Relationship-Based Access Control: Its Expression and Enforcement Through Hybrid Logic

Glenn Bruns
Bell Labs, Alcatel-Lucent
Glenn.Bruns@alcatel-lucent.com

Philip Fong Ida Siahaan
University of Calgary
{pwlfong, isriaha}@ucalgary.ca

Michael Huth
Imperial College London
M.Huth@imperial.ac.uk

ABSTRACT

Access control policy is typically defined in terms of attributes, but in many applications it is more natural to define permissions in terms of relationships that resources, systems, and contexts may enjoy. The paradigm of relationship-based access control has been proposed to address this issue, and modal logic has been used as a technical foundation.

We argue here that hybrid logic – a natural and well-established extension of modal logic – addresses limitations in the ability of modal logic to express certain relationships.

We identify a fragment of hybrid logic to be used for expressing relationship-based access-control policies, show that this fragment supports important policy idioms, and demonstrate that it removes an exponential penalty in existing attempts of specifying complex relationships such as "at least three friends" . We also capture the previously studied notion of *relational* policies in a static type system.

Categories and Subject Descriptors

D. Software [**D.4 OPERATING SYSTEMS**]: D.4.6 Security and Protection Subjects: Access controls

General Terms

Security

Keywords

Access-control models, Hybrid Logic, Relationship-based Access Control

1. INTRODUCTION

Access control is typically specified and enforced in terms of attributes: authenticated properties that must be possessed by the requester of a resource, the context of the request or the resource itself in order to grant access.

But there are many applications in which the decision of granting access should not primarily be based on attributes (e.g. whether a VPN connection is on or off) but rather on relationships that resources, systems, and contexts may enjoy. For example, a teenager may want to share pictures from a concert only with friends who actually went to the event. Expressing such policies through attributes is hard to do even within a monolithic and closed system, and is simply not feasible in distributed and open systems.

The paradigm of relationship-based access control [12, 10, 8] has been proposed to address this shortcoming of attributes. This research gives us a first understanding of appropriate *semantic* notions for relationship-based access control (see e.g. [8]). Yet, despite the initial progress reported in [10], it is less clear what appropriate syntactic counterparts these policies should have.

Ideally, a policy language for relationalship-based access control should be

- expressive enough to capture important policy idioms
- not so expressive as to make reasoning intractable
- intuitive for expressing and enforcing access control,
- formal, to support policy analysis, implementation, and optimization, and
- based on robust mathematical foundations.

Recent work [8, 10] has proposed the use of modal logics as such a mathematical foundation. We entirely agree with the spirit of that proposal, that a policy be specified as a formula of some logic. But the work in [10] already recognized that modal logic alone cannot express some pertinent relationships. Features that appear to be lacking include:

- the ability to bind a node to a principal in a relationship graph
- graded variants of modalities, e.g. "at least four friends"
- the ability to evaluate sub-policies from the perspective of a named principal, and
- the ability to *efficiently* compute a policy decision by evaluating a formula on a relationship graph.

The latter point is crucial, since existing attempts to realize the aforementioned features appear to do so at the expense of losing efficieny of policy evaluation.

We argue here that a natural and well-established extension of modal logic – *hybrid logic* [3] – can overcome these shortcomings and provide a robust mathematical foundation for relationship-based access control.

A key contribution of this paper is the recognition that fragments of hybrid logic are well-suited to the needs of relationship-based access control. Another key contribution of this paper is the demonstration that our proposed use of hybrid logic eliminates a known exponential penalty incurred in expressing important policy idioms such as "at

$$
\begin{array}{lcl}
t & ::= & n \mid x \\
\phi & ::= & t \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \langle i\rangle\phi \mid \langle -i\rangle\phi \mid \\
 & & @_t\,\phi \mid {\downarrow}x\,\phi
\end{array}
$$

Figure 1: The syntax of a simple hybrid logic HL, where n ranges over *Nom*, x ranges over *Var*, p ranges over AP, and i ranges over I.

least two colleagues" in existing modal logics for relationship-based access control [10].

As further evidence of the utility of hybrid logic, we also devise a fragment of this hybrid logic that gives rise only to policies whose access-control decisions depend only on the connectedness structure between the owner and the requester of a resource in a network graph.

Outline of paper.

We present a hybrid logic suitable for access control in Section 2. We explain, in Section 3, how this hybrid logic can be used for an access-control model based on relationships. Section 4 is devoted to examples of useful policy idioms written in this hybrid logic. A local model-checking algorithm for policy decisions is given in Section 5. In Section 6, a type system is developed for a fragment of our hybrid logic and it is shown that type-safe such formulas determine so-called *relational* policies. Related work is reflected upon in Section 7. A paper summary and remarks about future work conclude the paper in Section 8.

2. HYBRID LOGIC

We define the syntax and semantics of hybrid logic, with a view towards its application in policy-based access control.

Syntax.

We take as given a set *Nom* of *nominal* symbols, an infinite set *Var* of *variable* symbols, a set I of *label* symbols, and a set AP of *propositional* symbols. Nominals and variables allow us to bind nodes in relationship graphs to principals. Labels represent the different relationships present in that graph. Using these sets of symbols, we define formulas of a hybrid logic HL in Fig. 1. Other logical symbols can be derived in the usual way. For example,

$$
\begin{array}{ll}
[i]\phi \stackrel{\text{def}}{=} \neg\langle i\rangle\neg\phi & \phi \vee \psi \stackrel{\text{def}}{=} \neg(\neg\phi \wedge \neg\psi) \\
\bot \stackrel{\text{def}}{=} p \wedge \neg p & \top \stackrel{\text{def}}{=} p \vee \neg p
\end{array}
$$

express a must modality for label i, disjunction, falsity, and truth (respectively).

Semantics.

We define models for HL, which express relationship graphs for access control, as well as valuations, which map variables to nodes in relationship graphs.

Definition 1. *1. A* model M *of* HL *is a triple*

$$(S, \{R_i \subseteq S \times S \mid i \in I\}, V) \tag{1}$$

where S is a non-empty set of nodes, R_i a binary relation on S for all $i \in I$, and $V\colon (Nom \cup AP) \to 2^S$ a total function with $V(n)$ a singleton for all n in Nom.

