Cassandra: Flexible Trust Management, Applied to Electronic Health Records

Moritz Y. Becker Peter Sewell
Computer Laboratory, University of Cambridge
JJ Thomson Avenue, Cambridge, United Kingdom
$\{$*moritz.becker, peter.sewell*$\}$*@cl.cam.ac.uk*

Abstract

We study the specification of access control policy in large-scale distributed systems. We present Cassandra, *a language and system for expressing policy, and the results of a substantial case study, a security policy for a national Electronic Health Record system, based on the requirements for the ongoing UK National Health Service procurement exercise.*

Cassandra *policies are expressed in a language based on Datalog with constraints. The expressiveness of the language (and its computational complexity) can be tuned by choosing an appropriate constraint domain.* Cassandra *is role-based; it supports credential-based access control (e.g. between administrative domains); and rules can refer to remote policies (for automatic credential retrieval and trust negotiation). Moreover, the policy language is small, and it has a formal semantics for query evaluation and for the access control engine.*

For the case study we choose a constraint domain $\mathcal{C}_0$ that is sufficiently expressive to encode many policy idioms. The case study turns out to require many subtle variants of these; it is important to express this variety smoothly, rather than add them as ad hoc features. By ensuring only a constraint compact *fragment of $\mathcal{C}_0$ is used, we guarantee a finite and computable fixed-point model. We use a top-down evaluation algorithm, for efficiency and to guarantee termination.*

The case study (with some 310 rules and 58 roles) demonstrates that this language is expressive enough for a real-world application; preliminary results suggest that the performance should be acceptable.

1. Introduction

In this paper we study the specification and enforcement of security policy in large-scale distributed systems. Previous work on trust management and role-based access control systems has argued that it is desirable to:

- factor out the policy from the application code, so that it can be easily understood, and changed over time;
- express policy not in terms of individuals, but via the indirection of roles;
- express policy in a language with a formally defined semantics, again to ease precise understanding, and also to support static analysis, to verify sanity properties;
- support distributed access control, basing authorisation on digital credentials, with policies that express automatic credential retrieval over the network and strategies to establish mutual trust between strangers; and
- be scalable – to large numbers of sites and entities, but also to cover different administrative domains with independent policies or local adaptations of a default policy.

There is a tension in the design of policy languages, however: they should be expressive (so intended policies can be written naturally), small and elegant, without ad hoc features (so policies can be easily understood), and also efficiently computable in practical examples.

To address this, we have designed a trust management system, Cassandra, in which the expressiveness of the language can be tuned by selecting an appropriate *constraint domain* – policies are expressed in an extension of Datalog$_{\mathcal{C}}$, or Datalog with constraints. We ground this research by working out a substantial real-world example, a security policy for a national Electronic Health Record (EHR) system, based on the requirements for the ongoing UK National Health Service procurement exercise. This is, to the best of our knowledge, among the most complex policy examples discussed in the literature. It has some 310 rules and 58 roles and, as we shall see, demands the full expressiveness of Cassandra.

For the case study we choose a constraint domain $\mathcal{C}_0$ that is sufficiently expressive to encode many policy idioms such as role hierarchy, separation of duties, role appointment, cardinality constraints, role validity periods, and distributed trust negotiation. Interestingly, the case study turns out to require many subtle variants of these idioms that cannot be expressed in other languages; it is important that we can express this variety smoothly, rather than add each one as an ad hoc feature. By using static groundness analysis we restrict policies so that only a *constraint compact* fragment of $\mathcal{C}_0$ is required, guaranteeing a finite and computable fixed-point model. We use a memoizing top-down evaluation algorithm, for efficiency and to guarantee termination.

Cassandra is role-based with parameterised roles and actions (supporting concise policies); it is declarative (making policies as clear as possible); it can express powerful role revocation policies including cascading revocation; it supports credential-based access control decisions between administrative domains; and rules can refer to remote policies (providing automatic credential retrieval and trust negotiation). Moreover, the policy language is small, and the system has a formal semantics for both query evaluation and for the access control engine.

Existing trust management systems possess subsets of these features; it is however the combination of all these features, together with Cassandra's tunable expressiveness, that makes it unique, and powerful enough for us to express the policies of the case study.

In §2 we discuss the background for the case study, and outline an example scenario of the use of the EHR system. In §3 we discuss how policies are specified in Cassandra, including brief outlines of the semantics and the evaluation algorithm. For lack of space we omit full technical definitions and theoretical results. Cassandra needs not just a language for expressing policies, but an API and operational semantics for the access control engine; this is given in §4. We demonstrate how some 'classic' policy idioms can be expressed in §5: particular forms of role hierarchy, separation of duties, and delegation. In §6 we return to the case study, outlining its main features, paying careful attention to what idioms are used and what expressiveness it demands of the policy language. The full policy is available on-line [5]. In §7 we describe our prototype implementation (though at the time of writing it does not cover all aspects); preliminary experiments suggest that performance should be reasonable for such a policy. Finally we discuss related work and conclude.

2. Background: Electronic Health Records

Electronic Health Record (EHR) schemes are now being developed in Europe, Canada and Australia to provide "cradle-to-grave" summaries of patients' records, linking clinical information across the entire health system [16]. In England, the National Health Service (NHS) has been planning since 1998 to develop a nationwide Integrated Care Records Service (ICRS) providing health care providers and patients with 24 hour on-line access to EHRs on a central data-spine [36, 19]. The deployment of what is now one of the largest, most complex and riskiest IT projects in the public sector world-wide is scheduled for between 2005 and 2010. While the potential benefits of such a system are obviously huge, the government's plans have also been subject of a fiercely controversial public debate (see e.g. [41, 15, 13, 39, 14, 17, 22]).

A main issue of concern is the confidentiality of patient health information. At stake is not just the privacy of sensitive personal information but the success of the entire project. Patients will refuse to share their data if they do not trust the system or do not have sufficient control over the use of their data. It is equally important to gain clinician buy-in which will fail if the system is cumbersome to use, if the access restrictions are too strict or the response times too high.

Any EHR access control policy will be a compromise between conflicting interests. The general framework for any such policy has to comply with relevant legislation and regulations such as the Data Protection Act, Mental Health Act, Human Fertilisation and Embryology Act, the Abortion Regulations and the Venereal Diseases Regulations. Every health organisation is now required to have a privacy and data protection officer, also called "Caldicott Guardian" (named after the recommendation of the committee chaired by Dame Caldicott in 1997 [35]), who is responsible for overseeing the organisation's security policy and investigating breaches of confidentiality. Anything beyond this is still highly contentious. As an example, the Caldicott committee recommended that access to patient-identifiable information should be on a "strict need-to-know basis". In contrast, common medical code of ethics and professional practice goes further and requires the patient's consent for accessing personal information [44]. Anderson [3] stresses the same point in his security policy commissioned by the British Medical Association, and further demands that patients should automatically receive notifications when their data are accessed. It now seems to be current consensus that patient consent should be the basis for access decisions although it is not yet clear when explicit consent has to be sought and when implicit consent can be assumed.

It should be sufficiently clear from this that the access control policy of the national EHR system will undergo frequent changes as the public debate evolves. Furthermore, health organisations will likely have customised policies, compatible with but different from the national one. It is therefore necessary to be able to specify the policy independently of the implementation instead of having it hard-coded into the access control engine.

2.1. A scenario

The national data spine is expected to hold records for up to 100 million patients (including deceased patients and patients who have moved abroad). Given the large scale of the data spine, we propose a three-level architecture (as recommended in [37]) instead of one single physical database.

At the top level we have the national *Master Patient Index* (MPI), a central service that only stores bindings linking patient identifiers with the *EHR service* (the second level), that is the location of their EHR. The local NHS health organisations (e.g. hospitals, doctors' practices) constitute the third level. The records kept on this level are called Electronic Patient Records (EPR). The shared central EHR items are typically summaries of local EPR items. *Registration Authorities* (RAs) issue credentials to NHS clinical and administrative staff and health organisations for authentication purposes.

The following scenario illustrates some of the more challenging requirements of security policies for our national EHR architecture. In the discussion of the case study in §6 we discuss how these requirements are met by our example policy (we use the role and action names from the policy in the text below).

Anson Arkwright goes to see Dr Zoe Zimmer, his family's General Practitioner (GP), for an HIV test. Dr Zimmer records the visit in a local EPR item by performing an `Add-record-item` action (with suitable parameters) but does not send information to the EHR data spine on Anson's request. Some time later, Bob Arkwright, Anson's father, visits Dr Zimmer because of heart problems. During the visit he also tells her that he believes his son Anson may be a hypochondriac. Dr Zimmer adds a record item to Bob's EPR about his heart condition and an item in Anson's EPR about his father's comments. The latter item is marked as containing third party information about his father, so as long as his father (or the Caldicott Guardian on his behalf) does not activate a matching `Third-party-consent` role, Anson will not be able to read it. (Note that we use roles not just to model job positions within an organisation but also to indicate state changes, e.g. giving third-party consent.)

Dr Zimmer also attempts to submit a summary of Bob's new EPR item to his shared EHR. She does not need to query the MPI because she already knows the location of his EHR. She first activates her `Clinician` role on this EHR service by submitting an RA-issued `NHS-clinician-cred` role credential along with the activation request. Subsequently, her `Add-EHR-record-item` action succeeds because the EHR service can deduce she is Bob's `Treating-clinician` (Bob has explicitly consented to treatment years ago and has not withdrawn his `Consent-to-treatment` role). Dr Zimmer also decides to refer Bob to a local hospital's cardiologist, Dr Hannah Hassan. As Bob's treating clinician on the EHR service, Dr Zimmer can enter a `Referrer` role enabling Dr Hassan to also become a treating clinician. Bob's consent is not needed but he has the power to deactivate the referral.

