Supporting Privacy Preferences in Credential-Based Interactions

Claudio A. Ardagna
Università degli Studi di Milano
26013 Crema - Italy
claudio.ardagna@unimi.it

Sabrina De Capitani di Vimercati
Università degli Studi di Milano
26013 Crema - Italy
sabrina.decapitani@unimi.it

Sara Foresti
Università degli Studi di Milano
26013 Crema - Italy
sara.foresti@unimi.it

Stefano Paraboschi
Università di Bergamo
24044 Dalmine - Italy
parabosc@unibg.it

Pierangela Samarati
Università degli Studi di Milano
26013 Crema - Italy
pierangela.samarati@unimi.it

ABSTRACT

Users can today enjoy the many benefits brought by the development and widespread adoption of Internet and related services conveniently accessing digital resources. Servers offering such resources typically require users to release information about them, which servers can then use for enforcing possible access policies on the offered services. A major problem in this context relates to providing users with the ability of determining which information to release to satisfy the server requests during their electronic interactions.

In this paper, we provide an approach for empowering the user in the release of her digital portfolio based on simple sensitivity labels expressing how much the user values different properties, credentials or combinations thereof, as well as on additional constraints that the user might impose on information disclosure. We provide a generic modeling of the problem and illustrate its translation in terms of a Weighted MaxSat problem, which can be conveniently and efficiently managed by off the shelf SAT solvers, thus resulting efficient and scalable.

Categories and Subject Descriptors

K.4.1 [**Computer and Society**]: Public Policy Issues—*Privacy, Regulation*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security

Keywords

Privacy, user preferences, portfolio management, credentials

WPES'10, October 4, 2010, Chicago, Illinois, USA.

1. INTRODUCTION

The development of Web and Internet technologies have produced an obvious great impact on society and in the life of people. This success makes even more critical the resolution of important open problems that still plague the user experience, such as, the need to manage a large number of accounts and the limited control over her personal information collected by servers. Effective and easy to use technological solutions should then be developed for enabling the user to access Web services conveniently, while maintaining control over her personal information. Digital certificates, or credentials, are considered by many as the most promising technique for responding to this need.

Credentials enable departing from traditional authentication means, avoiding the need for users to remember logins and passwords, at the same time increasing the difficulty for adversaries of impersonating users and improperly acquire access privileges. Companies like IBM and Microsoft are investing in the development of anonymous credentials that can offer robustness and ease of use of certificates at the same time providing support for privacy better than currently used X.509 certificates (whose use entails disclosure of the user identity). The continuing development of Idemix technology by IBM, together with the acquisition of Credentica in 2008 by Microsoft and the public release in 2010 of an SDK for the use of U-Prove credentials, show the concrete interest paid to credential technology and in particular to anonymous credentials.

Effective use of such credential technologies 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. The research and development communities have addressed this requirement by investigating support of credential-based access control. However, most attention has been devoted on the server side of the problem, proposing a variety of models and languages for expressing access control policies at the server, as well as strategies for their evaluation and communication to the client. The client side aspect of the problem has been typically handled assuming a symmetric approach (based on access control rules) for regulating the release of user private information, and possibly enabling the client to engage in negotiation with the server. However, access control-like specifications do not completely fit the possible

protection requirements at the client side, where users need approaches to express in an intuitive and easy to use way their natural perception of the privacy value of their 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, supporting selective release of properties within certificates as allowed by novel anonymous credentials. Our model allows a user to assign sensitivity labels 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. Also, our model allows the user to specify additional constraints that she may want to impose on information disclosure when interacting over the Web. Expressing sensitivity labels, our approach allows for minimizing information release. We translate then the problem of minimizing user private information disclosed to the counterpart into a weighted satisfiability problem that common SAT solvers are able to manage with a limited amount of resources. This translation offers an efficient and effective enforcement of the user-friendly specifications captured in our model. This way our approach provides both usability for the user and efficiency in the enforcement.

The remainder of this paper is organized as follows. Section 2 illustrates the organization of the client portfolio. Section 3 defines server requests and client information disclosure responding to them. Section 4 illustrates how users can specify privacy preferences and requirements on the different elements of their portfolio and introduces the problem of minimizing information disclosure, satisfying all privacy constraints. Section 5 describes the translation of the problem of computing a minimum disclosure to an instance of the Weighted Max-SAT problem. Section 6 presents experimental results obtained by using a SAT solver on our problem formulation, which confirm the efficiency and scalability of our solution. Section 7 discusses related work. Finally, Section 8 presents our conclusions.

2. CLIENT PORTFOLIO

The information that a client can provide to acquire services from a server is collected within a *portfolio*. The portfolio contains both certified and uncertified *properties*. Certified properties are organized within *credentials*, signed by third parties, while uncertified properties are self-signed statements that form a *declaration* [2, 5]. We now describe in more details the components of the client portfolio.

2.1 Credentials

Credentials are organized by types, 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. Hierarchy $\mathcal{H}$ has a unique root, denoted with $*$, such that $t_i \preceq_{isa} *$, for each $t_i \in \mathcal{T}$. Our model allows referring to credentials at the granularity of instance or type. For example, *id* is an abstraction of credential types *id_card* and *driver_license* (i.e., *id_card* $\preceq_{isa}$ *id* and *driver_license* $\preceq_{isa}$ *id*).

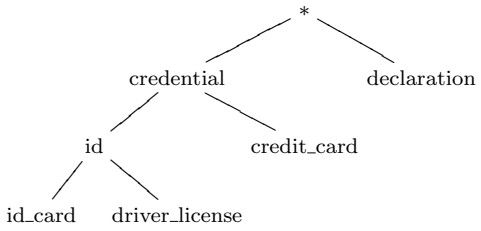

Figure 1: An example of hierarchy of credential types

Credentials *myId* and *myLicense* are instances of *id_card* and *driver_license*, respectively, and, consequently, also instances of *id*. Each credential instance c is characterized by a unique identifier, a set $\{p_1,\ldots,p_n\}$ of certified properties, a type, and an issuer. In the following, we will denote with c, *properties*(c), and *type*(c) the credential identifier, the set of properties certified by c, and the type of c, respectively.

