View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Teeside University's Research Repository

# Attribute Based Access Control for Big Data applications by Query Modification

Jim Longstaff
School of Computing
Teesside University
Middlesbrough, England
j.j.longstaff@tees.ac.uk

Joanne Noble
School of Computing
Teesside University
Middlesbrough, England
j.e.noble @tees.ac.uk

***Abstract*—we present concepts which can be used for the efficient implementation of Attribute Based Access Control (ABAC) in large applications using maybe several data storage technologies, including Hadoop, NoSQL and relational database systems. The ABAC authorization process takes place in two main stages. Firstly a sequence of permissions is derived which specifies permitted data to be retrieved for the user's transaction. Secondly, query modification is used to augment the user's transaction with code which implements the ABAC controls. This requires the storage technologies to support a high-level language such as SQL or similar. The modified user transactions are then optimized and processed using the full functionality of the underlying storage systems. We use an extended ABAC model (TCM2) which handles negative permissions and overrides in a single permissions processing mechanism. We illustrate these concepts using a compelling electronic health records scenario.**

***Keywords-Attribute Based Access Control, Identity and Access Management, Enterprise Information Systems, Hadoop, NoSQL***

## I. INTRODUCTION

In this paper we introduce an extended model of Attribute Based Access Control which we call the Tees Confidentiality Model version 2 (TCM2) [1] [2]. We particularly focus on its use in large applications which may include several storage technologies, e.g. relational database, relational data warehouse, Hadoop, and NoSQL. An electronic health records (EHR) scenario, in which a patient wishes to restrict access to sensitive data across all storage systems, is used for illustration.

Firstly, we note that health data about individuals may be stored for different purposes in different systems. Medical data, appointments, and insurance data may be stored in relational databases and relational data warehouses; medical monitoring data and archived data may be stored in NoSQL or Hadoop systems. An example of a Hadoop health data application is described in [3].

Any of this data could potentially indicate the existence of a condition or event that the patient wishes to restrict access to. For example, an appointment at a particular clinic could indicate a sensitive or embarrassing medical condition without accessing medical records. A patient might require finely-controlled access to such data, which might have consequences if other family members or insurance companies were to discover it. Such a situation is described in the scenario in section 3 below.

Additionally, there may be circumstances where restricted data must be accessed and communicated due to legal requirements, e.g. for communicable diseases, or when data about a third party who may be at risk is held. To enable this, overriding restrictions for appropriate persons or authorities must be provided, across all storage systems.

This paper offers a technical solution to these issues in the form of modifying user transaction code to include Attribute Based Access Control (ABAC) parts to implement patient consent directives. For such controls to be relatively simply generated, data access must be via a high level language, such as SQL or similar. Many "SQL on Hadoop" and NoSQL systems support such languages, making the use of such controls a possibility. The sophisticated optimizations and processing of the underlying systems can then come into play to access the data the user is authorized to view and use.

An authorization system is said to implement a particular authorization model. To-date, the most widely-used authorization model has been Role Based Access Control, or RBAC, which is used in operating systems, databases, access control systems for specialized applications, and development environments. Attribute Based Access Control or ABAC, is generally seen as the way forward for authorization model research, see e.g. [4]. The central idea of ABAC is that access can be determined based on various attribute values presented by a subject. Permissions (often called rules) specify conditions under which access is granted or denied. A comprehensive description of ABAC is given in [5], and approaches for combining RBAC and ABAC are outlined in [6].

This paper is structured as follows. Section 2 describes how the techniques presented in this paper could be further researched and developed to form the basis of a comprehensive ABAC authorization framework for big data applications. Section 3 presents a motivating example of a healthcare records scenario that includes the specification of consent directives and overrides; the value of supporting overrides is particularly illustrated. Section 4 gives the TCM2 permissions to implement the consent directives. Following this, a description of TCM2 model is given in section 5. The

key definitions for permissions processing are presented in section 6, together with the permissions sequences for an example user transaction. The SQL implementation of the user transaction and TMC2 controls is given in section 7. A comparison with other research is offered in section 8, after which conclusions and references follow.

## II. TOWARDS ABAC FOR BIG DATA

### A. Models and Languages

The ABAC model, whether it is TCM2 as described in section 5, or an alternative, should work with concepts which are understandable by end-users, analysts and developers alike. This should apply to the model elements (users, protected objects, attributes), and application objects (well-recognized real-world concepts such as patient, medication, insurance claim, etc.). Applications design approaches e.g. Entity Relationship analysis, relational database normalization, object-oriented modelling, invariably produce well-defined concepts.

