Policy Negotiation for Co-owned Resources in Relationship-Based Access Control

Pooya Mehregan and Philip W. L. Fong
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada

ABSTRACT

The collaborative nature of content development has given rise to the novel problem of multiple ownership in access control, such that a shared resource is administrated simultaneously by co-owners who may have conflicting privacy preferences and/or sharing needs. Prior work has focused on the design of unsupervised conflict resolution mechanisms.

Driven by the need for human consent in organizational settings, this paper explores interactive policy negotiation, an approach complementary to that of prior work. Specifically, we propose an extension of Relationship-Based Access Control (ReBAC) to support multiple ownership, in which a policy negotiation protocol is in place for co-owners to come up with and give consent to an access control policy in a structured manner. During negotiation, the draft policy is assessed by formally defined availability criteria: policy satisfiability, resiliency and feasibility, which all belong to the second level of the polynomial hierarchy. We devised two algorithms for verifying policy satisfiability, both employing a modern SAT solver for solving subproblems. The performance is found to be adequate for mid-sized organizations.

Keywords

Multiple Ownership; ReBAC; Negotiation; Availability.

1. INTRODUCTION

One of the most interesting recent developments in the area of Access Control is the study of ***multiple ownership***. Specifically, the many and diverse scenarios of resource sharing in social computing applications have brought about a fundamental change in our understanding of Discretionary Access Control (DAC). In the classical formulations of DAC (e.g., the Graham-Denning model [11, 19]), every object has a primary administrator known as its ***owner***. Ownership, in the context of Access Control, is about administrative rights and not property rights. The owner is the user who has full administrative privilege to determine the access control policy of the object. While partial administrative privileges can be delegated, the delegatees are considered acting on behalf of the owner, and thus fully trusted. This classical view, however, has been significantly revised due to the emergence of new access control requirements in social computing.

Social computing applications provide support for collaborative authoring of contents. Contents that are originally created by one user are enriched and annotated by other users. Examples of such features include liking, tagging and commenting. Some contents are even co-authored by multiple users. An example is the befriending operation, which creates a joint declaration of two users' relationship. An object of this kind has privacy bearing on multiple parties, and thus they all want to have a say on the access control policy of that object. These parties, who are called ***co-owners*** in this work, may have different or even conflicting privacy and availability preferences. In other words, co-owned objects are not the ones *accessed/shared* by multiple parties, but the ones that are *administered* by multiple parties. The study of multiple ownership involves the design and analysis of schemes for supporting co-owners in administrating access control policies in a collaborative manner.

There is now a significant and growing body of literature on multiple ownership [23, 26, 30, 24, 12, 13, 14, 15, 25, 21], most notable of which are the seminal work of Squicciarini *et al* [23, 24] and that of Hu *et al.* [12, 13, 14, 15]. A common theme in their works is to devise unsupervised conflict resolution mechanisms for reconciling the potentially conflicting privacy preferences and sharing needs of the various co-owners. These mechanisms are usually blackboxes in the sense that they are mathematically complex enough to escape comprehension by typical end-users [21].

In this work, we venture away from mainstream social media (Facebook, Twitter, Instagram, etc.), and study multiple ownership in the setting of ***organizational computing*** — medical records systems, documents sharing and co-authoring, business process support, etc. For instance, a patient, her treating clinicians and the hospital administration all share a privacy stake in a medical record. They co-own the record because they all want a say in the access control policy of the record. Again, colleagues from different departments who co-author a report share an interest in the access control of that report. As co-owners representing different departments, they may have different or even conflicting views on how the report shall be protected.

We propose in this work a framework for supporting multiple ownership in the family of Relationship-Based Access Control (ReBAC) models by Fong *et al.* [8, 10, 3, 22], through interactive policy negotiation with tool support. ReBAC is a general-purpose access control paradigm de-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SACMAT'16, June 05-08, 2016, Shanghai, China

© 2016 ACM. ISBN 978-1-4503-3802-8/16/06…$15.00

DOI: http://dx.doi.org/10.1145/2914642.2914652

signed for supporting organizational computing. It features a finer granularity of control over classical Role-Based Access Control (RBAC) models, and offers natural support for delegation of trust [8]. A first, large-scale implementation of ReBAC in an open-source medical records system was recently attempted [22]. We therefore ground the present work on ReBAC, as this forces us to confront the reality of fine-grained access control as well as trust delegation.

1.1 Rationale for Interactive Negotiation

A uniqueness of this work is the design and implementation of tool support for interactive negotiation of access control policies. We envision that co-owners engage in a conscious process of negotiating the access control policy of their co-owned objects. We fully acknowledge the importance of unsupervised conflict resolution schemes proposed in previous work, and we are in no way advocating interactive negotiation as a replacement for unsupervised conflict resolution. We, however, recognize the complementary role of interactive policy negotiation because of two reasons.

First, there is evidence that, in some circumstances, users feel the need to engage in policy negotiation. For example, the empirical study reported in [17] reveals that SNS users do negotiate boundary regulations. Even in the absence of much needed tool support, users find other methods to negotiate. For instance, they simply exchange emails to negotiate the suitability of a group photo before disclosing it. The authors of [17] advocate the need for supporting expectation discussion and privacy negotiation among co-owners *before* conflicts arise. The study reported in [2] also highlights the need for negotiation in photo sharing, and the fact that users make use of offline detours in the absence of tool-support for such negotiations. We believe that interactive negotiation empowers users to directly collaborate with one another. In many cases, conflicts occur because users are unaware of other co-owners' privacy preferences; otherwise, they display eagerness in accommodating the privacy preferences of other co-owners [17]. In collegial environments, such as those in organizational computing, interactive negotiation offers the opportunity for users to collaboratively arrive at a mutually agreeable policy.

Second, the accountability requirements of the application domain may demand the express consent of the co-owners in publishing information. In organizational settings such as a healthcare facility, in which co-owners may bear legal responsibilities for regulating information disclosure, conscious human consent in policy negotiation would be required. In addition, collaboration in organizational computing is usually highly structured, mirroring the structure of the organization. The number of co-owners are therefore much fewer than typical social computing applications. Interactive policy negotiation that reflects the accountability requirements of organizational computing is thus feasible.

Our position is that both unsupervised conflict resolution and interactive policy negotiation have their respective roles. Which approach to adopt must be determined on a case by case basis, depending on the application domain (social or organizational computing), nature of the co-ownership (number of co-owners), or accountability requirements (legal liability). In this work, we will study the less-explored design space of interactive negotiation, especially in the development of tools for supporting such negotiation.

1.2 Contributions

The concrete contributions of this work are the following:

1. An administrative model supporting multiple ownership is formulated for ReBAC (§3). Access control policies are specified in terms of graph patterns in order to facilitate policy negotiation and policy analysis. Administrative operations that require user consent and policy negotiation are explicitly identified.

2. A novel policy negotiation protocol is proposed for co-owners to arrive at and give consent to an access control policy (§4). Our design offers two unique features. First, structured revision of the policy under negotiation allows co-owners to express their privacy preferences through counteracting the policy components contributed by other co-owners. Second, the need for sharing is assessed against mechanically verified availability criteria. The two features jointly provide a balance of privacy and sharing.

3. Three formal criteria of availability (i.e., satisfiability, feasibility and resiliency) have been formulated for assessing negotiated policies (§6). These availability criteria are inspired by the analogous concepts recently proposed for workflow authorization systems [4, 28, 16, 5]. We show that all of the three criteria belong to the second level of the polynomial hierarchy [1, Chap. 5].

4. Focusing on the availability criterion known as policy satisfiability, we devise two decision procedures for assessing availability (§7). Rather than naively reducing policy satisfiability to Quantified Boolean Formula (QBF) satisfiability, our custom algorithms make use of a modern SAT solver to attack subproblems.

5. The performance of the two algorithms are evaluated in §8, in which we empirically justify the superiority of our custom algorithms over generic QBF solvers, compare the relative merits of the two algorithms, and demonstrate that their performance is adequate for mid-sized organizations of up to 100,000 users. To put this magnitude in perspective, Microsoft and Google employ 94,000 and 55,419 employees respectively.

2. RELATED WORK

The need for addressing the diverse privacy concerns of multiple privacy stakeholders has recently caught the attention of security researchers. The first work to address this problem is that of Squicciarini *et.al* [23]. Their work emphasizes the need to honour the privacy preferences of all co-owners. They used Mechanism Design (Clarke-Tax) and Game Theory in the setting of an auction to perform conflict resolution. Conflict resolution as proposed in this work is relatively complex for naive end-users. This shortcoming is addressed in their follow-up work [24], in which they apply majority-voting instead of the auction.

One of the early works which underlines the multiple ownership problem in online social networks is [26]. In this work, the privacy consequences of disregarding the co-owners' privacy preferences have been demonstrated by means of successful inference attacks.

