BILINGUAL PUBLISHING GROUP

**Journal of Electronic & Information Systems**

https://journals.bilpubgroup.com/index.php/jeis

**ARTICLE**

# Extending ABAC Authorisation for Complex Applications: Negative Permissions, Hierarchical Attribute Values, Break Glass Override

***Jo Noble*** **, *Jim Longstaff*** *

*School of Computing, Engineering and Digital Technologies, Teesside University, Middlesbrough TS1 3BX, UK*

## ABSTRACT

Much of the commercial and research writings on healthcare Real-Time Location Systems (RTLS) have concentrated on the considerable benefits of this technology for locating assets, patients and staff, with many applications developed, particularly for asset tracking. Less has been written about who is authorised to query and analyse healthcare RTLS data, including historical location data. We categorise use cases for the interactions which have been proposed and indicate further desirable authorisations which could be provided by extending Attribute Based Access Control (ABAC). The ABAC extensions we propose are: to authorise access to information protected at increasing levels of security by invoking break-glass overrides; to enable the denial of access through the use of separately-specified negative permissions, possibly at different levels of security; and supporting hierarchically-structured attribute values for users (e.g., teams and roles), operations and protected objects. We describe algorithms for rule selection and processing, largely at the conceptual level. They take account of architect-specified attribute weightings, the order of rule evaluation, and query modification if the database forms the basis for implementation. We show that our sorting strategy can reduce rule search space by over 90% compared to a naïve policy evaluation, while enabling fine-grained emergency access that current ABAC frameworks lack. We illustrate with an example of technician teams locating and maintaining sophisticated medical equipment in a large healthcare setting. Privacy, confidentiality and security become especially sensitive if technicians need to inspect medical data produced by the equipment, maybe in emergency situations. When fully implemented, our approach does not require additional application programming to provide the extended authorisation functionality described.

*CORRESPONDING AUTHOR:

Jim Longstaff, School of Computing, Engineering and Digital Technologies, Teesside University, Middlesbrough TS1 3BX, UK;
Email: J.J.Longstaff@tees.ac.uk

ARTICLE INFO

Received: 26 February 2026 | Revised: 15 July 2026 | Accepted: 30 July 2026 | Published Online: 11 August 2026
DOI: https://doi.org/10.30564/jeis.v8i2.13211

CITATION

Noble, J., Longstaff, J., 2026. Extending ABAC Authorisation for Complex Applications: Negative Permissions, Hierarchical Attribute Values, Break Glass Override. Journal of Electronic & Information Systems. 8(2): 61–80. DOI: https://doi.org/10.30564/jeis.v8i2.13211

***Keywords:*** Real Time Location System; Attribute Based Access Control; Patient Privacy; Healthcare Information Systems

# 1. Introduction

## 1.1. RTLS in Health Service Computing

Healthcare providers are under increasing pressure to embrace digital transformation to cope with healthcare demands and to improve an aging infrastructure[1]. Harnessing solutions such as the Internet of Things (IoT) within healthcare systems improves quality of patient care and makes financial and efficiency savings[1–3]. Use of connected internet-accessible devices provides the means to collect, store and share data relating to patients' health, experiences and care, and the management of the healthcare facility. One such technological solution is Real-Time Location System (RTLS) based tracking of assets, staff and patients, which provides wide-ranging benefits, especially when integrated with other IT systems.

We begin by summarising the capabilities of some advanced commercially available healthcare RTLS systems and applications:

- Supporting staff and patient safety. Summoning assistance very quickly when faced with physical threats or other emergencies. Alerting patients entering restricted areas, especially relevant to patients with dementia. The capability to automatically lock doors in extreme circumstances such as a suspected baby abduction. Enforcing hand hygiene compliance. Generating contact tracing reports in the event of possible infection from persons or equipment. Quickly locating medical equipment.
- Supporting wayfinding, enabling patients to find their way to clinics and facilities using turn-by-turn directions.
- Supporting analytics to improve workflow. The objectives include reducing wait times, identifying overcrowding, scheduling, and cost reduction by optimising the use of resources.

These capabilities suggest that many kinds of users could profitably use the resulting data. The data generated by RTLS becomes particularly sensitive where knowledge about a patient's condition or their history could be inferred from their locations at given times. The data can also establish whether a patient has been attended by medical staff during critical time periods. For these and other reasons, access to RTLS healthcare data must be carefully controlled by authentication and authorisation. Comparatively little has been written about solutions for the privacy issues that tracking of assets, staff and patients introduces[4–6], aside from cybersecurity solutions to protect the data held from malicious attacks[7–9]. The legal requirements for access to healthcare data, including break-glass emergency access, were addressed in the work of Longstaff[10] and elsewhere. Further reading on potential RTLS applications in healthcare and patient safety can be found in Blackstock, Holt, the BBC article and McCracken[11–14].

## 1.2. Authorisation Systems

An authorisation system, based on a model such as Attribute Based Access Control (ABAC), authorises access to data, applications, and system functionality. The central idea of ABAC is that access is determined based on various attribute values presented by a user. Permissions (often referred to as rules) specify conditions under which access is granted or denied. ABAC is an established access control approach[15] with an OASIS XACML standard[16]. Commercial software is available, e.g., Axiomatics[17], and many research variations have been developed, e.g., Talegaon et al.[18]. Utilising an ABAC architecture enables the decoupling of applications from authorisation polices, with benefits including centralised policy design and reductions in application coding.

## 1.3. Methodology and History

The objective of this paper is to show how an enhanced ABAC scheme can facilitate complex authorisations in large applications, such as RTLS healthcare. The project has its origins in previous projects to model and implement electronic patient records and access controls using object-oriented databases and SQL Server. These were carried out in collaboration with healthcare professionals and reported in conferences and other publications, including an ACM SACMAT

paper. A major project to investigate patients' specifying access control preferences on their medical records was funded by the Information Commisioner's Office. We also carried out a small-scale RTLS project with a local industrial company. Based on the experience and evaluations of these projects, we embarked on the present project to investigate ABAC access control for a large RTLS healthcare application. We firstly conducted extensive literature research into academic and commercial RTLS applications, followed by design, modelling and implementation work. The evaluations of the user aspects of our previous projects directly apply to this project. We have carried out further implementation and testing of the new ABAC rule selection scheme reported in this paper, concluding that substantial performance can be achieved over published schemes.

