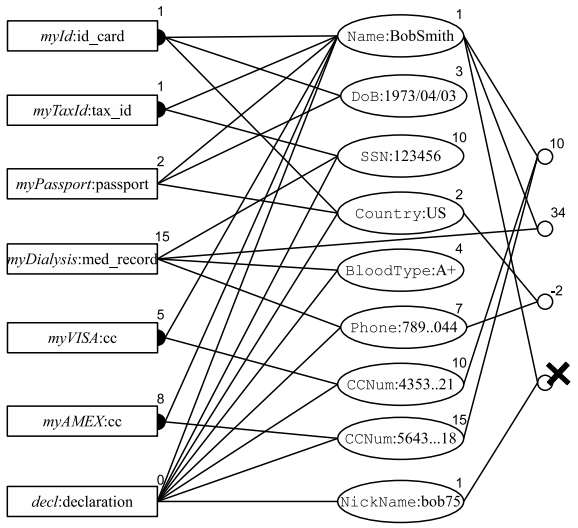
Definition 7.1 (Portfolio Graph – Extended) *Let $G(V_C\cup V_P,E_C)$ be a portfolio graph, $\mathcal{A}$ and $\mathcal{L}$ be a set of associations and disclosure constraints over $\mathcal{P}\cup\mathcal{C}$, respectively. A* portfolio graph *extended by associations and disclosure constraints $G(V_C\cup V_P\cup V_A\cup V_L,E_C\cup E_A\cup E_L,\ell)$ is a labeled graph, having an additional vertex for each association in $\mathcal{A}$ and each disclosure constraint in $\mathcal{L}$, and an additional edge connecting each association and disclosure constraint to the properties and credentials contained in it. The labeling function ℓ assigns a label $\ell(v)$ to each vertex $v\in(V_C\cup V_P\cup V_A)$, corresponding to $[\lambda(v),\phi(v)]$.*

Our definition of labeling function ℓ assumes that each credential, property, and association has a single sensitivity label that is independent from its context restrictions. Basically, sensitivity labels model how much the client values the release of her information and this evaluation is not related to the domain of the information itself. The definition of ℓ as the composition of λ and ϕ functions provides high flexibility and modularity. In fact, a client can choose to adopt sensitivity labels only, context restrictions only, or their combination. We note that our model can be simply extended to permit the client to define different sensitivity labels (one for each context restriction) for the same property, credential, and association. In this case, it is necessary to refer to traditional conflict resolution policies to manage inconsistencies among multiple sensitivity labels that may apply to the same server request [15].

Example 7.1 *Consider the portfolio graph in Figure 2, the sensitivity labels, associations, and disclosure constraints in Figure 4, and the hierarchy of categories in Figure 5. Figure 6 illustrates an example of an extended portfolio graph. For simplicity, sensitivity labels are indicated next to the vertices, while the complete labeling function ℓ of the graph, including also context restrictions, is reported in Figure 6(b).*

8 Disclosure modeling

A disclosure represents a subset of the client portfolio, which is communicated to the server for satisfying a request $\mathcal{R}$. 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, and disclosure constraints that are violated by it. 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 (or a declaration) certifying them, additional properties included in atomic credentials, and sensitive associations. 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 and credentials forming a sensitive association are disclosed, the sensitive association must be considered disclosed (*association exposure*); *4)* if all properties and credentials forming a disclosure constraint are disclosed, the disclosure constraint must be considered violated (*constraint violation*). These properties are captured by the following definition of disclosure graph.



(a)

PROPERTIES

ℓ(Name):	[1, ⟨∗,∗⟩]
ℓ(DoB):	[3, ⟨∗,∗⟩]
ℓ(SSN):	[10, ⟨*Government*,{*payment*,*services*}⟩, ⟨*Financial*,{*payment*,*services*}⟩, ⟨*Medical*,{*treatment*,*diagnosis*}⟩, ⟨*InsuranceCompanyA*,{*services*}⟩]
ℓ(Country):	[2, ⟨∗,∗⟩]
ℓ(BloodType):	[4, ⟨*Medical*,{*treatment*,*diagnosis*}⟩, ⟨*InsuranceCompanyA*,{*services*}⟩]
ℓ(Phone):	[7, ⟨∗,∗⟩]
ℓ(*myVISA*.CCNum):	[10, ⟨∗,{*payment*}⟩, ⟨*BankC*,∗⟩]
ℓ(*myAMEX*.CCNum):	[15, ⟨∗,{*payment*}⟩, ⟨*BankC*,∗⟩]
ℓ(NickName):	[1, ⟨∗,∗⟩]

CREDENTIALS

ℓ(*myId*):	[1, ⟨∗,∗⟩]
ℓ(*myTaxId*):	[1, ⟨*Government*,{*payment*,*services*}⟩, ⟨*Financial*,{*payment*,*services*}⟩]
ℓ(*myPassport*):	[2, ⟨∗,∗⟩]
ℓ(*myDialysis*):	[15, ⟨*Medical*,{*treatment*,*diagnosis*}⟩, ⟨*InsuranceCompanyA*,{*services*}⟩]
ℓ(*myVISA*):	[5, ⟨∗,{*payment*}⟩, ⟨*BankC*,∗⟩]
ℓ(*myAMEX*):	[8, ⟨∗,{*payment*}⟩, ⟨*BankC*,∗⟩]
ℓ(*decl*):	[0, ⟨∗,∗⟩]