The ABAC model must be capable of generating queries in the high-level, usually SQL-like language supported by the underlying storage systems, which might be relational database, NoSQL, Hadoop or other big data systems. Many systems support a variant of SQL. In particular there must be a mapping from the ABAC Protected Objects to the data stored in the underlying systems, which is presented and accessed using the SQL variant.

### B. ABAC Permissions design

Permissions are developed and maintained throughout an application's lifecycle by security architects. Permissions specification necessarily follows the development of the application model. User stories and Use Case models indicate typical users. Class Diagrams or Entity Relationship Diagrams indicate Protected Objects. ABAC permissions can be developed and tested alongside application functionality. Provision should also be made for dynamic authorization, where a new permission can be added to a mostly stable base permissions base as the need arises (e.g. to grant or deny access to a particular employee).

### C. Implementation

The ABAC permissions must be capable of generating queries to stored data in the high-level, usually SQL-like languages supported by the underlying storage systems. Examples of systems include Apache Hive, Drill, Spark SQL, Impala, and IBM Big Data. Similarly NoSQL systems such as Vertica, MongoDB, CouchDB and Cassandra, also support SQL-like languages for both tabular and JSON representations of data

The storage systems will therefore support an SQL-like schema for stored objects which may be in the form of relational tables, JSON documents. Each storage model has its advantages and disadvantages. Normalized relational tables will usually contain exactly the data specified in a user query, but might require joins to produce it. JSON documents might contain much more data than is requested, some of it denied by ABAC permissions. This would require complex filtering to produce and make available the permitted data in the retrieved documents.

## III. HEALTHCARE SCENARIOS

### A. Summary

This scenario was suggested by a Consultant Transplant Surgeon during the design and development of an Electronic Health Records (EHR) system. The "consent directives" indicate privacy concerns that have been expressed by patients in the past.

The scenario concerns a fictitious patient, Alice, and her GP, Fred. Alice is 50; the major events in Alice's medical history are:

- She had a pregnancy termination when she was 16
- Was diagnosed diabetic at 25
- End Stage renal failure when she was 45
- Renal transplant at 48
- Acutely psychotic at 49
- Crush fracture of T12 aged 50

Let us now suppose, not unreasonably, that Alice expresses the desire to place the following consent directives on the availability of her EHR data about two of these conditions:

1. My GP (Fred) can see all my data
2. Nobody must know about my termination except my GP, any Gynaecological Consultant, and the Consultant Renal Transplant Surgeon (Bill) who operated on me.
3. My GP, Consultant Renal Transplant Surgeon (Bill) and Consultant Orthopaedic Surgeon (Bob) can see my psychosis data, but no-one else.

To show the power of our ABAC model, consider the following contrived requirement (but still one which an EHR authorization system should be capable of handling):

4. I do not wish the members of the hospital team who carried out my termination operation to be ever able to see my psychosis data, except if they are viewing in a psychiatric role. (This directive to be in force throughout the careers of those professionals concerned).

We must add to these directives that they must be capable of being overridden in carefully controlled and audited ways. An example of overriding follows.

### B. Transaction Example

Consider the following transaction which requires access to restricted data, and illustrates the need for an override capability. The clinician user is a transplant surgeon, querying Alice's medical data.

Alice has been scheduled for a transplant (one of the major events listed). Tests lead the surgeon to suspect a previous pregnancy (if the tissue type of the father is similar to the graft

a very serious rejection may ensue), but the EHR termination data is denied to him. Alice refuses to confirm a previous pregnancy to the surgeon.

The transplant surgeon elects to override, to attempt to discover information about previous pregnancies. He first uses a Level 1 TP Override (described in section 5C) which is available to all healthcare professionals. This does not yield any data, because a Level 2 deny permission has been placed on the termination data. However a message is displayed, just for a user in the transplant surgeon role, saying that he can and should use a Level 2 TP Override. He does this, and discovers the termination data he needs. This allows for a specific form of treatment to be planned.

## IV. Scenario Permissions Representation

As an introduction to our TCM2 model we now give examples of permissions for the scenario. We call our permissions T Permissions, or TPs, to distinguish them from RBAC permissions, and other ABAC rule formulations. TPs consist of sets of classifier values; an example of a classifier value is <UserRole, Psychiatrist>. Classifier values can represent information other than attribute values, as is explained in section 5 below.

The permissions below are written using an informal notation which represents the data structures used for implementing TCM2.