We distinguish between two classes of credentials: *atomic* and *non-atomic* [2]. Atomic credentials can only be released as a whole, that is, their release entails the disclosure of all the properties they certify. By contrast, non-atomic credentials support the selective release of individual properties extracted from the credential. Atomic credentials (e.g., X.509 certificates) are the most common kind of credential used today in distributed systems. By contrast, non-atomic credentials (e.g., U-Prove and Idemix [6, 7]) are based on modern technologies and permit also to certify the possession of an instance of a given credential type, without disclosing the properties within it.

All the certified properties in the client portfolio may also be declared, that is, included in self-signed credential *decl*, which may also contain additional properties that are not included in any credential. Credential type *decl* is represented in the hierarchy of credential types as a direct descendant of the root. Figure 1 illustrates an example of hierarchy of credential types. Clearly, *decl* is non-atomic, but no proof of existence can be released for it.

2.2 Properties

Each property p is characterized by a unique identifier, a type (organized in a 1-level hierarchy), a name, and a value. In the following, we will denote with p and *type*(p) the property identifier and its type, respectively.

We distinguish between two classes of properties: *credential-independent* and *credential-dependent* [2]. Credential-independent properties depend only on the client, and not on the specific credential certifying their value. For instance, date of birth is a property of the client, and possible occurrences of the property in different credentials refer all to the same data. By contrast, credential-dependent properties are associated with the client with reference to a specific credential. For instance, credit card number is a property that depends on the specific instance of credential type *credit_card*. Different instances of credit cards will all refer to their specific credit card number, and therefore to a different property. Credential-dependent properties might have different occurrences of the same type, depending on the existence of different credentials including them, while credential-independent properties have a single occurrence. Note that

PROPERTIES

Id	Dep.	Type	Value
Name		*Name*	Bob
DoB		*DoB*	1975/10/23
Address		*Address*	3155, 5th Ave, New York, USA
Country		*Country*	USA
VISANum	✓	*CCNum*	4353...21
MCNum	✓	*CCNum*	5643...18
Phone		*Phone*	789-231-044
eMail		*eMail*	bob@abc.com
NickName		*NickName*	bob75

CREDENTIALS

Id	Atom.	Type	Properties
myId	✓	*id_card*	Name,DoB,Address
myLicense		*driver_license*	Name,DoB,Country
myVISA	✓	*credit_card*	Name,VISANum
myMC	✓	*credit_card*	Name,MCNum
decl		*declaration*	Name,DoB,Address Country,VISANum,MCNum Phone,eMail,NickName

Figure 2: An example of portfolio

each instance of a property (both credential dependent and independent) has a different identifier p.

EXAMPLE 2.1. *Figure 2 illustrates an example of a portfolio, including properties* Name, DoB, Address, Country, Phone, eMail, *and* NickName, *which are credential-independent, and* VISANum *and* MCNum, *which are credential-dependent and both of type* CCNum. *The portfolio contains credentials* myId (*of type* id_card), myLicense (*of type* driver_license), myVISA *and* myMC (*both of type* credit_card), *and a declaration* decl *including all the properties in the portfolio. The only non-atomic credential, besides* decl, *is* myLicense.

3. SERVER REQUEST AND CLIENT DISCLOSURE

We consider an open Web-based scenario that consists of remote communications between a client and a server. The client starts the communication by requesting access to a resource available on the server. In turn, the server sends a *request* for a set of properties to the client, to evaluate and enforce the policies protecting the resource.

We assume the server request $\mathcal{R}$ to be expressed as the disjunction of *simple requests* $R_1 \vee \ldots \vee R_n$. Each simple request R is a conjunction $r_1 \wedge \ldots \wedge r_m$ of *terms* r of the form $type.\{pt_1,\ldots,pt_i\}$, where *type* is a credential type and $\{pt_1,\ldots,pt_i\}$ is a set of property types. In the following, we use $type(r)$ to refer to element *type* of term r and $properties(r)$ to refer to the set $\{pt_1,\ldots,pt_i\}$. Each term r in the request prescribes the disclosure of a set of properties from a single credential c in the client portfolio, such that c represents an instance of abstraction $type(r)$ and, for each type in $properties(r)$, a property of the same type in $properties(c)$ is released, as formally defined in the following.

DEFINITION 3.1 (TERM SATISFACTION). *Let $\mathcal{C}$ and $\mathcal{P}$ be the set of credentials and properties, respectively, in the client portfolio, and $\mathcal{R}$ be a request. Credential $c \in \mathcal{C}$ satisfies a term r in $\mathcal{R}$ iff the following conditions hold:*

- $type(c) \preceq_{isa} type(r)$;
- $\forall pt \in properties(r)$, $\exists p \in properties(c)$: $type(p)=pt$.

Since the server does not know the client portfolio, terms in the request can only correspond to types appearing in the hierarchy of credential types, shared with the client. Therefore, each term can be seen as a compact representation of a set of alternative requests for credentials in the client portfolio. As an example, consider three terms, $r_1=id.\{\texttt{Name}\}$, $r_2=credential.\{\texttt{Name}\}$, and $r_3=*.\{\texttt{Name}\}$. The first term requires the release of property Name, certified by a credential of type *id*, that is either *id_card* or *driver_license*. The second term requires the release of property Name, certified by any credential. The third term requires the release of property Name that can be either certified by any credential or declared by the client.

Given a request $\mathcal{R}$ formulated by the server, the client must determine a *disclosure* $\mathcal{D}$, that is, a subset of her portfolio, to be communicated to the server for satisfying $\mathcal{R}$. Note that the disclosure of a set of properties in the client portfolio also implies the release of a set of credentials certifying them, and possibly of a set of additional properties (those certified by atomic credentials in $\mathcal{D}$). Therefore, while each disclosure $\mathcal{D}$ is a subset of properties and credentials, not all subsets of properties and credentials represent a proper disclosure, as formally defined in the following.