In the seminal work of Hu and Ahn [12, 14], each co-owner submits her preferred privacy policy along with a number as how (s)he perceives the co-owned object to be sensitive. A conflict resolution mechanism is then employed to aggregate the preferred privacy policies and sensitivities of the co-owners into a single privacy policy. They are the

first to make explicit the connection between policy composition (e.g., XACML) and conflict resolution mechanisms for multiple-ownership. In their follow-up work [13], they address the need for sharing, which is usually overlooked in the face of privacy. A quantitative scheme is then proposed to trade off between privacy and the need for sharing. In another follow-up work [15], they propose a game-theoretic analysis of their proposed model, showing there exist a Nash Equilibrium (NE) for multiple ownership problem.

An adaptive conflict resolution mechanism is proposed in [25]. This mechanism estimates if a user would concede based on her willingness to change action. Adaptiveness is captured in our work in the negotiation protocol (§4).

Mehregan and Fong make the important observation that multiple-ownership often arises "accidentally" when an object is composite, and when co-owners have privacy interests over various parts of the composite objects [21]. Therefore, resolving conflicting privacy preferences is genuinely needed only when indivisible objects are involved. We therefore focus on indivisible (atomic) objects in this paper.

Our approach is also comparable to [30], in which co-owners collaboratively author a privacy policy for a co-owned object, using a mechanism akin to a virtual whiteboard. Policy components come in two types: *weak* and *strong*. Weak policies are negotiable, and strong ones are not. This work is unique in prior literature in that it is the only one that involves negotiation among co-owners in coming up with a policy for the co-owned object. Our work is therefore comparable to this work. The drawback for [30] is that, when a strong policy is introduced, nobody would be able to edit/remove it except the co-owner who has contributed it. In our work, users can counteract restrictive atomic policies with permissive atomic policies (§4).

The ReBAC/MO model of §3 is a descendant of the line of ReBAC models of [8, 10, 3, 9]. Here we depart from [8, 10, 3, 9], which employs modal logic (and its extension, hybrid logic) for the specification of ReBAC policy. Instead, we revive the suggestion of [7] (see also [10, §4]) in using birooted graph patterns as the basis of policy specification. In both this work and that of [7], using graph patterns for policy specification facilitates policy analysis (in our case availability analyses). Our novelty includes (a) the identification of co-owners in atomic policies, (b) combination of privacy preferences of different co-owners, and (c) adoption of syntactic restrictions to facilitate policy negotiation and availability analyses.

The three availability criteria formulated in this work are inspired by the recent, exciting advancement of availability analyses in workflow authorization systems: satisfiability [4, 28, 5], resiliency [28], feasibility [16]. Compared to the analogous notions in the literature, our definitions are novel on three counts. First, they are defined to assess the availability of a resource in a ReBAC system, rather than the ease in which a workflow specification can be instantiated. Second, resiliency and feasibility are defined here via the notion of graph mutation. This is unique in comparison to workflow resiliency and feasibility, which are defined in terms of the departure of users. Third, the three notions of availability formulated here target the satisfaction of multiple-owners policies, whereas previous notions of availability attempt to satisfy security constraints in workflow specifications.

3. AN ADMINISTRATIVE MODEL OF MULTIPLE OWNERSHIP FOR REBAC

This section presents a ReBAC model that supports multiple ownership. §3.1 and §3.2 review notations previously defined in the ReBAC literature [7, 8, 10, 3]. §3.3 formalizes multiple-owners policies in terms of graph patterns. §3.4 presents a multiple-owners version of the ReBAC model, and highlights two prototypical administrative operations that are guarded by negotiation protocols.

3.1 Social Networks

A key component of the protection state for ReBAC is a social network [8, 10, 3]. Social networks are modelled as edge-labelled, directed graphs, in which vertices represents users, and edges represent interpersonal relationships. Edge labels denote the type of relationships that the edges signify (e.g., friend, parent, etc).

Definition 1 *Given a finite, non-empty set $\mathcal{L}$ of* **relation identifiers***, a* **graph** *G is a pair $\langle V, \{R_l\}_{l\in\mathcal{L}}\rangle$, where V is the set of vertices, and R is an indexed family of binary relations over the vertex set, such that each $R_l \subseteq V \times V$ is a binary relation denoting the edges of type l.*

We write $V(G)$, $L(G)$ and $R_l(G)$ to denote respectively the vertex set (V), the label set ($\mathcal{L}$), and the type-l edge set (R_l) of G.

A graph G_1 is a ***subgraph*** of G_2 iff $V(G_1) \subseteq V(G_2)$, $L(G_1) \subseteq L(G_2)$, and $R_l(G_1) \subseteq R_l(G_2)$ for every $l \in L(G_1)$. In that case, we write $G_1 \subseteq G_2$. A bijective function $f : V(G_1) \to V(G_2)$ is a ***graph isomorphism*** iff $L(G_1) = L(G_2)$, and for every $l \in L(G_1)$, for every $u, v \in V(G_1)$, we have $(u, v) \in R_l(G_1) \Leftrightarrow (f(u), f(v)) \in R_l(G_2)$. In such a case, we write $G_1 \simeq_f G_2$. G_1 and G_2 are ***isomorphic***, and we write $G_1 \simeq G_2$, if the above bijection f exists.

Given a finite, non-empty set $\mathcal{L}$ of edge labels, and a countably infinite universe $\mathcal{V}$ of vertices, we write $\mathcal{G}[\mathcal{V}, \mathcal{L}]$ to denote the set of all graphs G for which $V(G)$ is a finite subset of $\mathcal{V}$ and $L(G) \subseteq \mathcal{L}$.

3.2 Access Scenarios and Graph Patterns

Suppose G is the current social network. When a requester $v \in V(G)$ attempts to access an object owned by some user $u \in V(G)$, we have an ***access scenario*** characterized by these three components: G, u and v. The spirit of ReBAC is that this access scenario shall satisfy some graph-theoretic properties imposed by the access control policy in order for access to be granted. We model an access scenario by a ***birooted graph*** [7, 10], which is a graph plus the explicit identification of two vertices (not necessarily distinct).

Definition 2 *A* **birooted graph** *BG is a triple $G_{(u,v)}$, where G is a graph, and $u, v \in V(G)$ are called the* **roots** *of BG. More specifically, u and v are called the* **owner root** *and* **requester root** *respectively.*

In this work, access control policies are specified in terms of ***graph patterns***, which are also modelled as birooted graphs. More specifically, a graph-theoretic property corresponds to whether a graph pattern is "*contained*" in the access scenario. This matching between a graph pattern and an access scenario is formalized by the notion of subgraph isomorphism between birooted graphs. Suppose $BG_1 = G1_{(u_1,v_1)}$ is a graph pattern, and $BG_2 = G_{2(u_2,v_2)}$ is an

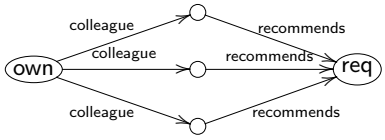
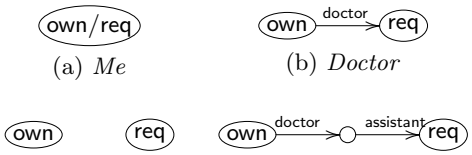


Figure 1: A birooted graph pattern that says "At least 3 colleagues of the owner recommend the requester." The owner and requester roots are labelled as own and req respectively.



(a) *Me* (b) *Doctor* (c) *Other* (d) *Assistant*

Figure 2: Four birooted graph patterns that are used for formulating policies in the clinic domain.

access scenario. BG_1 is a ***(birooted) subgraph*** of BG_2 iff $u_1 = u_2$, $v_1 = v_2$, and $G_1 \subseteq G_2$. Note that the two roots of a birooted subgraph must be identical to those of the birooted supergraph. A bijective function $f : V(G_1) \to V(G_2)$ is an ***isomorphism*** of BG_1 and BG_2 iff $f(u_1) = u_2$, $f(v_1) = v_2$, and $G_1 \simeq_f G_2$. We write $BG_1 \simeq BG_2$ iff such a bijection f exists. We write $BG_1 \lesssim BG_2$ iff there exists a birooted subgraph $BG_3 \subseteq BG_2$ such that $BG_1 \simeq BG_3$. That is, BG_1 is isomorphic to some subgraph of BG_2.

We write $\mathcal{B}[\mathcal{V}, \mathcal{L}]$ to denote the set of all birooted graphs $G_{(u,v)}$ for which $G \in \mathcal{G}[\mathcal{V}, \mathcal{L}]$. As a convention, we specify graph patterns using birooted graphs from $\mathcal{B}[\mathbb{N}, \mathcal{L}]$ ($\mathbb{N}$ is the natural number set).

Example 1 *Fig. 1 depicts a graph pattern asserting that the owner has at least three distinct colleagues who recommend the requester. Such a graph pattern can be used for expressing delegation of trust as well as level of trust.*

The next example illustrates how a policy can be specified by a disjunctive combination of graph patterns.