2. A valuation $g\colon Var \to S$ *is a total function. For some such g, we write $g[x \mapsto s]$ for the valuation that maps x to s, and maps t to $g(t)$ if $t \neq x$.*

$$
\begin{array}{lcllcl}
M,s,g & \models & x & \stackrel{\text{def}}{=} & s = g(x) \\
M,s,g & \models & n & \stackrel{\text{def}}{=} & V(n) = \{s\} \\
M,s,g & \models & p & \stackrel{\text{def}}{=} & s \in V(p) \\
M,s,g & \models & \neg\phi & \stackrel{\text{def}}{=} & M,s,g \not\models \phi \\
M,s,g & \models & \phi_1 \wedge \phi_2 & \stackrel{\text{def}}{=} & M,s,g \models \phi_1 \text{ and } M,s,g \models \phi_2 \\
M,s,g & \models & \langle i\rangle\phi & \stackrel{\text{def}}{=} & M,s',g \models \phi \text{ for some } (s,s') \in R_i \\
M,s,g & \models & \langle -i\rangle\phi & \stackrel{\text{def}}{=} & M,s',g \models \phi \text{ for some } (s',s) \in R_i \\
M,s,g & \models & @_n\,\phi & \stackrel{\text{def}}{=} & M,s^*,g \models \phi \text{ where } V(n) = \{s^*\} \\
M,s,g & \models & @_x\,\phi & \stackrel{\text{def}}{=} & M,g(x),g \models \phi \\
M,s,g & \models & {\downarrow}x\,\phi & \stackrel{\text{def}}{=} & M,s,g[x \mapsto s] \models \phi
\end{array}
$$

Figure 2: Satisfaction relation $M, s, g \models \phi$ specifying that formula ϕ of hybrid logic HL is true in node s of model M under valuation g.

The intuition behind function V is that $V(p)$ is the set of nodes at which p is true, and that $V(n)$ is the set containing the unique node in the relationship graph of M "named" n by V. Valuations g are total functions from variables to nodes. Note that attributes of principals can be expressed in models of HL through propositions in set AP.

The meaning of formulas in HL is defined by a satisfaction relation $M, s, g \models \phi$, defined inductively in Figure 2.

Each nominal and variable is true at a single node. The meaning of nominals is specified through function V; the meaning of variables through valuation g. Propositions, on the other hand, are true at zero, one, or more nodes.

The meanings of conjunction and negation are standard, as is the meaning of the modalities. Formula $\langle i\rangle\phi$ holds at s if there is some R_i-successor s' of s such that ϕ holds at s'. Dually, $\langle -i\rangle\phi$ holds at s if there is some R_i-predeccesor $\tilde{s}$ of S such that ϕ holds at $\tilde{s}$.

We now discuss the *hybrid* operators of HL. Intuitively, $@_t\,\phi$ jumps to the node named by t, whereas ${\downarrow}\,x\,\phi$ binds the current node to variable x. Formally, formula $@_t\,\phi$ says that ϕ holds at the unique node identified by t with respect to function V (if t is a nominal) or valuation g (if t is a variable). Formula ${\downarrow}x\,\phi$ holds at node s if ϕ holds at s, but where now valuation g is updated so that x identifies s.

Notational conventions.

Operator ${\downarrow}x$ is a binding operator with the usual notions of free and bound variables. We fix some notation for satisfaction checks.

Definition 2. *Let ϕ be a formula of* HL.

1. *If ϕ contains no free variables, we write $M, s \models \phi$ for any $M, s, g \models \phi$, as the evaluation of latter expression is independent of the choice of valuation g.*
2. *If ϕ contains only free variables $x_1, \dots, x_n$, we write $M, s, [x_1 \mapsto s_1, \dots, x_n \mapsto s_n] \models \phi$ for any $M, s, g[x_1 \mapsto s_1, \dots, x_n \mapsto s_n] \models \phi$, as the evaluation of the latter expression is again independent of the choice of g.*
3. *If ϕ is the Boolean combination of formulas of the form $@_t\,\psi$, we write $M, g \models \phi$, as then either $M, s, g \models \phi$ is true at all nodes s or false at all nodes s.*

We now use our hybrid logic HL to define an access-control model, focusing on the policy-decision point of such a model.

3. ACCESS-CONTROL MODEL

We outline here an archetypical model for relationship-based access control that uses hybrid logic for expressing access-control policies.

Protection state.

A protection state is simply a model M of HL in which the elements of S denote principals. The binary relations R_i represent interpersonal relations tracked by the access-control system. Each label i in I identifies one type of relationship (e.g., "parent"), such that (s_1, s_2) is in R_i iff principals s_1 and s_2 participate in a relationship of type i. In short, $(S, \{R_i \mid i \in I\})$ represents a directed, poly-relational social network. The valuation V specifies attributes of the principals. Each propositional symbol p in AP denotes an attribute. If a principal s in S has the attribute p, then s is in $V(p)$. Similarly, globally significant principals (e.g., trusted authorities) are identified by nominals via V.

Fragment of HL *for specifying access-control policies.*

Principals own resources and seek access on resources. We associate with each object *obj* a formula ϕ in HL that expresses a relationship between two parties: the *owner* and the *requester*. Then the requester is permitted access to *obj* if the owner and requester are in the relationship specified by ϕ. This approach has been suggested in [8] already, for a different logic, and it is what makes the access-control model *relationship-based*.

We propose to use two distinguished variables own and req to denote the principal who is the owner and the requester of the implicit object, respectively.

Definition 3. *Let* $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ *be the set of formulas of* HL *that*

- *contain at most* own *and* req *as free variable, and*
- *are Boolean combinations of formulas of form* $@_{\mathsf{own}}\, \phi$ *and* $@_{\mathsf{req}}\, \psi$.

We refer to formulas of $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ as *policies*. A common policy pattern is a conjunction

$$@_{\mathsf{own}}\, \phi \wedge @_{\mathsf{req}}\, \psi \tag{2}$$

This specifies an access-control policy of the resource owner own via two sub-policies. Sub-policy $@_{\mathsf{own}}\, \phi$ represents the perspective of the owner in the protection state, while $@_{\mathsf{req}}\, \psi$ represents the requester's perspective. Note that (2) is consistent with using only one such perspective. For example, dropping the second conjunct amounts to choosing ψ to be $\top$ and so the righthand conjunct is redundant.