DEFINITION 3.2 (DISCLOSURE). *Let $\mathcal{C}$ and $\mathcal{P}$ be the set of credentials and properties, respectively, in the client portfolio. A disclosure $\mathcal{D}$ of the portfolio is a set $\{d_1,\ldots,d_j\}$ of elements of the form $d=c.\{p_1,\ldots,p_i\}$ such that the following conditions hold:*

1. $\forall d \in \mathcal{D} \implies \{p_1,\ldots,p_i\} \subseteq properties(c)$ (certifiability);
2. $\forall d \in \mathcal{D}$ *s.t. c is atomic* $\implies \{p_1,\ldots,p_i\}=properties(c)$ (atomicity).

The above definition states that a subset of the client portfolio represents a disclosure if: *1)* each disclosed property is certified by (at least) a credential (*certifiability*); and *2)* if a property of an atomic credential is disclosed, all the properties in the credential are disclosed (*atomicity*).

4. 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 or different credentials enjoy therefore different values in this respect.

We capture such intuitive approach in expressing information privacy preferences as *sensitivity labels* that the user can associate with the different components (or combinations thereof) in her portfolio and therefore how much the user values their disclosure. A user may also specify different preferences for different servers, by assigning a different label to each component in its portfolio, depending on the server requesting the release. The request is then evaluated on the instance of the labels determined by the server with whom the user is interacting. For simplicity and without

loss of generality, in the following we will consider a user specifying a unique set of labels for all servers.

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

Sensitivity of properties and credentials. The first step for the user to specify how much she values information in her portfolio (to the goal of minimizing information disclosure when interacting with others) is to associate a sensitivity label λ 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, $\lambda(\texttt{eMail}) > \lambda(\texttt{Name})$.
- $\lambda(c)$: 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 an FBI certificate including properties `Name` and `Address`. 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 the demographic information of the user. 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.

Sensitivity of associations. The sensitivity label associated with each portfolio element defines how much the user values the release of the specific element. Of course, in the case of credentials, the value of the release of the complete credential is the composition of the labels of all the elements involved, that is, the credential existence and all the properties within the credential. In general, releasing a set of elements entails a sensitivity corresponding to the combination of the sensitivity of all the elements involved. There are however cases where merging some elements might produce an information release that 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 a user `Name`, her `Address`, and one of her credit cards (e.g., *myVISA*) can be considered more sensitive than the composition of the sensitivity labels of the three. In fact, not only it discloses the three values, but also the fact that they are in association (i.e., the address and credit card of the user). $\lambda(\{\texttt{Name}, \texttt{Address}, myVISA\})$ 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 a complete `Address` 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 address includes the country where the user lives. Hence, releasing the address together with the country does not release additional information. $\lambda(\{\texttt{Address}, \texttt{Country}\})$ expresses the sensitivity to be *removed* when the two pieces of information are joined. In this specific example, there is essentially a functional dependency between the two elements and therefore $\lambda(\{\texttt{Address}, \texttt{Country}\}) = -\lambda(\texttt{Country})$. In general, $\lambda(a)$ can take any negative value, provided that its absolute value be at most equal to the composition of all the elements but the most sensitive in a. In fact, an association has a sensitivity that is at least the one of the most sensitive element.

As for the specification of the sensitivity label of an association, we note that the user could explicitly define it (specifying the *additional* sensitivity given by the joint release of the involved properties), 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 involved. For instance, with reference to the portfolio in Figure 2, assume that $\lambda(\texttt{Name})=1$, $\lambda(\texttt{Address})=5$, and $\lambda(myVISA)=3$. If the sensitivity associated with the combined release of `Name`, `Address`, and *myVISA* is equal to 14, the sensitivity label of the association $\lambda(\{\texttt{Name}, \texttt{Address}, myVISA\})$ will be computed as $14-\lambda(\texttt{Name})-\lambda(\texttt{Address})-\lambda(myVISA) = 5$.

Disclosure constraints. In addition to specifying the sensitivity labels associated with credentials and properties in the portfolio, or combinations thereof, the user may want to specify additional constraints that cannot be simply expressed with a sensitivity label. We consider two kinds of constraints.

- *Forbidden views.* Some associations of the portfolio elements 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 forbidden views as a set of properties and/or credentials that should never be released together. For instance, a user might have in its portfolio both a real `Name` and a `NickName`, each one with

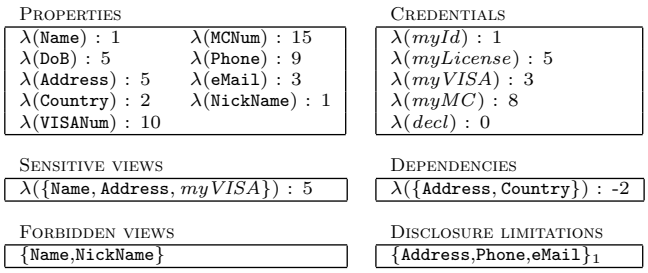

Figure 3: Portfolio sensitivity specification

a sensitivity label (to be considered when the element is released), but their association should never be disclosed.

- *Disclosure limitations.* In some cases, the user might wish to put a limitation of the kind *at most n of these elements* on the release of properties and credentials. Given a disclosure limitation $a \subseteq \mathcal{C} \cup \mathcal{P}$, we denote with a_n the constraint limiting the disclosure to at most n elements in a. For instance, a user may specify disclosure limitation {Address,Phone,eMail}$_1$, meaning that at most one among its contacts (i.e., Address, Phone, and eMail) can be released. Note that the forbidden view constraints above can be seen as a specific kind of disclosure limitation constraints, stating that at most $n-1$ among n portfolio elements can be released. However, we prefer to maintain their modeling separate, as we believe forbidden views to be a case deserving special consideration.