At the hospital, a `Receptionist` registers Bob as a patient by activating a `Register-patient` role. After his outpatient visit with Dr Hassan, the receptionist registers him with a surgical team in the same hospital for a heart bypass operation. For this purpose, the receptionist activates appropriate `Register-team-episode` and `Register-ward-episode` roles, thereby establishing a legitimate treating clinician relationship between Bob and the surgical team and the ward nurses. During surgery, abnormal liver values are found. Years ago, Bob activated an `Access-denied-by-patient` role to conceal the contents of all items in his EHR concerning an alcohol-related liver problem from everybody except clinicians treating him as GP. The head of the surgical team, Dr Lily Littlewood, decides to "break the seal" to view Bob's restricted EHR item by performing the action `Force-read-EHR-item`. This can be done by any clinician with a legitimate relationship but will be marked in the audit to be investigated by the hospital's Caldicott Guardian.

Unfortunately, the team encounters further complications during the operation and Bob needs to be kept in artificial coma. Dr Zimmer agrees to appoint Bob's wife, Carol, to be his agent by activating a `Register-agent` role on the EHR service. Carol then requests to activate the `Agent` role on the hospital's service. This succeeds after some trust negotiation between the hospital and the EHR service: the hospital responds to Carol's request by requesting an agent registration credential from Bob's EHR service; the EHR service replies by requesting a health organisation credential; the hospital agrees by sending an RA-issued health organisation credential to the EHR service; and finally the EHR service sends the originally requested `Register-agent` credential to the hospital certifying that Carol is indeed Bob's agent.

When Bob is woken and released, he attempts to revoke the agent registration for his wife but fails because it was Dr Zimmer who registered Carol. On Bob's request, Dr Zimmer deactivates her `Register-agent` role for Carol. If Carol is active with an `Agent` role on the EHR service at that moment, cascading revocation causes that role to be deactivated immediately as well.

3. Policy Specification in Cassandra

Access control in large-scale heterogeneous distributed computing environments is fundamentally different from access control in a single administrative domain. In the latter case, all users are known, so authorisation can be based on identity authentication. In the former case, entities (from single users to entire organisations) wish to share their resources with previously unknown entities. In the trust management approach [9], credentials asserting attributes about their holders are used as the basis of access control decisions: for example, a clinician may be granted access to an EHR service on the basis of a clinician employment credential issued by a registration authority. As access control policies in such environments can be complex, they are best described in a high-level policy specification language. The Cassandra language is summarised in Figure 1; this section gives an informal introduction to the language, the query semantics,

Predicate names $p ::= \mathsf{canActivate} \mid \mathsf{hasActivated} \mid \mathsf{permits} \mid \mathsf{canDeactivate} \mid$
$\mathsf{isDeactivated} \mid \mathsf{canReqCred}$, and user-defined predicate names
Policy rule: $E_{loc}@E_{iss}.p_0(\vec{e}_0) \leftarrow loc_1@iss_1.p_1(\vec{e}_1),\ \dots,\ loc_n@iss_n.p_n(\vec{e}_n),\ c$
Credential (rule): $E_{loc}@E_{iss}.p_0(\vec{e}_0) \leftarrow c$
Aggregation rule: $E_{loc}@E_{loc}.p(\mathsf{agg\text{-}op}\langle x\rangle, \vec{y}) \leftarrow E_{loc}@iss.q(\vec{x}), c$
where agg-op is group or count

$\mathcal{C}_{eq}$ expressions: $e ::= x \mid E$
$\mathcal{C}_{eq}$ constraints: $c ::= \mathsf{true} \mid \mathsf{false} \mid e = e' \mid c \wedge c' \mid c \vee c'$.

$\mathcal{C}_0$ expressions $e ::= x \mid E \mid N \mid C \mid () \mid (e_1, .., e_n) \mid \pi_i^n(e) \mid R(e_1, .., e_n) \mid A(e_1, .., e_n) \mid$
$f(e_1, .., e_n) \mid \emptyset \mid \Omega \mid \{e_1, .., e_n\} \mid e - e' \mid e \cap e' \mid e \cup e'$
$\mathcal{C}_0$ constraints $c ::= \mathsf{true} \mid \mathsf{false} \mid e = e' \mid e \neq e' \mid e < e' \mid e \subseteq e' \mid c \wedge c' \mid c \vee c'$
and derivable constraints: $c ::= \dots \mid e \in e' \mid e \notin e' \mid e \in [e_1, e_2] \mid [e_1, e_2] \subseteq [e_1', e_2']$
$\mathcal{C}_0$ types $\tau ::= entity \mid int \mid const \mid unit \mid \tau_1 \times \dots \times \tau_n \mid role(\tau) \mid action(\tau) \mid set(\tau)$

Access-control operations:
$\mathsf{doAction}(A(\vec{e})),\ \mathsf{activate}(R(\vec{e})),\ \mathsf{deactivate}(E_v, R(\vec{e})),\ \mathsf{reqCred}(E_s@E_{iss}.p(\vec{x}) \leftarrow c)$

Figure 1. Collected Syntax

and the evaluation algorithm (the language has a formal definition but for lack of space we do not give it, or the supporting theorems, in this paper). Section 4 describes the semantics of the access control operations, defining it precisely in terms of the query semantics.

Roles and actions The Cassandra policy language is role-based to make access control administration simpler and more scalable. Our *roles* and *actions* (generalised privileges) are parameterised for higher expressiveness [24, 34]: e.g. the role `Clinician`(*org*, *spcty*) has parameters for the health organisation and the specialty of the clinician; the action `Read-record-item`(*pat*, *id*) has parameters specifying the item identifier of a patient *pat*'s health record. Note that Cassandra's notion of role is more general than e.g. in the RBAC96 model [42], where a role is typically a job position within an enterprise's hierarchy; for example, entering a parameterised `Access-denied-by-patient` role may indicate that a record item is concealed by the patient, depending on the surrounding policy.

Predicates In role-based access control (RBAC), relations are used to define role membership, session activations and role permissions. In Cassandra, these and other access control relations are specified implicitly by rules defining six special predicates (their precise operational semantics is formalised in §4):

1. $\mathsf{canActivate}(e, r)$ expresses the fact that the entity e can activate role r (i.e., e is a member of r).
2. $\mathsf{hasActivated}(e, r)$ indicates that the entity e has activated role r. The distinction between the predicates canActivate and hasActivated corresponds to the distinction between role membership and session activation in traditional RBAC.
3. $\mathsf{permits}(e, a)$ says that the entity e is permitted to perform action a. This differs from the standard notion of role-permission assignment in two ways. Firstly, the parameter e allows constraints to refer directly to the subject of the activation. Secondly, permits has no parameter for a role associated with the action, thus allowing more flexible permission specifications, e.g. a permission that is conditioned on the activation (or perhaps merely membership) of more than one single role.
4. Cassandra rules can also specify role deactivation: if $\mathsf{canDeactivate}(e_1, e_2, r)$ holds, the entity e_1 has the power to deactivate e_2's role activation r.
5. The deactivation of one role can trigger deactivations of other (local) roles. This form of cascading revocation is especially important in the context of delegated or appointed roles. The deactivation of a role can make predicates of the form $\mathsf{isDeactivated}(e, r)$ become true, indicating the triggered deactivation of the entity e's role r.
6. Finally, access control across different administrative domains is based on attribute-asserting credentials. In Cassandra, these attributes are simply constrained Cassandra predicates. To protect the information contained in predicates, we use $\mathsf{canReqCred}(e_1, e_2.p(\vec{e}))$ to say that the entity e_1 is allowed to request credentials issued by the entity e_2 and asserting the predicate $p(\vec{e})$.

Apart from these six predicates with a special access control meaning, policy writers can introduce further user-defined auxiliary predicates.

Rules Many existing policy languages are based on Datalog [1] (Horn clauses without function symbols) because it is a rule-based, declarative language and widely understood. However, standard Datalog is not very expressive, so many systems extend the language with ad hoc features. We take a different approach and base our language on recursive Datalog$_\mathcal{C}$ in which the expressiveness – and conversely, the computational complexity – is parameterised on the chosen *constraint domain* $\mathcal{C}$ [27]. The language can thus be adapted to a wide range of applications without changing its base semantics.

Datalog$_\mathcal{C}$ extends Datalog rules with a constraint from some fixed constraint domain, which may contain variables occuring in the head predicate or the body predicates of the rule (but may not contain any predicate names). A *constraint domain* $\mathcal{C}$ is a language of first order formulae containing at least true, false and the identity predicate "=" between $\mathcal{C}$-expressions (variables, entities and possibly other constructs, including function symbols). Furthermore, it must be closed under variable renaming, conjunction ($\wedge$) and disjunction ($\vee$).

The policy rule language in Cassandra extends Datalog$_\mathcal{C}$'s predicates for the purpose of credential-based trust management. A credential can be seen as a constrained predicate, vouched for by an *issuing* entity and stored at a *location* entity. Correspondingly, Cassandra predicates are therefore prefixed with a location and an issuer. A *policy rule* is therefore of the form

$$E_{loc}@E_{iss}.p_0(\vec{e}_0) \;\leftarrow\; loc_1@iss_1.p_1(\vec{e}_1),\; \ldots\;,\; loc_n@iss_n.p_n(\vec{e}_n),\; c$$

where the p_i are predicate names and the $\vec{e}_i$ are (possibly empty) expression tuples (that may contain variables) matching the parameter types of the predicate. c is a constraint from some fixed constraint domain, and may contain variables occurring in the head predicate or the body predicates of the rule (but may not contain any predicate names). The *location* and the *issuer* of the rule, E_{loc} and E_{iss}, are entities, and the loc_i and iss_i are entities or entity typed variables.