SENSITIVE VIEWS

ℓ({Name,*myVISA*.CCNum,*myAMEX*.CCNum}):	[10, ⟨*BankC*,∗⟩]
ℓ({Name,*myDialysis*}):	[34, ⟨*HospitalA*,{*treatment*,*diagnosis*}⟩]

DEPENDENCIES

ℓ({Phone,Country}):	[−2, ⟨∗,∗⟩]

(b)

Figure 6 Portfolio graph in Figure 2 extended with the sensitivity specifications in Figure 4 and context restrictions

Definition 8.1 (Disclosure Graph) *Let* $G(V_C \cup V_P \cup V_A \cup V_L, E_C \cup E_A \cup E_L, \ell)$ *be a portfolio graph. A subgraph* $G^d(V^d_C \cup V^d_P \cup V^d_A \cup V^d_L, E^d_C \cup E^d_A \cup E^d_L, \ell)$ *of* G *where* $V^d_C \subseteq V_C$, $V^d_P \subseteq V_P$, $V^d_A \subseteq V_A$, $V^d_L \subseteq V_L$, $E^d_C \subseteq E_C$, $E^d_A \subseteq E_A$, *and* $E^d_L \subseteq E_L$ *is a* disclosure graph *iff the following properties hold:*

1. $v_p \in V^d_P \Longrightarrow \exists\ v_c \in V^d_C$ *s.t.* $(v_c, v_p) \in E^d_C$;
2. $v_c \in V^d_C$ *s.t. credential* v_c *is atomic* $\Longrightarrow \forall v_p \in V_P$: $(v_c, v_p) \in E_C$, $v_p \in V^d_P$ *and* $(v_c, v_p) \in E^d_C$;
3. $v_a \in V_A$ *s.t.* $\forall (v_a, v) \in E_A$, $v \in V^d_P \cup V^d_C \Longrightarrow v_a \in V^d_A$ *and* $(v_a, v) \in E^d_A$;
4. $v_l \in V_L$ *s.t.* $\forall (v_l, v) \in E_L$, $v \in V^d_P \cup V^d_C \Longrightarrow v_l \in V^d_L$ *and* $(v_l, v) \in E^d_L$.

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 (4, resp.) states that if all the vertices representing the properties and credentials composing an association (disclosure constraint, resp.) belong to the disclosure graph, then also the vertex representing the association (disclosure constraint, resp.) and the edges between the association (disclosure constraint, resp.) and the involved properties and credentials belong to the graph.

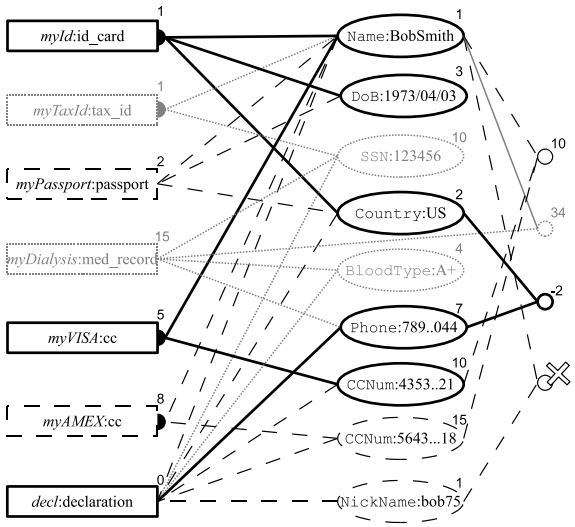
We note that not all the disclosures can be communicated to the server, since context restrictions and disclosure constraints limit their release. A disclosure is *legitimate* with respect to a set of context restrictions *Ctx* if: *i)* it includes only vertices characterized by context restrictions that dominate *Ctx*, and *ii)* it does not violate any disclosure constraint. Formally, a legitimate disclosure is defined as follows.

Definition 8.2 (Legitimate Disclosure Graph) *Let* $G^d(V^d_C \cup V^d_P \cup V^d_A \cup V^d_L, E^d_C \cup E^d_A \cup E^d_L, \ell)$ *be a disclosure graph and Ctx a set of context restrictions.* G^d *is a* legitimate disclosure graph *w.r.t. Ctx iff:*

- $\forall v \in (V^d_C \cup V^d_P \cup V^d_A)$, $\phi(v) \succeq_{ctx} Ctx$;
- $V^d_L = \emptyset$.

We note that only legitimate disclosures w.r.t. a set of context restrictions *Ctx* can be released to satisfy a request characterized by these disclosure restrictions. 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 $G^d(V^d_C \cup V^d_P \cup V^d_A \cup V^d_L, E^d_C \cup E^d_A \cup E^d_L, \ell)$, the sensitivity $\lambda(G^d)$ of the disclosure graph is $\lambda(G^d) = \sum_v \lambda(v)$, with v in $V^d_C \cup V^d_P \cup V^d_A$.