Example 2 *Consider a clinic domain, in which vertices are users, a* doctor *edge relates a patient to her family doctor, and an* assistant *edge relates a doctor to his assistant. The four graph patterns in Fig. 2 can serve as building blocks for specifying the policies below.*

- *"Only the owner can access." This policy is captured by the graph pattern Me in Fig. 2(a). The pattern requires that the owner is identical to the requester.*
- *"Either the owner or her family doctor can access." This policy demands a match between the access scenario and either one of the patterns Me or Doctor (Fig. 2(b)).*
- *"The owner, her family doctor, or the family doctor's assistant can access." This policy demands a match with one of the following patterns: Me, Doctor, and Assistant (Fig. 2(d)).*
- *"Everyone can access." The policy requires a match with either Me or Other (Fig. 2(c)). In short, access is granted if (a) the requester is the owner (Me), or (b) if the requester is not the owner (Other). That is, access is always granted.*

3.3 Policies and Authorization

In the original conception of ReBAC, an access control policy specifies how a legitimate accessor of a resource shall be related to the owner of that resource [8, 10, 3]. When we move to the context of multiple ownership, the notion of an access control policy shall be expanded: a policy specifies how a legitimate accessor shall be related to each of the co-owners. For example, co-owner u_1 demands that the accessor shares 5 common friends with u_1, but co-owner u_2 demands that the accessor be within a distance of 4 in the social network. The example underlines the compositional nature of access control policies when multiple owners are involved. We define here a simple language for specifying multiple-owner policies, by way of graph patterns. How policy negotiation is conducted among the owners to arrive at a policy that everyone can agree on is discussed in the next sections.

Fixing a universe $\mathcal{V}$ of vertices and a universe $\mathcal{PV}$ of legitimate graph patterns, the abstract syntax of a ***policy*** ϕ is defined as follows:

$$\phi ::= \alpha \mid \neg\phi \mid \phi \wedge \phi$$
$$\alpha, \beta ::= \mathsf{acc}(BG, u)$$

where each of α and β is called an ***atomic policy***, which is composed of an ***anchor*** $u \in \mathcal{V}$ together with a graph pattern $BG \in \mathcal{PV}$ (u need not be a vertex in BG). We also define the usual derived form $\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$. The intuitive meaning of an atomic policy $\mathsf{acc}(BG, u)$ is that owner u demands the requester to be related to u by a graph-theoretic relationship captured by the graph pattern BG. A policy is essentially a boolean combination of atomic policies.

The ***anchor set*** of policy ϕ, denoted by $anchors(\phi)$, is the set of all anchors appearing in the atomic policies of ϕ.

Authorization is defined in terms of the ***satisfaction relation*** ($\models$) between a graph G, a vertex $v \in V(G)$, and a policy ϕ for which $anchors(\phi) \subseteq V(G)$:

- $G, v \models \mathsf{acc}(BG, u)$ iff $BG \lesssim G_{(u,v)}$.
- $G, v \models \neg\phi$ iff it is not the case that $G, v \models \phi$.
- $G, v \models \phi_1 \wedge \phi_2$ iff $G, v \models \phi_1$ and $G, v \models \phi_2$.

Lastly, we write $\mathcal{P}[\mathcal{V}, \mathcal{PV}]$ to denote the set of all policies ϕ with anchors from $\mathcal{V}$ and graph patterns from $\mathcal{PV}$.

3.4 Putting It All Together

We define a ***ReBAC model with multiple ownership (ReBAC/MO)***. The model is intentionally simplified for the purpose of highlighting only the essential elements.

Definition 3 (Scheme) *A* **protection scheme** Σ *is a tuple* $\langle \mathcal{L}, \mathcal{U}, \mathcal{O}, \mathcal{PV} \rangle$*, where* $\mathcal{L}$ *is a finite set of relation identifiers,* $\mathcal{U}$ *and* $\mathcal{O}$ *are countably infinite sets representing respectively the universes of* **users** *and* **objects***, and* $\mathcal{PV} \subseteq \mathcal{B}[\mathbb{N}, \mathcal{L}]$ *is the* **policy vocabulary***, which specifies the set of legitimate graph patterns that can be used for formulating access control policies.*

Definition 4 (State) *Given a protection scheme* $\Sigma = \langle \mathcal{L}, \mathcal{U}, \mathcal{O}, \mathcal{PV} \rangle$*, a* **protection state** γ *is a tuple* $\langle U, O, G, \omega, \pi \rangle$*, where* $U \subseteq \mathcal{U}$ *is a finite set of active users,* $O \subseteq \mathcal{O}$ *is a finite set of active objects,* $G \in \mathcal{G}[\mathcal{U}, \mathcal{L}]$ *is the current social network of the active users (i.e.,* $V(G) = U$*),* $\omega : O \to 2^U$ *maps objects to their co-owners, and* $\pi : O \to \mathcal{P}[U, \mathcal{PV}]$ *maps objects to their access control policies. It is further required that* $anchors(\pi(o)) \subseteq \omega(o)$ *in order for* γ *to be well-formed.*

Definition 5 (Request) *Suppose* $\gamma = \langle U, O, G, \omega, \pi \rangle$. *An* **access request** *is characterized by a pair* $(o, v) \in O \times U$. *Request* (o, v) *is* **authorized** *in* γ *iff* $G, v \models \pi(o)$.

We outline here two prototypical administrative transitions. A fully specified system would offer more transitions (e.g., creation of users). These two prototypical transitions are singled out to highlight the need for different policy negotiation protocols.

Given a function $f : X \to Y$, we write $f[a \mapsto b]$ to denote the function $g : X \cup \{a\} \to Y \cup \{b\}$ for which $g(x) = b$ if $x = a$ and $g(x) = f(x)$ otherwise.

The first transition, $\mathsf{create}(o, U')$, creates a new object o with co-owners from user set U'. The following rule specifies when the transition is well-formed.

$$\frac{\begin{array}{cc} o \in \mathcal{O} \setminus O & \emptyset \subset U' \subseteq U \\ O' = O \cup \{o\} & \omega' = \omega[o \mapsto U'] \\ \multicolumn{2}{c}{\pi' = \pi \left[o \mapsto \bigvee_{u \in U'} \mathsf{acc}(Me, u)\right]} \end{array}}{\langle U, O, G, \omega, \pi \rangle \xrightarrow{\mathsf{create}(o, U')} \langle U, O', G, \omega', \pi' \rangle} \text{(CREATE)}$$

Note that initially o is accessible only to users in U'.

The second transition, $\mathsf{release}(o, \phi)$, re-publishes the existing object o with the policy ϕ. Well-formedness of the transition is specified in the following rule.

$$\frac{\begin{array}{c} o \in O \qquad \mathit{anchors}(\phi) \subseteq \omega(o) \\ \pi' = \pi[o \mapsto \phi] \end{array}}{\langle U, O, G, \omega, \pi \rangle \xrightarrow{\mathsf{release}(o, \phi)} \langle U, O, G, \omega, \pi' \rangle} \text{(RELEASE)}$$

While the rules above specify when a transition is well-formed, they do not articulate the protocols by which users sign up for co-ownership, and how they agree on a policy for a co-owned object. Such are the topics of the next section.

4. NEGOTIATION PROTOCOLS

The two administrative transitions, create and release, are supposed to be guarded by negotiation protocols that allow the co-owners to agree on the creation or publishing of objects. In particular, the two transitions are guarded respectively by two different protocols, simple consenting (§4.1) and policy negotiation (§4.2).

4.1 Simple Consenting

Facebook uses simple consenting in the befriending operation. When a user u befriends another user v, the system sends a befriending invitation v. If the latter gives consent, then a friendship relationship is recorded in the system. We call this protocol ***simple consenting***. In a similar vein, when an object is created in our model, all the co-owners are notified, and their consents are sought before the object creation process is complete.

Our model differs from Facebook on how the access control policy of a co-owned object is administrated. In Facebook, simple consenting is used only for forging co-ownership, but the access control policy of a co-owned object is not formulated collaboratively. When u_1 befriends u_2, the access control policy of the friendship relationship is not part of the consenting process. Once friendship has been forged, u_1 may unilaterally specify a policy ϕ_1 (for which $\mathit{anchors}(\phi_1) = \{u_1\}$), and u_2 may independently specify another policy ϕ_2 (again, $\mathit{anchors}(\phi_2) = \{u_2\}$). The friendship relationship is now visible to any requester who satisfies $\phi = \phi_1 \vee \phi_2$. This lack of collaboration allows a co-owner (u_1) to formulate a highly liberal policy (ϕ_1) without the consent of another co-owner (u_2), thereby compromising the privacy of the latter.

With this consideration, the create transition of our model initializes the policy $\pi(o)$ of a newly created object o to a highly restrictive default:

$$\bigvee_{u \in \omega(o)} \mathsf{acc}(Me, u) \tag{1}$$

When one or more co-owners of o attempt to update $\pi(o)$, they will have to perform the release operation, which is guarded by a policy negotiation protocol, in which all co-owners collaboratively work out an access control policy.

4.2 Policy Negotiation

The policy negotiation framework to be presented below is designed to allow co-owners to collaboratively arrive at an access control policy while balancing two considerations.

1. *Privacy preferences.* The co-owners want to make sure that only certain users may access the co-owned objects. The participatory nature of the negotiation process allows the co-owners to reflect their *subjective* privacy preferences in the final access control policy.
2. *Sharing needs.* An object is created because the co-owners desire to make it available to other users. The negotiation process allows each co-owners to state a desirable level of availability for the co-owned object. This specified level of availability will be used as an *objective* measure of policy quality.