A policy rule of the simple form

$$E_{loc}@E_{iss}.p_0(\vec{e}_0) \leftarrow c$$

is called a *credential rule* or just a *credential*. If it is sent over the network, it can be thought of as a certificate asserting $p_0(\vec{e}_0)$, signed and issued by E_{iss}, and belonging to and stored at E_{loc}. The location and the issuer of a rule are usually identical; only in the case of a credential rule can they be different, as E_{loc} may hold a "foreign" credential signed by a different entity E_{iss}. A Cassandra *policy* of an entity E_{loc} is a finite set of rules (including credentials) with location E_{loc}.

In standard Datalog$_\mathcal{C}$, a predicate can be deduced if there is a rule with a matching head, such that the rule's body predicates can be deduced while satisfying the constraint of the rule. In Cassandra, a body predicate $B@C.p(\vec{e})$ can refer to a remote location, if B is not equal to the local entity, say A. To deduce the predicate, A will contact B over the network and delegate authority to B to deduce the predicate. Such a remote query amounts to a credential request: B will first try to deduce $B@B.\mathsf{canReqCred}(A, C.p(\vec{e}))$ before attempting to deduce the requested predicate.

To illustrate this, consider the following example where likes is some user-defined predicate. Suppose A's policy contains the rules

$$\begin{aligned} R_1 &\equiv A@A.\mathsf{likes}(A,x) \leftarrow x@y.\mathsf{likes}(y,x), x \neq y \\ R_2 &\equiv A@B.\mathsf{likes}(B,A) \leftarrow \mathsf{true} \end{aligned}$$

(so R_2 is a foreign credential from B), and C's policy contains

$$R_3 \equiv C@D.\mathsf{likes}(D,C) \leftarrow \mathsf{true}$$

(so R_3 is a foreign credential from D). Intuitively, R_1 means: A likes an entity x if x's policy proves that some other entity y says that he likes x. More formally, we can deduce $A@A.\mathsf{likes}(A,x)$ on A if we can deduce $x@y.\mathsf{likes}(y,x)$ on the service x provided that x is not equal to y. If, in the context of evaluation, x turns out to be A, then the body becomes

$$A@y.\mathsf{likes}(y,A) \leftarrow A \neq y,$$

to be deduced locally on A, by finding a matching credential (foreign since $A \neq y$). R_2 is such a credential, thus A has proved $A@A.\mathsf{likes}(A,A)$. Otherwise, if x turns out to be different from A, say $x = C$, A automatically requests a credential from C of the form

$$C@y.\mathsf{likes}(y,C) \leftarrow C \neq y.$$

At this point, Cassandra's trust negotiation mechanism comes into play: C first tries to prove

$$C@C.\mathsf{canReqCred}(A, y.\mathsf{likes}(y,C)) \leftarrow C \neq y$$

on its own policy to see whether A is allowed to request such a credential. The result of this deduction is either false (A cannot request such a credential) or some constraint on the variable y, say $y \neq E$ (A is allowed to get such a credential provided $y \neq E$). In the latter case, C will then try to prove

$$C@y.\mathsf{likes}(y,C) \leftarrow C \neq y \wedge y \neq E.$$

This is satisfied by R_3, so C will reply to A with the answer $C@D.\mathsf{likes}(D,C)$, upon which A can finally prove $A@A.\mathsf{likes}(A,C)$.

In the remainder of the paper, we will omit the rule location E_{loc} if it is clear from the context. We will also omit location and issuer prefixes loc_i and iss_i from body predicates if they are equal to the rule location.

Constraint domains All $\mathsf{Cassandra}$ entities on the network must agree on a common constraint domain $\mathcal{C}$. For example, the least expressive constraint domain $\mathcal{C}_{eq}$ is the one where the only atomic constraints are equalities between variables and entities. Choosing this trivial constraint domain reduces the expressiveness of the language to standard Datalog or Horn clauses without function symbols.

The design of the much richer constraint domain $\mathcal{C}_0$ was guided by the EHR case study. In $\mathcal{C}_0$, atomic expressions can be variables, entities, integers, constants of various types, the empty set $\emptyset$ and the universal set Ω. Compound expressions are built from the atomic ones recursively: tuples $(e_1, .., e_n)$, tuple projections $\pi_i^n(e)$, roles $R(e_1, .., e_n)$, actions $A(e_1, .., e_n)$, function applications $f(e_1, .., e_n)$, and the set expressions $\{e_1, .., e_n\}$, $e - e'$, $e \cap e'$ and $e \cup e'$. $\mathcal{C}_0$-constraints include equalities $e = e'$, inequalities $e \neq e'$, integer orders $e < e'$, set containments $e \subseteq e'$ and, of course, finite conjunctions and disjunctions of these. The function symbols are interpreted by a fixed set of side-effect-free functions that may return environment-dependent data. For example, for our case study we have functions to access data fields of health record items such as $\texttt{Get-EHR-item-author}(pat, id)$, and a function $\texttt{Current-time}()$ that returns the current time.

A type system for $\mathcal{C}_0$ where the types τ are of the form int, $entity$, $const$, $unit$, $\tau_1 \times ... \times \tau_n$, $role(\tau)$, $action(\tau)$ and $set(\tau)$ ensures that expressions and constraints are well-formed. Moreover, without typing one could trivially encode undecidable properties, with policies that perform arithmetic on unary-encoded integers.

We also allow constraints that can be defined in terms of the existing ones, such as non-membership $e \notin e'$, integer ranges $e \in [e_1, e_2]$ and range containments $[e_1, e_2] \subseteq [e_1', e_2']$.

Aggregation rules Policies often require negative conditions in the premise of a rule, e.g. that an entity has not activated a particular role, or that no entity has activated the role. The former example can be expressed by introducing negated body predicates; this method however cannot express the latter example, which implicitly involves universal quantification. Instead, we introduce rules with *aggregation operators* [40] with which both examples can be expressed. Aggregation is also useful for grouping and cardinality constraints, e.g. constraints on the set of all role activations of a particular entity, or on the number of all such activations. For aggregation, the constraint domain $\mathcal{C}$ is required to contain equalities over set and integer constants and variables. An *aggregation rule* is of the form

$$E_{loc}@E_{loc}.p(\mathsf{agg\text{-}op}\langle x\rangle, \vec{y}) \leftarrow E_{loc}@iss.q(\vec{x}), c$$

where the aggregation operator $\mathsf{agg\text{-}op}$ is either group or count. The predicate $E_{loc}@iss.q(\vec{x})$ must be one that can be satisfied with only finitely many different parameters, and $\vec{x}$ must contain x. The first argument of p is instantiated to the (finite) set of all different values of x that satisfy the body. If $\mathsf{agg\text{-}op} = \mathsf{count}$, it is instantiated to the cardinality of that set. For example,

$$\begin{aligned}&\mathsf{group\text{-}active\text{-}doctors}(\mathsf{group}\langle x\rangle, spcty) \leftarrow \\ &\quad \mathsf{hasActivated}(x, \texttt{Doctor}(spcty))\end{aligned}$$

finds the set of all active doctors with specialty $spcty$. Similarly,

$$\begin{aligned}&\mathsf{count\text{-}specialties}(\mathsf{count}\langle spcty\rangle, x) \leftarrow \\ &\quad \mathsf{hasActivated}(x, \texttt{Doctor}(spcty))\end{aligned}$$

counts the number of different specialties in which x is active as a doctor.

The restriction that the body predicate of an aggregation rule must be local (its location must be equal to E_{loc}) is necessary because aggregation requires complete knowledge of the predicate. Answers from remote entities are always sound but may be incomplete as they are subject to $\mathsf{canReqCred}$ restrictions. The restriction that the body predicate can only be satisfied with finitely many different parameters and that x is mentioned in it ensures that aggregation is finite. In terms of negation, this corresponds to semi-positive policies, thus avoiding negation-related issues such as undecidability and semantic ambiguity.

The separation-of-duties example in §5 shows how aggregation can be used to express universally quantified negation. Note that the kind of negation we can express via aggregation can only occur in the body of a rule and never in the head. In particular, we cannot express explicit prohibitions (as in e.g. Halpern and Weissman's logic [25], Ponder [18] or in FAF [28]); rather, we assume that everything is prohibited unless it is explicitly permitted (closed world assumption).

3.1. Semantics and Evaluation

$\mathsf{Cassandra}$ provides direct support for access control decisions: the access control engine issues policy queries and bases decisions on the query answers, as specified in §4. A typical query issued by the access control engine (queries are never made directly by users) would be $\mathsf{hasActivated}(x, \texttt{Doctor}(spcty)) \leftarrow spcty \neq \texttt{GP}$. The answer of a query is a $\mathcal{C}$-constraint describing all (potentially infinitely many) variable instantiations that satisfy the query according to the policy. In this example, the answer would be a constraint specifying all non-GP doctors and their associated specialties. For a completely ground query (all variables forced to a unique value), the answer constraint would be either true or false. In general, a query has the same form as a credential rule, i.e. $E_{loc}@E_{iss}.p_0(\vec{e}_0) \leftarrow c$. The answer constraint to such a query may depend on policies of several different entities, as the rules can refer to remote policies.

A policy evaluation algorithm takes a query as input and returns the answer constraint. We require such an algorithm to terminate for all queries and policies, to be efficient, and of course to be sound and complete with respect to the language semantics.

We define the language semantics by an immediate consequence operator $T_{\mathcal{C},\mathcal{P}}$ where $\mathcal{C}$ is the global constraint domain and $\mathcal{P}$ the (naturally disjoint) union of the policy rules of all entities. $T_{\mathcal{C},\mathcal{P}}$ is an extension of the immediate consequence operator for standard Datalog with which the deductive closure, the set of all deducible ground facts, of a Datalog program can be computed [1]. In Cassandra, credential rules correspond to Datalog facts, hence the deductive closure contains the set of all deducible credentials rules. $T_{\mathcal{C},\mathcal{P}}$ is thus a function between sets $\mathcal{I}$ of credential rules: $T_{\mathcal{C},\mathcal{P}}(\mathcal{I})$ is the set of all credential rules that can be deduced from $\mathcal{I}$ in one single step. We omit the formal definition [6] due to lack of space but point out that $T_{\mathcal{C},\mathcal{P}}$ assumes of $\mathcal{C}$ only the existence of certain operations, satisfiability test, subsumption (implication) test and an existential quantifier elimination procedure, and is otherwise independent of $\mathcal{C}$.