### 1.4. Contents

The paper is structured as follows:

In Section 2, we define use cases centred on the tracking of assets, staff and patients actioned by a range of NHS personnel, such as technicians, clinicians, and management. These cover use cases which have appeared in the commercial and research literature, and also potentially useful more complex ones. They serve as examples throughout the paper.

Section 3 presents our enhancements to conventional ABAC models and practise. These support authorisation by individuals (subjects), teams and roles, with levels of protection reached by break-glass overrides. The complexity of such functionality has received comparatively little attention in the literature. Authorising functionality for identities/individuals for decentralised access control is an active research area [19]. Also, using separately-specified negative permissions (sometimes called negative authorisation) is often preferred over rule models with complemented conditions, particularly for large applications [20]. Our ABAC extensions are based on an object-oriented model, TCM2 [21], which provides a basis for developing ABAC and other access control models.

In Section 4, we develop the representation of our use case examples in the form of enhanced ABAC (TCM2) permissions. In doing this, we include additional data required for the ABAC/TCM2 permissions processing described in subsequent sections.

Section 5 introduces a formal methods representation of our enhanced ABAC model, which we illustrate with the use case examples.

Section 6 presents the additional processing required to implement our ABAC model extensions. A difficulty previously noted in ABAC is the proliferation of permissions (rules) in industrial-scale applications. We describe an efficient way of permissions (rule) selection which involves assigning preferences to attributes, attribute values, and permissions.

Finally, the remaining sections offer evaluations, reviews of related work, and a conclusion.

## 2. RTLS Use Cases

### 2.1. Use Case Classification

We now give several use cases of the kind that have appeared in healthcare RTLS research and commercial literature. They demonstrate the complex interplay of users, teams and roles which need to be supported, either at the present time or in the future. They may be categorised as follows:

- C1—Querying current location/data for tracked objects.
- C2—Alerting to an emergency.
- C3—Monitoring occurrence of events within time periods.
- C4—Querying and analysing historical location/data for tracked objects.

To illustrate the power of our ABAC approach, we add the following:

- C5—Denying access to tracked object data but permitting access at a higher level of security,

We list the use cases by role, individual user or team (user or team) + role, and indicate the use case category as specified above. C5 involves the use of negative permissions, which is an extension of the usual notion of ABAC. Data about the tracked object and (detected) location is to be queried and returned unless otherwise indicated. In several cases, medical data might be retrieved, sometimes by further break glass queries.

### 2.2. By Role

UC1. As a doctor or nurse, I need to find medical equipment to provide treatment. (C1)

*Locating medical equipment is perhaps the most common of current RTLS healthcare applications.*

UC2. As an administrator or technician, I need to locate equipment. (C1)
*This is concerned with the locating of assets of any type.*

UC3. As an RTLS app, I need to alert hospital staff if a vulnerable patient leaves their ward. (C2)
*There are many stories of tragedies involving mental health patients absconding or wandering into prohibited areas[11].*

UC4. As a member of the medical or security staff, I need emergency access to location data to find a patient who has wandered off. (C1)
*We are treating this as a break-glass (or override) query, which will be subject to additional auditing.*

UC5. As a nurse, I want to be able to summon assistance in an emergency. (C2)
*This is stated as a necessary and important RTLS functionality.*

UC6. As a receptionist, switchboard operator, or administrator, I need to access a patient's location at ward/department level, to respond to queries. (C1)
*Illustrates a general access query returning a zone, which is less accurate than a detected location.*

UC7. As a member of the medical staff, I need to be able to determine how long a patient has been waiting in a consultation room. (C3)
*A time-monitoring application, one of many. E.g., suggesting whether a patient has been attended to by a nurse or doctor.*

UC8. As an RTLS Technician, I wish to access detailed Location Records to identify faults and maintenance requirements. (C4)
*No details of objects (person or asset) associated with tags to be provided.*

### 2.3. By User/Team

UC9. As a member of the Hospital Management Team, I wish to query and analyse historical Location Records to identify inventory needs by zone. (C4)
*It would be unusual in this scenario to authorise access without a role being specified, but this is possible using our ABAC scheme.*

### 2.4. By User/Team and Role

UC10. As a member of the Hospital Management Team with the Administrator role activated, I wish to query and analyse historical location and medical records to identify instances where persons (patients, medical staff, non-medical personnel) have come into contact with an infected person or asset to a specified level of proximity. (C4)
*This has been stated to be an important function of both current and future RTLS systems.*

### 2.5. Complex Authorisations with Negative Permissions

We use the following examples to illustrate the type of functionality which would be useful in large, complex applications. We show in subsequent sections how our ABAC extensions handle these examples.

Any Technician is authorised to locate and maintain Transport Class equipment. Technician Team 1 is to be prevented from carrying out pump maintenance. Technician Team 3 is permitted to locate and perform pump maintenance, but not infusion pump maintenance. However, Bob, a member of Technician Team 3, is permitted to perform infusion pump maintenance, but only on a Level 1 Override. (C5)

## 3. TCM2 Model Basics

### 3.1. Model Overview

In previous publications, we have called our extensions to the ABAC model TCM2. Previous descriptions of our TCM2 model can be found by Longstaff and Noble[21], with

more detail in the work of Longstaff and Howitt [22].

TCM2 supports hierarchically-structured attribute values (called Classifier Values) for users, operations and protected objects; authorising access to information protected at increasing levels of security; and enabling the denial of access through the use of separately-specified negative permissions.

## 3.2. Model Elements

In **Figure 1**, all rectangles represent classes, and rounded rectangles additionally represent classifiers (corresponding to ABAC attributes); the single-headed arrows are one-to-many associations, and the lines are one-to-one associations. The dashed line indicates the potential inclusion of a classifier and its value in a T Permission.