Example 8.1 *Figure 7 represents an example of a legitimate disclosure graph* G^d *w.r.t. context restrictions* $Ctx = \{\langle \text{Medical}, \{\text{payment}\}\rangle\}$ *for the portfolio graph in Figure 6. In the figure, the vertices and edges in the portfolio graph that also belong to the disclosure graph are represented with a bold black line, the*



(a)

PROPERTIES

ℓ(Name):	[1, ⟨∗,∗⟩]
ℓ(DoB):	[3, ⟨∗,∗⟩]
ℓ(SSN):	[10, ⟨*Government*,{*payment*,*services*}⟩, ⟨*Financial*,{*payment*,*services*}⟩, ⟨*Medical*,{*treatment*,*diagnosis*}⟩, ⟨*InsuranceCompanyA*,{*services*}⟩]
ℓ(Country):	[2, ⟨∗,∗⟩]
ℓ(BloodType):	[4, ⟨*Medical*,{*treatment*,*diagnosis*}⟩, ⟨*InsuranceCompanyA*,{*services*}⟩]
ℓ(Phone):	[7, ⟨∗,∗⟩]
ℓ(*myVISA*.CCNum):	[10, ⟨∗,{*payment*}⟩, ⟨*BankC*,∗⟩]
ℓ(*myAMEX*.CCNum):	[15, ⟨∗,{*payment*}⟩, ⟨*BankC*,∗⟩]
ℓ(NickName):	[1, ⟨∗,∗⟩]

CREDENTIALS

ℓ(*myId*):	[1, ⟨∗,∗⟩]
ℓ(*myTaxId*):	[1, ⟨*Government*,{*payment*,*services*}⟩, ⟨*Financial*,{*payment*,*services*}⟩]
ℓ(*myPassport*):	[2, ⟨∗,∗⟩]
ℓ(*myDialysis*):	[15, ⟨*Medical*,{*treatment*,*diagnosis*}⟩, ⟨*InsuranceCompanyA*,{*services*}⟩]
ℓ(*myVISA*):	[5, ⟨∗,{*payment*}⟩, ⟨*BankC*,∗⟩]
ℓ(*myAMEX*):	[8, ⟨∗,{*payment*}⟩, ⟨*BankC*,∗⟩]
ℓ(*decl*):	[0, ⟨∗,∗⟩]

SENSITIVE VIEWS

ℓ({Name,*myVISA*.CCNum,*myAMEX*.CCNum}): [10, ⟨*BankC*,∗⟩]
ℓ({Name,*myDialysis*}): [34, ⟨*HospitalA*,{*treatment*,*diagnosis*}⟩]

DEPENDENCIES

ℓ({Phone,Country}): [−2, ⟨∗,∗⟩]

(b)

Figure 7 An example of a legitimate disclosure graph w.r.t. *Ctx*={⟨*Medical*,{*payment*}⟩} over the portfolio in Figure 6

vertices and edges that are relevant for Ctx (i.e., such that $\phi(v) \succeq_{ctx} Ctx$*) are represented with a dashed black line, all the other vertices and edges are represented with a dotted light gray line. The sensitivity of the disclosure* $\lambda(G^d)$ *is computed as the composition of the sensitivity labels of bold vertices, that is,* λ(Name) + λ(DoB) + λ(Country) + λ(Phone) + λ(myVISA.CCNum) + λ(myId) + λ(myVISA) + λ(decl) + λ({Country,Phone}) = 27.

Intuitively, a request $\mathcal{R}$ is satisfied by a legitimate disclosure w.r.t. $\phi(\mathcal{R})$ if at least one of the simple requests r in $\mathcal{R}$ is satisfied. A simple request r is satisfied by a legitimate disclosure w.r.t. $\phi(\mathcal{R})$ 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* $\succeq_T$ *type*(c). Graphically, a legitimate disclosure w.r.t. $\phi(\mathcal{R})$, 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 8.3 (Satisfying Disclosure) *Let* r *be a simple request,* $\phi(r)$ *its context restriction,* $G^r(V^r_T \cup V^r_P, E^r_T)$ *the corresponding request graph, and* G^d *a legitimate disclosure graph w.r.t.* $\phi(r)$ *(Definition 8.2).* G^d satisfies G^r*, denoted* $G^d \models G^r$*, iff there exists a subgraph* $G^{d'}(V^{d'}_C \cup V^{d'}_P, E^{d'}_C)$ *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. $\forall v_t \in V^r_T$, $\exists f(v_t) \in V^{d'}_C \wedge v_t \succeq_T type(f(v_t))$;
2. $\forall v_p \in V^r_P$, $\exists f(v_p) \in V^{d'}_P \wedge v_p = f(v_p)$;
3. $\forall (v_t, v_p) \in E^r_T$, $(f(v_t), f(v_p)) \in E^{d'}_C$.

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, since G^d possibly includes atomic credentials and 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 legitimate disclosure w.r.t. $\phi(\mathcal{R})$ (Definitions 8.1 and 8.2).

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

Example 8.2 *Consider the disclosure graph* G^d *in Figure 7 and the request graphs* G^{r_1} *and* G^{r_2} *in Figure 3. 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 legitimate disclosure graphs w.r.t. $\phi(\mathcal{R})$ that satisfy the server request, the client is interested in the one that minimizes the 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, the disclosure either does not satisfy the request or the sensitivity label of the disclosure increases). In other words, a disclosure graph G^d is minimal if there does not exist any disclosure graph $G^{d'}$, subgraph of G^d, that satisfies the request with lower sensitivity.