The deductive closure or *fixed-point model* of $\mathcal{P}$ is the unique least fixed-point of $T_{\mathcal{C},\mathcal{P}}(\mathcal{I})$, which can be computed in the usual iterative bottom-up fashion if it is finite. However, it is not obvious whether the fixed-point model is finite. Indeed, it is easy to show that many simple looking constraint domains, such as one that supports untyped tuple constructors, or one with negative gap-order constraints ($x - c < y$, where c is a positive integer constant), enable the construction of undecidable policies.

Constraint compactness [45] is a sufficient condition on constraint domains to guarantee a finite and hence computable fixed-point model for any finite global policy set $\mathcal{P}$. A constraint domain $\mathcal{C}$ is said to be constraint compact if any infinite set of $\mathcal{C}$-constraints in which only finitely many variables and constants occur has a finite subset subsuming the entire set, that is, for every constraint c in the infinite set there is a constraint c' in the finite set such that c implies c'.

Unfortunately, constraint compactness severely restricts the expressiveness of the constraint language and is also often hard to prove. We use *static groundness analysis* [4] to restrict policies in such a way that variables occuring in specific constructs will always have been grounded (so a unique value can be deduced for each) by the time existential quantifier elimination is performed on them, given the query patterns from §4, so these constructs can be ignored. In the case of $\mathcal{C}_0$ we require that variables occuring in function arguments or in set expressions are ground at quantifier elimination time. This reduces $\mathcal{C}_0$ to a constraint domain that we have proved to be constraint compact.

Static groundness analysis is also used to ensure that the location prefix of body predicates becomes ground by the time we evaluate it: otherwise the evaluator would have to query many different entities (all, in the worst case), which is clearly unpractical.

In the context of constraint logic databases, queries are usually evaluated against the complete fixed-point model, pre-computed iteratively in a bottom-up fashion. This would not be an acceptable evaluation strategy for Cassandra: firstly, the constraints may contain (side-effect free) function calls that depend on the environment, for example for getting the current time, and therefore cannot be pre-computed; secondly, the fact that rule bodies can refer to remote predicates would require a distributed form of bottom-up evaluation which would be highly unpractical; and thirdly, the model would have to be re-computed after every activation or deactivation of roles as role activation and deactivation modify policies.

Instead, Cassandra uses a modified version of Toman's top-down CLP evaluation algorithm [45] based on SLG resolution, a memoization strategy [11]. The usual top-down algorithms based on SLD resolution as used in Prolog systems are unsuitable because they do not guarantee termination even in cases where the fixed-point model is finite. The SLG algorithm, on the other hand, preserves the termination properties of the bottom-up evaluation method. We have extended the algorithm in [45] to deal with goals referring to remote entities [6]. The evaluation algorithm is sound and complete with respect to the fixed-point semantics and is far more efficient than the bottom-up method, especially when queries are fully instantiated as is often the case in Cassandra.

4. Access Control Semantics

The only way to query the policy is through the interface of the access control engine. We define an operational semantics for the four operations that an entity E_r can request from an entity E_s acting as a Cassandra service: performing an action, activating a role, deactivating a role, and requesting a credential. As these four operations may have side-effects on the policies, their semantics is most naturally specified by a labelled transition system with transitions between global policy sets $\mathcal{P}$. The labels of the transition system are parameterised by the requester E_r, the service E_s, the requested operation, and a set of credentials C_r belonging to the requester that are submitted to E_s along with the request. (In an implementation, these would be signed certificates.) The fact that these credentials now belong to E_s is expressed by renaming their location to E_s. We write

$$\begin{aligned}&\mathit{Submit}_{E_s}(C_r) = \\ &\quad \{E_s.E_{iss}@p \leftarrow c \mid E'.E_{iss}@p \leftarrow c \in C_r\}\end{aligned}$$

for the renamed set of credentials.

Performing an action E_r attempts to perform the (ground) action $A(\vec{e})$ on a service E_s.

$$\mathcal{P} \xrightarrow{E_r, E_s, \mathsf{doAction}(A(\vec{e})), C_r} \mathcal{P}$$

is a valid transition if $E_s@E_s.\mathsf{permits}(E_r, A(\vec{e}))$ is deducible from $\mathcal{P} \cup \mathit{Submit}_{E_s}(C_r)$. (Note that deduction may involve This transition has no effect on the state.

Role activation E_r attempts to activate the (ground) role $R(\vec{e})$ on a service E_s.

$$\mathcal{P} \xrightarrow{E_r, E_s, \mathsf{activate}(R(\vec{e})), C_r} \mathcal{P}'$$

(where $\mathcal{P}' = \mathcal{P} \uplus \{E_s@E_s.\mathsf{hasActivated}(E_r, R(\vec{e}))\}$) is a valid transition if the role has not already been activated and if $E_s@E_s.\mathsf{canActivate}(E_r, R(\vec{e}))$ is deducible from $\mathcal{P} \cup \mathit{Submit}_{E_s}(C_r)$. As a result of this transition, a new hasActivated credential rule is added to E_s's policy.

Role deactivation E_r requests to deactivate the "victim" E_v's role $R(\vec{e})$ on a service E_s. This transition rule also implements cascading deactivation.

$$\mathcal{P} \xrightarrow{E_r, E_s, \mathsf{deactivate}(E_v, R(\vec{e})), C_r} \mathcal{P} - \mathit{Victims}_s$$

is a valid transition if E_v has actually activated $R(\vec{e})$ and if $E_s@E_s.\mathsf{canDeactivate}(E_r, E_v, R(\vec{e}))$ is deducible from $\mathcal{P} \cup \mathit{Submit}_{E_s}(C_r)$. The set $\mathit{Victims}_s$ is the set of all hasActivated credential rules in E_s's policy for which a corresponding isDeactivated credential can be derived under the assumption that the predicate $E_s@E_s.\mathsf{isDeactivated}(E_v, R(\vec{e}))$ holds. As a result of the transition, all role activations in $\mathit{Victims}_s$ are removed from E_s's policy. Note that $\mathit{Victims}_s$ contains only role activations with location E_s; Cassandra does not support distributed cascading revocation across the network, as is proposed in Oasis [26, 48]. Such a mechanism would be very hard to implement on a wide scale as it would require a reliable event infrastructure and the management of a much bigger state recording which entities have to be notified about which deactivation events.

Requesting Credentials E_r requests the credential $E_s@E_{iss}.p(\vec{x}) \leftarrow c$ from a service E_s. If E_{iss} and E_s are identical, E_s computes an exact answer of the requested predicate p and sends a freshly signed certificate to E_r. Otherwise, E_s sends all foreign certificates matching the request to E_r. (Note that the credentials that E_r receives as a result have constraints that are in general more restricted than the requested constraint c.) More formally:

$$\mathcal{P} \xrightarrow{E_r, E_s, \mathsf{reqCred}(E_s@E_{iss}.p(\vec{x}) \leftarrow c), C_r} \mathcal{P} \cup \mathit{Creds}$$

is a valid transition provided the following: let c_0 be the satisfiable answer of the query

$$E_s@E_s.\mathsf{canReqCred}(E_r, E_{iss}.p(\vec{x}) \leftarrow c)$$

against $\mathcal{P} \cup \mathit{Submit}_{E_s}(C_r)$. If $E_s = E_{iss}$, Creds only contains the credential $E_r@E_s.p(\vec{x}) \leftarrow c'$ where c' is the satisfiable answer of the query $E_s@E_s.p(\vec{x}) \leftarrow c_0$ against $\mathcal{P} \cup \mathit{Submit}_{E_s}(C_r)$. Otherwise, $E_s \neq E_{iss}$, in which case Creds is the set of all foreign $p(\vec{x})$-credentials belonging to E_s but issued by E_{iss} whose constraints are at least as restrictive as c_0. (The credential locations must of course be renamed from E_s to E_r.)

5. Expressing Policy Idioms

Cassandra is powerful enough to express common RBAC policy idioms such as role hierarchy, separation of duties and role delegation, so there is no need to support them specifically with ad hoc language extensions. This approach not only keeps the language definition simpler and more uniform, it also enables the policy designer to express variants and combinations of these idioms that cannot be expressed in other policy languages. Naturally, these advantages come at a cost: rules may look more complicated than in languages with built-in application-tailored features. Here we give encodings of the three above mentioned idioms in their pure forms. The EHR case study exhibits variants of role appointment, as well as other idioms such as cardinality constraints, role validity periods and distributed trust negotiation.