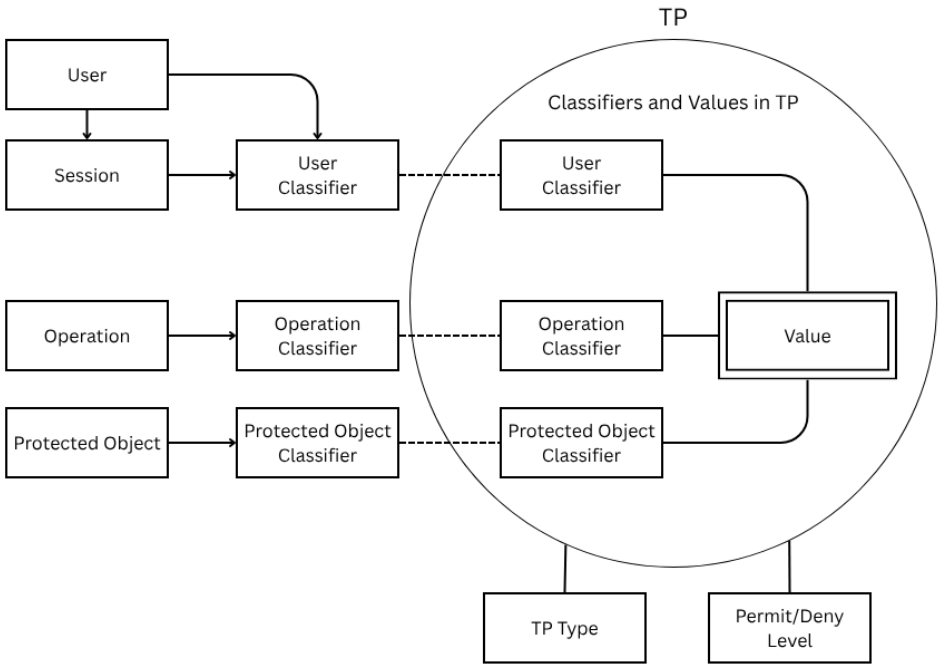


**Figure 1.** TCM2 Model.

The following are depicted in **Figure 1**:

**User**—Users are teams, persons or applications. Examples are RTLS Technician, Hospital Security Officer, RTLS_App_CalcWait.

**Session**—Denotes an individual User Session and is associated with Active Session User Classifiers and their classifier values.

**Protected Object (PO)**—This can be any object which is subjected to authorisation controls, e.g., a data model object, an entire database, an application. It is accessed and used by the operations it provides. In this paper, the protected object is Location_Data, which is an aggregation of other classes. A component of Location_Data, e.g., Tracked Object can also be a Protected Object.

**Operation**—This can range from a read or update operation defined for a Protected Object (class), to executing an application. Examples are Read (Location_record), Read_Zone2level (Location_record), Execute_RTLS_App_CalcWait.

**Classifier**—A Classifier describes an authorisation model object. In their simplest form, classifiers take the form of attributes in ABAC. Examples of classifiers are User_id, UserName, UserRole, PO_id, PO_Type, PO_Location. A security architect must define the importance of classifiers regarding authorisation, for example, User_id is preferred over User_Role when determining access to a Protected Object. This importance level is held in the variable cfiesq, Section 5.2.

**Classifier Value (cvalue)**—Classifier values can be structured in hierarchies pictured as inverted trees. For example, the classifier value <UserRole, Doctor> could have daughter roles <UserRole, GP>, <UserRole, Opthamolo-

gist>, <UserRole, Psychiatrist>, etc. Classifier Value Hierarchies provide additional structure representing relative importance for determining authorisation outcomes. They often correspond to class diagram structures for the application, but not necessarily. The variable posin_CVH is used to hold the position of a classifier value in the tree. Classifier values which are not hierarchically structured can be present.

**T Permission (TP)**—This corresponds to an ABAC permission or rule. It is a set of classifier values, mapped onto a permit or deny value, and a Permit/Deny Level (see Section 3.3 below). A TP states that a user is permitted/denied to execute an operation on a protected object (often a defined set of data). TP examples are given in Sections 4.3–4.4 below.

### 3.3. Deny/Permit Levels

TPs can be specified at increasing levels of power, which are called Deny and Permit Levels (DP Levels). A transaction must have a DP Level specified when it is executed—the default level is Level 1 (L1). The qualifying TPs would include the Deny TPs and Permit TPs for that level and lower levels, the Deny TPs of higher levels, but not the Permit TPs of higher levels. A Deny TP operates at its own level and all lower levels. Therefore, data could be denied to users at Level 1 who might be able to access it at Level 2 by overriding (if so authorised); more sensitive data might be only available to more senior users who were authorised to override to higher levels.

### 3.4. T Permissions Consistency

Authorisation requires a consistent set of TPs. During policy development, the TCM2 system will check each new TP with the already-validated TP set and will alert the security architect to any discrepancies for resolution before the TP is accepted. A consistent TP set will have the following properties:

- No totally identical TPs exist, within the same or differing DP Levels.
- A Deny TP and a Permit TP which are otherwise identical cannot exist in the same DP Level.
- A Deny TP can exist with an otherwise identical Permit TP existing at a higher DP Level. (An authorised user could therefore override access to the more-restricted information.)

Messages can be attached to TPs for display when the TP is matched during transaction execution. Messages can give more information, directions, and advice to override access to sensitive information at a higher level.

## 4. TCM2 RTLS Scenario

### 4.1. Scope

We use this scenario to illustrate authorisation by role, individual user and team. Role structures with inheritance are used in large complex organisations to facilitate access to functionality and data. Additionally authorisations might be defined for individual identities, maybe by creating new roles, resulting in role proliferation. Individuals may be assigned to teams, where the teams have their own hierarchical structures. Members of teams may be at different stages of training/competence, which may affect the functionality they can activate. Additionally, there may be a team leader and deputy leader. There might be specific authorisations for a team assigned to a particular department. Therefore, there can, in general, be a very rich interplay of authorisations, requiring a well-defined approach for TP searching and selection.

We assume all technicians have authorisation to alert medical staff if they discover malfunctions in pumps. They can examine anonymised patient readings to discover errors.

### 4.2. Classifier Values for RTLS Scenario

The following classifier values are used in the examples given in this paper.

#### 4.2.1. Classifier Values for User/Team

Firstly, the User/Team classifier has an importance rating of 1 (this is its value in the classifier sequence (cfiersq) featured in Section 5.2) (**Figure 2**).