Authorization decision.

The access-control system arrives at an authorization decision as follows. Given a protection state (i.e., a model) M, a requesting principal r has permission to access an object *obj* in *Obj* that is controlled by principal o according to a policy *pol* in $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ iff the following condition holds:

$$M, [\mathsf{own} \mapsto o, \mathsf{req} \mapsto r] \models \mathit{pol} \tag{3}$$

Intuitively, the only free variables own and req of formula *pol* of $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ get bound to the principals named by the owner and requester, respectively, and this formula is then evaluated with those bindings.

Thus, checking whether an access is granted amounts to checking the satisfaction of a $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ formula with respect to a valuation. We discuss a local model-checking algorithm for $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ below.

Policy functions.

Through the semantics of hybrid logic, which has been defined as a satisfaction relation, every policy ϕ in $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ induces a *policy function*, which maps protection states to binary *permission relations.*

Definition 4. *1. A policy function P is a total function $M = (S_M, \dots) \mapsto P(M)$ from models M to binary relations $P(M) \subseteq S_M \times S_M$.*

2. The policy function induced by a policy ϕ of $\mathsf{HL}(\mathsf{own}, \mathsf{req})$*, written* $\mathsf{p}[\phi]$*, is defined as follows:*

$$\mathsf{p}[\phi] = \{(o, r) \in S_M \times S_M \mid M, [\mathsf{own} \mapsto o, \mathsf{req} \mapsto r] \models \phi\}$$

The intuition behind policy functions is that a requester r can access an object of owner o in protection state M if (o, r) is in $P(M)$ – and is denied access if (o, r) is not in $P(M)$. Note that a policy function maps a collection of relations (plus a mapping for nominals and propositions) over a set of principals to a single permission relation over the same set.

4. EXAMPLE POLICIES

We now provide example policies, starting with basic ones.

Basic policies.

If the owner of an object defines the policy to be

$$@_{\mathsf{own}}\, \langle \mathit{friend} \rangle \mathsf{req} \tag{4}$$

then every friend of the resource owner is granted access to that resource. Note that this policy specifies no constraints from the perspective of the requester. Similarly, the formula

$$@_{\mathsf{own}}\, \langle \mathit{friend} \rangle (\mathsf{req} \vee \langle \mathit{friend} \rangle \mathsf{req}) \tag{5}$$

captures a "friend or a friend of a friend" policy. The formula

$$@_{\mathsf{own}}\, (\langle \mathit{teammate} \rangle \mathsf{req} \wedge \langle \mathit{friend} \rangle \mathsf{req}) \tag{6}$$

specifies that all friends who are teammates get access. This example already hints at the usefulness of hybrid logic. The use of the variable req allows us to refer to some principal who is *both* a teammate and a friend of the owner.

If req behaved like a normal proposition that could be true at zero or more nodes, then we could not capture this semantic conjunction in the logic; the existential quantification of $\langle i \rangle$ does not distribute through conjunction.

The policies we have specified so far are variants or adaptations of policies written the logic E of [10]. To see the true benefits of using HL, we write more complex policies next.

More complex policies.

Consider a teacher who wants to confine access to her picture to friends who are teachers, but who also wants to grant access to only those friends of such friends who are not taught by friends. This is hard to express in English, but there is a true case of an English teacher who had to resign because her picture could be seen by student friends of friends – unbeknownst to her. Hybrid logic can express such a policy unambiguously. One possible formula is

$$@_{\mathsf{own}}\,(\langle \mathit{friend}\rangle(\mathsf{req} \wedge \mathit{isTeacher}) \vee \langle \mathit{friend}\rangle(\mathit{isTeacher} \wedge \langle \mathit{friend}\rangle\mathsf{req} \wedge \neg\langle \mathit{student}\rangle\mathsf{req})) \quad (7)$$

Note that this formula is indeed in $\mathsf{HL}(\mathsf{own}, \mathsf{req})$. The first disjunct specifies that access is granted if the requester is a friend who is a teacher, where $\mathit{isTeacher}$ is in AP. The second disjunct specifies that access is granted if the requester is a friend of a teacher friend of the owner, but where the requester is also not a student of that teacher friend.

We point out that this policy does not prevent the scenario where access is granted to *some* student of a teacher friend, since the may modality is an existential quantification.

This policy also illustrates the utility of propositions (attributes), as we here want to express that a friend is *some* teacher, not necessarily the teacher or student of the owner or requester. This property could be expressed through $\langle \mathit{teacher}\rangle\top$, someone is a teacher if they teach someone. But such an encoding becomes ackward for unary relations such as $\mathit{isDiabetic}$, so we do include propositions in our logic HL.

Dual policies.

The last example illustrates that we can compose negative and positive permissions. For example, the policy

$$@_{\mathsf{own}}\,\langle \mathit{friend}\rangle(\mathsf{req} \wedge \neg \mathit{Alice}) \quad (8)$$

says access is granted to all friends, except to *Alice*, a principal that is a nominal in *Nom*. Note that this policy makes no assumption about whether or not *Alice* is actually a friend.

The policy expressed in (8) serves also as an example for how one might "dualize" a policy written from the perspective of one of the principals into an equivalent one written from the perspective of the other principal. The policy

$$@_{\mathsf{req}}\,(\langle -\mathit{friend}\rangle\mathsf{own} \wedge \neg \mathit{Alice}) \quad (9)$$

is intuitively equivalent to that in (8), but of form $@_{\mathsf{req}}\,\phi$.

Graded modalities.

A nice feature of hybrid logic is that it allows us to express so called *graded* may modalities [3] as syntactic sugar of the logic. To see this let n be a positive natural number, and let $\langle i\rangle_n$ be the corresponding graded may modality for label i. The intuition of $\langle i\rangle_n\phi$ is that it holds in node s iff there are *at least* n many R_i-successors of s at which ϕ holds.