EXAMPLE 4.1. *Figure 3 illustrates an example of a labeling function λ for the portfolio illustrated in Figure 2. The figure also reports one sensitive view* {Name, Address, myVISA}*, since the combined release of the specified elements could result in unsolicited advertisement; one dependency* {Address, Country}*, since the knowledge of the complete address also discloses the country; one forbidden view* {Name,NickName}*, since the client does not want to release both its name and its nickname; and one disclosure limitation* {Address,Phone,eMail}$_1$*, since the client wants to release at most one of its contacts.*

The sensitivity label $\lambda(\mathcal{D})$ of a disclosure $\mathcal{D}$ is computed by composing the labels of the credentials and properties in $\mathcal{D}$, and of the exposed associations. An association a is exposed by $\mathcal{D}$ if all properties and all credentials in a appear in $\mathcal{D}$. Also, a disclosure is *valid* if it does not violate any disclosure constraint specified by the client. We note that only valid disclosures can be released.

Given a server request $\mathcal{R}$, the client is interested in determining, if it exists, a valid disclosure $\mathcal{D}$ that satisfies $\mathcal{R}$, while minimizing the sensitivity label of the disclosure. A disclosure $\mathcal{D}$ satisfies a request $\mathcal{R}$ if at least one of the simple requests R in $\mathcal{R}$ is satisfied by $\mathcal{D}$. A simple request R is satisfied by $\mathcal{D}$ iff, for each term r in R, there exists an element $c.\{p_1,\ldots,p_i\} \in \mathcal{D}$ that satisfies r (Definition 3.1). Formally, the problem of computing a minimum valid disclosure is formulated as follows.

PROBLEM 4.1 (MIN-DISCLOSURE). *Given a set $\mathcal{C}$ of credentials and a set $\mathcal{P}$ of properties composing the client portfolio, a set $\mathcal{A}$ of associations, a set $\mathcal{V}$ of forbidden views, a set $\mathcal{L}$ of disclosure limitations, a labeling function λ, and a server request $\mathcal{R}$, find a* minimum disclosure *$\mathcal{D}$ of the portfolio w.r.t. $\mathcal{R}$ that satisfies the following requirements:*

- *$\mathcal{D}$ is valid w.r.t. $\mathcal{V}$ and $\mathcal{L}$;*
- *$\mathcal{D}$ satisfies $\mathcal{R}$;*
- *$\nexists$ a valid disclosure $\mathcal{D}'$ s.t. $\mathcal{D}'$ satisfies $\mathcal{R}$ and $\lambda(\mathcal{D}') < \lambda(\mathcal{D})$.*

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

THEOREM 4.1. *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 and the set $\mathcal{C}$ of credentials 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. Each element in the finite set U translates to a credential-independent property p with sensitivity label $\lambda(p)=0$. Any subset $s \in S$ of U translates to an atomic credential c in $\mathcal{C}$ with sensitivity label $\lambda(c)=1$ certifying all the properties corresponding to the elements in s. Let us now consider request $\mathcal{R}=\bigwedge_{p \in U} *.type(p)$, where $*$ is the root of the credential type hierarchy. A minimum disclosure $\mathcal{D}$ corresponds to a solution S' for the corresponding minimum set cover problem. In particular, each disclosed credential corresponds to a subset in S'. Hence, the Min-Disclosure problem is NP-hard. □

EXAMPLE 4.2. *Let us consider a request* $\mathcal{R}$= $r_1 \wedge r_2$ = id.{Name,Address} ∧ cc.{Name,CCNum}. *The first term r_1 can be satisfied only by releasing atomic credential* myId. *In fact, credential* myLicense*, even if of type* id*, does not certify property* Address. *The second term r_2 can be satisfied by releasing either* myVISA *or* myMC. *The valid disclosure that satisfies $\mathcal{R}$ with minimum sensitivity label is* $\mathcal{D}$={myId.{Name,DoB,Address}, myVISA.{Name,VISANum}}. *Note that $\mathcal{D}$ releases property* DoB *that is not explicitly required by the server. Also, it exposes sensitive view* {Name,Address,myVISA}. *The sensitivity label of $\mathcal{D}$ is therefore* $\lambda(\mathcal{D})=\lambda(\text{myId}) + \lambda(\text{Name}) + \lambda(\text{DoB}) + \lambda(\text{Address}) + \lambda(\text{myVISA}) + \lambda(\text{VISANum}) + \lambda(\{\text{Name},\text{Address},\text{myVISA}\}) = 30$.

In the following, we present an approach for solving the Min-Disclosure problem.

5. COMPUTING A MINIMUM DISCLOSURE

Our approach to solve the Min-Disclosure problem (Problem 4.1) is based on its translation into an instance of the Weighted Max-SAT problem and on the availability of SAT

solvers, known to be able to efficiently solve large problems. This translation interprets credentials and properties as Boolean variables, and associations, disclosure constraints, and the server request as clauses. The Weighted Max-SAT problem can be formulated as follows: *given a set of clauses, find a truth assignment that maximizes the sum of weights of satisfied clauses.*

Any instance of the Weighted Max-SAT problem modeling an instance of our problem consists of two parts: *i)* a set of clauses representing the client portfolio and its sensitivity specifications (Section 5.1); and *ii)* a clause specifying the server request (Section 5.2). Note that the first part is generated only once at initialization time, and is updated if credentials are inserted into or removed from the portfolio. The second part is instead dynamically generated at each request. In the remainder of this section, we describe how these clauses are defined.

5.1 Client Portfolio

We define two sets of Boolean variables, representing credentials and properties in the client portfolio. A truth assignment to the variables of the problem represents a disclosure $\mathcal{D}$, where only credentials and properties represented by true variables are released.

Since not all truth assignments correctly represent a disclosure, we define a set of clauses that implements the certifiability and atomicity properties in Definition 3.2.