Here its classifier values are (unique) names, suitable for expository purposes. They are shown with their position in the Classifier Value Hierarchy (posin_CVH, Section 3.2).

#### 4.2.2. Classifier Values for UserRole

The UserRole classifier has an importance rating (cfiersq) of 2 (**Figure 3**).

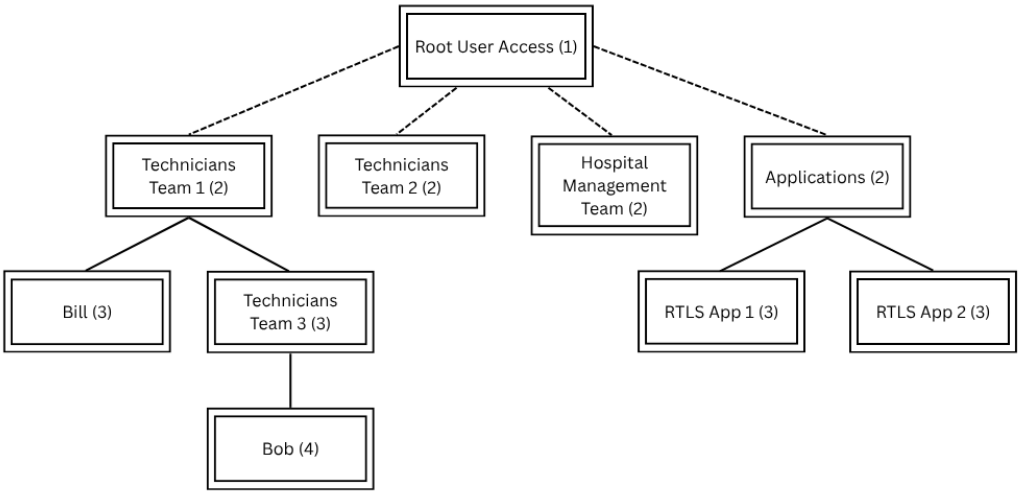


**Figure 2.** User/team classifier value hierarchy showing importance ratings.

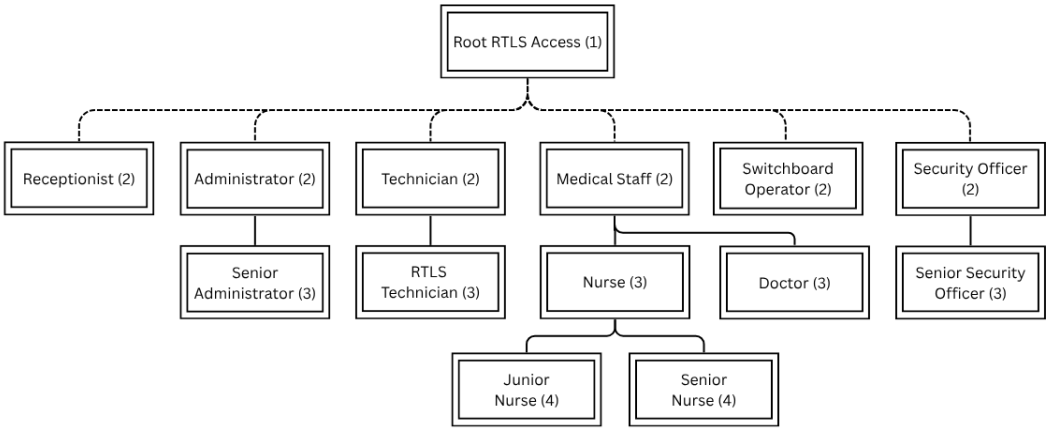


**Figure 3.** UserRole classifier value hierarchy showing importance ratings.

### 4.2.3. Classifier Values for PO with Location

This is a partial model indicating the required data for location tracking in healthcare facilities.

For authorisation purposes, they are regarded as a PO with the classifier PO_Type and classifier values as shown. Authorising access to Location_Data protected objects also provides access to the other protected objects shown (**Figure 4**).

**Location Data**—The aggregated data for location tracking, which needs to be authorised for access in most cases.

**Location**—The actual detected location, which is the most precise location identifiable.

**Zone**—Zones are hierarchically structured. Larger zones can contain smaller ones. A location may be associated with a single zone, which itself may be contained within a zone hierarchy.

**Location_record**—A Location_record is created when a movement of a tracked object is detected. Attributes include the precise Location, Date/Time, Tag_id, for an object being tracked. The time interval between creation of Location_records can be set as a RTLS parameter. When near real-time data is required, this interval will be short.

**Tracked Object**—This contains Tag_id and other tag data, which is inherited by its specialisations.

**Patient**—Data provided from patient records. Just the data necessary for location tracking applications. This might include details of mobility, vulnerability, a brief statement of current medical condition, and urgent medications. This and any further data might require privileged access facilitated by break glass override permissions.

**Staff**—Specialisations include Healthcare Profession-

als, managers and administrators, and technical staff. Just the data needed for location tracking is included. Attributes include Role (hierarchically structured). Some staff may be users, as defined for the TCM2 model.