For example, a policy that grants access if there are at least 3 friends of the owner who satisfy ϕ can be specified as

$$@_{\mathsf{own}}\,\langle \mathit{friend}\rangle_3\phi \quad (10)$$

To see that $\langle i\rangle_n\phi$ is expressible in HL, take $n+1$ many variables x, y_1, y_2, $\dots y_n$ that do not occur in ϕ and define

$$\begin{aligned}\langle i\rangle_n\phi \;\stackrel{\text{def}}{=}\; & \downarrow x\,\langle i\rangle\downarrow y_1\,(\phi \wedge \qquad (11)\\ & @_x\,\langle i\rangle\downarrow y_2\,(\neg y_1 \wedge \phi \wedge\\ & @_x\,\langle i\rangle\downarrow y_3\,(\neg y_1 \wedge \neg y_2 \wedge \phi \wedge\\ & \dots\\ & @_x\,\langle i\rangle\downarrow y_n\,(\neg y_1 \wedge \neg y_2 \wedge \dots \wedge \neg y_{n-1} \wedge \phi))\dots)\end{aligned}$$

This standard encoding says that at the current node, named x, there is some R_i-successor y_1 of x that satisfies ϕ, and there is some R_i-successor y_2 of x that is different from y_1 and satisfies ϕ, and so on. This clearly renders the desired semantic effect of having at least n many R_i-successors of the currrent node that satisfy ϕ.

Graded modalities give us the ability to count exactly as well. To specify that there are exactly n R_i-successors that satisfy ϕ, we may write

$$\langle i\rangle_{=n}\phi \stackrel{\text{def}}{=} \langle i\rangle_n\phi \wedge \neg\langle i\rangle_{n+1}\phi \quad (12)$$

One can model a rudimentary trust level in a network of friends by asking whether the requester is connected to the owner by a path of *friend* labels of length at most k.

We can specify such policies inductively for $k \geq 1$ as

$$\begin{aligned}\mathsf{depth}[\mathit{friend}, 1] &\stackrel{\text{def}}{=} \langle \mathit{friend}\rangle\mathsf{req} \qquad (13)\\ \mathsf{depth}[\mathit{friend}, k+1] &\stackrel{\text{def}}{=} \mathsf{depth}[\mathit{friend}, k] \vee \langle \mathit{friend}\rangle\mathsf{depth}[\mathit{friend}, k]\end{aligned}$$

We can combine this trust mechanism with graded modalities to express a policy that grants access if there are at least two colleagues who have sufficient trust in the requester:

$$@_{\mathsf{own}}\,\langle \mathit{colleague}\rangle_2\mathsf{depth}[\mathit{friend}, 3] \quad (14)$$

Next, let us now consider a policy cf_k based on common friends. It grants access to the owner, to his friends, and to those who have at least $k > 0$ common friends. Through graded may modality, we can express this in $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ as

$$\mathsf{cf}_k \stackrel{\text{def}}{=} @_{\mathsf{own}}\,(\mathsf{req} \vee \langle \mathit{friend}\rangle\mathsf{req} \vee \langle \mathit{friend}\rangle_k\langle \mathit{friend}\rangle\mathsf{req}) \quad (15)$$

The leftmost disjunct specifies that the owner has access, the second disjunct says that friends of the owner have access, and the third disjunct says that access is granted to requesters who have at least k many friends in common with the owner. The encoding assumes that R_{friend} is symmetric. If not, the last modality should be an inverse one.

Here is an example policy of $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ that has non-dual subpolicies from the perspective of owner and requester:

$$@_{\mathsf{own}}\,(\langle \mathit{friend}\rangle\mathsf{req} \wedge \langle \mathit{friend}\rangle_3\top) \wedge @_{\mathsf{req}}\,\langle \mathit{friend}\rangle_5\neg\mathsf{own} \quad (16)$$

This policy grants access if the requester is a friend of the owner, the owner has at least three friends (counting or not counting the requester), and if additionally the requester has at least five friends other than the owner. Intuitively, in order to express this policy in $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ we seem to require both operators $@_{\mathsf{req}}$ and $@_{\mathsf{own}}$ as the policy involves a form of counting in both nodes named by own and req.

5. LOCAL MODEL CHECKING

In our setting, we are given a model M as protection state, and a policy *pol* in $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ as specification of access control. We then wish to decide whether $M, [\mathsf{own} \mapsto o, \mathsf{req} \mapsto r] \models \mathit{pol}$ for specified nodes o and r.

This form of evaluation is a kind of *local* model checking. The term "local" is used because the intuition is that the model M is explored only as needed from the nodes of interest, those named by own and req. Local model checking is a good fit for our needs as only those portions of the model that are potentially needed to make an access-control decision are explored.

Description of algorithm.

We now describe our local model-checking algorithm for $\mathsf{HL}(\mathsf{own}, \mathsf{req})$. Its pseudo-code is depicted in Figure 3. The

```
MC($s$,$g$,$\phi$) {
  case {
    $\phi$ is $x$ : return (areEqual($s$,$g(x)$));
    $\phi$ is $n$ : return (isElem($s$,$V(n)$));
    $\phi$ is $p$ : return (isElem($s$,$V(p)$));
    $\phi$ is $\neg\psi$ : return (!MC($s$,$g$,$\psi$));
    $\phi$ is $\psi_1 \wedge \psi_2$ : return (MC($s$,$g$,$\psi_1$) && MC($s$,$g$,$\psi_2$));
    $\phi$ is $\langle i\rangle\psi$ : return MCmay($s$,$g$,$\psi$,$i$,fwd);
    $\phi$ is $\langle -i\rangle\psi$ : return MCmay($s$,$g$,$\psi$,$i$,bwd);
    $\phi$ is $@_n\,\psi$ : let $t$ = hd($V(n)$) in return MC($t$,$g$,$\psi$);
    $\phi$ is $@_x\,\psi$ : let $t$ = $g(x)$ in return MC($t$,$g$,$\psi$);
    $\phi$ is $\downarrow x\,\psi$ : let $h$ = $g[x \mapsto s]$ in return MC($s$,$h$,$\psi$);
   }
}

MCmay($s$,$g$,$\phi$,$i$,direction) {
  if (direction == fwd) { $X$ = [ $s'$ | $(s,s')$ in $R_i$ ];
  } else                { $X$ = [ $s'$ | $(s',s)$ in $R_i$ ]; }
  for (all $s'$ in $X$) {
      if (MC($s'$,$g$,$\phi$)) { return true; }
   }
   return false;
}

policyDecision($o$,$r$,$\phi$) {  // $\phi$ in HL(own,req)
  let $g$ = [own $\mapsto$ $o$, req $\mapsto$ $r$] in {
    return MC($o$,$g$,$\phi$);
  }
}
```

Figure 3: Local model checking algorithm for HL.

model M is implicit but its structure and local nodes and valuations are explicit in the code.