- *Certifiability*: each disclosed property must be certified by (at least) a credential, that is, the credential existence is also disclosed. We need a clause for each property $p \in \mathcal{P}$ stating that if $p=1$ then there exists at least a credential c such that $c=1$ and $p \in properties(c)$. Formally:

$$\neg p \vee (p \wedge (\bigvee_{c \in \mathcal{C}: p \in properties(c)} c))$$

- *Atomicity*: if an atomic credential is disclosed, all its properties are disclosed. We need a clause for each atomic credential $c \in \mathcal{C}$ stating that if $c=1$, then for all $p \in properties(c)$, $p=1$. Formally:

$$\neg c \vee (c \wedge (\bigwedge_{p \in properties(c)} p))$$

All the clauses describing the constraints on the portfolio structure are mandatory (*hard clauses*), that is, any solution must satisfy all of them to be considered a disclosure. As a consequence, we assign them a special weight value $\top$ [10] that can be interpreted as $+\infty$: any truth assignment must satisfy all of them.

The sensitivity label of each property, credential, and association in the portfolio is modeled through a weighted clause (*soft clauses*). We note that the Weighted Max-SAT problem is aimed at maximizing the weight of satisfied clauses, while our problem (Problem 4.1) minimizes the weight of disclosed properties and credentials, and of exposed associations. We then define a negative clause for each property, credential, and association as follows.

- *Property*: $\forall p \in \mathcal{P}$: $\neg p$.
- *Credential*: $\forall c \in \mathcal{C}$: $\neg c$.
- *Association*: $\forall a = \{p_i, \ldots, p_j, c_k, \ldots, c_l\} \in \mathcal{A}$: $\neg(p_i \wedge \ldots \wedge p_j \wedge c_k \wedge \ldots \wedge c_l)$.

The weight of each soft clause corresponds to the sensitivity label of the property, credential, or association it represents. The weight of a disclosure is then obtained as the sum of the weights of violated soft clauses (i.e., released credentials and properties, and exposed associations), which is minimized.

Forbidden views and disclosure limitations are instead modeled as hard clauses, since they cannot be violated by any truth assignment. In particular, we define a clause for each forbidden view and a clause for each disclosure limitation, as follows.

- *Forbidden view*: $\forall v = \{p_i, \ldots, p_j, c_k, \ldots, c_l\} \in \mathcal{V}$: $\neg(p_i \wedge \ldots \wedge p_j \wedge c_k \wedge \ldots \wedge c_l)$.
- *Disclosure limitation*: $\forall l = \{p_i, \ldots, p_j, c_k, \ldots, c_l\}_n \in \mathcal{L}$:
  $S = \{s \in 2^l : |s| \leq n\}$, $\bigvee_{s \in S} (\bigwedge_{x \in s} x \bigwedge_{x \in (l \setminus s)} \neg x)$
  if we consider $n = 1$, the clause reduces as follows:
  $\bigvee_{x \in l} (x \bigwedge_{x_i \in (l \setminus \{x\})} \neg x_i)$.

We note that the number of clauses modeling the client portfolio is linear in the number of properties, credentials, associations, and disclosure constraints composing it. In fact, we need to define $|\mathcal{P}|$ certifiability clauses, at most $|\mathcal{C}|$ atomicity clauses, $|\mathcal{P}|+|\mathcal{C}|+|\mathcal{A}|$ soft clauses representing sensitivity labels, and $|\mathcal{L}|+|\mathcal{V}|$ hard clauses representing disclosure constraints. Therefore, the number of clauses necessary to describe the client portfolio is $O(|\mathcal{P}|+|\mathcal{C}|+|\mathcal{A}|+|\mathcal{L}|+|\mathcal{V}|)$.

Figure 4 illustrates the clauses modeling the portfolio in Figure 2, considering the sensitivity specification in Figure 3.

5.2 Server Request

Given a server request $\mathcal{R}=R_1 \vee \ldots \vee R_n$ where each simple request is of the form $R=r_1 \wedge \ldots \wedge r_m$, we first need to translate each term r into an equivalent Boolean formula that enforces Definition 3.1. Basically, the Boolean formula representing term r is obtained as the disjunction of all credentials c in the client portfolio such that $type(c) \preceq_{isa} type(r)$, in conjunction with all properties p in $properties(c)$ such that the type of p appears in $properties(r)$. Formally, the Boolean expression translating term r is as follows.

$$\bigvee_{c \in \mathcal{C}: type(c) \preceq_{isa} type(r)} c \wedge (\bigwedge_{\substack{p \in properties(c), \\ pt \in properties(r): \\ type(p)=pt}} p)$$

Note that a credential c, even if it represents an instance of $type(r)$, is not included in the clause, if c does not certify at least one property for each of the types specified in r. This preliminary check is implemented during the translation of our problem into a Weighted Max-SAT instance.

The clause representing the whole request is obtained by adequately composing the translated terms, according to $\mathcal{R}$.

$$\bigvee_{R \in \mathcal{R}} (\bigwedge_{r \in R} (\bigvee_{c \in \mathcal{C}: type(c) \preceq_{isa} type(r)} c \wedge (\bigwedge_{\substack{p \in properties(c), \\ pt \in properties(r): \\ type(p)=pt}} p)))$$

This clause is hard and must then be satisfied by any truth assignment. In fact, a truth assignment that does not satisfy this clause represents a disclosure that does not satisfy the

Certifiability

Clauses	Weight
$(\neg\texttt{Name})\vee(\texttt{Name}\wedge(myId\vee myLicense\vee myVISA\vee myMC\vee decl))$	$\top$
$(\neg\texttt{DoB})\vee(\texttt{DoB}\wedge(myId\vee myLicense\vee decl))$	$\top$
$(\neg\texttt{Address})\vee(\texttt{Address}\wedge(myId\vee decl))$	$\top$
$(\neg\texttt{Country})\vee(\texttt{Country}\wedge(myLicense\vee decl))$	$\top$
$(\neg\texttt{VISANum})\vee(\texttt{VISANum}\wedge(myVISA\vee decl))$	$\top$
$(\neg\texttt{MCNum})\vee(\texttt{MCNum}\wedge(myMC\vee decl))$	$\top$
$(\neg\texttt{Phone})\vee(\texttt{Phone}\wedge decl)$	$\top$
$(\neg\texttt{eMail})\vee(\texttt{eMail}\wedge decl)$	$\top$
$(\neg\texttt{NickName})\vee(\texttt{NickName}\wedge decl)$	$\top$