**Asset**—Specialisations for the Equipment Class are shown.

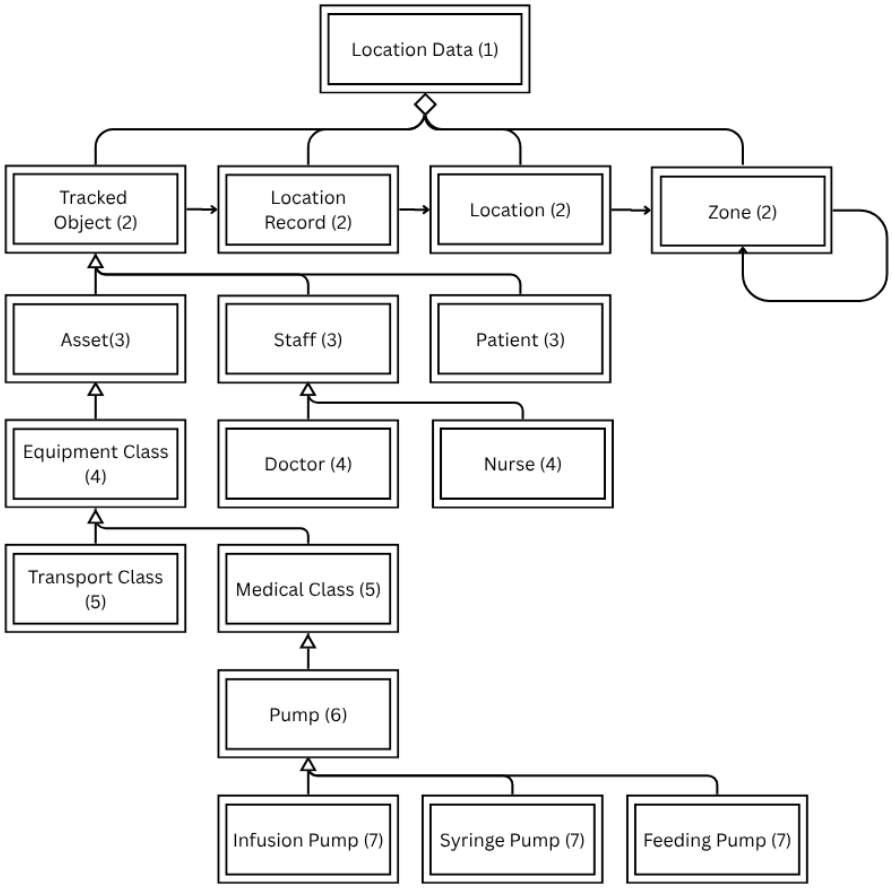


**Figure 4.** PO_Type classifier values showing importance ratings.

## 4.3. TPs for Scenario

We use a notation which is suitable for explanatory purposes, representing database queries on the data models presented above.

Most of the operations are applications which access Location_Data, and were developed by security architects using dialogues (which we do not show).

The following TPs are specified for the set of hospital users and the Location_Data object and its defined operations. A component of a TP (a classifier value) signifies a selection on that component. Projections on POs indicate data which is to be returned to the operation.

**UC1.** Doctor, Nurse Finding Medical Equipment.

TP1 Permit_TP (L1):

{<UserRole, Medical_Staff>,
<UserLocation, Hospital_Zone>,
<Op_id, Find_Equipment ()>,
<PO_Type, Medical_Equipment>,
PROJECT Medical_Equipment,
PROJECT Location}

This permits Medical Staff (and all its subclassifer values, e.g., Doctor, Nurse) to use the Find_Equipment application for Medical_Equipment (and subclassifier values) at security Level 1.

T Permissions matching and firing involves determining whether a TP matches active session classifier values. So if the following active classifier values describe the transaction:

(<UserName, Jill>, <UserRole, Nurse>, UserLocation, Hos-

pital_Zone_3>, <Time, 21:32>, <Op_id, FindEquipment ()>, <PO_Type, Medical_Equipment>)

Then the previous TP would grant access to this data, because for every classifier in the TP there is at least one classifier value common to the TP and the active session values. (Paraphrase of an expression from the TCM2 formal specification), Note that the cvalue Medical Staff is an ancestor of Nurse in the classifier value hierarchy for UserRole.

**UC3.** As an RTLS app, I need to alert Hospital Staff if a vulnerable patient leaves their ward.

TP3 Permit_AlertTP (L1):
{<User_id, MonitorAlertApp>,
<Op_id, MonitorAlert ()>,
<PO_Type, Patient>,
PROJECT Location,
PROJECT Patient}

**UC4.** Any member of hospital staff to find a vulnerable patient in an emergency (break glass override).

TP4 Permit_TP (L1_Ovr):
{<UserRole, HospitalStaff>,
<UserLocation, Hospital_Zone>,
<Op_id, AlertFindVulnerablePatient (PatientName or tag_id)>,
<PO_Type, Patient>,
PROJECT Location,
PROJECT Patient}

**UC5.** Nurse summoning emergency assistance.

TP5 Permit_AlertTP (L1):
{<UserRole, Nurse>,
<UserLocation, Hospital_Zone>,
<Op_id, NurseAlert (tag_id)>,
<PO_Type, Nurse>,
PROJECT Location,
PROJECT Nurse}

**UC8.** RTLS Technician accessing Location_records to identify faults and maintenance requirements.

TP8 Permit_TP (L1):
{<UserRole, RTLS_Technician>,
<UserLocation, Hospital_Zone>,
<Op_id, R>,
<PO_Type Asset>,
PROJECT Location}

**UC9.** Hospital Management Team member analysing historical Location Records to identify inventory needs by zone.

TP9 Permit_TP (L1):
{<User_id, Hospital_Management_Team>,
<Op_id, R>,
<PO_Type Location_Data>,
PROJECT Location}

**UC10.** A hospital manager/Administrator (Bill) with the Administrator role analysing historical Location_Records to identify possible instances of infection.

TP10 Permit_TP (L1)
{<User_id, Bill>,
<UserRole, Administrator>,
<UserLocation, Hospital_Zone>,
<Op_id, IdentifyContacts (Infected_Person_tag_id)>
<PO_Type, Location_Data>,
PROJECT Person,
PROJECT Location}

## 4.4. TPs to Illustrate Processing

Again, referring to the model diagrams, we express the following use cases.

Any Technician is authorised to locate and maintain Transport Class equipment. Technician Team 1 is to be prevented from carrying out pump maintenance. Technician Team 3 is permitted to locate and perform pump maintenance, but not infusion pump maintenance. However, Bob, a member of Technician Team 3, is permitted to perform infusion pump maintenance, but only on a Level 1 Override.