Firstly, the EHR data for any patient is normally made available to

a) Healthcare professionals (HCPs) such as clinicians, doctors, and administrators who have a Legitimate Relationship (LR) with the patient. This means that the patient is registered with or has been referred to them.

b) Additionally, all HCPs can exercise a Level 1 TP Override facility, to exceptionally access restricted data, when they have reason to do so. Naturally, all access and overrides will be logged, and subject to audit.

The following TPs authorize this access:

TP1 Permit_TP (N):
{<UserRole, HCP>,
<LR, yes>,
<Op_id, R_A>,
<PO_Type, EHR>}

TP2 Permit_TP(L1_Ovr):
{<UserRole, HCP>,
<LR, yes>,
<Op_id, R_A>,
<PO_Type, EHR>}

TP1 represents the granting of read and append access to EHR data for a clinician-user in the role of Healthcare Professional ("HCP"), under normal ("N") processing where no override has been used. A Legitimate Relationship ("LR") must exist, meaning that the patient is registered with the clinician, or has been referred to the clinician for treatment. TP2 permits access for any HCP to any EHR data if the user has exercised a Level 1 Override.

The TPs which implement the consent directives given in section 3A, in the order in which they are expressed, are

TP3 Deny_TP(L2):
{<UserRole, HCP>,
<PO_Coll_id, Alice_TerminationData >,
<PO_Type, EHR>}

TP4 Permit_TP (N):
{<User_id, Fred>,
<UserRole, GP>,
<Op_id, R_A>,
<PO_Coll_id, Alice_TerminationData >,
<PO_Type, EHR>}

TP5 Permit_TP (N):
{<UserRole, GC>,
<Op_id, R_A>,
<PO_Coll_id, Alice_TerminationData >,
<PO_Type, EHR>}

TP6 Permit_TP (N):
{<User_id, <Bill >,
<Op_id, R_A>,
<PO_Coll_id, Alice_TerminationData >,
<PO_Type, EHR>}

TP7 Deny_TP(L2):
{<UserRole, HCP>,
<PO_Coll_id, Alice_PsychiatryData >,
<PO_Type, EHR>}

TP8 Permit_TP (N):
{<User_id, Fred>,
<UserRole, GP>,
<Op_id, R_A>,
<PO_Coll_id, Alice_PsychiatryData >,
<PO_Type, EHR>}

TP9 Permit_TP (N):
{<User_id, < Bill, Bob>
<Op_id, R_A>,
<PO_Coll_id, Alice_PsychiatryData >,
<PO_Type, EHR>}

TP10 Permit_TP(N):
{<User_Coll_id, TermTeam>,
<UserRole, Psychiatrist>,
<Op_id, R_A>,
<PO_Coll_id, Alice_TerminationData >,
<PO_Type, EHR>}

Deny TPs are negative permissions which prevent access. These can be very detailed, for specific users and data, TP3 denies (at Level 2 – see section 5C) any kind of access to Alice's termination data to HCPs. However if authorized by another TP, e.g. TP12 below, a transplant surgeon could use TP Override at Level 2 to cancel the effect of the deny permission TP3.

We assume that HCPs are never granted a Level 2 Override, which provides access to very sensitive data denied at Level 2. The permissions which generate the message to the transplant surgeon upon L1 Override, and which provide the L2 Override for the transaction scenario in section 3B, are

TP11 Deny_TP (L1):
{<UserRole, TransplantSurgeon>,
<LR, yes>,
<PO_Coll_id, Alice_TerminationData >
<PO_Type, EHR>}

TP12 Permit_TP(L2_Ovr):
{<UserRole, TransplantSurgeon>,
<LR, yes>,
<Op_id, R_A>,
<PO_Coll_id, Alice_TerminationData >
<PO_Type, EHR>}

The message associated with the TP11 permission could only be sent to a Transplant Surgeon who has an established LR with the patient. Also the transplant surgeon could only access the data upon Level 2 Override if he possesses an LR.

## V. TCM2 Overview

In this section we summarize further aspects of the TCM2 authorization model. More detailed expositions can be found in [2] [7].

### A. Classifiers

The TCM2 model is based on the RBAC concepts of users, operations and protected objects [8]; however these concepts now have classifiers, as illustrated in the previous section. The simplest form of classifier corresponds to an attribute, as used in ABAC [5] [9]. User classifiers can take the role of parameters in parameterized RBAC; extended classifiers are defined for combinations of user, operation and protected object, and collection classifiers can be created to facilitate authorizations for collections of objects. Classifier values are structured into hierarchies, which can be represented as inverted trees, with less-specialized values placed nearer the root. Classifier values can be provided by several mechanisms (stored database values, generator programs, and external applications). A classifier ordering is determined by the security architect or analyst, to indicate importance for matching (ie deciding the authorization outcome). For example if the classifier User_id was deemed to be more important than UserRole when deciding authorization, then a permission with a User_id value match would be preferred to another permission (not containing the User_id value) which was matched by a UserRole value.