Atomicity

Clauses	Weight
$(\neg myId)\vee(myId\wedge\texttt{Name}\wedge\texttt{DoB}\wedge\texttt{Address})$	$\top$
$(\neg myVISA)\vee(myVISA\wedge\texttt{Name}\wedge\texttt{VISANum})$	$\top$
$(\neg myMC)\vee(myMC\wedge\texttt{Name}\wedge\texttt{MCNum})$	$\top$

Properties

Clauses	Weight
$\neg\texttt{Name}$	1
$\neg\texttt{DoB}$	5
$\neg\texttt{Address}$	5
$\neg\texttt{Country}$	2
$\neg\texttt{VISANum}$	10
$\neg\texttt{MCNum}$	15
$\neg\texttt{Phone}$	9
$\neg\texttt{eMail}$	3
$\neg\texttt{NickName}$	1

Credentials

Clauses	Weight
$\neg myId$	1
$\neg myLicense$	5
$\neg myVISA$	3
$\neg myMC$	8

Sensitive views

Clauses	Weight
$\neg(\texttt{Name}\wedge\texttt{Address}\wedge myVISA)$	5

Dependencies

Clauses	Weight
$\neg(\texttt{Address}\wedge\texttt{Country})$	-2

Forbidden views

Clauses	Weight
$\neg(\texttt{Name}\wedge\texttt{NickName})$	$\top$

Disclosure Limitations

Clauses	Weight
$(\texttt{Address}\wedge\neg\texttt{Phone}\wedge\neg\texttt{eMail})\vee(\neg\texttt{Address}\wedge\texttt{Phone}\wedge\neg\texttt{eMail})\vee(\neg\texttt{Address}\wedge\neg\texttt{Phone}\wedge\texttt{eMail})$	$\top$

Figure 4: SAT clauses for the client portfolio

request and therefore does not grant access to the service. The weight associated with this clause is then $\top$.

Example 5.1. *Let us consider the request $\mathcal{R}$ in Example 4.2, its translation into a clause for the Weighted Max-SAT problem requires to previously reformulate r_1 and r_2, as follows:*

- r_1=id.{Name,Address} *is translated as:* myId∧Name∧Address*;*
- r_2=cc.{Name,CCNum} *is translated as:* (myVISA∧Name∧VISANum)∨(myMC∧Name∧MCNum).

It is important to note that the computation of the clause modeling the server request should be efficient, to avoid delays in the access to the service. In fact, this translation must be executed any time the client needs to acquire a service. We note however that this is the unique clause that needs to be computed at request time.

5.3 Session Management

In the above 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 information already disclosed to the server and ensure minimality of the information released overall, rather than the specific information released at each server request (i.e., at each accessed functionality) within the session. At each request, information released in previous requests within the session needs to be considered, to ensure both minimal information release to the server as well as the respect of disclosure constraints (i.e., forbidden views and disclosure limitations). For instance, with reference to Example 4.1, suppose a client discloses its NickName to the server. By providing stateful capability, the release of Name will be prohibited, since otherwise the forbidden view would be violated. As another 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 released to the server for the flight reservation, to minimize the information released overall. For instance, if requested a credit card or an id document, she might want to stick with the ones just released. Consideration of information already released is of outmost importance, in particular for disclosure constraints that should be evaluated with respect to the information communicated in the whole session (as opposed to the specific individual interaction steps).

Our Weighted Max-SAT formulation of the problem allows to address the issue above, providing the ability of maintaining the *communication state* of the client with the server. As a matter of fact, information already disclosed to the server can be simply expressed by injecting in the instance of Weighted Max-SAT problem to be evaluated with respect to a request, a set of clauses of the form $\langle p, \top\rangle$ ($\langle c, \top\rangle$, resp.) for each property (credential, resp.) already disclosed. In such a way, we capture properties and credentials previously disclosed and therefore they are taken into consideration in the calculation of the weight of the truth assignment, as well as in the evaluation of the disclosure constraints.

Example 5.2. *Consider a Web-based reservation system that permits, within a unique session, to book a flight and rent a car. Given the server request $\mathcal{R}$=$r_1\wedge r_2$=*id.{Name,Address} ∧ cc.{Name,CCNum}*, the minimum valid disclosure $\mathcal{D}$=*{myId.{Name,DoB,Address}, myVISA.{Name,VISANum}}* in Example 4.2 permits the user to book her flight. As soon as the user tries to rent a car, the server issues request $\mathcal{R}'$=*driver_license.{Name,Country} ∨ ∗.{NickName}*, where the second term applies to registered users. This request can be satisfied by disclosing either $\mathcal{D}_1$=*myLicense.{Name,Country}* or $\mathcal{D}_2$=*decl.{NickName}*. If we do not consider the communication state, $\lambda(\mathcal{D}_1)$=8 and $\lambda(\mathcal{D}_2)$=1, thus the client discloses $\mathcal{D}_2$. On the contrary, if we consider the communication state including the disclosure of $\mathcal{D}$, $\lambda(\mathcal{D}_1)$=5, since credential independent property* Name

has already been released by $\mathcal{D}$ and the disclosure of Country *activates dependency* {Address,Country}. *Disclosure $\mathcal{D}_2$, combined with $\mathcal{D}$, violates forbidden view* {Name,NickName} *and therefore is not valid. The client then discloses $\mathcal{D}_1$.*

Note that the stateful session approach described above can also be used to support the consideration of a generic history of previous communications, providing the client with the ability to remember previous releases to the server and therefore minimizing the information released overall within a generic time/operation window set by the client. The simple clauses above representing the information already released to each server would be injected in the formulation of the Weighted Max-SAT problem. Note that the clauses describing the user portfolio (certifiability, atomicity, sensitivity, and disclosure constraints) do not have to change.