The T Permissions are:

TP12 Permit_TP (L1)
{<UserRole, Technician>,
<UserLocation, Hospital_Zone>,
<Op_id, LocateMaintain ()>,
<PO_Type, Transport_Class>,
PROJECT Location,
PROJECT Transport_Class}

TP13 Deny_TP (L1)
{<User_id, Technician_Team_1>,
 <UserRole, Technician>,
 <UserLocation, Hospital_Zone>,
 <Op_id, LocateMaintain ()>,
 <PO_Type, Pump>}

TP14 Permit_TP (L1)
{<User_id, Team_3>,
 <UserRole, Technician>,
 <UserLocation, Hospital_Zone>,
 <Op_id, LocateMaintain ()>,
 <PO_Type, Pump>,
PROJECT Location,
PROJECT Pump}

TP15 Deny_TP (L1)
{<User_id, Team_3>,
 <UserRole, Technician>,
 <UserLocation, Hospital_Zone>,
 <Op_id, LocateMaintain ()>,
 <PO_Type, Infusion_Pump>}

TP16 Permit_TP (L2 (L1_Ovr))
{<User_id, Bob>,
 <UserRole, Technician>,
 <UserLocation, Hospital_Zone>,
 <Op_id, LocateMaintain ()>,
 <PO_Type, Infusion_Pump>,
PROJECT Location,
PROJECT Infusion_Pump}

### 4.5. Transactions to Illustrate Processing

The following transactions are used to illustrate T Permissions processing in the following sections.

T1. Bob, a member of Technician Team 3, searches for the Infusion Pump IP36 he has been asked to inspect. (The data about the pump and its location is not returned.)

T2. Bob performs the same query using L1 override to L2. This time the data is returned.

For both these transactions, the active classifier values are:

(<UserName, Bob>, <UserRole, Technician>, UserLocation, Hospital_Zone>, <Time, 21:32>, <Op_id, LocateMaintain ()>, <POid, IP36>, <PO_Type, Infusion_Pump>)

T3. A member of Team 3 (not Bob) located in the Hospital_Zone requests details of pumps, some of which may be infusion pumps and some of other types:

(<UserName, Team_3>, <UserRole, Technician>, UserLocation, Hospital_Zone>, <Time, 21:32>, <Op_id, LocateMaintain ()>, <PO_Type, Pump>)

## 5. T Permissions Processing Principles

In this section, we summarize further aspects of the TCM2 formal model. Longstaff and Howitt[22] define more advanced forms of classifiers and classifier value matching than are covered in this paper, including range matching.

TPs processing depends on two main principles: TP Match, and Nearer Match, which we describe below.

### 5.1. TP Match

Firstly, a T Permission will match (i.e., qualify to potentially authorise a transaction) if all its classifier values are contained in the active classifier values defining the transaction. This can be expressed formally using the B specification language notation by the following definition:

TP_Matched(tp, acvals) $\hat{=}$
bool(dom(acvals ∩ ad[tp]) = dom(tp))
where:
acvals is the transaction active classifier values (classifier values specifying the transaction, example given in Section 4.5), and tp is a validated T Permission.

The set ad[tp] contains the original ancestor classifier values as well as the set of all descendant classifier values. Its definition is: ad ∈ cvalues × cvalues, with the constraints that any pair (cvalue1, cvalue2) would have the same classifier, and cvalue1 is the ancestor of cvalue2. An example was given in Section 3.2.

This definition includes all derived TPs—these are TPs not explicitly appearing in the TPs set, but which can be generated by inserting classifier values lower down in the appropriate value hierarchies.

Therefore, 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.

In the examples in **Table 1**, both TP15 and TP16 would qualify to potentially authorise transaction T1 (see Section 4.5 for more information about transaction T1).

**Table 1.** Example of TP Match.

| Examples | |
|---|---|
| T1 Bob, member of Technician Team3, searches for Infusion Pump IP36<br><br>U(<ser_id, Bob>, <UserRole,Technician><br><UserLocation, Hospital_Zone_3>,<br><Time,21:32>, <Op_id, LocateMaintain ()>,<br><POid, IP36>, <PO_Type, Infusion_Pump>) | TP15 Deny_TP (L1)<br><br>{<User_id, Team_3>,<br><UserRole, Technician>,<br><UserLocation, Hospital_Zone>,<br><Op_id, LocateMaintain ()>,<br><PO_Type, Infusion_Pump>} |
| TP16 Permit_TP (L1_ovr)<br><br>{<User_id, Bob>,<br><UserRole, Technician>,<br><UserLocation, Hospital_Zone>,<br><Op_id, LocateMaintain ()>,<br><PO_Type, Infusion_Pump>,<br>PROJECT Location PROJECT Infusion_Pump} | dom(TP16) = {User_id, UserRole, UserLocation, Op_id, PO_Type}<br><br>dom(T1 ∩ TP16) = {User_id, UserRole, UserLocation, Op_id, PO_Type}<br><br>These are identical, therefore, TP16 matches. Same result for TP15, because <User_id, Team_3>,<User_id, Bob> is in ad. |

## 5.2. Nearer Match TP

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

There is an ordering cfiersq on the classifiers that is set by the security architect in consultation with the users, and is a mapping of the set of integers 1,2,3,4… to the set of classifiers.

cfiersq ∈ iseq (cfiers)

cfiersq = (1..User_id, 2..UserRole, 3..UserLocation, 4..Op_id, 5..PO_Asset_id, 6..PO_Type, 7..Time)

For two TPs in our examples:

$CFIER_L$(TP16) = User_id
$NCFIER_L$(TP16) = 1
$VCFIIER_L$(TP16) = <User_id, Bob>

$CFIER_L$(TP15) = User_id
$NCFIER_L$(TP15) = 1
$VCFIIER_L$(TP15) = <User_id, Team3>

Therefore, given a set of matched TPs tps, the (set of) nearest match(es) is determined as follows.

To compare two TPs tp1, tp2 to find the nearest match (from the set of all TPs which match the active classifier values), we first look at $NCFIER_L$(tp1) and $NCFIER_L$(tp2).

If these are not the same, then the most important classifier, i.e., the lowest number, takes priority. If they are the same, then we consider $VCFIER_L$ (tp1) and $VCFIER_L$ (tp2).

If these are not the same, then we determine if they are related through the ancestor/descendant relationship, and if so, the descendant value takes priority.

If they are the same, then the number and value of the next most important classifier are compared.

Therefore:

We also require the posin_CVH values for the relative importance of determining authorisation outcomes. These are determined by the security architect.

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.