Role Hierarchy A project leader is more senior than both a production engineer and a quality engineer. Both production engineer and quality engineer are more senior than the engineer role. This example is taken from [43]. The hierarchy graph can be directly represented by canActivate dependencies. We extend the example by using roles that have a "department" parameter:

$\mathsf{canActivate}(x, \mathtt{Prod\text{-}eng}(dep)) \leftarrow$
$\quad \mathsf{canActivate}(x, \mathtt{Proj\text{-}leader}(dep))$

$\mathsf{canActivate}(x, \mathtt{Qual\text{-}eng}(dep)) \leftarrow$
$\quad \mathsf{canActivate}(x, \mathtt{Proj\text{-}leader}(dep))$

$\mathsf{canActivate}(x, \mathtt{Eng}(dep)) \leftarrow$
$\quad \mathsf{canActivate}(x, \mathtt{Prod\text{-}eng}(dep))$

$\mathsf{canActivate}(x, \mathtt{Eng}(dep)) \leftarrow$
$\quad \mathsf{canActivate}(x, \mathtt{Qual\text{-}eng}(dep))$

Separation of Duties To encode separation-of-duties constraints it is necessary to be able to express negated conditions such as "x has *not* activated role $R(y)$", where x and y will have been instantiated by the time the condition is processed. We will write this condition in the body as

$\neg\mathsf{hasActivated}(x, R(y))$ as shorthand for the user-defined aggregation condition $\mathsf{existsActivation}_R(0, x, y)$, defined by a rule

$$\begin{aligned}&\mathsf{existsActivation}_R(\mathsf{count}\langle x'\rangle, x, y) \leftarrow\\ &\quad \mathsf{hasActivated}(x', R(y)), x' = x\end{aligned}$$

Clearly, $\mathsf{existsActivation}_R(0, x, y)$ holds if and only if x has not activated $R(y)$. As an example for dynamic n-wise parameter-centric separation of duties, suppose that nobody can work on two projects at the same time if they both belong to a set of n pairwise mutually conflicting projects. With a function `Conflict`() that returns this set of conflicting projects, this can be encoded as

$$\begin{aligned}&\mathsf{canActivate}(x, \mathtt{Projmem}(p)) \leftarrow\\ &\quad \neg\mathsf{hasActivated}(x, \mathtt{Projmem}(p')),\\ &\quad \{p, p'\} \subseteq \mathtt{Conflict}()\end{aligned}$$

Delegation and Appointment Delegation can be viewed as a special case of the more general appointment mechanism where the appointer is required to be a member of the appointed role [48]. Appointment can easily be encoded in Cassandra. It is also possible to encode different versions of delegation, such as transitive delegation where the delegatee is able to further delegate the role or finitely transitive delegation where the delegation chain is of a fixed finite length (cf. [49]). Here we only show the encoding of a simple version of appointment. A manager M can appoint A as an employee by activating an "appointer role" $\mathtt{AppointEmployee}(A)$. This then enables A to activate the "employee appointed by M" role $\mathtt{Employee}(M)$.

$$\begin{aligned}&\mathsf{canActivate}(mgr, \mathtt{AppointEmployee}(emp)) \leftarrow\\ &\quad \mathsf{hasActivated}(mgr, \mathtt{Manager}())\\ &\mathsf{canActivate}(emp, \mathtt{Employee}(appointer)) \leftarrow\\ &\quad \mathsf{hasActivated}(appointer, \mathtt{AppointEmployee}(emp))\end{aligned}$$

Furthermore, A's employee role is revoked automatically when $\mathtt{AppointEmployee}(A)$ is deactivated:

$$\begin{aligned}&\mathsf{isDeactivated}(emp, \mathtt{Employee}(appointer)) \leftarrow\\ &\quad \mathsf{isDeactivated}(appointer, \mathtt{AppointEmployee}(emp))\end{aligned}$$

We also have to specify who is allowed to revoke the appointment role. With grant-dependent revocation, only the appointer herself can revoke it:

$$\begin{aligned}&\mathsf{canDeactivate}(x, appointer, \mathtt{AppointEmployee}(emp)) \leftarrow\\ &\quad x = appointer\end{aligned}$$

Grant-independent revocation, on the other hand, allows every manager to revoke employee roles:

$$\begin{aligned}&\mathsf{canDeactivate}(x, appointer, \mathtt{AppointEmployee}(emp)) \leftarrow\\ &\quad \mathsf{hasActivated}(x, \mathtt{Manager}())\end{aligned}$$

In some cases, all roles appointed by M should be revoked whenever M is revoked from her role herself, in this case the manager role. We can encode a cascading chain of revocations as follows:

$$\begin{aligned}&\mathsf{isDeactivated}(mgr, \mathtt{AppointEmployee}(emp)) \leftarrow\\ &\quad \mathsf{isDeactivated}(supermgr, \mathtt{AppointManager}(mgr))\end{aligned}$$

6. Expressing the Case study in Cassandra

We have drafted a complete Cassandra policy for a national EHR system, based mainly on the Output Based ICRS Specification [38] (a 570-page document given to potential suppliers during the current procurement process), reports from NHS pilot projects of the Electronic Records Development and Implementation Programme (ERDIP) [37, 23], and various Department of Health documents [20, 21]. The policy comprises 310 rules, 58 roles and 10 actions. Of the 310 rules, there are 84 canActivate and canDeactivate rules each, 52 isDeactivated, 28 permits and 24 canReqCred rules. The remaining 38 are user-defined aggregate rules. The case study suggests that common policy idioms such as appointment hardly occur in their pure forms in practice; rather, it is necessary to be able to express different variants of them. Interestingly, one of the most commonly mentioned policy idioms in the literature, separation of duties, was not encountered in this case study. The rules can be roughly divided into the following categories:

Permissions assignment: many of the permits rules are straightforward parameterised role-action assignments, e.g. "patients can annotate their own record items". Others require more than one role-related prerequisite condition, e.g. "clinicians can force-read record items concealed by a patient if they have activated their clinician role and if they are member of a workgroup (clinical team or ward) currently treating the patient". The last condition is also an example of an auxiliary or derived role: the `Group-treating-clinician` role need not be activated when using the rule; it is sufficient that it *can* be activated. The permits rules concerning reading record items are typically also conditioned on consent and (absence of) access denial role activations (activated by other entities); such rules cannot be easily expressed in languages in which the subject parameters of the head and the body are the same, e.g. RT [33] or Oasis [26, 48].

Consent: access to health records is primarily based on explicit patient consent. Consent may be required for initial treatment, for referrals and for disclosure of third-party information. We implement consent as a form of appointment: by activating a consent role, a patient "appoints" a clinician to be e.g. a `Treating-clinician` with a legitimate relationship. To prevent frivolous users from unsolicitedly activating myriads of consent roles, the user must first have been requested to activate that role. These consent requests are again implemented as a form of appointment, but now the other way round: by activating a consent request role, the clinician enables the patient to activate a consent role. Consent is thus implemented as a two-stage appointment mechanism.

Registration is an administrative task that takes on many forms in our case study: e.g. MPI-administrators enter newly born patients into the MPI, receptionists register new patients, human resource managers employ clinicians and other staff, head nurses assign nurses to wards, and heads of clinical teams assign clinicians to their respective teams. It is easy to see that registration can again be implemented using variants of the appointment encoding given in §5. Variants include combinations with cardinality restrictions ("patients can register at most three distinct agents acting on their behalf") and uniqueness constraints ("a patient can only be registered if no one has already activated the registration role for that patient"). The two mentioned variants make use of Cassandra's aggregation operators.

Referral is implemented as a form of delegation. Our case study exhibits two kinds of patient referral. On the EHR service, no patient consent is required, and referral chains are of unbounded length. On the level of the local health organisation, we decided to implement a stricter alternative: a local treating clinician can refer the patient to an external clinician (who will then have restricted rights to read the local EPR record items) only with explicit patient consent, and delegation chain can only be of length one.

Access denial: this is a policy idiom motivated by the requirement that patients may conceal their data. By activating an access denial role, patients can fine-tune the rights to access their records. The corresponding permits rules need to check that no such access denial role has been activated; this requires universally quantified negation, expressed with the help of aggregation operators.

Deactivation: canDeactivate rules specify who can deactivate which roles. Although these rules are rather straightforward, it is important that deactivation can be specified flexibly. For example, revocation of patient agent registrations is asymmetric in the sense that patients can only revoke the agents they have appointed themselves (grant-dependent revocation), whereas Caldicott Guardians can revoke not only the agents they have appointed for a patient but also those appointed by the patient himself (variant of grant-independent revocation). Cascading deactivation, specified by isDeactivated rules, is used to automatically deactivate a role if some other role is deactivated. For example, the revocation of a patient's registration in the hospital triggers the deactivation of all roles that have something to do with that patient, including agent registrations, inpatient episode registrations, legitimate relationships with clinicians, access denial roles, and consent roles.

Credential management: credential-based trust negotiation and credential protection are governed by the interaction between canReqCred rules and rules with remote body predicates. The scenario in §2.1 gives an example of multi-phase automatic trust negotiation. canReqCred rules are also used for regulating direct credential requests from entities. For example, agent credentials can be requested by certified health organisations, and also by the agent himself. The location parameter of Cassandra predicates facilitates very flexible forms of automatic credential retrieval: unlike most other systems, credential locations are not restricted to the issuer or the credential subject. For example, a credential of the form

$$\begin{array}{l}\texttt{RA}.\mathsf{hasActivated}(\texttt{RA-adm},\\ \quad\texttt{NHS-health-org-cred}(\texttt{Org},\texttt{Start},\texttt{End}))\end{array}$$

may be found at the location Org which is neither the issuer (RA) nor the subject (RA-adm).

The users of the system – patients, clinicians and administrative staff – are modelled as entities whose policies consist only of the credentials they acquire over time. We assume the existence of an infrastructure for the identification of individuals. The NHS number identifies patients uniquely, but we do not specify the exact authentication mechanism (e.g. a password, a name certificate, or a smartcard-stored key). Further we assume that network communication is secured.

In the following we outline the policy for the EHR architecture proposed in §2 and illustrated in the scenario (§2.1) and give examples of policy rules that illustrate particular features of Cassandra or express a variant of an interesting policy idiom.

Master Patient Index Recall that the MPI is a central national service that, for each patient, stores the location of his EHR. This MPI patient/EHR-service binding can be queried by patients, health professionals and organisations and then used to retrieve the EHR items from the EHR service. This approach keeps the size of the central service manageable and can cope with high access rates since our MPI policy is very simple and the amount of transferred data small.

There will be an estimated number of 300 million GP appointments, 70 million inpatient episodes and out-patient hospital attendances, and about 30 million other health episodes and encounters per year [38]. For each of these events, the MPI will typically be consulted not more than two or three times, hence the expected number of accesses will be around one billion per year. The service therefore needs to be able to handle at most a few thousand accesses per second if we take peak times into account. The MPI policy defines only four roles and 17 rules.