Definition 8.4 (Minimal Disclosure) *Let G be a portfolio graph and $\mathcal{R}$ be a request. A disclosure graph $G^d(V_C^d \cup V_P^d \cup V_A^d \cup V_L^d, E_C^d \cup E_A^d \cup E_L^d, \ell)$ is a* minimal disclosure *w.r.t. $\mathcal{R}$ iff:*

- *G^d satisfies $\mathcal{R}$ ($\exists r$ in $\mathcal{R}$ such that $G^d \models G^r$, Definition 8.3);*
- *$\nexists$ a disclosure graph $G^{d'}$ of G such that $G^{d'}$ satisfies $\mathcal{R}$; $V_C^{d'} \cup V_P^{d'} \cup V_A^{d'} \subset V_C^d \cup V_P^d \cup V_A^d$; and $\lambda(G^{d'}) < \lambda(G^d)$.*

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

Problem 8.1 (Min-Disclosure) *Given a portfolio graph G and a request $\mathcal{R}$, find a* minimum disclosure *graph $G^d(V_C^d \cup V_P^d \cup V_A^d \cup V_L^d, E_C^d \cup E_A^d \cup E_L^d, \lambda)$ of G w.r.t. $\mathcal{R}$ that satisfies the following requirements:*

- *G^d satisfies $\mathcal{R}$ ($\exists r$ in $\mathcal{R}$ such that $G^d \models G^r$, Definition 8.3);*
- *$\nexists$ a disclosure graph $G^{d'}$ of G such that $G^{d'}$ satisfies $\mathcal{R}$ and $\lambda(G^{d'}) < \lambda(G^d)$.*

We note that any minimum disclosure graph is also a minimal disclosure graph, while the contrary is not true.

Example 8.3 *With reference to the request graph G^{r_1} in Figure 3(a), the disclosure graph G^d in Figure 7 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 8.1, 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`,`Country`}*, can be satisfied by disclosing either* myId *or* myPassport*. The release of* myId *discloses an additional property (i.e.,* `DoB`*). The disclosure of (non-atomic) credential* myPassport *does not require the release of property* `DoB`*, which incurs in additional cost, and is therefore preferred.*

The Min-Disclosure problem is NP-hard, as stated by the following theorem.

Theorem 1 *The Min-Disclosure problem is NP-hard.*

Proof: The proof is a reduction from the NP-hard problem of the Minimum Set Cover, formulated as follows: *given a collection S of subsets of a finite set U, determine a subset $S' \subseteq S$ such that every element in U belongs to at least one member of S', and the cardinality of S' is minimized.*
Given the set $\mathcal{P}$ of properties, the set $\mathcal{C}$ of credentials, and the set $\mathcal{A}$ of associations composing the client portfolio, and a request $\mathcal{R}$, the correspondence between the Min-Disclosure problem and the minimum set cover problem can be defined as follows. The set $\mathcal{P}$ of properties includes a property for each element in U, a property `s_id` for each $s \in S$, and a property p_0. Properties in $U \cup \{p_0\}$ are credential independent, while properties `s_id` are credential dependent. The set $\mathcal{C}$ of credentials is defined as the set S of subsets of elements in U. Specifically, credential c representing a subset s certifies the properties composing s and property `s_id`. The set $\mathcal{C}$ of credentials also includes an additional credential of type *prb* that certifies property p_0. All the credentials in $\mathcal{C}$ are atomic. The set $\mathcal{A}$ of associations is composed of a single dependency $A = U \cup \{p_0\}$. The sensitivity label of all the properties and credentials in the portfolio, but p_0, is equal to zero, $\lambda(p_0)=1$ and $\lambda(A)=-1$. Let us now consider request $\mathcal{R}_1 = prb.p_0 \wedge *.\texttt{s\_id}$. If the disclosure $\mathcal{D}$ has sensitivity label equal to 0, then it represents a solution S' for the corresponding minimum set cover problem, where the disclosed credential is the unique set covering U. If the disclosure $\mathcal{D}$ has sensitivity label equal to 1, we consider a new request $\mathcal{R}_2 = prb.p_0 \wedge *.\texttt{s\_id} \wedge *.\texttt{s\_id}$ that tries to find a set cover composed of two subsets in S. The number of terms in the request is iteratively increased by one until a solution to the problem is found or the number of terms in the request is equal to $|S| = |\mathcal{C}|$. Hence, the Min-Disclosure problem is NP-hard. □

9 Communication state management

In the previous discussion, we focused on minimizing the information released by the client in response to a request from a server. However, Web-based information systems typically support a navigational access to resources. For instance, a client, guided by a Web interface, can browse a catalog and then possibly buy an item within it. The client might then be requested to release first some information (credentials/properties) for accessing and browsing the catalog, and then further information for completing the purchase. In such a case, it is important for the client to maintain track of the *communication state* (i.e., information already disclosed to the server) within a generic time/operation window set by the client for two main purposes: *i)* to minimize the amount of information released overall; or *ii)* to prevent the server from linking different accesses to the same client (i.e., prevent *linkability*). In the following, we will discuss how these requirements can be managed by our graph-based model.