The model also includes an override operation which allows a user to acquire a more specialized classifier value (if he was specifically authorized to use this type of override); e.g. a JuniorPsychiatrist might acquire the role (ie classifier value) of ConsultantPsychiatrist in an emergency situation.

### B. T Permissions

Other TPs may be derived using the classifier value hierarchies for each classifier present in a TP. Ranges of classifier values can also be specified in TPs.

TCM2 builds the permission, checks that it doesn't repeat or conflict with existing permissions, and then generates an explanation of the permission for validation.)

### C. Deny levels

Deny TPs are specified at increasing levels of power, called Deny Levels. A deny level contains deny permissions specified at lower deny levels. Therefore data could be denied to users who might be able to access it by Level 1 Override (if so authorized), whereas more sensitive data might be only available to more senior users who were authorized to override at Level 2.

### D. TP Sets

TPs can be defined as having membership in separate, independent T Permission Sets, or TP Sets. TP Sets can be used separately to determine authorization, or combined.

Representation of different levels of processing can be accomplished with TP Sets, e.g. government regulations (TPS1), consumer-specified directives (TPS2), and directives specified by proxies for consumers (TPS3). Therefore TPS1 authorizations can be preferred to TPS2 authorizations, if this is what the security architect requires.

In the examples above, access to health records is provided by one TP Set.

## VI. Matched Permissions Sequences

### A. Overview

A full formal specification of TCM2 has been developed using the B Method [10], and extracts from this specification are included in this section. Permissions processing depends on two main principles: TP Match, and Nearest Match, which we describe below.

### B. TP Match

Firstly, a T Permission will match (ie qualify to authorize a transaction) if all its classifier values are contained in the transaction. Additionally, a TP will match if one of its derived TPs matches.

This can be expressed formally using B by the following definition:

$$\textbf{\textit{TPPermitAccess}}\ (tp, acvals) \triangleq \textbf{\textit{bool}}\ (\textbf{\textit{dom}}\ (acvals \cap \textbf{\textit{ad}}\ [tp]) = \textbf{\textit{dom}}\ (tp))$$

where acvals is the transaction active classifier values (classifier values specifying the transaction, example given in section 3B), and tp is a T Permission which permits access. The set ad[tp] contains the original ancestor classifier values as well as the set of all descendant classifier values. That is, access is granted if for every classifier in the domain of tp there exists at least one classifier value in common between the active classifier values acvals and the classifier values of tp and all their descendants. Similarly for TPDenyAccess.

### C. Nearest Matched TP

The second principle concerns determining which of two TPs (taken from a set of Matched TPs) is the stronger or nearer match to a transaction. This Nearest Match TP would then have a higher priority in determining the authorization outcome.

A TP is a set of classifier values. There is an ordering cfiersq on the classifiers that is set by the security architect and is a mapping of the set of integers 1,2,3,4....to the set of classifiers.

$$\textbf{\textit{cfiersq}} \in \textbf{iseq}\ (\textit{cfiers})$$

Given the ordering on the classifiers then for any set of classifier values cvs there exists a classifier for that set which is the most important classifier i.e. the lowest in the ordering.

$$CFIER_L\,(cvs) = \textbf{\textit{cfiersq}}\ (\textbf{\textit{min}}\ (\textbf{\textit{cfiersq}}^{-1}[\textbf{\textit{dom}}\ (cvs)])$$

There also exists an associated ordering number for that classifier and an associated value:

$$NCFIER_L\,(cvs) = \textbf{\textit{min}}\ (\textbf{\textit{cfiersq}}^{-1}[\textbf{\textit{dom}}\ (cvs)])$$
$$VCFIER_L\,(cvs) = cvs\ [CFIER_L\,(cvs))]$$

Therefore, given a set of matched TPs tps the (set of) nearest match(es) is given by

$$\begin{aligned}
&\textbf{\textit{NearestMatch}}(tps) \triangleq \{nmtp \mid nmtp \in tps \wedge \\
&\neg\exists tp.\ (tp \in tps \wedge \\
&(\\
&\quad NCFIER_L(nmtp - tp \cap nmtp) > \\
&\quad NCFIER_L(tp - tp \cap nmtp) \\
&\vee \\
&\quad VCFIER_L(nmtp - tp \cap nmtp) \mapsto \\
&\quad VCFIER_L(tp - tp \cap nmtp) \in \textbf{\textit{ad}}) \\
&)\\
&\}
\end{aligned}$$

where ad is the ancestor/descendant relationship.