A new patient/EHR-service binding can be entered by an MPI administrator, by activating the Register-patient(*pat*, *ehr-srv*) role, if no binding for that patient exists yet. This negative condition holds if the aggregation parameter of the user-defined aggregation predicate count-patient-regs is 0. Patient registration can be seen as a variant of appointment in which there appointees (patients) can only be appointed (registered) once.

$$\begin{aligned}
&\textsf{canActivate}(adm, \texttt{Register-patient}(pat, \textit{ehr-srv})) \leftarrow \\
&\quad \textsf{hasActivated}(adm, \texttt{MPI-admin}()), \\
&\quad \textsf{count-patient-regs}(0, pat) \\
&\textsf{count-patient-regs}(\textsf{count}\langle x\rangle, pat) \leftarrow \\
&\quad \textsf{hasActivated}(x, \texttt{Register-patient}(pat, \textit{ehr-srv}))
\end{aligned}$$

The MPI policy also contains rules for removing patient/EHR-service bindings and various rules for requesting credentials containing the binding. One examples for credential request is given below. The rule authorises health organisations to request a credential asserting the binding if they in turn can (and are willing to) provide an RA-approved NHS health organisation credential. This is an example for encoding a particular distributed trust negotiation strategy in Cassandra. Cassandra's evaluation algorithm ensures that trust negotiation strategies always terminate, even in the presence of mutually recursive rules.

$$\begin{aligned}
&\textsf{canReqCred}(org, \texttt{MPI}.\textsf{hasActivated}(x, \\
&\qquad \texttt{Register-patient}(pat, \textit{ehr-srv}))) \leftarrow \\
&\quad org@ra.\textsf{hasActivated}(x, \\
&\qquad \texttt{NHS-health-org-cred}(org, start, end)), \\
&\quad ra \in \texttt{NHS-registration-authorities}(), \\
&\quad \texttt{Current-time}() \in [start, end]
\end{aligned}$$

EHR services The EHR records are physically stored on EHR services that could be deployed below the Strategic Health Authority (SHA) level. There are 28 SHAs in England, each covering one to two million living patients. So if there are four EHR services per SHA, each will keep the complete health records of up to a million patients (including deceased or moved abroad). The EHR service policy defines 18 roles, five actions and 98 rules.

Each patient is associated with exactly one health record consisting of a set of items, indexed by some item ID. Only health professionals can create new items but patients can annotate existing items. The system function `Get-EHR-item-author`(*pat, id*) returns the author of the record item *id* of patient *pat*. Similar functions exist for getting the health organisation of the author, a set of subjects the item relates to, the time the item was created, and the set of third parties whose consent must be sought before the patient may view the item.

A clinician can generally only read (`Read-EHR-item`) a patient's record item if either she is the item's author or she is currently treating the patient. The rule for the latter case also specifies that she must be treating the patient in a specialty that allows her to read the subjects of the item. Furthermore, she is denied access if the patient has activated a matching `Access-denied-by-patient` role. The parameters of this role specify which items are affected, to whom access should be denied, and a validity date range for the access restriction. Recall the scenario from §2.1, where Bob conceals all record items concerning an alcohol-related liver problem from everybody except clinicians treating him as GP. This can be accomplished by activating the role $\texttt{Access-denied-by-patient}(what, whom, 0, \infty)$ where $what = (\texttt{Bob}, \Omega, \Omega, \Omega, \{\texttt{Liver}, \texttt{Drugs}\}, 0, \infty)$ and $whom = (\Omega, \Omega, \Omega - \{\texttt{GP}\})$. The rather complex user-defined aggregation rule for count-access-denied-by-pat is used in the permits rule to ensure nobody has activated a matching access denial role. This condition holds if the aggregation parameter is 0.

$$\begin{aligned}
&\textsf{permits}(cli, \texttt{Read-EHR-item}(pat, id)) \leftarrow \\
&\quad \textsf{hasActivated}(cli, \texttt{Clinician}(org, spcty)), \\
&\quad \textsf{canActivate}(cli, \texttt{Treating-clinician}(pat, org, spcty)), \\
&\quad \textsf{count-access-denied-by-pat}(0, (pat, id), (org, cli, spcty)), \\
&\quad \texttt{Get-EHR-item-subjects}(pat, id) \subseteq \\
&\qquad \texttt{Permitted-subjects}(spcty)
\end{aligned}$$

$$\begin{aligned}
&\textsf{count-access-denied-by-pat}(\textsf{count}\langle x\rangle, (pat, id), \\
&\qquad (org, reader, spcty)) \leftarrow \\
&\quad \textsf{hasActivated}(x, \texttt{Access-denied-by-patient}(\\
&\qquad what, whom, start, end)), \\
&\quad what = (pat, ids, orgs, authors, subjects, \\
&\qquad \textit{from-time}, \textit{to-time}) \land \\
&\quad whom = (orgs1, readers1, spctys1) \land \\
&\quad \texttt{Get-EHR-item-org}(pat, id) \in orgs \land \\
&\quad \texttt{Get-EHR-item-author}(pat, id) \in authors \land \\
&\quad \texttt{Get-EHR-item-subjects}(pat, id) \cap subjects \neq \emptyset \land \\
&\quad \texttt{Get-EHR-item-time}(pat, id) \in [\textit{from-time}, \textit{to-time}] \land \\
&\quad (id \in ids) \land (org \in orgs1) \land (reader \in readers1) \land \\
&\qquad (spcty \in spctys1) \land (\texttt{Current-time}() \in [start, end])
\end{aligned}$$

The `Clinician` role can be activated by submitting a currently valid clinician approval credential issued by a registration authority to the EHR service. Clinicians have a legitimate relationship with a patient if they can activate the `Treating-clinician` role (an example of an *auxiliary* role that need not be activated, hence the canActivate condition instead of hasActivated in the permits rule above). This is permitted if the patient or his agent have previously given consent to this treatment (activated a `Consent-to-treatment` role), in the case of emergency (active `Emergency-clinician` for the patient), or if the clinician has been referred by another treating clinician (who has activated `Referrer` for her). Finally, if the health organisation of the clinician supports Cassandra and is NHS-approved, she can simply submit a credential that proves she is a member of a workgroup involved in the treatment of the patient. This allows local hospitals to manage workgroup memberships themselves. The rule for this case is shown below and is an example of distributed delegation of authority. Note how the rule specifies that the clinician's `Group-treating-clinician` role credential has to be submitted directly to the EHR service, whereas the `NHS-health-org-cred` role credential is retrieved automatically, so we have two different automatic credential retrieval strategies within one rule.

$$\begin{aligned}
&\mathsf{canActivate}(cli, \texttt{Treating-clinician}(pat, org, spcty)) \leftarrow \\
&\quad org.\mathsf{canActivate}(cli, \\
&\qquad \texttt{Group-treating-clinician}(pat, group, spcty)), \\
&\quad org@ra.\mathsf{hasActivated}(x, \\
&\qquad \texttt{NHS-health-org-cred}(org, start, end)), \\
&\quad ra \in \texttt{NHS-registration-authorities}(), \\
&\quad \texttt{Current-time}() \in [start, end]
\end{aligned}$$

Patients (and similarly, their agents) can read their record items if they are not concealed by a clinician (nobody has activated a matching `Access-denied-by-clinician` role) or by the patient himself, e.g. if a patient does not want to see any items concerning "Cancer". Furthermore, all third parties affected by the item must have given their explicit consent to disclosure. The set of all such third parties whose consent has been registered is found using the group aggregation operator (the actual permits rule for patient access is not shown here):

$$\begin{aligned}
&\mathsf{third\text{-}party\text{-}consent}(\mathsf{group}\langle party\rangle, pat, id) \leftarrow \\
&\quad \mathsf{hasActivated}(x, \texttt{Third-party-consent}(party, pat, id))
\end{aligned}$$

Patient agents are registered on the patient's EHR service, either by the patient or by his GP. A registered agent can act on behalf of the patient on the EHR service but can also request an agent credential to be used at e.g. a local Cassandra-supporting hospital. Our revocation policy for agents is a combination of grant-dependent (patients can deactivate their own agent registrations) and grant-independent revocation (the GP can also deactivate her patients' registrations, e.g. if a patient is deemed incompetent). The agent registration rule for patients is a another variant of appointment with a restriction on the number of appointments: patients may appoint at most three different agents. (The aggregation rule for agent-regs(*n*, *pat*) counts the number of agent registrations, and is not shown here.)

$$\begin{aligned}
&\mathsf{canActivate}(pat, \texttt{Register-agent}(agent, pat)) \leftarrow \\
&\quad \mathsf{hasActivated}(pat, \texttt{Patient}()), \\
&\quad \mathsf{agent\text{-}regs}(n, pat), \\
&\quad n < 3
\end{aligned}$$

Other patient-centred issues such as consent to treatment are also managed at the EHR service level. The policy is designed in such a way that users are forced to activate at least one of the main roles, `EHR-service-admin`, `Clinician`, `Emergency-clinician`, `Patient`, `Agent` or `Third-party`, before they can perform any action or activate a registration, consent or access configuration role.

Local Health Organisations/EPR services It is expected that local health organisations will gradually migrate to electronic systems that conform to national standards and are compatible with the nation-wide services. This process may take a long time, so none of the national services puts any policy related requirements on local systems running in the health organisations. This is also the reason why the EHR cannot be deployed at this level. However, our EHR service policy is designed in such a way that if a health organisation can produce credentials and has a compatible policy-enforcement system, working with the EHR service can be more efficient. Health organisations can be as small as single GP practices but could also be entire NHS acute trusts with up to half a million registered patients. The policy of our model hospital defines 31 roles, five actions and 168 rules. It is mainly concerned with staff and patient registration (a `Receptionist` can register new patients by activating `Register-patient`(pat)) and the trust management of local electronic patient records (EPR). The rules for accessing the local EPR are similar to those for the shared EHR on the EHR services, so we do not give any explicit examples of policy rules here.