9.1 Minimizing disclosure

When the client-server interaction is characterized by a sequence of requests for data by the server, the client may want to ensure minimality of the information released overall, rather than the specific information released at each server request within a generic time/operation window. As an example, consider a Web-based reservation system, where the client first books a flight and then rents a car. When completing the car reservation, the client might want to take into consideration the information already disclosed to the server for the flight reservation, to minimize the information released overall. Therefore, if the server requires a credit card or an id document, the client might want to stick with the one just released. The information already disclosed by the client has also to be taken into consideration to ensure the disclosure constraints are not violated. For instance, with reference to the sensitivity specification Example 5.1, suppose a client discloses its `NickName` to the server. By providing stateful capability, the subsequent release of `Name` will be prohibited, since otherwise the disclosure constraint would be violated.

Our graph-based modeling of the client portfolio allows to address the issues above, providing the ability of maintaining the communication state of the client with the server (i.e., the set of properties and credentials already released). Each time window is characterized by a unique disclosure graph that is possibly updated at each request by including additional properties/credentials. The disclosure graph of a time window must satisfy all the requests, while minimizing for each request the additional information released with respect to the communication state. Given a request, if the disclosure graph modeling the communication state satisfies it, the client does not need to release additional information; otherwise, the disclosure graph is extended to include other properties/credentials in the portfolio. In such a way, we capture properties and credentials previously disclosed and therefore they are taken into consideration in the calculation of the sensitivity label of the disclosure, as well as in the evaluation of the disclosure constraints.

Example 9.1 *Consider a Web-based medical system, in category* HospitalA*, that permits to reserve and pay for an examination, and to consult a diagnosis. Given the server request in Example 4.1, disclosure* $\mathcal{D}$=\{myId.\{`Name`,`DoB`,`Country`\}, myVISA.\{`Name`,`CCNum`\}, decl.`Phone`\} *in Example 8.1 permits the user to reserve and pay for her examination. As soon as the user tries to consult a diagnosis, the server issues request* $\mathcal{R}'$=id.\{`Name`\}*. This request can be satisfied by disclosing either* $\mathcal{D}_1$=myId.\{`Name`,`DoB`,`Country`\} *or* $\mathcal{D}_2$=myPassport.\{`Name`\}*. If we do not consider the communication state,* $\lambda(\mathcal{D}_1)$=7 *and* $\lambda(\mathcal{D}_2)$=1*, and the client discloses* $\mathcal{D}_2$*. On the contrary, if we consider the communication state including the disclosure of* $\mathcal{D}$*,* $\lambda(\mathcal{D}_1)$=0*, since atomic credential* myId *has been already released in* $\mathcal{D}$*.*

9.2 Preventing linkability

Web-based interactions usually assume clients and servers to be unknown to each other. At each server request, the client discloses a subset of its portfolio that could be exploited by the server to associate different disclosures with the same client. As a consequence, besides minimizing the amount of information disclosed at each server request, the client may be interested in preventing linkability. At each request, information released within a time window needs to be considered, to ensure that the same subset of properties/credentials is not disclosed more than once. For instance, consider an online music store, where the user orders, with two subsequent operations, two different songs that can be payed either by credit card or with a gift card. For the purchase of the first song, she releases her credit card, while for the purchase of the second one, she discloses an anonymous gift card, thus preventing linkability.

Since, as already discussed, our graph-based modeling of the client portfolio provides the ability of maintaining the communication state of the client with the server, it also allows the client to identify a satisfying disclosure that minimizes the risk of linkability. To this aim, given the communication state, the client needs to compute a disclosure graph that both satisfies the current request and minimizes its intersection with properties and credentials in the communication state. The larger is the intersection among disclosures, the higher is the linkability risk. Therefore, whenever a request can be satisfied only by releasing a set of properties/credentials that has an intersection with the communication state, the client can decide to either proceed or terminate the communication. For instance, if the intersection with the communication state is represented by property `Country` the client may decide to proceed, since many different users live in the same country. On the contrary, if the intersection with the communication state is represented by property `SSN` the client may decide to terminate the communication, since `SSN` is a unique identifier.

Example 9.2 *Consider a Web-based medical system, in category* HospitalA*, that permits to reserve two examinations. Given a server request* $\mathcal{R}$=$r_1 \vee r_2$*, with* r_1=(id.\{`Name`,`DoB`\} $\wedge$ *.\{`Phone`\}) *and* r_2=(credential.\{`SSN`\} $\wedge$ *.\{`NickName`\})*, disclosure* $\mathcal{D}$=\{myId.\{`Name`,`DoB`,`Country`\},decl.`Phone`\} *satisfies* r_1 *and permits the user to reserve her first examination. As soon as the user tries to reserve the second examination, the server issues the same request* $\mathcal{R}$*. In this case the user wants to protect linkability, and therefore minimize the intersection between data released in different requests (still minimizing the sensitivity label of each single release).* $\mathcal{R}$ *can be satisfied by disclosing either the properties and credentials in the communication state (i.e.,* $\mathcal{D}$*), or one between* $\mathcal{D}_1$=\{myPassport.\{`Name`,`DoB`\},decl.`Phone`\} *and* $\mathcal{D}_2$=\{myDialysis.\{`SSN`\},decl.\{`NickName`\}\}*. If we do not consider the communication state, since* $\lambda(\mathcal{D})$=12*,*

$\lambda(\mathcal{D}_1)$=13, and $\lambda(\mathcal{D}_2)$=26, the client discloses $\mathcal{D}$. On the contrary, if we consider the communication state including the disclosure of $\mathcal{D}$, $\mathcal{D}_2$ is released since $\mathcal{D}\cap\mathcal{D}_2=\emptyset$.