The successful completion of the policy negotiation process for object o produces a ***settlement***, which is comprised of two components:

1. A ***settlement policy*** ϕ for which $\mathit{anchors}(\phi) \subseteq \omega(o)$, and all co-owners in $\omega(o)$ have expressed consent over the adoption of ϕ.
2. Every co-owner $u \in \omega(o)$ would have specified an ***availability criterion*** κ_u, and the system would have verified that the settlement policy ϕ satisfies κ_u for each owner u.

While the settlement policy expresses the privacy preferences of the co-owners, the availability criteria express their sharing needs.

The negotiation process proceeds in rounds. In each round, every owner u will be given a chance to contribute changes to the settlement policy ϕ in a controlled manner, as well as revising her availability criterion κ_u. Negotiation concludes successfully when the settlement policy ϕ is verified by the system to satisfy the availability criterion κ_u of every owner u. These availability criteria will be discarded once the negotiation concludes. Only the settlement policy is recorded.

Further details of the settlement policy and availability criteria are given in §4.2.1 and §4.2.2 respectively. Details of the negotiation protocol are given in §4.2.3.

4.2.1 Syntax of Settlement Policy

To facilitate the negotiation of policies and the verification of availability criteria, we constrain the syntax of an access control policy to the following form:

$$\phi = (\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_m) \wedge (\neg\beta_1 \wedge \neg\beta_2 \wedge \ldots \wedge \neg\beta_n) \tag{2}$$

where α_i and β_j are atomic policies.

We define additional notations to facilitate discussion. We write $atom^+(\phi)$ and $atom^-(\phi)$ to denote respectively the atom sets $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ and $\{\beta_1, \beta_2, \ldots, \beta_n\}$. We define the following notations for the set of positive (resp. negative) atoms that are anchored at user u.

$$atom^+(\phi, u) = \{\alpha \in atom^+(\phi) \mid u \in anchors(\alpha)\}$$
$$atom^-(\phi, u) = \{\beta \in atom^-(\phi) \mid u \in anchors(\beta)\}$$

We also write $pat^+(\phi, u)$ and $pat^-(\phi, u)$ to denote respectively the birooted graph sets $\{BG \mid \mathsf{acc}(BG, u) \in atom^+(\phi, u)\}$ and $\{BG \mid \mathsf{acc}(BG, u) \in atom^-(\phi, u)\}$. In short, $pat^+(\phi, u)$ is the set of positive graph patterns with u as the anchor, and $pat^-(\phi, u)$ is the analogue for negative graph patterns.

A comparison between the above syntactic form and the form of policies used in Facebook for controlling the visibility of a friendship relationship is in order. The four policies of Example 2 are all ***positive policies*** [7, 10]. This means each of them can be expressed as a disjunctive combination of finitely many graph patterns. It should be obvious to the reader that the four policies of Example 2 are but organizational variants of Facebook's standard policies: "only me," "friends," "friends of friends," and "everyone." This means Facebook's standard policies can also be represented by disjunctions of finitely many graph patterns. As discussed in §4.1, the policy that controls the visibility of the friendship relationship between users u and v has the form $\phi = \phi_u \vee \phi_v$, where each of ϕ_u and ϕ_v is a disjunction of atomic policies (because they are positive policies). Consequently, the access control policy of a friendship relationship in Facebook has the following form:

$$\phi = (\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_m) \tag{3}$$

Two differences between our syntactic form, (2), and that of Facebook, (3), can be observed. First, our syntactic form allows negative graph patterns. A co-owner may counteract the positive graph patterns contributed by other co-owners through the specification of negative graph patterns. Second, the negotiation of policies of the above form is regulated by a policy negotiation protocol, while ϕ_u and ϕ_v are set unilaterally by u and v.

4.2.2 Availability Criteria

The sharing needs of a co-owner $u \in \omega(o)$ of a resource o is expressed in terms of an availability criterion κ_u. Intuitively, κ_u expresses how widely accessible u wants o to be. Co-owner u may choose to impose availability criterion κ_u on one of two formulas: (a) the settlement policy ϕ, or (b) the formula $\phi[u]$ defined as follows:

$$\phi[u] = \left(\bigvee_{\alpha \in atom^+(\phi, u)} \alpha \right) \wedge \left(\bigwedge_{\beta \in atom^-(\phi)} \neg\beta \right)$$

In short, $\phi[u]$ is obtained from the settlement policy ϕ by selecting *only* those positive atoms anchored at u, but including *all* the negative atoms. Intuitively, $\phi[u]$ specifies the legitimate accessors who are preferred by u. For brevity, we assume that an availability criterion is imposed on the settlement policy ϕ. The reader should keep in mind that in reality a co-owner may choose between ϕ or $\phi[u]$.

Three different families of availability criterion are proposed in this work. The first notion of availability is that of ***satisfiability***, which requires that a resource be accessible by a specific number of users in the present protection state.

Example 3 (Satisfiability) *Object o shall be accessible by at least* 35 *users in the social network.*

Resiliency is a second family of availability criteria. Intuitively, resiliency mandates a level of availability even when the protection state is "slightly" perturbed by administrative actions. Resiliency is more demanding than satisfiability, in the sense that resiliency requires not only that a resource is accessible by some specific number of users in the present protection state, but also that the resource remains available as such when the protection state mutates. In the context of ReBAC, such mutations take the form of the introduction of new relationships (edges) or the dissolution of existing relationships. It is, however, unrealistic to expect that availability can be maintained against arbitrary mutation of the social network. Thus resiliency prescribes availability over a specific extent of mutation to the protection state.

Example 4 (Resiliency) *Even if Bob (one of the owner of resource o) were to acquire up to* 10 *additional patients (or lose up to* 10 *of my existing patients), object o shall remain accessible by at least* 25 *users.*

A third notion of availability, ***feasibility***, is a relaxation of satisfiability. A resource may not meet the requirement of satisfiability (accessible by a specific number of users), but it meets the demand of feasibility if the social network can be "repaired" to meet the requirement of satisfiability. Again, repairs are realized by edge addition or removal in the social network.

Example 5 (Feasibility) *I shall be able to make object o accessible by at least* 50 *users if the company hires no more than* 10 *additional members for my department (or fires no more than* 10 *existing colleagues).*

The formal definition of availability criteria and their mechanical verification are deferred to §6 and §7 respectively.

4.2.3 Protocol in Details

The negotiation of the access control policy of a co-owned object o proceeds in rounds. Throughout the process, the protocol maintains a policy ϕ as the candidate for $\pi(o)$, as well as an availability criterion κ_u for each co-owner $u \in \omega(o)$. The policy ϕ is always in the form of (2).

As mentioned before, ϕ is initialized to the default policy (1). The availability criterion κ_u of every co-owner u is initialized to 1-satisfiability (i.e., accessible to at least one user). The default policy trivially satisfies 1-satisfiability.

Round i proceeds according to the following steps:

1. **Notify.** Every co-owner $u \in \omega(o)$ is presented with the following: (i) the policy ϕ obtained in the previous round, including the positive and negative graph patterns ($pat^+(\phi, v)$ and $pat^-(\phi, v)$) of every co-owner v, (ii) the availability criterion κ_u adopted by co-owner u in the previous round, and (iii) a boolean flag indicating whether ϕ satisfies κ_u.
2. **Consent or Revise.** Each co-owner u will now be given an opportunity to take one of two options:
   (a) *Consent.* If κ_u is satisfied, u may choose to give consent to adopt ϕ as the policy of o.

(b) *Revise.* Otherwise, co-owner u may choose to revise (i) κ_u and/or (ii) the graph pattern sets $pat^+(\phi, u)$ and $pat^-(\phi, u)$. In short, co-owner u may only edit the atomic policies of ϕ for which u is an anchor.

The consent/revise step of different co-owners may be conducted concurrently.

3. **Verify.** If all co-owners give consent, then the protocol concludes successfully. Otherwise, at least one co-owner has revised either her availability criterion or her graph patterns. The system will now perform verification of the current policy ϕ against each of the availability criterion κ_u (see §6 for details). The result is reported to the co-owners in the next round.

5. AN EXAMPLE

The various features of our scheme are illustrated in the following example. A medical doctor $\mathcal{D}$ suspects that the unique conditions of her patient $\mathcal{P}$ are the symptoms of a new cardiovascular disease. $\mathcal{D}$ wants to share the medical record of $\mathcal{P}$ with two researchers, $\mathcal{CR}$ and $\mathcal{UR}$, who work in a pharmaceutical company and a university respectively. The following is a transcript of how they negotiate the access control policy ϕ of the co-owned medical record.

Round #1. In anticipation of referral, $\mathcal{D}$ adds a positive atom $\alpha_{\mathcal{D}}$ to ϕ, to grant access to those cardiologists belonging to the same health district as she does. $\mathcal{D}$ also imposes on $\phi[\mathcal{D}]$ a satisfiability requirement ($\kappa_{\mathcal{D}}$), demanding that at least two users (i.e., herself and a cardiologist) can access.