### D. Normal TP Processing Example

The Initially-Matched set of TPs, and the Nearest Match TP sequence (following removal of all Override TPs) are:

**Initially-Matched TPs**

| | |
|---|---|
| TP1 | Permit_TP (N) |
| TP2 | Permit_TP (L1_Ovr) |
| TP3 | Deny_TP (L2) |
| TP7 | Deny_TP (L2) |
| TP11 | Deny_TP (L1) |
| TP12 | Permit_TP (L2_ovr) |

**Nearest-Matched TPs (no overrides)**

| | | |
|---|---|---|
| TP1 | Permit_TP (N) | 1 |
| TP3 | Deny_TP (L2) | 2 |
| TP7 | Deny_TP (L2) | 3 |
| TP11 | Deny_TP (L1) | 4 |

The match strength is indicated in ascending order, starting with the weakest (i.e. 1).

Processing the Nearest-Match TP sequence authorizes the retrieval of all data except the Termination and Psychosis data. The strongest match, TP11, will exactly match the transaction, and will deny access to the Termination data for Transplant Surgeons; it will generate a message, though, just for TransplantSurgeons. TP7 and TP3 deny access to the Psychiatric and Termination data for all HCPs. TP1 permits access to all data except the Psychiatric and Termination data for all HCPs.

### E. Override TP Processing Examples

Consider the transaction from section 3.B. The same initially-matched TPs are returned. However on applying Level 1 Override (L1_ovr) the sequence of Nearest-Matched TPs shown below is obtained: processing this sequence determines that access is again permitted to all data except the Termination and Psychosis data.

If a TP Level 2 Override (L2_Ovr) is used, the indicated sequence is obtained: these TPs authorize access to the termination and unrestricted data, while still denying access to the psychosis data.

**Nearest Match TPs (L1_ovr)**

| | | |
|---|---|---|
| TP1 | Permit_TP (N) | 1 |
| TP2 | Permit_TP (L1_Ovr) | 2 |
| TP3 | Deny_TP (L2) | 3 |
| TP7 | Deny_TP (L2) | 4 |
| TP11 | Deny_TP (L1) | 5 |

**Nearest Match TPs (L2_ovr)**

| | | |
|---|---|---|
| TP1 | Permit_TP (N) | 1 |
| TP2 | Permit_TP (L1_Ovr) | 2 |
| TP3 | Deny_TP (L2) | 3 |
| TP7 | Deny_TP (L2) | 4 |
| TP12 | Permit_TP (L2_Ovr) | 5 |

## VII. SQL Permissions Implementation

### A. Overview

Firstly, we note that there must be a mapping from the TCM2 conceptual model (protected objects, classifier value hierarchies) to the models implemented by the underlying storage systems. This can be straightforward for normalized relational models, less so for models not in BCNF. JSON documents can provide hierarchical structures if so designed, and indexing of documents and other structures can be used to retrieve permitted data.

We now describe the central features of just one approach which generates and processes Nearest Match TP Sequences to provide access to authorized data. This approach uses SQL, and requires all the permissions which potentially apply to be presented in one relational table or materialized view. A sequence of Nearest Match permissions is derived by a single SQL query on this view. From this Nearest Match sequence, WHERE clause statements are constructed which are added to

the user's transaction, itself programmed using SQL. This is an application of the Query Modification technique pioneered by Stonebraker [11]. The examples which follow have been programmed in Transact SQL for Microsoft SQL Server 2014.

The full advantages of database technology for large-scale implementations are not addressed in detail here. One example would be the use of optimized, stored execution plans for known transactions, which would only need re-building for TCM2 purposes if the Initially-Matched TP set for the transaction changed.

### *B. The TP Relational Model*

Base tables, not described here, exist to hold data about Classifiers, ClassifierValues, and permissions, and have appropriate constraints, storage organizations and indexes. Whatever the underlying data model, a set of TPs can be represented as a single indexed view, which contains the columns

TP_id: TP identifier

PorD: indicates whether the permission, if matched, permits or denies access

L: indicates which level access is granted or denied

Ovr: indicates an override permission (which must have a PorD value of "permit")

and for each classifier value and classifier that can appear in a TP:

| | |
|---|---|
| Classifier id: | Classifier identifier |
| Classifier_UserPrec | The security architect-defined relative importance (see sections 5A and 6C) |
| ClassifierValue: | The actual Classifier Value |
| PCVH | The position of the ClassifierValue in the Classifier Value Hierarchy (a higher position indicates more specialised value, see section 5A) |

If no ClassifierValue for that classifier exists in the TP, then a dummy ClassifierValue and a dummy PCVH value is entered in the table. The dummy data is chosen so as not to affect the outcome of the ORDER BY clause in the SQL query in section 7C.