Note that the approach to prevent linkability is applicable also when different servers may exchange client information. In this case, the client needs to maintain a unique communication state for each domain, including the information communicated to any server in the domain itself.

10 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 a 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, associations and disclosure constraints, 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. Figure 8 illustrates the pseudocode of our heuristics. It takes as input the portfolio graph G and a server request $\mathcal{R}=r_1\vee\ldots\vee r_i$, characterized by context restriction $\phi(\mathcal{R})$, and computes a minimal disclosure graph G^d that satisfies $\mathcal{R}$, if such a disclosure exists.

The algorithm first removes from the portfolio graph all the properties and credentials that cannot belong to a disclosure that is legitimate w.r.t. $\phi(\mathcal{R})$, since $\phi(v)\not\succeq_{ctx}\phi(\mathcal{R})$. Also, it removes from the portfolio graph the associations and disclosure constraints including at least one of the properties and/or credentials removed from G, since these associations and constraints cannot be violated by any legitimate disclosure. The associations that are characterized by context restrictions that do not dominate $\phi(\mathcal{R})$ are instead transformed into disclosure constraints, since the joint release of the properties/credentials in the association must be forbidden in the considered context. The algorithm assigns a sensitivity label equal to $\lambda(G)$+1 to each disclosure constraint in G. In this way, disclosure constraints can be treated by the algorithm in the same way as associations. If a disclosure has sensitivity label greater than $\lambda(G)$, it violates at least a disclosure constraint and is therefore not legitimate.

INPUT
$\mathcal{R}=r_1\vee\ldots\vee r_i$: server request
$\phi(\mathcal{R})$: context restriction of the request
$G(V_C\cup V_P\cup V_A\cup V_L, E_C\cup E_A\cup E_L, \ell)$: extended portfolio graph

OUTPUT
$G^d(V^d_C\cup V^d_P\cup V^d_A\cup V^d_L, E^d_C\cup E^d_A\cup E^d_L, \ell)$: disclosure graph satisfying $\mathcal{R}$

MAIN
/* **Pre-filtering** */
for each $v\in(V_C\cup V_P)$ **do**
  if $\phi(v)\not\succeq_{ctx}\phi(\mathcal{R})$ **then**
    for each $v_a\in V_A : (v_i,v_a)\in E_A$ **do**
      $E_A := E_A \setminus \{(v_j,v_a)\in E_A : v_j\in V_C\cup V_P\}$
      $V_A := V_A \setminus \{v_a\}$
    for each $v_l\in V_L : (v_i,v_l)\in E_L$ **do**
      $E_L := E_L \setminus \{(v_j,v_l)\in E_L : v_j\in V_C\cup V_P\}$
      $V_L := V_L \setminus \{v_l\}$
    $E_C := E_C \setminus \{(v,v_i)\in E_C : v_i\in V_C\cup V_P\}$
    if $v\in V_C$ **then** $V_C := V_C \setminus \{v\}$
      else $V_P := V_P \setminus \{v\}$
for each $v_a\in V_A$ **do**
  if $\phi(v_a)\not\succeq_{ctx}\phi(\mathcal{R})$ **then**
    $E_L := E_L \cup \{(v,v_a)\in E_A : v\in V_C\cup V_P\}$
    $E_A := E_A \setminus \{(v,v_a)\in E_A : v\in V_C\cup V_P\}$
    $V_L := V_L \cup \{v_a\}$
    $V_A := V_A \setminus \{v_a\}$
/* model disclosure constraints as associations */
$threshold := \lambda(G)+1$
for each $v\in V_L$ **do** $\lambda(v) := threshold$
/* **Find a minimal disclosure** */
$G^d := \emptyset$
for each $r_i\in\mathcal{R}$ **do**
  $G^{d_i} :=$ **Compute_Disclosure**$(G^{r_i}, G, threshold)$
  if $G^{d_i}\neq\emptyset \wedge \lambda(G^{d_i})<\lambda(G^d)$ **then** $G^d := G^{d_i}$
if $G^d\neq\emptyset$ **then return**(G^d)
  else return(error)

COMPUTE_DISCLOSURE$(G^r, G, threshold)$
$G^d := G$
$opt := G^d$
while $opt=G^d$ **do**
    $G^d := opt$
    for each $v\in V^d_C$ **do** /* try to remove credentials */
      $G^{d'} :=$ **Remove_Cred**(v, G^d)
      if $G^{d'}\models G^r \wedge \lambda(opt)>\lambda(G^{d'})$ **then** $opt := G^{d'}$
    for each $v\in V^d_P$ **do** /* try to remove properties */
      $G^{d'} :=$ **Remove_Prop**(v, G^d)
      if $G^{d'}\models G^r \wedge \lambda(opt)>\lambda(G^{d'})$ **then** $opt := G^{d'}$
if $\lambda(opt)>threshold$ **then return**$(\emptyset)$ /* the disclosure is not legitimate */
else return(opt)