The algorithm MC has as argument a local node s, a valuation g, and a formula ϕ of HL. Its body is a case analysis of the top-level operator of ϕ. We assume that $V(n)$ and $V(p)$ are implemented as lists, that isElem is a membership test for such lists, that hd returns the first element of a non-empty list, and that areEqual can check for node equality.

The algorithm does a recursive decent until it encounters formulas that are variables, nominals, or propositions. The cases of the forward and backward modalities require a call to a sub-routine MCmay.

Routine MCmay first computes the set of all R_i-successors or predecessors of node s, where that choice is decided by an inspection of a parameter value that indicates whether the modality is a forward one $\langle i\rangle$ or a backward one $\langle -i\rangle$.

It then iterates through all these nodes until one of them makes the formula true, in which case it does return true. If no such node makes the formula true, it returns false.

A genuine implementation of algorithm MC would *not* precompute the sets X, but would generate new elements of X on demand until either a witness for truth has been found, or all elements of X are revealed not to be such witnesses. Also, to make this efficient each reached node would keep a hash table of subformulas it has already evaluated so that each subformula is evaluated at most once in each node.

We can use that local model-checking algorithm to implement a policy decision point in policyDecision. It takes as input a node o representing the owner of the object, a node r representing the requester of that object, and a formula ϕ of HL(own, req) representing the access-control policy. Then it creates the valuation that binds own and req to o and r, respectively. Finally, it returns the result of the local model check on ϕ under valuation g.

6. RELATIONAL POLICIES

Previous work [2, 10] has noted that what distinguishes relationship-based access control from traditional access-control paradigms is its extensive use of *relational policies.* Intuitively, a relational policy is one in which the authorization decision is based *solely* on how owner and requester are connected to one another (e.g., friend, friend-of-friend, etc). Therefore, a relational policy does not base its authorization decision on the requester's identity (e.g., John can access), attributes (e.g., managers can access), or social positions (e.g., all those who have at least 100 friends can access).

The goal of this section is to identify a syntactic fragment of the logic HL(own, req) that captures relational policies. This then allows policy developers to efficiently verify that their policy specifications determine relational policies.

Relational policies.

We begin by formalizing what it means for a policy function to be relational. We first define some auxiliary concepts that use only the binary relations R_i of models.

Definition 5. *Fix models $M = (S_M, \{R_i^M \subseteq S_M \times S_M \mid i \in I\}, V_M)$ and $N = (S_N, \{R_i^N \subseteq S_N \times S_N \mid i \in I\}, V_N)$.*

1. *These models are* isomorphic *via a bijection $f\colon S_M \to S_N$, if, for all $i \in I$: $(s,t) \in R_i^M$ iff $(f(s), f(t)) \in R_i^N$.*
2. *Nodes s,t of S_M are* connected*, written $s \overset{M}{\leftrightsquigarrow} t$, if either $s = t$, or inductively, there is an s' in S_M with $(s,s') \in R_i^M \cup (R_i^M)^{-1}$, for some $i \in I$, and $s' \overset{M}{\leftrightsquigarrow} t$.*
3. *The* shared component *of nodes s,t in M, written $\mathsf{SC}(M,s,t)$, is the relational structure $(S, \{R_i \subseteq S \times S \mid i \in I\})$ defined as follows:*

$$S \overset{\text{def}}{=} \{s,t\} \cup \{s' \in S_M \mid s \overset{M}{\leftrightsquigarrow} s', s' \overset{M}{\leftrightsquigarrow} t\}$$
$$R_i \overset{\text{def}}{=} R_i^M \cap (S \times S)$$

There are two parts to what it means to be a relational policy function. First, the policy function must be *topology-based*: it must consume neither attribute information (i.e., valuations are not considered) nor "identity information" (i.e., isomorphic labeled graphs cannot be distinguished) when an authorization decision is made.

Definition 6. *A policy function P is* topology-based *if, for all models M and N, and all bijections $f : S_M \to S_N$, whenever M and N are isomorphic via f, then $P(N) = \{(f(s), f(t) \mid (s,t) \in P(M)\}$.*

Second, the policy function must be *local*: changes in permission to a model must reflect a change in connectivity between the owner and requester.

Definition 7. *A policy function P is* local *iff, for all models M and N, and all $s,t \in S_M \cap S_N$, we have that*

$$\mathsf{SC}(M,s,t) = \mathsf{SC}(N,s,t) \text{ implies } P(M)(s,t) = P(N)(s,t).$$

The definition of "local" demands that any change in an access decision must imply that either s and t have gone from being disconnected to being connected (or vice versa), or the shared component of s and t have been altered.

Such a requirement ensures that any change of authorization decision is not caused merely by the social positions of the requester or the owner (i.e., where exactly they are in the relational structure). We note that this definition of local-ness is equivalent to the one given in [10].

$$\begin{array}{lcl}\psi & ::= & \top \mid \bot \mid x \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \\ & & \langle i\rangle\psi \mid \langle -i\rangle\psi \mid [i]\psi \mid [-i]\psi \mid @_x\,\psi \mid \downarrow x\,\psi \\ \phi & ::= & \top \mid \bot \mid @_{\mathsf{own}}\,\psi \mid @_{\mathsf{req}}\,\psi \mid \neg\phi \mid \phi\wedge\phi \mid \phi\vee\phi\end{array}$$

Figure 4: Syntax of candidate formulas for a relational fragment of $\mathsf{HL}(\mathsf{own},\mathsf{req})$, with x from Var.

Example. The following formulas express local policies:

$$@_{\mathsf{own}}\,(\langle\mathsf{child}\rangle\mathsf{req} \wedge [\mathsf{child}]\mathsf{req}) \quad (17)$$

$$@_{\mathsf{own}}\,\langle\mathsf{friend}\rangle(\mathsf{req} \wedge \langle\mathsf{spouse}\rangle\top) \quad (18)$$

Formula (17) demands the requester to be the *only* child of the owner. Formula (18) requires the requester to be a *married* friend of the owner.