### *C. Determining the Nearest Match Sequence*

The following SQL expression will determine the Nearest Match Sequence for the transaction in section 3B:

```
SELECT * FROM TP
WHERE
dbo.fnCvMatch('User_id','John', User_id) > 0 AND
dbo.fnCvMatch('UserRole','TransplantSurgeon', UserRole) > 0
AND
dbo.fnCvMatch('Op_id', 'RA', Op_id) > 0 AND
dbo.fnCvMatch('PO_Type', 'EHR', PO_Type) > 0
ORDER BY User_id__PCVH desc,PO_id_PCVH desc,
UserRole_PCVH desc, PO_Type_PCVH,
PO_ClinicianOfCare_PCVH desc, PO_Site_PCVH desc,
PO_StartDate_PCVH desc, PO_EndDate_PCVH desc
```

The function fnCvMatch ('Classifier',' tran_tpcv', tpcv) is present for every classifier value in the transaction acvals. It does the following:

- Returns 0 if the tran_tpcv is a dummy value, meaning the tran_tpcv isn't part of the tpcv ClassifierValueHierqachy being tested.
- Returns 0 if tran_tpcv is not present in the tpcv ClassifierValueHierachy.
- Returns an integer > 0 if the tran_tpcv is present in the tpcv ClassifierValueHierachy.

There are versions of fnCvMatch defined for particular data types. Also this function can use a classifier value provided by external systems. It can be implemented efficiently in several ways, depending on the tables used to represent the ClassifierValueHierachy.

The SELECT FROM WHERE clause directly implements the permissions matching described in section 6B. (Note that permissions review can be accomplished by modifying this part of the query – the ORDER BY clause is not required.)

The ORDER BY clause contains PCVH data appearing in the order of importance of the corresponding classifier defined by the security architect. It directly implements the Nearest Match definition described in section 6C.

### *D. The Health Events Data Model*

These examples are programmed for a single PO (ie protected object) table containing health events data for patients. This table design is based on a data model for a GP system. Examples of health events, each represented by a single row, range from operations, prescriptions, to telephone communications. One column enables health events to be associated with a diagnoses of a particular condition, allowing e.g. a prescription to be associates with the condition of asthma.

We note that this PO table would be suitable for Hadoop and NoSQL implementation. Very large volumes of data could be quickly loaded into these systems. If a JSON document structure is supported there is the potential for grouping all data relating to a sensitive condition, facilitating the processing of TCM2 classifier collections such as <PO_Coll_id, Alice_TerminationData > from the scenario presented in sections 3 and 4.

### *E. Transaction query*

The transaction consists of John the Transplant Surgeon, querying Alice's EHR to discover data about previous pregnancies. This data is denied to him by permission TP7 from section 4, under normal processing.

```
SELECT * FROM PO
WHERE
Patient_id = 2220 /*Alice*/ and [PO_Type] = 'EHR'
```

### *F. Query with TMC2 controls*

The previous query has now been augmented with code derived from the Nearest Match TP sequence in section 6D. When regarded as a database query, this sequence describes

all the user – operation – protected objects permitted by the transaction. When the augmented SQL is combined with the transaction SQL, there is very often huge scope for query optimization, as is illustrated in the example below.

```
SELECT * FROM  PO
WHERE
Patient_id = 2220 AND PO_Type = 'EHR'
AND
(-- TP1 PO part
 PO_Type = 'EHR')
AND NOT
  (-- TP3 PO part
   PO_Type = 'EHR'
     AND
   PO_id IN
        (SELECT PO_id FROM AliceTerminationData)
  )
 AND NOT
   (-- TP7  PO part
    PO_Type = 'EHR'
      AND
    PO_id IN
        (SELECT PO_id FROM AlicePsychiatricData)
   )
 AND NOT
    (-- TP11 PO part
     PO_Type = 'EHR'
       AND
     PO_id IN
        (SELECT PO_id FROM AliceTerminationData)
    )
```

Expressed in SQL, rather than the internal data structures of the ABAC system, this query is optimized to:

```
SELECT * FROM  PO
WHERE
Patient_id = 2220 AND PO_Type = 'EHR'
AND PO_id NOT IN
        (SELECT PO_id FROM AliceTerminationData)
AND PO_id Not IN
        (SELECT PO_id FROM AlicePsychiatricData)
```

Similarly, the Level 2 Override permissions example would be optimized to:

```
SELECT * FROM  PO
WHERE
Patient_id = 2220 AND PO_Type = 'EHR'
AND PO_id NOT IN
        (SELECT PO_id FROM AlicePsychiatricData)
```

## VIII. Related Work (Authorisation Models)

TCM2 has a number of similarities with our previously published TCM work [12] [13], in which role is treated as an application concept, and similar overrides are proposed. Also, the previous TCM papers have described design and processing strategies for permission types, but not for dynamic authorization involving individual permissions, as has been presented in this chapter.