REMOVE_CRED(v, G^d)
for each $v_a\in V^d_A : (v,v_a)\in E^d_A$ **do** /* remove associations */
  $E^d_A := E^d_A \setminus \{(v,v_a)\}$
  $V^d_A := V^d_A \setminus \{v_a\}$
for each $v_l\in V^d_L : (v,v_l)\in E^d_L$ **do** /* remove constraints */
  $E^d_L := E^d_L \setminus \{(v,v_l)\}$
  $V^d_L := V^d_L \setminus \{v_l\}$
for each $v_p\in V^d_P : (v,v_p)\in E^d_C$ **do** /* remove uncertified properties */
  if $\nexists\ v_c\in V^d_C : v_c\neq v \wedge (v_c,v_p)\in E^d_C$ **then**
    Remove_Prop(v_p, G^d)
/* remove the credential */
$E^d_C := E^d_C \setminus \{(v_i,v)\in E^d_C : v_i\in V^d_P\}$
$V^d_C := V^d_C \setminus \{v\}$
return(G^d)

REMOVE_PROP(v, G^d)
for each $v_a\in V^d_A : (v,v_a)\in E^d_A$ **do**
  $E^d_A := E^d_A \setminus \{(v,v_a)\}$ /* remove associations */
  $V^d_A := V^d_A \setminus \{v_a\}$
for each $v_l\in V^d_L : (v,v_l)\in E^d_L$ **do**
  $E^d_L := E^d_L \setminus \{(v,v_l)\}$ /* remove constraints */
  $V^d_L := V^d_L \setminus \{v_l\}$
for each $(v,v_c)\in E^d_C$ s.t. v_c is atomic **do**
  Remove_Cred(v_c, G^d) /* remove atomic credentials certifying v */
/* remove the property */
$E^d_C := E^d_C \setminus \{(v,v_i)\in E^d_C : v_i\in V^d_C\}$
$V^d_P := V^d_P \setminus \{v\}$
return(G^d)

Figure 8 Algorithm computing a minimal disclosure

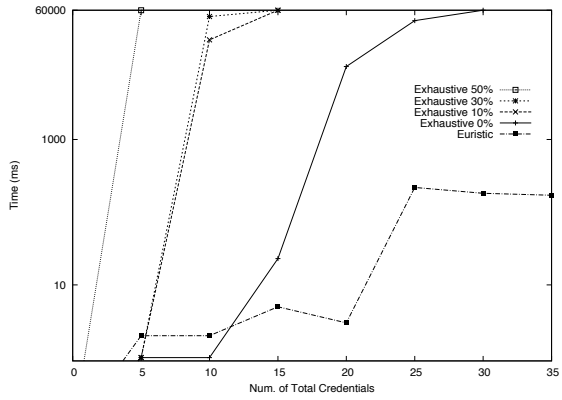


Figure 9 Execution time of the heuristic and the exhaustive algorithms

For each simple request r_i in $\mathcal{R}$, the algorithm computes the corresponding minimal disclosure G^{d_i} through function **Compute_Disclosure**. Function **Compute_Disclosure** initializes the minimal disclosure G^d for simple request r to G, which corresponds to release all the credentials and properties in the portfolio that are relevant for $\mathcal{R}$. The function then iteratively evaluates the sensitivity label of the disclosure graphs obtained by removing from G^d either a credential (function **Remove_Cred**) or a property (function **Remove_Prop**). Among the graphs obtained, the algorithm selects the one (*opt*) with minimum sensitivity that satisfies r, which becomes the new disclosure graph G^d. The process of reducing G^d 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 r or has a higher sensitivity label). If the computed minimal disclosure has sensitivity label greater than $\lambda(G)$, function **Compute_Disclosure** returns an empty disclosure graph, otherwise it returns the computed minimal disclosure, which is legitimate and satisfies the simple request r. Among all the (non empty) minimal disclosures G^{d_i} computed for the simple requests composing $\mathcal{R}$, the algorithm selects the one with lowest sensitivity label, which is finally returned.

To assess the efficiency and effectiveness of our heuristics, 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 heuristics with the optimum, we also implemented an exhaustive algorithm solving the Min-Disclosure problem (Problem 8.1). Experiments have been run on a PC with two Intel Xeon Quad 2.0GHz L3-4MB, 12GB RAM, 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, the number of disclosure constraints, and the sensitivity of credentials, properties, and associations. Overall, the heuristic algorithm was able to produce the optimum in 88% of the cases, and when the optimum was not identified, the distance from the optimum was on average 7.21% above the optimum. Figure 9 compares the execution time of our heuristics 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%, 30%, 10%, and 0% of the credentials to be non-atomic. As expected, the heuristics was always able to produce an answer in less than 130ms, whereas the exhaustive algorithm requires exponential time in the size of the portfolio, with a strong dependence on the number of non-atomic credentials.