The following formulas express policies that are not local:

$$@_{\mathsf{req}}\,\langle\mathsf{spouse}\rangle\top \quad (19)$$

$$@_{\mathsf{own}}\,[\mathsf{child}]\mathsf{req} \quad (20)$$

Formula (19), which is a relaxation of (18), requires the requester to be married. It is not local as a change of authorization decision only requires the introduction (or elimination) of a spouse edge in the neighbourhood of the requester, which may otherwise be disconnected from the owner.

Formula (20), which is a relaxation of (17), grants access either if the owner has no child, or if the requester is the only child. The following explains why it is not a local policy. Suppose we start with a model M in which the owner has no child, and the requester is disconnected from the owner. The requester has access in M. Now the owner gives birth to a child, resulting in a new model N, in which the owner is incident to a child edge. The requester loses his or her access in protection state N, but the owner and the requester remain disconnected. **(End of Example.)**

We can now define the technical notion of relational policy.

Definition 8. *A policy function is* relational *iff it is both topology-based and local.*

A relational fragment of $\mathsf{HL}(\mathsf{own},\mathsf{req})$.

It is easy to show that every formula in $\mathsf{HL}(\mathsf{own},\mathsf{req})$ expresses a topology-based policy. To obtain relational policies, the challenge is to ensure that formulas are constructed to express *only* local policies. To better appreciate the nature of this challenge, observe the following:

Proposition 1. *The family of local policies contains* $\mathsf{p}[\top]$ *and* $\mathsf{p}[\bot]$ *and is closed under boolean combinations.*

Thus complex local policies can be built up from primitive ones using the boolean connectives offered by $\mathsf{HL}(\mathsf{own},\ \mathsf{req})$. It is the modal operators $\langle i\rangle$ and $\langle -i\rangle$ that may break local-ness of policies. The reason is that, in general, local-ness is not preserved by relational composition.

Suppose P_1 and P_2 are policy functions. We write $P_1 \circ P_2$ to denote the policy function P such that $P(M) = \{(s,t) \in S_M \times S_M \mid \exists s' \in S_M\colon (s,s') \in P_1(M), (s',t) \in P_2(M)\}$. Even though policy functions P_1 and P_2 may both be local, $P_1 \circ P_2$ need not be local. An example is when P_2 is $\mathsf{p}[\top]$. Thus $\langle i\rangle\psi$ may not be local even if ψ is local. This is illustrated by formula (19).

We define a relational fragment of $\mathsf{HL}(\mathsf{own},\mathsf{req})$ to ensure that modal operators preserve local-ness. This relational fragment is obtained in two steps. First, we constrain the syntax of $\mathsf{HL}(\mathsf{own},\mathsf{req})$ by the grammar in Figure 4. Second, we impose a type system (Figure 5) to further constrain formula construction, such that only properly typed formulas belong to the fragment.

Definition 9. *The hybrid logic fragment* HL_{rel} *contains those formulas* ϕ *as defined in Figure 4 such that:*

- *For all subformulas* $@_{\mathsf{own}}\,\psi$ *of* ϕ*, we have* $\psi : \mathsf{Local}(\mathsf{req})$.
- *For all subformulas* $@_{\mathsf{req}}\,\psi$ *of* ϕ*, we have* $\psi : \mathsf{Local}(\mathsf{own})$.

We now explain the two steps of restricting $\mathsf{HL}(\mathsf{own},\mathsf{req})$ in turn. First, the syntax of ϕ in Figure 4 reiterates the requirement of $\mathsf{HL}(\mathsf{own},\mathsf{req})$, that policy formulas are boolean combinations of subformulas of the form "$@_{\mathsf{own}}\,\psi$" or "$@_{\mathsf{req}}\,\psi$". As long as these subformulas express local policies, then Proposition 1 ensures that ϕ is local.

Second, the inference rules in Figure 5 involve two type judgments of form "$\psi : \mathsf{Local}(x)$" and "$\psi : \mathsf{OC}(x)$", where x is typically either own or req. Intuitively, the derivation of judgment $\psi : \mathsf{Local}(\mathsf{req})$ means ψ can be used within $@_{\mathsf{own}}$ to form a local policy. Similarly, the derivation of judgment $\psi : \mathsf{Local}(\mathsf{own})$ ensures that $@_{\mathsf{req}}\,\psi$ is a local policy. The judgment based on label $\mathsf{OC}(x)$ is for typing subformulas ψ that are "*owner-checkable (OC)*" as defined in [10]. Although OC formulas are not local, they can be combined with local formulas in a conjuction to yield local formulas.

Theorem 1 (Type Soundness). *If* $\psi : \mathsf{Local}(\mathsf{req})$*, then* $\mathsf{p}[@_{\mathsf{own}}\,\psi]$ *is a local policy. Similarly, if* $\psi\colon \mathsf{Local}(\mathsf{own})$*, then* $\mathsf{p}[@_{\mathsf{req}}\,\psi]$ *is a local policy.*

To see an application of this theorem, consider the policy in (18) of form $@_{\mathsf{own}}\,\psi$. The first conjunct type checks as $\mathsf{Local}(\mathsf{req})$, the second one as $\mathsf{OC}(\mathsf{req})$. So their conjunction and also ψ both type check as $\mathsf{Local}(\mathsf{req})$. Theorem 1 ensures the policy itself is local. The policy in (17) can be type checked in a similar fashion.

An attempt to type check the policy of form $@_{\mathsf{own}}\,\psi$ in (19), however, fails: our type system can assign type $\mathsf{OC}(\mathsf{own})$ to ψ but this cannot be lifted to type $\mathsf{Local}(\mathsf{own})$. Policy (20) fails to type check to $\mathsf{Local}(\mathsf{req})$ for a similar reason.

The encoding of graded modalities $\langle i\rangle_n\phi$ given in (11) will type check to $\mathsf{Local}(x)$ so long as ϕ type checks to $\mathsf{Local}(x)$.

Our typing rules can also type check the depth-first search tree encoding of finitary relational policies [10, Appendix A].

In summary, we have identified a relational fragment HL_{rel} via a grammar and a type system. This allows policy developers to efficiently verify whether the policy formula under development is indeed relational.