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

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

$$NCFIER_L(cvs) = min(cfiersq^{-1}[dom(cvs)])$$

$$VCFIER_L(cvs) = cvs[CFIER_L(cvs)]$$

$$\text{NearerMatch(tps)} \triangleq \{\text{nmtp} \mid \text{nmtp} \in \text{tps} \wedge \neg\exists \text{tp}.(\text{tp} \in \text{tps} \wedge (\text{NCFIER}_L(\text{nmtp}) > \text{NCFIER}_L(\text{tp}) \vee \text{VCFIER}_L(\text{nmtp}) \mapsto \text{VCFIER}_L(\text{tp}) \in \text{ad}))\}$$

where ad is the ancestor/descendant relationship.

TP16 is a Nearer Match than TP15 because
$\text{NCFIER}_L(\text{TP16}) = \text{NCFIER}_L(\text{TP15})$
and the pair
$(\text{VCFIER}_L(\text{TP16}), \text{VCFIER}_L(\text{TP15}))$
is not contained in ad.

# 6. T Permissions Processing Algorithms

## 6.1. Overview

The algorithms would be used at the early stages of authorisation determination, and are largely described at the conceptual level. Their potential use with schemes described in the literature are examined later. Note that in the following, overrides are authorised by TPs in exactly the same way as other overrides. An override will only succeed if an exact match with an override Permit TP occurs. In an operational application, both attempted overrides (where an override button is supported and used) and successful overrides (where an override T Permission has fired) are recorded for audit purposes. The existence of an override TP can be used to generate a message indicating the existence of more sensitive data, and providing an override button and advising on its use[10]. Time constraints on access are implemented using classifiers, and attempted violations are recorded and analysed, as is any override activity.

## 6.2. Sort T Permissions

The TPs in the policy set undergo at least a two-stage sorting process, and are then maintained in the resulting order. The first stage sorts the TPs according to their Most important Classifier value ratings NCFIER, given below. This order is consistent with the cfiersq sequence and will almost universally start with User_id and UserRole. This sort stage produces a list in which the $\text{NCFIER}_L$ comparisons in Section 5.2 have been effected. The second stage evaluates the $\text{VCFIER}_L$ comparison, just for classifiers appearing more than once in the first stage list. Following this, we now have a list in which the most restrictive TPs appear at the beginning. Note that this process will not absolutely guarantee a unique ordering of TPs according to restrictiveness because it is possible that two TPs might have the same $\text{NCFIER}_L$ and $\text{VCFIER}_L$ values. This is unlikely to occur when working with the most important classifiers, such as User or UserRole, which have low cfiersq ratings. Therefore, in most cases, a two-stage sort is sufficient for reducing the search space for later matching. Should a more precise list be required, the further stages described in Section 5.2 can be followed. Matching T Permissions can now be identified quickly by searching the list.

The TP Sort Criteria for our example are:

User_id/posin_CVH
UserRole/posin_CVH
PO_id/posin_CVH
PO_Type/posin_CVH
TP_Type
Permit or Deny
TP_DP_Level

The sorted T Permission list for our example below is therefore:

TP12 Permit_TP (L1)
TP13 Deny_TP (L1)
TP14 Permit_TP (L1)
TP15 Deny_TP (L1)
TP16 Permit_TP (L1_ovr)

## 6.3. Matched T Permission Lists

This is carried out for individual classifiers, and may involve searching classifier value hierarchies. An example might be matching a transaction with a <UserRole, Senior_Nurse> classifier value, which would be authorised by a TP with an ancestor value of <UserRole,Medical_Staff>. This permission would need to be found by searching the

UserRole classifier value hierarchy with <UserRole,Senior_Nurse>. The search algorithms can utilise the strict sort order of the TP set and also the position of classifer values in the hierarchy (posin_CVH).

| **T1. L1 (tp_sort_list)** | **T2. L2 (tp_sort_list)** | **T3. L1 (tp_sort_list)** |
|---|---|---|
| TP14 Permit_TP (L1) | TP14 Permit_TP (L1) | TP14 Permit_TP (L1) |
| TP15 Deny_TP (L1) | TP15 Deny_TP (L1) | TP15 Deny_TP (L1) |
| | TP16 Permit_TP (L1_ovr) | |

The remaining classifier values (for operations and POs) are not hierarchically structured in these examples, and therefore would not alter the sort order.

## 6.4. Handling Override T Permissions

The sorted TP list might contain TPs which operate at different DP Levels. Depending on the DP Level at which the transaction is to be executed, certain TPs will be discarded during the search, according to the following criteria.

L1 Processing:

Leave L1 Permits. Remove L2 and higher Permits. Leave Denys at all levels.

L2 Processing:

Leave L1 and L2 Permits. Remove L3 and higher Permits. Leave all Denys.

L3 Processing:

Leave L1, L2 and L3 Permits. Remove L4 and higher Permits. Leave all Denys.

(Repeat for higher levels).

## 6.5. T Permissions Searching

Search TPs from most restrictive to most permissive until a match is found. This TP, together with any permissions below it, determines the authorisation outcome. The search process terminates when a TP matching the query target (not necessarily the PO) is found; otherwise, it continues, reaching higher nodes in relevant hierarchies. Note that the classifier precedence is preserved in the result of the search. The matching TPs are indicated below for our examples.

| **T1. L1 (tp_auth_list)** | **T2. L2 (tp_auth_list)** | **T3.L1 (tp_auth_list)** |
|---|---|---|
| TP15 Deny_TP (L1) | TP16 Permit_TP (L1_ovr) | TP14 Permit_TP(L1) |
| | | TP15 Deny_TP(L1) |

For Transaction T3, the last most restrictive TP will be a Deny TP (for infusion pumps) and the prior matched TP a Permit TP (for all the pumps). Then the authorisation will consist of evaluating this list of two TPs.

## 6.6. Database Implementation

Considering a centralised authorisation system, there are two ways of implementing authorisation enforcement. The first is to augment a query language representation of the transaction with additional restrictions derived from the matched T Permissions list—this was covered in the study of Longstaff and Noble[21]. The second is to insert authorisation restrictions into the query execution plan(s) produced by the data storage system, at an early stage of optimisation. Further optimisations, such as choosing indexes and access procedures, would follow. For often-run transactions, pre-optimised and pre-compiled execution plans could be used.

The SQL representations are shown as follows.