Company researcher $\mathcal{CR}$ contributes a positive atom $\alpha_{\mathcal{CR}}$, to grant access to all members of his research team. $\mathcal{CR}$ worries that 3 of his team members may leave the team due to personal reasons, and he anticipates that the research on the new disease needs at least a team of 5 researchers. $\mathcal{CR}$ then imposes $\kappa_{\mathcal{CR}}$, a resiliency requirement, on $\phi[\mathcal{CR}]$, demanding that at least 5 users (i.e., team members) can access even if the social network is changed by 3 edges (i.e., team members departing).

University researcher $\mathcal{UR}$ contributes a positive atom $\alpha_{\mathcal{UR}}$, to grant access to his graduate students. $\mathcal{UR}$ is expecting one new PhD student to join her research group in the upcoming Fall semester. By that time, she wants to make sure that at least herself and three of her PhD students can access the document. So $\kappa_{\mathcal{UR}}$ is set up to be a feasibility requirement, to be imposed on $\phi[\mathcal{UR}]$, demanding that there is a way to change 1 edge (i.e., the incoming student) in the social network so that 4 users (i.e., herself and 3 PhD students) have access.

$\mathcal{P}$ does not add any positive atom, nor does he alter the default availability criterion. Yet he does not want anyone associated with, or funded by, the insurance company $\mathcal{I}$ to see his record. So he introduces a negative atom $\beta_{\mathcal{P}}$ to deny access to such individuals.

It turns out that every cardiologist in the same health district as $\mathcal{D}$ is funded by insurer $\mathcal{I}$, and thus $\kappa_{\mathcal{D}}$ is violated by $\phi[\mathcal{D}]$. In addition, all but one of the PhD students of $\mathcal{UR}$ are funded by scholarships from $\mathcal{I}$, so $\kappa_{\mathcal{UR}}$ is violated by $\phi[\mathcal{UR}]$. A second round of negotiation is therefore needed.

Round #2. Noticing that $\alpha_{\mathcal{D}}$ is too restrictive, doctor $\mathcal{D}$ now revises $\alpha_{\mathcal{D}}$ to include cardiologists in both her own health district and neighboring health districts. Such a widening of access "counteracts" the limiting effect of $\beta_{\mathcal{P}}$.

University researcher $\mathcal{UR}$ figures that requiring the record to be accessible by 3 PhD students is probably too conservative. The project can be completed even if only two PhD students have access. So she revises $\kappa_{\mathcal{UR}}$ to a feasibility requirement, demanding that there is a way to change 1 edge in the social network so that 3 users have access (1 less than before). In short, $\mathcal{UR}$ "counteracts" the limiting effect of $\beta_{\mathcal{P}}$ by lowering her sharing expectation.

All the availability criteria check out, and negotiation concludes successfully.

6. AVAILABILITY CRITERIA

An important contribution of this work is the formulation of formal availability criteria as a means for expressing sharing needs, as well as the mechanical verification of such criteria to provide feedback to the negotiating co-owners. In this section, we formalize the three notions of availability in §4.2.2: satisfiability, resiliency and feasibility. While the spirit of these notions is inspired by their analogues in workflow authorization systems [4, 28, 16, 5], our technical formulation via graph mutation is a novel contribution.

In the following, a policy needs *not* be in the constrained form of (2). We do not anticipate that imposing restriction (2) affects our complexity results (Theorems 1, 2 and 3).

6.1 Satisfiability

The first notion of availability is that of ***satisfiability***.

Problem: Satisfiability (MO-SAT)

Instance: a positive integer k, a graph G, a policy ϕ for which $anchors(\phi) \subseteq V(G)$

Question: Does there exist k distinct vertices $v \in V(G)$ for which $G, v \models \phi$?

Fixing the protection state (i.e., G), policy ϕ is said to be ***k-satisfiable*** iff $\langle k, G, \phi \rangle \in$ MO-SAT. Intuitively, if a co-owner demands ϕ to be k-satisfiable, then there shall exists k potential requesters who are able to gain access in the current protection state. Obviously, a $(k+1)$-satisfiable policy is also k-satisfiable.

Not only MO-SAT, but all the three availability criteria presented in this section belong to the second level of the polynomial hierarchy [1, Chap. 5].

Theorem 1 MO-SAT *is in* $\mathbf{\Delta_2^p}$*, and is both* **NP***-hard and* **coNP***-hard.*

See [20, Chap. 5, pp. 55–57] for a proof.

6.2 Feasibility

We need another notation before we can proceed: $d(G, G')$ is the ***(edge) distance*** between the two graphs G and G', which is defined to be $\Sigma_{l \in \mathcal{L}} |R_l(G) \,\Delta\, R_l(G')|$, where $S \,\Delta\, T$ is the symmetric difference of sets S and T.

Relaxing the definition of satisfiability gives rise to a second notion of availability — ***feasibility***, which demands that the protection state can be "repaired" to render the policy k-satisfiable.

Problem: Feasibility (MO-FEA)

Instance: a non-negative integer δ, a positive integer k, a graph G, and a policy ϕ for which $anchors(\phi) \subseteq V(G)$

Question: Is there a graph G' for which $L(G') = L(G)$, $V(G') = V(G)$, and $d(G, G') \leq \delta$, such that there exists k distinct $v \in V(G')$ for which $G', v \models \phi$?

Obviously, we get back MO-SAT if $\delta = 0$. Fixing graph G, we say that ϕ is (δ, k)***-feasible*** iff $\langle \delta, k, G, \phi \rangle \in$ MO-FEA. Every k-satisfiable policy is (δ, k)-feasible.

Theorem 2 MO-FEA *is* $\mathbf{\Sigma_2^P}$*-complete.*

A proof of the theorem is given in [20, Chap. 5, pp. 57–58].

6.3 Resiliency

Strengthening satisfiability yields the notion of ***resiliency***, which demands that a policy remains k-satisfiable even if the protection state is mutated.

Problem: Resiliency (MO-RES)

Instance: a non-negative integer δ, a positive integer k, a graph G, and a policy ϕ for which $anchors(\phi) \subseteq V(G)$

Question: Is it the case that for every G' such that $L(G') = L(G)$ and $V(G') = V(G)$ and $d(G, G') \leq \delta$, there exists k distinct vertices $v \in V(G')$ for which $G', v \models \phi$?

Fixing G, we say that ϕ is (δ, k)***-resilient*** iff $\langle \delta, k, G, \phi \rangle \in$ MO-RES. A (δ, k)-resilient policy is also k-satisfiable.

Theorem 3 MO-RES *is* $\mathbf{\Pi_2^P}$*-complete.*

Consult [20, Chap. 5, pp. 59–60] for a proof.

7. VERIFYING AVAILABILITY

As the three availability criteria belong to the second level of the polynomial hierarchy, we do not expect that they can be solved in polynomial time. A natural starting point would be to reduce the problems to ***Quantified Boolean Formula (QBF)*** satisfiability, and then deploy a QBF solver to attack the problems [1, Chap. 5]. Experimental results suggest that this line of attack is not viable (§8.2). In this work, we focus on solving MO-SAT, since it belongs to $\mathbf{\Delta_2^P}$, and thus it has a higher degree of tractability. We defer to future work the exploration of more tractable subproblems of MO-FEA and MO-RES as well as their decision procedures.

As MO-SAT is both **NP**- and **coNP**-hard, it is not likely it belongs to **NP** [1, Chap. 2]. Consequently, we do not expect that we can reduce MO-SAT to the Boolean satisfiability (SAT) problem, and then employ a SAT solver to attack MO-SAT. Rather, we make use of a SAT solver as a *subroutine* for crafting two verifiers for MO-SAT. As MO-SAT $\in \mathbf{\Delta_2^P}$, this is theoretically viable.

7.1 Reducing an Atomic Policy to SAT

The two MO-SAT algorithms to be examined in §7.2 both invoke a SAT solver for testing the satisfiability of atomic policies. We describe here a reduction of the satisfiability of atomic policies to SAT. This reduction will be used as a subroutine in the algorithms of §7.2.

We use Sat4j as our SAT solver [18]. Sat4J allows users to formulate not only regular clauses, but also what Wang and Li [28] call ***Pseudo Boolean (PB)*** constraints, which count the number of satisfied literals in a clause:

$$(l_1 \vee l_2 \vee \ldots \vee l_m)_{\leq k} \tag{4}$$

$$(l_1 \vee l_2 \vee \ldots \vee l_m)_{=k} \tag{5}$$

Specifically, (4) is satisfied if at most k of the literals are satisfied, and (5) is satisfied if exactly k literals are satisfied.

Algorithm 1, Reduce, reduces 1-satisfiability to SAT. Specifically, Reduce takes three parameters:

1. G is the social graph of the ReBAC protection state.
2. $\mathsf{acc}(H_{(x,y)}, u)$ is the atomic policy for which 1-satisfiability is to be tested.
3. $\{Y_j\}_{j \in V(G)}$ is an indexed family of propositional variables, one for each vertex in G.