One difference is that our model hospital is workgroup based, with two kinds of workgroups, teams and wards. Patients are treated in clinical teams, usually headed by a consultant, and during inpatient episodes patients also belong to a ward, usually headed by a head nurse. As such, the legitimate relationship between patient and clinician is not based on explicit consent to treatment as in the case of the EHR services but only requires a receptionist or the head of the workgroup to register the patient with a workgroup by activating `Register-team-episode` or `Register-ward-episode`. Any clinician belonging to the workgroup automatically becomes a treating clinician. Clinicians can also request workgroup credentials for supporting requests to access a patient's EHR item on an EHR service.

Registration Authorities RAs could be located on the NHS cluster level, of which there are five in England. A typical cluster comprises up to 2000 Primary Care Practices and 100 Acute Hospital Trusts and other health organisations. An RA should be able to cope with up to 200,000 registered health professionals [38]. The policy of our model RA ("RA-East") defines five roles and 27 rules.

For example, the person *cli* is registered as a clinician with specialty *spcty* working for the health organisation *org* if somebody activates the appointment role `NHS-clinician-cred`(*org, cli, spcty, start, end*). RA credentials are required to be time-limited, so all RA roles have a start and an end date among their parameters. The role can be activated by local administrators, and only if *org* is registered as a current health organisation on this RA. Furthermore, the validity period must be covered by the validity period of the organisation's registration:

$$\begin{array}{l}\mathsf{canActivate}(adm,\\ \quad \texttt{NHS-clinician-cred}(org, cli, spcty, start, end)) \leftarrow\\ \mathsf{hasActivated}(adm, \texttt{RA-admin}()),\\ \mathsf{hasActivated}(x, \texttt{NHS-health-org-cred}(org, start', end')),\\ [start, end] \subseteq [start', end']\end{array}$$

The registration can be revoked by deactivating the role. This is an example of a variant of time-limited appointment with grant-independent revocation, as deactivation can be done by any active administrator (note that x occurs only in the head of the rule):

$$\begin{array}{l}\mathsf{canDeactivate}(adm, x,\\ \quad \texttt{NHS-clinician-cred}(org, cli, spcty, start, end)) \leftarrow\\ \mathsf{hasActivated}(adm, \texttt{RA-admin}())\end{array}$$

The registration is cancelled automatically if the health organisation loses its registration for the relevant period. This isDeactivated rule implements a rather complex variant of cascading deactivation (that cannot be expressed in e.g. Oasis): the cancellation should only proceed if there is no other registration for the health organisation that is valid during the relevant period. This negative condition is ensured by the user-defined aggregation predicate other-NHS-health-org-regs whose first parameter is 0 if and only if the condition holds.

$$\begin{array}{l}\mathsf{isDeactivated}(adm,\\ \quad \texttt{NHS-clinician-cred}(org, cli, spcty, start, end)) \leftarrow\\ \mathsf{isDeactivated}(x,\\ \quad \texttt{NHS-health-org-cred}(org, start', end')),\\ \mathsf{other\text{-}NHS\text{-}health\text{-}org\text{-}regs}(0, x, org, start', end'),\\ [start, end] \subseteq [start', end']\end{array}$$

$$\begin{array}{l}\mathsf{other\text{-}NHS\text{-}health\text{-}org\text{-}regs}(\mathsf{count}\langle y\rangle, x, org, start, end) \leftarrow\\ \mathsf{hasActivated}(y,\\ \quad \texttt{NHS-health-org-cred}(org, start', end')),\\ [start, end] \subseteq [start', end'] \wedge\\ (x \neq y \vee start \neq start' \vee end \neq end')\end{array}$$

A registered clinician can request a credential asserting that someone has activated the corresponding registration role for her during a given period. Similarly, EHR services, other RAs as well as the health organisation employing the clinician can also request the credential. The rule for the latter case is shown below.

$$\begin{array}{l}\mathsf{canReqCred}(org, ra.\mathsf{hasActivated}(x,\\ \quad \texttt{NHS-clinician-cred}(org, cli, spcty, start, end)) \leftarrow\\ \mathsf{hasActivated}(y, \texttt{NHS-health-org-cred}(org, start', end')),\\ \texttt{Current-time()} \in [start', end']\end{array}$$

As we can see, in this variant of appointment, the appointee (the clinician) never actually activates the appointed role `NHS-health-org-cred`. The appointer (the RA-admin) only enables her to request a credential certifying the appointment. Similar rules exist for Caldicott Guardians (`NHS-Caldicott-guardian-cred`) and health organisation authentication (`NHS-health-org-cred`) credentials. The RA policy further contains rules for the appointment of new administrators.

7. Discussion

Implementation and performance A prototype of Cassandra has been implemented in OCaml. The access control engine forms the core of the implementation. It provides a simple user interface for reading in policies and handling user requests concerning actions, credentials, and role activation and deactivation. On a request, the access control engine uses the policy evaluation engine, an implementation of the top-down evaluation algorithm from §3, to decide whether to grant or to deny access. If access is granted, the state (the global set of policies) might be changed, as specified in §4. The globally chosen constraint domain can be plugged into the access control and policy evaluation engines as an independent module. Constraint domain modules must only provide implementations for standard functions such as conjunction, satisfiability and implication checking, and existential quantifier elimination. We have implemented the constraint domain $\mathcal{C}_0$, including a type inference mechanism that allows us to omit explicit variable typing.

At the time of writing, role deactivation and credential requests and the static groundness analyser are still in the process of being implemented. Furthermore, the current prototype only simulates the distributed system, and issued credentials are implemented without encryption and public key signatures.

The prototype was tested with the EHR policies from §6 by going through various scenarios. The initial test results were promising: even the most complex requests were handled within fractions of a second. We believe it would be feasible to use Cassandra to enforce our EHR policy on a nationwide system, despite its relative complexity. Of course, authoritative results can only be produced after completion of a less naive implementation and under more realistic settings; we have for example so far only tested the system with up to 10,000 patients. Even though Datalog$_{\mathcal{C}}$ queries in $\mathcal{C}_0$ may theoretically be intractable, the test results with the EHR policy suggest that the worst-case does not occur in practice: the policy seems complex but a closer analysis reveals that recursion is very shallow and that nearly all variables become ground at an early stage. This means that neither the evaluation engine nor the constraint solving procedures need to work very hard. The current implementation is rather inefficient in that credential rules for role activations are stored in a linear list. If a hashtable is used instead, the cost of almost all requests would be nearly independent of the number of patients in the system; for the rest, the cost would be at most linear in the number of patients.

Related work A large amount of work has been done on policy specification in a non-trust-management context (e.g. Ponder [18], GTRBAC [30], FAF [28], RCL2000 [2], RDM2000 [49], just to name a few): these languages do not support credential-based authorisation, credential retrieval and distributed trust negotiation, or they do not focus on distributed resource sharing at all.

The notion of credential-based trust management was introduced by Blaze et al. with PolicyMaker [10, 9] and its successor, KeyNote [8, 7]. PolicyMaker's credentials and policies are fully programmable in that the choice of the policy specification language is left open to the user. This makes it very flexible but makes policy compliance checking undecidable in general. In KeyNote, policies are written in a specific assertion language containing simple string and arithmetic constraints on environment variables. Both PolicyMaker and KeyNote do not handle credential retrieval. Unlike Cassandra, these systems do not act as a protective layer around the resources; they are just called by the application to check policy compliance. The application is responsible for interpreting the answer and the actual policy enforcement. One of the distinctive aspects of Cassandra is that not only the policy language but also the access control semantics is formally specified, with exact definitions for the conditions and consequences of role activation and deactivation, action and credential requests. The policy language together with the access control semantics provide a rigorous framework unifying dynamic RBAC, role revocation, distributed trust management and trust negotiation. Existing distributed trust management systems have a narrower scope and specify at most the policy specification language, not the access control semantics.

The Cassandra policy specification language was inspired by Oasis [48, 26], a role-based trust management system in which Datalog-like rules specify which credentials are prerequisite for role activation and deactivation. Oasis does not retrieve credentials automatically and does not guarantee termination of queries. One of the main contributions of the work on Oasis was the development of the concept of appointment as a generalisation of delegation. Previous systems have concentrated on delegation (e.g. RDM2000 [49], a rule-based language with built-in role delegation), which, as our case study suggests, is far less useful in practice. Standard appointment can be expressed in Oasis with its built-in appointment mechanism. However, as the subject of the role activation prerequisites must be equal to the subject of the conclusion of the rule, it is not impossible to encode the different variants of appointment we encountered.

RDM2000 can express role revocation policies including revocation authorisation and cascading revocation in much the same way as Cassandra. Oasis is the only other trust management system we are aware of that supports rule-based cascading role revocation. However, its revocation mechanism is not expressive enough for some of the revocation policies in our case study. In particular, Oasis's revocation triggers are restricted to those role activations that were actually used to activate the role; furthermore, revocation is triggered whenever a role membership condition ceases to hold, even if other current conditions might allow the activation to stay alive. Cassandra's isDeactivated mechanism is expressive enough to capture such policies. In Oasis, revocations can be automatically propagated across the network via an event infrastructure. In contrast, Cassandra's revocations are explicitly restricted to local role activations. We believe that distributed revocation is too difficult to implement reliably and securely on a wide scale; for example, services would need to keep state about which other services to notify about revocations, and would need to apply at other services for the right to receive particular revocations.