In the original TCM, hierarchies of classifier collections formed the basis of permissions processing, and permissions design. Also inheritance of permissions within classifier collection hierarchies was specified using permission types. In this chapter, hierarchies of classifier values, with permissions inheritance always assumed, replaces classifier collection hierarchies. This is a major difference between TCM2 and the TCM. Also the TCM has no concept of permission-triggered messages.

Relational database query modification was developed for the INGRES database system [11]. The basic idea is that a user interaction with the data base is modified to an alternate form in which the specification of authorization controls are included. This modification takes place in a high level interaction language. Hence, the processing of a resulting interaction can be accomplished with no further regard for protection. The sophistication of the underlying storage technology can be brought to bear to support enterprise applications.

Regarding emergency access to data, the Break-Glass approach [14] provides emergency accounts giving access to normally restricted data. The difficulties of such an approach are discussed in [15], which integrates a Break Glass approach into access control software; emergency level access is supported. These emergency levels are similar to the ‘deny levels’ concept in our model.

Investigations into Attribute Based Access Control, or ABAC, have been undertaken to address the inflexibility to change of RBAC models, and for use in distributed applications [5] [16] [17]. Access decisions are based on attributes that the user can be proved to have. In ABAC, different parties must reach trust agreements over attribute definitions, which can be more straightforward than agreeing consistent role definitions. ABAC provides good support for context, such as time of day. ABAC has been sometimes referred to as Policy Based Access Control or Claims Based Access Control. ABAC research, particularly focusing on attribute integrity and security, has been referred to as the ‘grand challenge’, and the future direction of authorization model development [4]. Applications in messaging and cryptography are described in [18] [19], and consistency and fault detection in rule structures is reported in [16].

XACML is an extensively developed and implemented ABAC approach, for which the underlying model has similarities with TCM2. XACML subjects, actions, and resources (corresponding to TCM2 users, operations, and protected objects) have attributes, on which authorization decisions are made. A comprehensive architecture involving PDPs, PEPs is defined for this. There is provision for extensions to be written into an XACML application, which could be used in an implementation of our model.

There are potential difficulties for permissions review/risk exposure for ABAC – potentially large numbers of rules, and their processing, must be considered. Huang, et al. [20] propose a combination of ABAC and RBAC, in which the permissions available to a user are the intersection of permissions provided by RBAC active roles and ABAC rules. Our model extends the ABAC approach in that classifiers (which can represent ABAC attributes) are defined for operations and protected objects, in addition to users. Note that there is no direct TCM2 equivalent to RBAC permissions, which are used in the presentation of ABAC models.

RBAC models for role administration (ie for assigning roles to users) have been extensively researched, eg the ARBAC02 model [21]. The ARBAC02 model includes models of organization structures for user pools and permission pools which are independent from role hierarchies. These concepts could in principle be modeled in our model by classifiers, which are themselves independent. The development of a system for administration is beyond the scope of this paper, and is a topic for continuing research.

Our model can straightforwardly support a central concept of usage control [22] [23] [24] in that mutable attributes can be modeled as classifiers, and can participate in permission types. The 'Legitimate Relationship' classifier featured in this paper is similar to a mutable attribute, and can determine and change access during a session. Also Legitimate Relationship shows how we can model a relationship between a user and a protected object.

## IX. CONCLUSIONS

We have summarized an approach to controlling access to data held and maybe duplicated and summarized across relational database, Hadoop and NoSQL systems. The mechanism involves generating additions to user transactions expressed in high-level SQL-like languages, to ensure that only data satisfying restrictions can be retrieved and used. To the best of our knowledge this approach has not appeared before in literature connected with big data applications. We anticipate that a comprehensive ABAC framework for big data applications can be developed based upon the concepts presented in this paper.

Several relational database demonstrators of our TCM2 model have previously been implemented by ourselves and others, and the approaches to complex authorizations and override positively evaluated within healthcare information system projects, and research and commercial ventures.