Algorithm 1: Reduce

input : G, $\mathsf{acc}(H_{(x,y)}, u)$, $\{Y_j\}_{j \in V(G)}$
output: a CNF formula ϕ

```
1  Initialize φ to an empty set of clauses;
2  foreach i ∈ V(H) do
3  |  foreach j ∈ V(G) do
4  |  |  Create fresh propositional variable X_{i,j};
5  foreach i ∈ V(H) do
6  |  φ := φ ∧ ( ⋁_{j∈V(G)} X_{i,j} )_{=1};
7  foreach j ∈ V(G) do
8  |  φ := φ ∧ ( ⋁_{i∈V(H)} X_{i,j} )_{≤1};
9  foreach l ∈ L(H) do
10 |  foreach i, j ∈ R_l(H) do
11 |  |  foreach p ∈ V(G) do
12 |  |  |  φ := φ ∧ ( ¬X_{i,p} ∨ ⋁_{q∈neigh^G(l,p)} X_{j,q} );
13 φ := φ ∧ X_{x,u};
14 foreach j ∈ V(G) do
15 |  φ := φ ∧ (¬X_{y,j} ∨ Y_j) ∧ (X_{y,j} ∨ ¬Y_j);
16 return φ;
```

The output of Algorithm 1 is a CNF formula ϕ, which contains, among other propositional variables generated within the algorithm, those variables Y_j's supplied by the caller. Formula ϕ is satisfiable iff there exists $v \in V(G)$ for which $H_{(x,y)} \lesssim G_{(u,v)}$. In addition, a satisfying assignment will turn Y_v to true, and all the other Y_j's to false. Having $\{Y_j\}_{j \in V(G)}$ as an input to the algorithm allows the caller to recover the identity of v with ease, and also makes it easy to embed ϕ in a larger SAT instance.

The returned formula ϕ tests if there exists a vertex $v \in V(G)$ as well as a birooted graph isomorphism f between $H_{(x,y)}$ and some subgraph of $G_{(u,v)}$. The mapping f is represented by the propositional variables $X_{i,j}$'s created on line 4. More specifically, $X_{i,j}$ is turned on iff f maps $i \in V(H)$ to $j \in V(G)$. Line 6 adds clauses to ensure that f is a function. Then on line 8, clauses are introduced to ensure that the f is injective. Function $\mathsf{neigh}^G(l, p)$ (read *neighbours*) used in line 12 is defined below:

Definition 6 $\mathsf{neigh}^G(l, p)$ *is a function with* $G \times L(G) \times V(G) \to 2^{V(G)}$ *signature and following definition:*

$$\mathsf{neigh}^G(l, p) = \{q \in V(G) \mid (p, q) \in R_l(G)\} \tag{6}$$

Line 12 add clauses to ensure that adjacency is preserved by f. Line 13 ensures $f(x) = u$. The clauses added on line 15 ensure that $f(y)$ is identical to the selection of vertex indicated by the Y_j's.

The design of Algorithm 1 is comparable to the reduction of graph isomorphism to SAT reported in [27], with four differences. First, PB constraints have been employed in our reduction for ensuring that f is an injective function, where regular clauses are used in [27] for similar purposes. In our preliminary experiments, the use of PB constraints have significantly enhanced the performance of SAT solving. Second, adjacency preservation is only enforced in one direction in our algorithm (line 12), as we are testing subgraph isomorphism rather than graph isomorphism (as in [27]). Third, additional clauses are introduced to handle the mapping of roots in our algorithms (lines 13–15). Fourth,

Algorithm 2: VertexEnumerator

input : k, G, ϕ
output: A boolean value

1 **foreach** $j \in V(G)$ **do**
2 | Create fresh propositional variable Y_j;
3 $vars := \{Y_j\}_{j \in V(G)}$;
4 $\psi := \left(\bigvee_{j \in V(G)} Y_j\right)_{=1}$;
5 $count := 0$;
6 **foreach** $v \in V(G)$ **do**
7 | **foreach** *atomic policy* α *in* ϕ **do**
8 | | $\varphi_\alpha := (Y_v) \wedge \psi \wedge \mathsf{Reduce}(G, \alpha, vars)$;
9 | | apply Sat4j on φ_α to obtain truth value for α;
10 | evaluate ϕ using truth values from line 9;
11 | **if** ϕ *evaluates to true* **then**
12 | | $count := count + 1$;
13 | **if** $count = k$ **then**
14 | | **return** *true*;
15 **return** *false*;

Algorithm 3: ModelEnumerator

input : k, G, ϕ
output: A boolean value

1 **foreach** $j \in V(G)$ **do**
2 | Create fresh propositional variable Y_j;
3 $vars := \{Y_j\}_{j \in V(G)}$;
4 $\psi := \left(\bigvee_{j \in V(G)} Y_j\right)_{=1}$;
5 **foreach** $\alpha \in atom^+(\phi)$ **do**
6 | $\varphi_\alpha := \mathsf{Reduce}(G, \alpha, vars)$;
7 $\varphi := \psi \wedge \mathsf{Disj}(\{\varphi_\alpha \mid \alpha \in atom^+(\phi)\})$;
8 $count := 0$;
9 invoke Sat4j to enumerate the models σ of φ;
10 **foreach** σ **do**
11 | let v be such that $\sigma(Y_v) = true$;
12 | $flag := true$;
13 | **foreach** $\beta \in atom^-(\phi)$ **do**
14 | | $\varphi_\beta = (Y_v) \wedge \psi \wedge \mathsf{Reduce}(G, \beta, vars)$;
15 | | invoke Sat4j on φ_β to get truth value for β;
16 | | **if** β *is true* **then**
17 | | | $flag := false$;
18 | | | break;
19 | **if** *flag* **then**
20 | | $count := count + 1$;
21 | **if** $count = k$ **then**
22 | | **return** *true*;
23 | add the unit clause $(\neg Y_v)$ to φ;
24 **return** *false*;

according to theorem 4 the clauses added in line 12 have significantly better space complexity than the ones in [27].

Theorem 4 *Let m and n be the number of vertices for the graph patterns and the social graph, respectively. Suppose we fix the average degree of the vertices. The space complexity of the adjacency preservation clauses in Algorithm 1 (line 12) is $\Theta(m^2 n)$ and in [27] is $\Theta(m^2 n^2)$.*

A proof is given in [20, Chap. 6, p. 76].

7.2 Two MO-SAT Verifiers

We pointed out that checking MO-SAT by a QBF solver is inefficient (§8.2). Instead, we propose custom algorithms that invoke a SAT solver as a subroutine. As MO-SAT is in $\mathbf{\Delta_2^P}$, only polynomially many invocations are required.

Modern SAT solvers are highly optimized, and are better studied than QBF solvers. In this work we work with Sat4j [18], which is a SAT solver written in JAVA. According to its developers, Sat4j is only 3.25 times slower than its C++ counterpart. It has also been employed in the past for automating various analyses in Access Control [28].

We present below two algorithms for deciding MO-SAT. In both cases, the input is an MO-SAT instance (k, G, ϕ), and the output is a boolean decision. As we shall see, Sat4j is not only used as an oracle for deciding NP-complete subproblems, but also used in the second algorithm as a model enumerator for pruning the search space.

7.2.1 Vertex Enumerator

Our first algorithm is listed in Algorithm 2. This algorithm does not make any assumptions on the syntax of the input formula ϕ. The full syntax in §3.3 can be used.

Algorithm 2 iterates through each vertex v of the social graph G, and checks if v satisfies the policy ϕ (i.e., v can successfully access the resource guarded by ϕ), hence, the name VertexEnumerator. If k users are found to satisfy ϕ, then ϕ is k-satisfiable. To evaluate ϕ, each atomic policy in ϕ is evaluated independently, and their truth values are substituted into ϕ (line 10). To evaluate an atomic policy α, line 8 reduces α to a SAT instance (via a call to Reduce), and adds clauses to select v as the requester (and to deselect other users). The resulting SAT instance is then solved by invoking Sat4j to obtain the truth value for α (line 9).

Algorithm 2 is basically a constructive proof that MO-SAT is in $\mathbf{\Delta_2^P}$. Its strength is generality: it works for all ϕ. Its weakness is that it examines every vertex of G without discretion, and thus leads to slower performance (§8).

7.2.2 Model Enumerator

Our second MO-SAT verifier is Algorithm 3, ModelEnumerator. This algorithm assumes that the input formula ϕ conforms to the constrained syntax of (2). This syntactic restriction is crucial to the design of the algorithm.

The design of Algorithm 3 is enabled by two features of modern SAT solvers, including Sat4j:

1. *Model enumeration.* The SAT solver does not only find one model (i.e., satisfying truth assignment) for the input CNF, but allows the caller to iterate through all models.
2. *Learned clauses.* During the enumeration of models, users may introduce additional clauses to the target formula, thereby eliminating certain models from subsequent enumeration.