The RT family of role-based trust management languages [33] bears some similarities to our system. RT can be seen as an extension/combination of SPKI/SDSI 2.0 [12] and Delegation Logic [31]. In RT, the Datalog-based rules, or credentials, as they are called, specify only the role membership relation: either directly, by role hierarchy, by (direct or attribute-based) delegation of authority, or any combination of these. As in Oasis, the subjects of the rule head and the body conditions are implicitly the same, which is sufficient to express delegation but not convenient for appointment policies. In RT's youngest offspring, $RT_1^{\mathcal{C}}$ [32], rules are translated into Datalog$_{\mathcal{C}}$. Constraints are used only to define a range on each role parameter; constraints between two parameters are not permitted in order to keep policies more comprehensible and to guarantee tractability. We find that a more liberal use of constraints is useful and necessary, as our EHR policy shows, and can still be efficient in practice. RT roles are prefixed with the issuing entity, just like Cassandra's predicates are, but do not specify the location where a matching credential may be found. RT solves this by statically specifying for each role name whether credentials defining such roles are stored with the issuer or the subject. Our EHR policy has rules in which predicates have locations different from both issuer and the subject entity.

SD3 is another Datalog-based trust management system [29]. It is similar to Cassandra in that SD3 predicates can also have an issuer and a location, which in the case of SD3 is an IP address. SD3 is a very general system that does not specify any access control meaning for its predicates and can be viewed as Cassandra without constraints, roles and access control semantics, but with automatic credential retrieval and trust negotiation. SD3 passes the proof tree from its highly optimised policy evaluation engine through a simple and small proof checker to reduce the size of its trusted computing base. This would be a technique that could also be applied to Cassandra.

The problem of trust negotiation has been addressed in

[46] where various different strategies are discussed. Their Credential Access Policy (CAP) corresponds to Cassandra's canReqCred rules. Cassandra's uniform treatment of rules during evaluation gives us trust negotiation "for free", with a strategy similar to the "Parsimonious Strategy" in [46]. It has been pointed out that this strategy may leak information about the policy without actually disclosing any credentials. [47] prevents this problem by adding another policy protection layer.

Conclusion and Future work We have developed a role-based policy specification system for access control in a distributed system, Cassandra, in which the expressiveness can be tuned according to need. In combination with the constraint domain $\mathcal{C}_0$, Cassandra's expressiveness surpasses that of existing systems while preserving a strong termination property. The language is truly policy-neutral in that it can express subtle variants of well-known policy idioms. It can express trust-management related policies such as credential-based authorisation, automatic trust negotiation and automatic credential retrieval strategies. Our work on a policy for a national EHR system, one of the most substantial trust management case studies we are aware of, demonstrates that Cassandra is expressive enough for large-scale real-world applications with highly complex policy requirements.

To gather more reliable test results, we will first complete the implementation of role deactivation and credential requests, and a static groundness analyser. In a further step, we plan to build a prototype that is truly distributed, uses digital certificates for sending credentials over the network and uses a relational database to accelerate the look-up of role activations. Such an implementation will enable us to test the EHR policy in a more realistic setting, with at least a few million policy entries. We will also produce web-based EHR user interfaces to illustrate how applications can be build on Cassandra.

Cassandra was designed to satisfy complex policy requirements and at the same time be simple enough that its language and access control semantics can be formally specified. We plan to use this formal framework to prove high-level meta-properties about specific policies or policy idioms. Along the same lines, we wish to formalise a low-level model of Cassandra that specifies the underlying network protocols, the public key infrastructure and the design of certificates.

Acknowledgments We acknowledge support from a Gates Cambridge Scholarship (Becker), a Royal Society University Research Fellowship (Sewell), EPSRC grant GRN24872, and EC FET-GC project IST-2001-33234 PEPITO. The authors thank Rishi Mukherjee for his insight and discussion on hospital management, and Arne Heizmann and Matthew Parkinson for corrections. We also thank the reviewers for their valuable comments.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases*. Addison-Wesley, 1995.

[2] G.-J. Ahn and R. Sandhu. Role-based authorization constraints specification. *ACM Transactions on Information and System Security*, 3(4):207–226, 2000.

[3] R. Anderson. A security policy model for clinical information systems. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 30–42, 1996.

[4] N. Baker and H. Sondergaard. Definiteness analysis for CLP(R). In *Australian Computer Science Conference*, pages 321–332, 1993.

[5] M. Y. Becker. Cassandra policies for a national EHR architecture. `http://www.cl.cam.ac.uk/users/mywyb2/policy`.

[6] M. Y. Becker and P. Sewell. Cassandra: distributed access control policies with tunable expressiveness. In *Policy Workshop*, June 2004.

[7] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. RFC 2704: The KeyNote trust management system version 2, 1999.

[8] M. Blaze, J. Feigenbaum, and A. D. Keromytis. KeyNote: Trust management for public-key infrastructures (position paper). *Lecture Notes in Computer Science*, 1550:59–63, 1999.

[9] M. Blaze, J. Feigenbaum, and A. D. Keromytis. The role of trust management in distributed systems security. In *Secure Internet Programming*, pages 185–210, 1999.

[10] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *IEEE Symposium on Security and Privacy*, pages 164–173, 1996.

[11] W. Chen and D. S. Warren. Towards effective evaluation of general logic programs. Technical report, State University of New York at Stony Brook, 1993.

[12] D. Clarke, J.-E. Elien, C. Ellison, M. Fredette, A. Morcos, and R. L. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.

[13] T. Collins. Doctors express alarm at plans to store patient data without consent (15/07/03). *Computer Weekly*, 2003.

[14] T. Collins. How the national programme came to be the health service's riskiest IT project (16/09/03). *Computer Weekly*, 2003.

[15] T. Collins and M. Simons. NHS plan branded a 'farce' (03/06/03). *Computer Weekly*, 2003.

[16] A. Cornwall. Electronic health records: an international perspective. *Health Issues*, 73, 2002.

[17] M. Cross. NHS spree revealed (12/06/03). *The Guardian*, 2003.

[18] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The Ponder policy specification language. In *Policy Workshop*, 2001.

[19] Department of Health, UK. Building the information core: Implementing the NHS plan. 2001.

[20] Department of Health, UK. Building the information core: Protecting and using confidential patient information. 2001.

[21] Department of Health, UK. Legal and policy constraints on electronic records. 2002.

[22] Foundation for Information Policy Research. NHS confidentiality consultation – FIPR response. February 2003. `http://www.cl.cam.ac.uk/users/rja14/fiprmedconf.html`.

[23] N. Gaunt. Confidentiality and consent: Use cases applicable to shared electronic health record. *S&W Devon ERDIP Project*, 2003.

[24] L. Giuri and P. Iglio. Role templates for content-based access control. In *Proceedings of the 2nd ACM Workshop on Role-Based Access Control (RBAC-97)*, pages 153–159, 1997.

[25] J. Y. Halpern and V. Weissman. Using first-order logic to reason about policies. In *Proceedings of the Computer Security Foundations Workshop (CSFW'03)*, 2003.

[26] R. Hayton, J. Bacon, and K. Moody. OASIS: Access control in an open distributed environment. In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, pages 3–14, 1998.

[27] J. Jaffar and M. J. Maher. Constraint logic programming: a survey. *Journal of Logic Programming*, 19/20:503–581, 1994.

[28] S. Jajodia, P. Samarati, M. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, 26(2):214–260, 2001.

[29] T. Jim. SD3: A trust management system with certified evaluation. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pages 106–115, 2001.

[30] J. B. D. Joshi, B. Shafiq, A. Ghafoor, and E. Bertino. Dependencies and separation of duty constraints in GTRBAC. In *Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 51–64. ACM Press, 2003.

[31] N. Li, B. N. Grosof, and J. Feigenbaum. Delegation Logic: a logic-based approach to distributed authorization. *ACM Transactions on Information and System Security*, pages 128–171, 2003.

[32] N. Li and J. C. Mitchell. Datalog with constraints: A foundation for trust management languages. In *Proceedings of the 5th International Symposium on Practical Aspects of Declarative Languages*, pages 58–73, 2003 2003.

[33] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130, 2002.

[34] E. C. Lupu and M. Sloman. Reconciling role-based management and role-based access control. In *ACM Workshop on Role-based Access Control*, pages 135–141, 1997.

[35] National Health Service, UK. The Caldicott committee: report on the review of patient-identifiable information. 1997.

[36] National Health Service, UK. Information for health: an information strategy for the modern NHS 1998-2005. 1998.

[37] National Health Service, UK. ERDIP evaluation: Technical options for the implementation of electronic health record nationally. 2002.

[38] National Health Service, UK. Integrated Care Records Service: Output based specification version 2. 2003.

[39] M. Palmer. A complex operation for the NHS spine (14/08/03). *Computer Weekly*, 2003.

[40] P. Revesz. *Introduction to constraint databases*. Springer Verlag, 2002.

[41] J. Rogers. GPs voice patient confidentiality concerns (20/05/03). *Computer Weekly*, 2003.

[42] R. Sandhu. Rationale for the RBAC96 family of access control models. In *Proceedings of the 1st ACM Workshop on Role-Based Access Control*, 1997.

[43] R. Sandhu, V. Bhamidipati, E. Coyne, S. Canta, and C. Youman. The ARBAC97 model for role-based administration of roles: Preliminary description and outline. In *Proceedings of 2nd ACM Workshop on Role-Based Access Control*, pages 41–54, 1997.

[44] A. Sommerville, N.-J. Macdonald, and R. Weston. *Medical Ethics Today: Its Practice and Philosophy*. British Medical Association, BMJ Publishing Group, 1993.

[45] D. Toman. Memoing evaluation for constraint extensions of datalog. *Constraints*, 2(3/4):337–359, 1997.

[46] W. Winsborough, K. Seamons, and V. Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*, volume 1, pages 88–102, 2000.

[47] W. H. Winsborough and N. Li. Towards practical automated trust negotiation. In *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks*, pages 92–103, 2002.

[48] W. Yao, K. Moody, and J. Bacon. A model of OASIS role-based access control and its support of active security. *ACM Transactions on Information and System Security*, 5(4), 2002.

[49] L. Zhang, G.-J. Ahn, and B.-T. Chu. A rule-based framework for role-based delegation and revocation. *ACM Transactions on Information and System Security*, 6(3):404–441, 2003.