6. EXPERIMENTAL EVALUATION

To prove the effectiveness and evaluate the efficiency of the solution proposed in the previous sections, we implemented a prototype written in C/C++, realizing the translation of any instance of our Min-Disclosure problem (Problem 4.1) into an equivalent instance of the Weighted Max-SAT problem as described in Section 5. The algorithm receives as input the set of credentials and properties in the client portfolio, a set of associations, a set of disclosure constraints, a labeling function λ, and a request $\mathcal{R}$, and computes a set of clauses that are given as input to the Yices SAT solver (http://yices.csl.sri.com). If $\mathcal{R}$ can be satisfied, the Yices SAT solver returns a valid disclosure $\mathcal{D}$ that satisfies $\mathcal{R}$ and minimizes the sensitivity label of $\mathcal{D}$.

6.1 Translation of the Problem

The Yices SAT solver tool does not support clauses with negative weights and therefore we need an alternative formulation to represent dependencies. To this purpose, we take advantage of the fact that the sensitivity of a release cannot be negative, since the sensitivity of the disclosure of a set of properties and credentials cannot be lower than the most sensitive element disclosed (see Section 4). Given a dependency a={p_i,...,p_j,c_k,...,c_l}, the sensitivity label of each property/credential in a is not modified, provided at least one property/credential in a is not disclosed. On the contrary, the sensitivity of the disclosure of the whole set of properties and credentials composing a needs to consider also the negative label $\lambda(a)$. We therefore reformulate the clauses modeling the sensitivity label of each property/credential in a and the clause modeling the sensitivity label of the dependency itself as follows.

- Each clause $\neg p$ such that $p \in a$ is modified as:

  $$\neg(p \wedge (\bigvee_{p_i \in a : p_i \neq p} \neg p_i \bigvee_{c \in a} \neg c))$$

  with weight $\lambda(p)$;

- Each clause $\neg c$ such that $c \in a$ is modified as:

  $$\neg(c \wedge (\bigvee_{p \in a} \neg p \bigvee_{c_i \in a : c_i \neq c} \neg c_i))$$

  with weight $\lambda(c)$;

From

Clauses	Weight
¬Address	5
¬Country	2
¬(Address∧Country)	-2

To

Clauses	Weight
¬(Address∧¬Country)	5
¬(Country∧¬Address)	2
¬(Address∧Country)	5

Figure 5: Modified clauses

- Clause $\neg(p_i \wedge \ldots \wedge p_j \wedge c_k \wedge \ldots \wedge c_l)$, with negative weight $\lambda(a)$, is modified by assigning it a weight equal to $\lambda(a)+\lambda(p_i)+\ldots+\lambda(p_j)+\lambda(c_k)+\ldots+\lambda(c_l)$.

Note that, if the same property/credential is involved in more than one dependency, the clause modeling the sensitivity label of the property/credential must consider all the dependencies in which it appears. Analogously, the clause modeling the sensitivity label of each dependency must consider the other ones.

It is possible to see that, if the dependencies of the considered instance are disjoint, the number of clauses of the corresponding Weighted Max-SAT instance is not affected by the transformation described above. In the case of non-disjoint dependencies, this number increases with the number of dependencies with at least a common property/credential.

EXAMPLE 6.1. *Let us consider the set of clauses modeling the client portfolio illustrated in Figure 4 that includes clause* ¬(Address∧Country) *with negative weight. To adopt Yices SAT-solver, it is necessary to reformulate the clauses that model the sensitivity labels of properties* Address *and* Country*, and the clause modeling the dependency as illustrated in Figure 5.*

6.2 Experimental Results

Experiments have been run on a PC with two Intel Xeon Quad 2.0GHz L3-4MB processors, 12GB RAM, four 1-Tbyte disks, and a Linux Ubuntu 9.04 operating system. We considered four different portfolio configurations, where 50%, 25%, 10%, and 0% of the credentials are non-atomic. Figure 6(a) illustrates the average time necessary to generate the clauses modeling the client portfolio, with a number of credentials between 5 and 35. As clear from the figure, the time necessary to generate the clauses representing the portfolio ranges between 0ms and 3ms. From the computed results, we also note that the number of Boolean variables is between 21 and 127 and that the number of (hard and soft) clauses generated is between 34 and 251.

For each portfolio configuration, we randomly generated 10 requests, composed of 2 to 6 terms, where each term refers to a credential type and may include multiple conditions. Requests were expanded as described in Section 3, resulting in considerably large clauses. We then measured the time necessary for: *i)* the translation of the request into a hard clause; *ii)* the computation of a truth assignment to Boolean variables through the Max-SAT solver; and *iii)* the translation of the truth assignment into a disclosure. Figure 6(b) illustrates the average time necessary for the evaluation of a request. As expected, the time necessary for the evaluation of a request increases with the portfolio size and does not depend on the number of non-atomic credentials. In all the experiments, the evaluation time remains below 5ms.

To assess the efficiency of our solution, we compared it with a brute-force exhaustive algorithm for computing a