The core idea of Algorithm 3 is to employ Sat4j to enumerate the models σ of $(\alpha_1 \vee \ldots \vee \alpha_m)$ (lines 9–10). Every such model identifies a v in G that can access so long as the negative atoms are ignored (line 11). The algorithm then invokes another instance of Sat4j to ensure that v does not satisfy any negative atom $\beta \in atom^-(\phi)$ (line 15). Once the model σ has been considered, a unit clause is introduced into ϕ to ensure that v is never considered again (line 23).

By employing $(\alpha_1 \vee \ldots \vee \alpha_m)$ to drive the process of identifying potential accessors, ModelEnumerator does not need to

consider every vertex in G, this could lead to a performance advantage over VertexEnumerator. See §8.3 for an empirical comparison of the two algorithms.

The subroutine Disj($\{\varphi_1, \ldots, \varphi_k\}$) invoked on line 7 takes a set of CNF formulas $\varphi_1, \ldots, \varphi_k$ as input, and returns a CNF formula that is equivalent to $\varphi_1 \vee \ldots \vee \varphi_k$. We illustrate the construction in the case of $k = 2$. The general case is a straightforward extension. Suppose $\varphi_1 = \bigwedge_{i=1}^{m_1} C_i^1$ and $\varphi_2 = \bigwedge_{i=1}^{m_2} C_i^2$ are conjunctions of respectively m_1 and m_2 clauses. By introducing two additional propositional variables x^1 and x^2, one can construct a CNF formula φ that is equivalent to their disjunction $(\varphi_1 \vee \varphi_2)$:

$$\varphi = (x^1 \vee x^2)_{=1} \wedge \left(\bigwedge_{i=1}^{m_1} (\neg x^1 \vee C_i^1)\right) \wedge \left(\bigwedge_{i=1}^{m_2} (\neg x^2 \vee C_i^2)\right)$$

8. EXPERIMENTS AND RESULTS

This section reports three experiments we conducted for evaluating the performance of our MO-SAT algorithms. The first experiment (§8.2) compares the performance between Vertex Enumerator and two QBF solvers GhostQ and RAReQS. The second experiment (§8.3) compares the performance between Vertex Enumerator and Model Enumerator. The third (§8.4) demonstrates that the performance of Model Enumerator is adequate for mid-sized organizations.

8.1 Experimental Setup

Experiment Platform.

Our experiments are conducted using a machine with the following configurations.

Hardware: Intel® Xeon® CPU E5-1650 v3 @ 3.50GHz (6 cores); 64 GB RAM; 1 TB SSD.

Software: Operating System: Fedora release 20 (Heisenbug); Sat4j algorithms are programmed and run by Java™ SE Runtime Environment (build 1.8.0_05-b13).

Datasets.

All experiments are executed on synthetic datasets (including social networks and access control policies). We list the dataset generation parameters below.

- *Rep.*: number of times an experiment is repeated for a given set of parameters;
- k_u: number of requesters needed to satisfy policy (i.e., aiming for k_u-satisfiability);
- $\#atom^+$: number of positive atoms (α) in policy (ϕ);
- $\#atom^-$: number of negative atoms (β) in policy (ϕ);
- $\#V_{pat^+}$: number of vertices in the graph patterns of positive atoms (α);
- $\#V_{pat^-}$: number of vertices in the graph patterns of negative atoms (β);
- $Prob._G$: the edge probability by which social graph G is generated;
- $Prob._{pat}$: the edge probability by which graph patterns are generated;
- $AvgDeg._G$: the average degree of vertices (for outgoing edges) that we require social graph G to possess;
- $AvgDeg._{pat}$: the average degree of vertices (for outgoing edges) that we require the patterns to possess;
- $|\mathcal{L}|$: number of edge labels used in graphs and graph patterns.

We use Erdős-Rényi model [6] to generate the social graphs and graph patterns. In this model, an edge from any vertex u to any vertex v is set with a fixed probability, independent from other edges. We use this model to obtain randomness along with low deviation from average degrees of vertices.

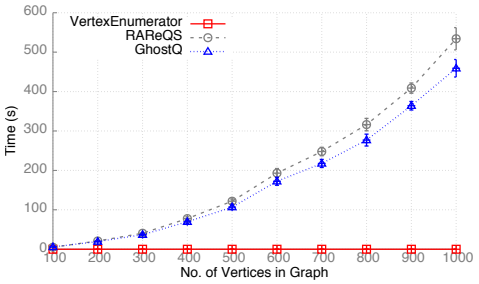


Figure 3: Vertex Enumerator versus QBF Solvers

There is also a validation phase after generating graph patterns. A graph pattern will be deemed "usable", if every edge in the graph pattern is connected to both roots. This guarantees the graph patterns represent ***local policies*** [7, 10]. "Unusable" graph patterns are discarded. Moreover, whenever an edge is being added, we choose its label uniformly at random from the available labels set.

We use the edge probability parameters ($Prob._G$ and $Prob._{pat}$) whenever we want the average degree of the vertices to grow proportionally to the size of graph. In other words, we keep the edge probability constant. On the other hand, whenever we want to keep the average outgoing edges degree of vertices constant, we use the average degree parameters ($AvgDeg._G$ and $AvgDeg._{pat}$).

In each round of experiment, we first create the dataset with the required parameters, and then all the algorithms are executed with the same dataset. Fairness in comparison is therefore guaranteed.

We consider 95% confidence interval in our experiments.

8.2 Vertex Enumerator vs QBF Solvers

This experiment compares the performance of Vertex Enumerator (Algorithm 2) against that of two non-clausal QBF solvers — GhostQ and RAReQS. In the latter cases, MO-SAT instances are first reduced to QBF satisfiability instances, and then the QBF solvers are invoked to solve them. We choose to work with non-clausal QBF solvers because converting the QBF encoding to CNF would increase the number of quantifier alternation of the resulting formulas, making the problem instances prohibitively challenging.

The experiment variable is the size of the social graph. The parameters for this experiment are as follows: • *Rep.*: 20; • k_u: 1; • $\#atom^+$: 1; • $\#atom^-$: 0; • $\#V_{pat^+}$: 5; • $Prob._G$: 0.1; • $Prob._{pat}$: 0.1; • $|\mathcal{L}|$: 1.

The average running time of the three solvers are plotted in Fig. 3. The performance of the QBF solvers are clearly inferior to VertexEnumerator. This confirms the need for custom MO-SAT algorithms.

8.3 Vertex Enumerator vs Model Enumerator

This experiment compares the performance of Vertex Enumerator (Algorithm 2) and Model Enumerator (Algorithm 3). The experiment is repeated for two different sizes of graph patterns in the positive atoms (α). The parameters for this experiment are listed below: • *Rep.*: 20; • k_u: 1; • $\#atom^+$: 3; • $\#atom^-$: 3; • $\#V_{pat^+}$: 5; • $\#V_{pat^-}$: 5; • $Prob._G$: 0.1; • $Prob._{pat}$: 0.1; • $|\mathcal{L}|$: 1; The size of social

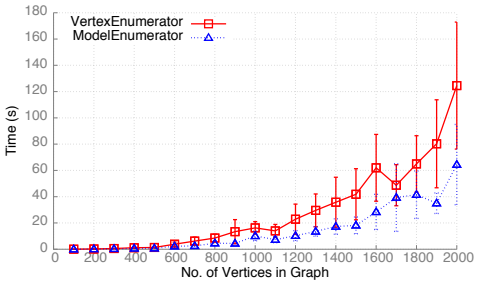


Figure 4: Vertex Enumerator versus Model Enumerator

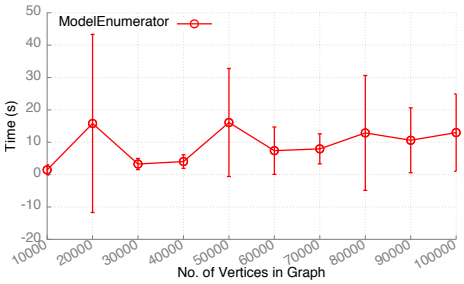


Figure 5: Model Enumerator with Large Social Graphs

graph is the variable of experiment, and we measure the performance of algorithms given different sizes of social graphs.

The average running times of the two algorithms are plotted in Fig. 4. Two observations can be made. First, Model Enumerator clearly out-performs Vertex Enumerator. This confirms our intuition that Model Enumerator is more selective in generating requester candidates than Vertex Enumerator. Second, the performance advantage of Model Enumerator widens as the size of social graph increases.

8.4 Large Social Graphs

This experiment evaluates if Model Enumerator is adequate for realistic social graphs for mid-sized organizations. The following are the parameters for this experiment: • $Rep.$: 20; • k_u: 1; • $\#atom^+$: 3; • $\#atom^-$: 3; • $\#V_{pat+}$: 5; • $\#V_{pat-}$: 5; • $AvgDeg._G$: 200; • $Prob._{pat}$: 0.5; • $|\mathcal{L}|$: 4;

In the previous two experiments, we employ only one edge label. That is done in order to make the problem instances challenging: edge labels provide useful information for the SAT solver. To make the social graphs more realistic in this third experiment, we follow [22] in adopting four edge labels. Moreover, we use more complex graph patterns by boosting the edge probabilities in graph pattern from 0.2 to 0.5. For realism, we also fix the average degree of vertices (outgoing edges) to two hundreds, instead of allowing it to grow proportionally to the size of the social graph (as in the previous two experiment). In an organizational setting, vertex degree is likely very stable.