7. RELATED WORK

Modal logics for access control.

Access-control logics for distributed systems can be interpreted as modal logics, with Kripke-style, possible-world semantics. In ABLP [1], every principal is a modality, and states in a model are epistemic states. ICL and its variants [11] can be compiled into formulas in the modal logic S4. Following [8, 10], $\mathsf{HL}(\mathsf{own},\mathsf{req})$ formulas are interpreted against models that capture principal attributes and social

$$\frac{\psi : \mathsf{Local}(x)}{\psi : \mathsf{OC}(x)} \quad (\text{O-Sub})$$

$$\frac{}{\top : \mathsf{OC}(x)} \quad (\text{O-Top})$$

$$\frac{}{\bot : \mathsf{OC}(x)} \quad (\text{O-Bot})$$

$$\frac{}{y : \mathsf{OC}(x)} \quad (\text{O-Var})$$

$$\frac{\psi : \mathsf{OC}(x)}{\neg\psi : \mathsf{OC}(x)} \quad (\text{O-Not})$$

$$\frac{\psi_1 : \mathsf{OC}(x) \qquad \psi_2 : \mathsf{OC}(x)}{\psi_1 \vee \psi_2 : \mathsf{OC}(x)} \quad (\text{O-Or})$$

$$\frac{\psi_1 : \mathsf{OC}(x) \qquad \psi_2 : \mathsf{OC}(x)}{\psi_1 \wedge \psi_2 : \mathsf{OC}(x)} \quad (\text{O-And})$$

$$\frac{\psi : \mathsf{OC}(x)}{\langle i\rangle\psi : \mathsf{OC}(x)} \quad (\text{O-May})$$

$$\frac{\psi : \mathsf{OC}(x)}{[i]\psi : \mathsf{OC}(x)} \quad (\text{O-Mus})$$

$$\frac{\psi : \mathsf{OC}(x) \qquad y \neq x}{@_y\,\psi : \mathsf{OC}(x)} \quad (\text{O-At})$$

$$\frac{\psi : \mathsf{OC}(x) \qquad y \neq x}{\downarrow y\,\psi : \mathsf{OC}(x)} \quad (\text{O-Dow})$$

$$\frac{}{\bot : \mathsf{Local}(x)} \quad (\text{L-Bot})$$

$$\frac{}{x : \mathsf{Local}(x)} \quad (\text{L-Var})$$

$$\frac{\psi_1 : \mathsf{Local}(x) \qquad \psi_2 : \mathsf{Local}(x)}{\psi_1 \vee \psi_2 : \mathsf{Local}(x)} \quad (\text{L-Or})$$

$$\frac{\psi_1 : \mathsf{OC}(x) \qquad \psi_2 : \mathsf{Local}(x)}{\psi_1 \wedge \psi_2 : \mathsf{Local}(x)} \quad (\text{L-And1})$$

$$\frac{\psi_1 : \mathsf{Local}(x) \qquad \psi_2 : \mathsf{OC}(x)}{\psi_1 \wedge \psi_2 : \mathsf{Local}(x)} \quad (\text{L-And2})$$

$$\frac{\psi : \mathsf{Local}(x)}{\langle i\rangle\psi : \mathsf{Local}(x)} \quad (\text{L-May})$$

$$\frac{\psi : \mathsf{Local}(x) \qquad y \neq x}{@_y\,\psi : \mathsf{Local}(x)} \quad (\text{L-At})$$

$$\frac{\psi : \mathsf{Local}(x) \qquad y \neq x}{\downarrow y\,\psi : \mathsf{Local}(x)} \quad (\text{L-Dow})$$

Figure 5: A type system for formulas from Fig. 4, identifying polices that are local. The typing rules for inverse modalities $\langle -i\rangle$ and $[-i]$, which closely parallel O-May, O-Mus and L-May, are omitted for brevity.

networks. The nodes of a model denote principals, modalities correspond to relationship types, and accessibility relations express interpersonal relations.

Modal logics for relationship-based access control.

In [10], a modal logic E for relationship-based access control was developed as an extension and improvement of a similar modal logic B [8]. So we focus our discussion on E here.

Its abstract syntax is given by

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \langle i\rangle\phi \mid \langle -i\rangle\phi \mid {\downarrow} p\,\phi \mid \phi_1 \otimes \phi_2$$

where p ranges over a set of propositional symbols with a reserved symbol a, and i ranges over a set of labels. In [10], the operator $\downarrow p$ is actually written $@_p$ but we chose the former notation here, as it will aid our comparative discussion.

The models for E are similar to those for HL. Semantically, propositions are treated in E like nominals in hybrid logic: they are true in exactly one node. Symbol a represents the requester of an access. The semantics of the $\downarrow p$ operator is actually that of the $\downarrow n$ operator in hybrid logic, except that $\downarrow \mathsf{a}\,\phi$ is interpreted as $\bot$. Intuitively, $\downarrow p\,\phi$ identifies p with the owner of the object.

Lastly, operator $\phi \otimes \psi$ holds if one can separate model M into two parts of node sets that only overlap for the owner and requester, and where ϕ holds in one of these parts and ψ holds in the other one.

This operator crucially increased the expressiveness of **B** so that, e.g., thresholds on the number of specific successors of a node can be expressed. Since one can encode graded modalities and similar counting mechanisms in HL, we do not need to add such an operator to our logic.

We view this fact as a big advantage of using HL instead of E. For $\phi \otimes \psi$ incurs a seemingly unavoidable exponential penalty, since there are exponentially many partitions of the set of nodes that need to be considered in its evaluation. The evaluation of HL over models, however, is linear in the size of the model and formula with appropriate caching in place.

Another advantage of HL is the greater flexibility of atomic formulas: HL offers nominals, variables, and propositions (i.e. attributes). Further, the operator $@_t$ is very useful as it specifies where a policy should be evaluated. In fact, by promoting $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ we suggest most policies would be Boolean combinations of policies of form $@_t\,\psi$.

Hybrid logics for access control.

We are not aware of much other work on using hybrid logic for access control. But in [5], a logical framework is used to design an authorization logic for time-dependent access-control decisions. The logic contains a variant of the $@_t$ operator, $A@I$, which allows the relativization of the truth of proposition A to the time interval I.