## ACKNOWLEDGMENT

The authors wish to thank Tony Howitt, Professor Mike Lockyer, Professor Michael Thick and Steve Dunne for advice and contributions. This work was supported in part by grants and contracts from the England National Programme for IT (part of the England National Health Service), particularly as part of the ERDIP and HRI Programmes (2000-2006).

## REFERENCES

[1] J. Longstaff, "Extending Attribute Based Access Control to Facilitate Trust in eHealth and Other Applications," in *Cyber Security and Privacy*, Springer. http://link.springer.com/book/10.1007/978-3-642-41205-9 , 2013, pp. 127-137.

[2] J. Longstaff and A. Howitt, "TCM2: Supporting dynamic authorization and overrides in Attribute Based Access Control," in *Case Studies in Secure Computing: Achievements and Trends*, B. Issac and N. Israr, Eds., Auerbach Publications, Taylor and Francis. ISBN 9781482207064, 2014.

[3] D. A. Teich and J. Vaughan, "SQL, Hadoop Make a Powerful Data Pair," TechTarget, Newton, MA, 2015.

[4] R. Sandhu, "The Authorization Leap from Rights to Attributes: Maturation or Chaos?," in *SACMAT'12, June 20–22, 2012, Newark, New Jersey.*, 2012.

[5] Hu, Vincent C, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations (Draft)," NIST, 2014.

[6] E. J. Coyne and T. R. Weil, "ABAC and RBAC: Scalable, Flexible, and Auditable Acces Management," *IEEE IT Professional 1520-9202/13,* June 2013.

[7] A. Howitt, "Formal Specification of the Tees Confidentiality Model," PhD thesis, Teesside University, 2008.

[8] ANSI, "Role Based Access Control, ANSI INCITS 359-2012," ANSI, 2012.

[9] D. R. Kuhn, E. J. Coyne and T. R. Weil, "Adding Attributes to Role-Based Access Control," *Computer, vol. 43 no. 6, IEEE.,* 2010.

[10] "B-Method," Available from www.methode-b.com, 2013.

[11] M. Stonebraker and E. Wong, "Access Control in a Relational Database management System by Query Modification," in *ACM 74 Annual Conference*, 1974.

[12] J. J. Longstaff, M. A. Lockyer and A. Howitt, "Functionality and implementation issues for complex authorization models," *Special issue (on Role Based Access Control) of the IEE Proceedings, Software, Vol 153, No 1. ISSN 1462-5970,* 2006.

[13] J. J. Longstaff, M. A. Lockyer and J. Nicholas, "The Tees Confidentiality Model an authorization model for identities and roles," in *Eighth ACM Symposium on Access Control Models and Technologies*, 2003.

[14] BREAK-GLASS (SPC), "Break-glass:An approach to granting emergency access to healthcare systems," BreakWhite paper, joint NEMA/COCIR/JIRA Security and Privacy Committee, 2004.

[15] A. D. Brucker and H. Petritsch, "Extending Access Control Models with Break-glass," in *ACM Symposium on Access Control Models and Technologies*, 2009.

[16] D. R. Kuhn, "Vulnerability Hierarchies in Access Control Configurations," in *4th Symposium on Configuration Analytics and Automation*, 2011.

[17] M. Blaze, J. Feigenbaum and J. Ioannidis, "The KeyNote Trust Management System Version 2," IETF RFC 2704, http://www1.cs.columbia.edu/~angelos/Papers/rfc2704.txt, 1999.

[18] J. e. a. Li, "Attribute-based Signature and its Applications," in *ASIACCS'10*, 2010.

[19] S. e. a. Yu, "Attribute Based Data Sharing with Attribute Revocation," in *ASIACCS'10*, 2010.

[20] J. Huang, D. M. Nicol, R. Bobba and J. H. Huh, "A Framework Integrating Attribute-based Policies into Role-Based Access Control," in *SACMAT'12*, 2012.

[21] S. Oh, R. Sandhu and X. and Zhang, "An Effective Role Administration Model Using Organization Structure," *ACM Transactions on Information and System Security, 9, 2.,* 2006.

[22] J. Park and R. Sandhu, "The UCONABC Usage Control Model," *ACM Transactions on Information and System Security, 7,1.,* 2004.

[23] X. Zhang, F. Parisi-Presicce, R. Sandhu and J. Park, "Formal Model and Policy Specification of Usage Control," *ACM Transactions on Information and System Security, 8, 4.,* 2005.

[24] H. Janicke, A. Cau and H. Zedan, "A note on the formalization of UCON," in *ACM Symposium on Access Control Models and Technologies.*, 2007.