We run Model Enumerator over social graphs of sizes 10,000 to 100,000 (increment 10,000). To put the scale in perspective, Microsoft and Google employ 94,000 and 55,419 employees, respectively. Fig. 5 depicts the average running times for Model Enumerator. Average problem solving times are within 50 seconds for social graphs of 100,000 vertices. This means that between successive rounds of policy negotiation, we can expect to have a delay of less than one minute for verifying availability criteria. This is well acceptable as Facebook-style consenting protocols (for befriending) work more in the style of email notification than instant messaging. Such performance indicates that our technology can be deployed comfortably in mid-sized organizations in the scale of, say, Microsoft or Google.

9. LIMITATIONS

A limitation of this work is that the following question remains open: What if the co-owners cannot settle on a policy in a timely fashion? This is known as termination or convergence guarantee in the literature. Convergence of collective access control decision is not a new topic, and is often conducted using Game Theory [15, 23].

One may be tempted to compare this work with Automated Trust Negotiation (ATN) [29, 31], and to expect that termination guarantees can be obtained in an analogous manner for this work. ATN is designed to enable the exchange of sensitive attributes among parties. These attributes are needed for authorizing access to a sensitive object in the paradigm of ***Attribute-based Access Control***. The number of attributes each principal possesses is finite; an object's policy is fixed; each policy consists of only finitely many attributes. These settings guarantee termination.

In contrast, our interactive negotiation protocol is proposed for co-owners to collaboratively devise a ReBAC policy that satisfies their availability criteria. The atomic policies are comprised of graph patterns, and thus the space of atomic policies is infinite. Therefore, termination guarantee becomes nontrivial unless we introduce further restrictions on the space of atomic policies or on the refinement of the settlement policy in successive rounds.

10. CONCLUSION AND FUTURE WORK

We argued that interactive policy negotiation has an important role in supporting multiple ownership in organizational computing. We proposed a novel, tool-supported policy negotiation framework, in which policies under negotiation are assessed by formal availability criteria, including satisfiability, resiliency and feasibility. Although these availability criteria belong to the second level of the polynomial hierarchy, we designed and evaluated algorithms for deciding satisfiability, and showed that they provide adequate performance for mid-sized organizations.

We highlight three possible future works. The first is to formulate more tractable subproblems for feasibility and resiliency. A possible compromise is to aim for subproblems that reside in Δ_2^p. We have seen, in the case of MO-SAT, that such subproblems are quite amenable to the design of efficient algorithms based on modern SAT solvers.

A second future work concerns the generalization of the language for specifying multiple-owners policies. Currently, we use birooted graph patterns as the basis of the language. This constrains us to consider only binary relationships between the requester and one of the co-owners. In some cases, we may need to express relationships between the requester and multiple co-owners (e.g., the requester shall be a mutual friend of co-owners u_1 and u_2).

A third future work is to identify conditions under which the negotiation protocol is guaranteed to terminate. One possibility is to incorporate incentives into the negotiation protocol so as to promote collaboration among co-owners [15, 23]. Another way is to introduce syntactic restrictions on the refinement of settlement policy in successive rounds.

Acknowledgments

This work is supported in part by an NSERC Discovery Grant (RGPIN-2014-06611) and a Canada Research Chair (950-229712).

11. REFERENCES

[1] Arora, S., and Barak, B. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[2] Besmer, A., and Richter Lipford, H. Moving beyond untagging: Photo privacy in a tagged world. In *Proceedings of CHI '10* (Atlanta, Georgia, USA, 2010), pp. 1563–1572.

[3] Bruns, G., Fong, P. W. L., Siahaan, I., and Huth, M. Relationship-based access control: Its expression and enforcement through hybrid logic. In *Proceedings of CODASPY '12* (San Antonio, TX, USA, Feb. 2012).

[4] Crampton, J. A reference monitor for workflow systems with constrained task execution. In *Proceedings of SACMAT '05* (Stockholm, Sweden, 2005), pp. 38–47.

[5] Crampton, J., Gutin, G., and Yeo, A. On the parameterized complexity and kernelization of the workflow satisfiability problem. *ACM TISSEC 16*, 1 (June 2013).

[6] Erdős, P., and Rényi, A. On random graphs i. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences 6* (1959), 290–297.

[7] Fong, P. W. L. Preventing Sybil attacks by privilege attenuation: A design principle for social network systems. In *Proceedings of IEEE S&P '11* (Oakland, CA, May 2011), pp. 263–278.

[8] Fong, P. W. L. Relationship-based access control: Protection model and policy language. In *Proceedings of CODASPY '11* (San Antonio, Texas, USA, Feb. 2011), pp. 191–202.

[9] Fong, P. W. L., Mehregan, P., and Krishnan, R. Relational abstraction in community-based secure collaboration. In *ACM CCS '13* (Berlin, Germany, 2013), pp. 585–598.

[10] Fong, P. W. L., and Siahaan, I. Relationship-based access control policies and their policy languages. In *Proceedings of SACMAT '11* (Innsbruck, Austria, June 2011), pp. 51–60.

[11] Graham, G. S., and Denning, P. J. Protection: Principles and Practice. In *Proceedings of AFIPS '72 (Spring)* (Atlantic City, New Jersey, 1972), pp. 417–429.

[12] Hu, H., and Ahn, G.-J. Multiparty authorization framework for data sharing in online social networks. In *Proceedings of DBSec '11* (Richmond, VA, USA, 2011), pp. 29–43.

[13] Hu, H., Ahn, G.-J., and Jorgensen, J. Detecting and resolving privacy conflicts for collaborative data sharing in online social networks. In *Proceedings of ACSAC'11* (Orlando, Florida, USA, 2011), pp. 103–112.

[14] Hu, H., Ahn, G.-J., and Jorgensen, J. Multiparty access control for online social networks: Model and mechanisms. *IEEE TKDE 25*, 7 (2013).

[15] Hu, H., Ahn, G.-J., Zhao, Z., and Yang, D. Game theoretic analysis of multiparty access control in online social networks. In *Proceedings of SACMAT '14* (London, Ontario, Canada, 2014), pp. 93–102.

[16] Khan, A. A., and Fong, P. W. L. Satisfiability and feasibility in a relationship-based workflow authorization model. In *Proceedings of ESORICS '12* (Pisa, Italy, Sept. 2012), pp. 109–126.

[17] Lampinen, A., Lehtinen, V., Lehmuskallio, A., and Tamminen, S. We're in it together: Interpersonal management of disclosure in social network services. In *Proceedings of CHI '11* (Vancouver, BC, Canada, 2011), pp. 3217–3226.

[18] Le Berre, D., and Parrain, A. The sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation 7* (2010), 59–64.

[19] Li, N., and Tripunitara, M. V. On Safety in Discretionary Access Control. In *Proceedings of IEEE S&P '05* (Oakland, CA, May 2005), pp. 96–109.

[20] Mehregan, P. *Multiple Ownership in Access Control*. PhD thesis, University of Calgary, March 2016.

[21] Mehregan, P., and .Fong, P. W. L. Design patterns for multiple stakeholders in social computing. In *Proceedings DBSec '14*. July 2014, pp. 163–178.

[22] Rizvi, S. Z. R., Fong, P. W., Crampton, J., and Sellwood, J. Relationship-based access control for an open-source medical records system. In *Proceedings of SACMAT '15* (Vienna, Austria, 2015), pp. 113–124.

[23] Squicciarini, A. C., Shehab, M., and Wede, J. Privacy Policies for Shared Content in Social Network Sites. *The VLDB Journal 19*, 6 (Dec. 2010), 777–796.

[24] Squicciarini, A. C., Xu, H., and Zhang, X. L. CoPE : Enabling collaborative privacy management in online social networks. *Journal of the American Society for Information Science 62*, 3 (2011), 521–534.

[25] Such, J. M., and Criado, N. Adaptive conflict resolution mechanism for multi-party privacy management in social media. In *Proceedings of WPES '14* (Scottsdale, Arizona, USA, 2014), pp. 69–72.

[26] Thomas, K., Grier, C., and Nicol, D. M. Unfriendly: Multi-party privacy risks in social networks. In *Proceedings of PETS '10* (Berlin, Germany, July 2010), pp. 236–252.

[27] Torán, J. On the resolution complexity of graph non-isomorphism. In *Proceedings of SAT'13* (Helsinki, Finland, July 2013), pp. 52–66.

[28] Wang, Q., and Li, N. Satisfiability and resiliency in workflow authorization systems. *ACM TISSEC 13*, 4 (Dec. 2010).

[29] Winsborough, W. H., and Li, N. Safety in automated trust negotiation. In *IEEE S&P '04* (Oakland, California, USA, May 2004), pp. 147–160.

[30] Wishart, R., Corapi, D., Marinovic, S., and Sloman, M. Collaborative privacy policy authoring in a social networking context. In *Proceedings of IEEE POLICY'10* (Fairfax, VA, July 2010), pp. 1–8.

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