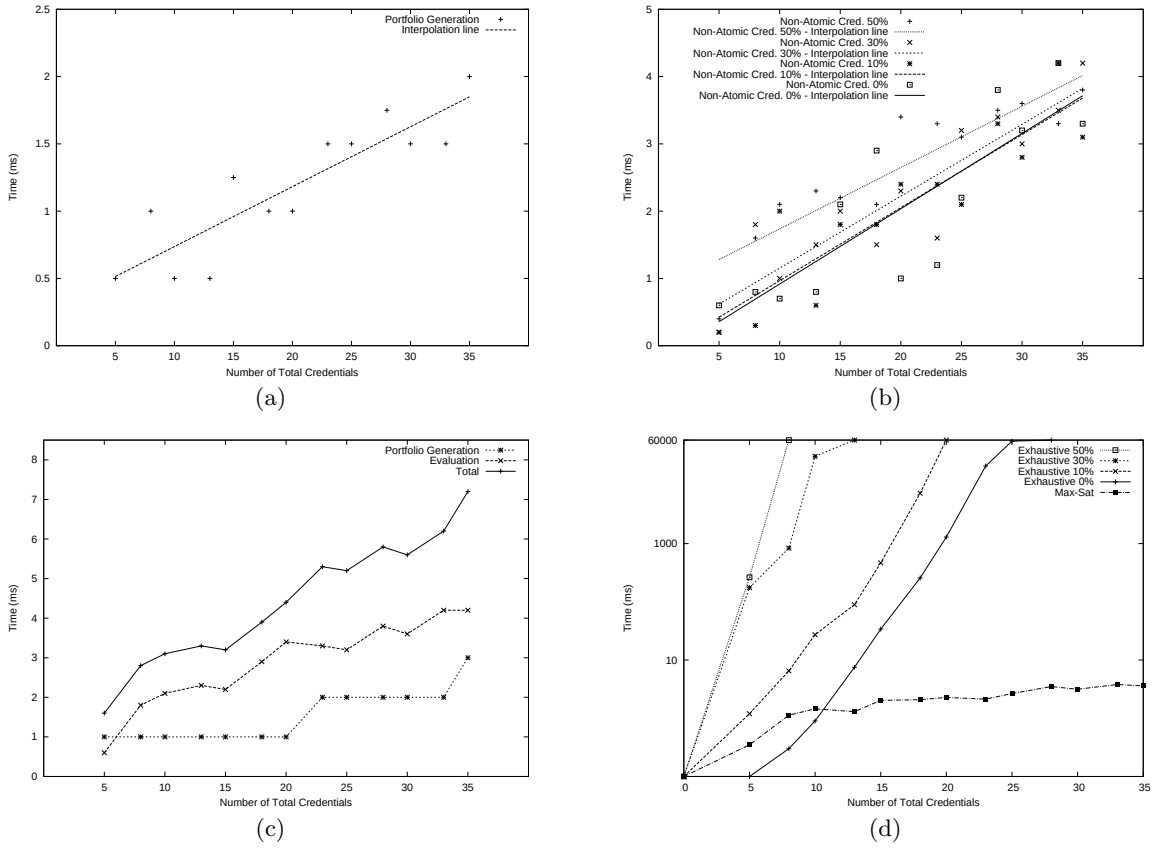

Figure 6: Experimental results

minimum disclosure. To this purpose, we calculate the total time our algorithm needs to produce a solution, as the sum of the time necessary to generate the clauses modeling the client portfolio and the average time needed for the evaluation of a request. Figure 6(c) shows the total time necessary to compute a minimum disclosure in the worst case scenario. Figure 6(d) compares the performance of our solution with the brute-force algorithm. As expected, the brute-force algorithm becomes quickly infeasible, with a strong dependence on the number of non-atomic credentials. On the contrary, the approach that uses the Max-SAT solver can compute, for all the considered configurations, a solution in less than 10ms.

7. RELATED WORK

Research on credential-based access control (e.g., [3, 5, 11, 14, 16, 19]) primarily focused on server side issues and proposed solutions for controlling access to resources, specifying and enforcing policies, and enabling negotiation strategies. 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 exploit emerging technologies (e.g., SAML [1], OpenID [9], and anonymous credentials [6, 7]) for determining which credentials and/or properties release to minimize the sensitive information communicated to the server. In the literature only few works have addressed this issue. In [15] the authors describe a solution that limits the release of personal data in trust negotiation. This approach takes advantage of the proposal in [4], which uses a Merkle hash tree structure to selectively release properties within credentials (or proofs over them). The work in [15] however permits neither to assign sensitivity labels to elements in the portfolio nor to define associations and disclosure constraints. 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. [12] 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. [18] 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 [18], our solution permits 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 of portfolio components have been first proposed in [2]. This paper considerably extends this previous work by supporting a richer approach for the specification of sensitivity labels, allowing also for negative sensitivity, capturing data dependencies and implications, as well as for additional constraints. Also, it 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. The approach results therefore more expressive, as well as more efficient, than the graph-based modeling and the heuristic approach in [2].

8. CONCLUSIONS

Credentials promise to play a crucial role in future Web systems, for the enhanced user convenience and security they offer. In this paper, we presented an approach for supporting user privacy preferences in a credential-based interaction system, thus helping to manage what would otherwise be a significant obstacle to the deployment of credential-based solutions. Our approach can be easily integrated with the client environment (e.g., within the browser) and can efficiently identify, even for large portfolios, the set of credentials and properties that minimizes the exposure of personal information. Our proposal provides a simple and expressive solution for expressing and enforcing users privacy preferences, and can therefore provide a foundation for the construction of a user-centered privacy-conscious future Internet experience.

9. ACKNOWLEDGMENTS

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).

10. 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. 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).

[4] D. Bauer, D. Blough, and D. Cash. Minimal information disclosure with efficiently verifiable credentials. In *Proc. of the 4th ACM Workshop on Digital Identity Management (DIM 2008)*, Alexandria, Virginia, USA, October 2008.

[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] F. Heras, J. Larrosa, and A. Oliveras. MiniMaxSAT: a new weighted Max-SAT solver. In *Proc. of the 10th International Conference on Theory and Applications of Satisfiability Testing (SAT 2007)*, Lisbon, Portugal, May 2007.

[11] 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.

[12] 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.

[13] A. Lee and M. Winslett. Towards an efficient and language-agnostic compliance checker for trust negotiation systems. In *Proc. of the 2008 ACM Symposium on InformAtion, Computer and Communications Security (ASIACCS 2008)*, March.

[14] A. 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.

[15] F. Paci, D. Bauer, E. Bertino, D. Blough, A. Squicciarini, and A. Gupta. Minimal credential disclosure in trust negotiations. *Identity in the Information Society*, 2(3):221–239, December 2009.

[16] T. Ryutov, L. Zhou, C. Neuman, T. Leithead, and K. 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.

[17] B. Smith, K. Seamons, and M. Jones. Responding to policies at runtime in trustbuilder. In *Proc. of the 5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004)*, Yorktown Heights, NY, USA, June 2004.

[18] D. Yao, K. Frikken, M. 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.

[19] T. Yu, M. Winslett, and K. 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.