11 Related work

Research on credential-based access control [4, 5, 14, 17, 19] and trust negotiation [10, 12, 13, 16, 18, 21, 22] primarily focused on server-side issues and proposed solutions for controlling access to resources, specifying and enforcing policies, and enabling negotiation. These solutions typically assume to adopt a symmetric approach at the client side, for regulating the release of user private information and possibly manage negotiation with the server. These approaches, however, do not allow the client to determine which credentials and/or properties to release to minimize the sensitive information communicated to the server. Also, they do not support emerging technologies, such as SAML [1], OpenID [9], and anonymous credentials [6, 7]. In the literature, only few works have addressed this issue. Chen et al. [8] propose a solution that associates costs with credentials and policies to minimize the cost of a credential release within a trust-negotiation protocol. Kärger et al. [11] describe a logic-based language for the specification of privacy preferences dictating a partial order among the client properties. Both 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. Yao et al. [20] 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 credentials, such that the total privacy score of disclosed credentials is minimal and the server access threshold is satisfied. Differently from [20], our solution allows the client to define its privacy preferences and to minimize the disclosure of sensitive information, independently from the server preferences. Also, our proposal provides a complete modeling of the client portfolio, including both sensitivity of associations and disclosure constraints. Sensitivity labels as a means for expressing privacy preferences on portfolio components have been first proposed in [2, 3]. In [2], we introduce a solution for protecting the privacy of the

users based on sensitivity labels, a graph-based modeling of the portfolio, and a heuristic approach to determine a disclosure minimizing released information. The work in [3] enhances the solution in [2] by supporting a richer approach for the specification of sensitivity labels and additional constraints. It also proposes a novel modeling of the problem exploiting its translation in terms of a Weighted Max-SAT problem and its resolution via existing SAT solvers. This paper considerably extends the works in [2, 3] by providing a revised graph modeling of the portfolio that extends and complements sensitivity labels with context restrictions. Also, it proposes an approach enabling the client to identify the information in the portfolio relevant for a given request, based on the competence area of the server and on the transaction. Finally, it presents a solution that permits the client to consider the communication state (i.e., previous releases in a generic time window set by the user) when computing a satisfying disclosure that either minimizes the total amount of released information or prevents linkability among requests.

12 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. Foresti, S. Paraboschi, and P. Samarati. Minimizing disclosure of private information in credential-based interactions: A graph-based approach. In *Proc. of the 2nd IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT 2010)*, Minneapolis, MN, USA, August 2010.

[3] C.A. Ardagna, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, and P. Samarati. Supporting privacy preferences in credential-based interactions. In *Proc. of the Workshop on Privacy in the Electronic Society (WPES 2010)*, Chicago, Illinois, USA, October 2010.

[4] 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 Transactions on Service Computing (TSC)*, 2010. (to appear).

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

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

[7] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proc. of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT 2001)*, Innsbruck, Austria, May 2001.

[8] W. Chen, L. Clarke, J. Kurose, and D. Towsley. Optimizing cost-sensitive trust-negotiation protocols. In *Proc. of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, Miami, FL, USA, March 2005.

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

[10] K. Irwin and T. Yu. Preventing attribute information leakage in automated trust negotiation. In *Proc. of the 12th ACM Conference on Computer and Communications Security (CCS 2005)*, Alexandria, VA, USA, November 2005.

[11] P. Kärger, D. Olmedilla, and W.-T. Balke. Exploiting preferences for minimal credential disclosure in policy-driven trust negotiations. In *Proc. of the 5th VLDB Workshop on Secure Data Management (SDM 2008)*, Auckland, New Zealand, August 2008.

[12] A. J. Lee, M. Winslett, J. Basney, and V. Welch. The Traust authorization service. *ACM Transactions on Information and System Security (TISSEC)*, 11(1):1–33, February 2008.

[13] J. Ni, N. Li, and W.H. Winsborough. Automated trust negotiation using cryptographic credentials. In *Proc. of the 12th ACM Conference on Computer and Communications Security (CCS 2005)*, Alexandria, VA, USA, November 2005.

[14] T. Ryutov, L. Zhou, C. Neuman, T. Leithead, and K. E. Seamons. Adaptive trust negotiation and access control. In *Proc. of the 10th Symposium on Access control Models and Technologies (SACMAT 2005)*, Stockholm, Sweden, June 2005.

[15] P. Samarati and S. De Capitani di Vimercati. Access control: Policies, models, and mechanisms. In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design*, volume 2171 of *LNCS*. Springer-Verlag, 2001.

[16] K. Seamons, M. Winslett, and T. Yu. Limiting the disclosure of access control policies during automated trust negotiation. In *Proc. of 8th Network and Distributed System Security Symposium (NDSS 2001)*, San Diego, CA, USA, April 2001.

[17] K. E. Seamons, W. Winsborough, and M. Winslett. Internet credential acceptance policies. In *Proc. of the Workshop on Logic Programming for Internet Applications*, Leuven, Belgium, July 1997.

[18] W. Winsborough, K. E. Seamons, and V. Jones. Automated trust negotiation. In *Proc. of the DARPA Information Survivability Conference & Exposition (DISCEX 2000)*, Hilton Head Island, SC, USA, January 2000.

[19] M. Winslett, N. Ching, V. Jones, and I. Slepchin. Assuring security and privacy for digital library transactions on the web: Client and server security policies. In *Proc. of the 4th International Forum on Research and Technology Advances in Digital Libraries (ADL 1997)*, Washington, DC, USA, May 1997.

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

[21] T. Yu and M. Winslett. A unified scheme for resource protection in automated trust negotiation. In *Proc. of the IEEE Symposium on Security and Privacy*, Berkeley, CA, USA, May 2003.

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