Local-ness in ReBAC policies.

Local-ness ensures that change of authorization decision is due to a change in "connectivity" between the owner and the requester of an object. The notion was originally formulated in [2] (an extension of [9]), in which a local policy is one such that, if there is a change of authorization decision due to the introduction of one new edge, then the new edge must be connected to both the owner and the requester. The definition was based on social networks that are undirected graphs, with no edge labels.

In [7], the definition was generalized to account for the introduction of "one or more" edges. The two definitions can be shown to be equivalent. In [10], the definition was adapted to account for social networks that are directed and poly-relational, such that the definition in [7] is merely a special case. In this work we adopt a formulation of local-ness that makes explicit the kind of graph structures that local policies cannot distinguish. One can show that this new formulation is equivalent to that of [10].

8. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed the use of hybrid logic for the specification and enforcement of access-control decisions in the relationship-based approach to access control.

Concretely, we presented a fragment of hybrid logic that is customized to the needs of relationship-based access control. We demonstrated that the models of that hybrid logic are appropriate as models of protection states in relationship-based access control. We showed how the semantics of hybrid logic on such models gives meaning to access-control policies written in that logic.

Then we discussed how this semantics can be implemented as a policy-decision point, via a local model-checking algorithm. Next, we featured numerous examples of policies and showed how they can be elegantly specified in our hybrid logic. Importantly, we showed how hybrid logic can express graded modalities such as "at least three friends".

To understand better connections with related work, we identified an attribute-free fragment of our hybrid logic, via a static type system, in which only policies can be specified that are *relational* in a technical sense from the literature.

We then discussed related work and how our approach improves on it. Let us finally point out future work.

Heterogeneous protection state.

It would be beneficial to let nodes model either *subjects or objects*, so that relationships can also be expressed between subjects and objects. An example of the sort of policy one could then write in HL is

$$@_{\mathsf{req}}\,\langle \mathit{sameFloor}\rangle \mathsf{resc} \wedge @_{\mathsf{own}}\,\langle \mathit{collaborator}\rangle \mathsf{req} \wedge$$
$$@_{\mathsf{resc}}\,\mathit{isPrinter} \wedge @_{\mathsf{file}}\,\mathit{isPDF}$$

This says that a print job should be granted those requesters req who are on the same floor as the resource resc, and where additionally the resource resc is a printer, the file file to be printed is in PDF format, and the owner own is a collaborator of the requester req.

Policy composition.

One may think of policies written in HL as just one aspect of control to resources. Different aspects of controlling access could then be combined. For example, in [8] there is a mechanism for determining an appropriate context for evaluating a policy function, where the context is an appropriate model for that policy. Since policies ϕ in HL determine policy functions as well, one can readily use the protection model of [8] with our hybrid logic HL.

Another example is when we have an attribute-free language, e.g. HL with empty AP. Then we may treat each HL policy as a "rule", and then combine rules in a PBel style policy composition language [4], perhaps with rules from other policy languages that refer to attributes.

Footprint of policy evaluation.

Our local model-checking algorithm can be made to run linearly in the size of the model. This low complexity is little comfort for social networks graphs with hundreds of millions of nodes. But policies used by people in social networks appear to have a small footprint, meaning that only few nodes and edges of the social relationship graph are reached and computed upon during local model checking. We illustrate this point with a Facebook style policy.

In Facebook, access policies are given by fixed options such as Everyone, Friends of Friends, Only Friends, and Customize [6]. The default policy is Everyone which involves no relations. Policies commonly use the options Only Friends or Customize. In the option Customize the owner builds lists of groups or lists of friends that are granted or denied access.

In contrast, option Friends of Friends, specified in (5), grants access to all friends and friends of friends. Its evaluation may visit all friends-of-friends nodes. But its equivalent policy

$$@_{\mathsf{own}}\,\langle \mathit{friend}\rangle(\mathsf{req} \vee \downarrow x\, @_{\mathsf{req}}\,\langle \mathit{friend}\rangle x)$$

only explores the owner's friends nodes.

Acknowledgments

We thank Ian Hodkinson for having pointed out to us a wealth of literature on hybrid logic and for always having been willing to discuss technical issues.

9. REFERENCES

[1] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, Sept. 1993.

[2] M. Anwar, Z. Zhao, and P. W. L. Fong. An access control model for Facebook-style social network systems. Technical Report 2010-959-08, Department of Computer Science, University of Calgary, July 2010. Submitted for review.

[3] C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*. Elsevier, 2007.

[4] G. Bruns and M. Huth. Access control via Belnap logic: Intuitive, expressive, and analyzable policy composition. *ACM Trans. Inf. Syst. Secur.*, 14(1):9, 2011.

[5] H. DeYoung. A logic for reasoning about time-dependent access control policies. Senior Research Thesis CMU-CS-08-131, School of Computer Science, Carnegie Mellon University, 20 May 2008.

[6] Facebook, 2011. `http://www.facebook.com/about/privacy/your-info-on-fb#controlprofile`.

[7] P. W. L. Fong. Preventing Sybil attacks by privilege attenuation: A design principle for social network systems. In *Proc. of the 2011 IEEE Symposium on Security and Privacy (S&P'11)*, pages 263–278, Oakland, CA, USA, May 2011.

[8] P. W. L. Fong. Relationship-based access control: protection model and policy language. In *CODASPY*, pages 191–202, 2011.

[9] P. W. L. Fong, M. Anwar, and Z. Zhao. A privacy preservation model for Facebook-style social network systems. In *Proc. of the 14th European Symp. on Research In Comp. Sec. (ESORICS'09)*, volume 5789 of *LNCS*, pages 303–320, Saint Malo, France, Sept. 2009.

[10] P. W. L. Fong and I. Siahaan. Relationship-based access control policies and their policy languages. In *SACMAT*, pages 51–60, 2011.

[11] D. Garg and M. Abadi. A modal deconstruction of access control logic. In *FOSSACS'2008*, volume 4962 of *LNCS*, pages 216–230, 2008.

[12] C. E. Gates. Access control requirements for Web 2.0 security and privacy. In *Proc. of IEEE Web 2.0 Privacy and Security Workshop (W2SP'07)*, Oakland, California, May 2007.