**Transaction T2 Data Retrieval**

```
SELECT Location FROM Tracked_Object
WHERE ID = 'IP6'
```

**Transaction T3 Data Retrieval**

```
SELECT Location FROM Pump
WHERE PO_TYPE ! = 'Infusion_Pump'
```

## 6.7. Further Implementation and Scalability

The sorted T Permissions list lends itself to large applications and scaling due to its fixed and predictable structure. T Permission processing takes place at the first stage of authorisation, and can be subjected to local implementations and optimisations. It can be applied within highly distributed networked environments; many applications now distribute

functionality and processing, even across countries or continents. Depending on the circumstances, ordering TPs in the way described in Section 6.2 could enable standadised distribution of TPs and processing to facilities where they would exclusively be used. Should more global authorisation be required, this could be achieved by combining TP sequences. A further possibility would be to maintain separate TP sets for users, teams and roles, which could be matched in parallel; this would depend on the nature of the application and an analysis of the benefits which would result.

Generally speaking, distributed database technology provides extensive functionality for implementing ABAC in large, distributed applications. This includes distributed query processing and optimisation, transaction management, and fault tolerance. Especially for providing and maintaining classifier value (or attribute value) provision across nodes in a network.

# 7. Alternative Processing Strategy Compared

Reducing the number of comparisons between attribute-value pairs in access requests and ABAC rules during rule matching is stated as an important implementation issue [23,24]. However, very little research has been reported in this area [23,25]. The PolTree ABAC system [23] rearranges the rule order in increasing relevance so that the rule which satisfies most requests is tested first, etc. Then it rearranges the order of attribute-value pairs in the rules so that the least frequently used attribute value pair is compared first, leading to early rejection of rules not satisfying the access request. This is based on the assumption that the access requests are uniform. Two tree structures for storing ABAC rules are presented to facilitate fast searching against access requests. Note there is no provision for hierarchical attribute value structuring and separately-specified negative permissions, which as we have shown requires maybe several TPs to be evaluated in a defined order.

We now offer a comparison between early most-frequently-used (Poltree type) rule selection and TP matching using the models in Section 4, which have three classifiers with hierarchically-structured classifier values. We only consider rule/TP ordering and assume that subsequent optimisations can be used with both schemes.

To do this, we need to approximate the corresponding number of TPs with hierarchical structuring to a flattened ABAC rule set. This is given by

$$\text{No of TPs} = \frac{\text{No of ABAC Rules}}{E[H(\text{User\_id})] \times E[(\text{UserRole})] \times E[(\text{PO\_Type})] \times M \times U}$$

Where:

E[H(Att)] Expected average coverage of leaf values for the attribute, taking account of all levels in the hierarchy.

M The mergeable fraction (often 0.3–0.7), i.e., the number of rules capable of being merged. All non-hierarchical attributes must be the same for merging to take place. Large-domain atomic attributes can completely negate hierarchy-based reduction.

U The utilisation factor: Designers don't always generalise maximally (often 0.5–0.8).

This is a reduction approximation consistent with concepts in ABAC, role mining, and combinatorial factorisation. It is an abstraction based on the Generalisation and Merging algorithms presented in Xu and Stoller's work [26]. We assume the attributes are independent, enabling multiplication of hierarchical attribute reduction estimates. It is accurate enough for our hypothetical scenario.

Considering the example models and transactions in Section 4, we make the following assumptions for a hypothetical full-scale implementation of the concepts in this paper (**Table 2**).

**Table 2.** Scenario characteristics.

| Characteristic | Value | Rationale/Assumption |
|---|---|---|
| No. of ABAC Rules in policy, with no hierarchical structuring. | 1,000 | Realistic for a large healthcare facility. |
| Equivalent number of TPs, with Team/User_id, UserRole, PO_Type hierarchically structured. | 100 | Reduction approximation with E[H(User_id)] = 2(shallow hierarchy)<br>E[(UserRole)] = 3(moderate hierarchy)<br>E[(PO_Type)]= 5(deep hierarchy)<br>M = 0.5(half of rules mergeable)<br>U = 0.7(corresponds to practice) |

**Table 2.** *Cont.*

| Characteristic | Value | Rationale/Assumption |
|---|---|---|
| Average no. of Authorisations by UserRole. | 70% | Many authorisations by medical and administrative staff are expected to be by role. |
| Average no. of authorisations by User/Team. | 20% | Medical and healthcare staff often work in teams. Specialised staff may have individual authorisations. |
| Average no of User/Team + UserRole Authorisations. | 10% | Specialist authorisations for team members who have activated a specific role. |

The numbers of rules and equivalent TPs seem to be intuitively sensible for the models in Section 4; other approximations can be derived by varying the input parameters to the reduction formula. Let us denote an ABAC Rule set ordered and searched by most-frequently-used as RuleF. We now estimate the number of RuleF/T Permission comparisons needed for comparing the example transactions and permissions base (Section 4), leading to a matched RuleF rule or TP.

Regarding TP hierarchical classifier value matching, this can be implemented efficiently utilising the strict TP multi-level sort order (Section 6.2), and classifier value levels within hierarchies: the number of transaction classifier value and TP classifier value comparisons can be substantially reduced. These techniques could not be used with the most-frequently-used ordering.

In **Table 3**, Transactions T1–T3 have no direct authorising mechanism in standard ABAC and would need to be implemented by applications accessing the relevant rule(s).

**Table 3.** Estimated performance comparison.

| Transaction | Transaction Authorisation Criteria | No. of RuleF Rules Compared (Linear Search on 1,000 Rules) | No. of TPs Compared (Enhanced Search on 100 TPs) | Comments |
|---|---|---|---|---|
| UC1, TP1 | UserRole | 50 | 3 | General query, accessing one frequently used rule. |
| UC9, TP9 | User/Team | 700 | 10 | Specialist user activity, just one of several transactions accessing a relatively rarely-used rule. |
| UC10, TP10 | User/Team + UserRole | 900 | 10 | Highly specialised analytical access. One transaction of several accessing rarely-used rules. |
| T1, TP15 | User/Team + UserRole, with negative TPs | 800(?) + additional processing | 10 | Additional processing costs for RuleF processing. Deny TP reached. |
| T2, TP16 | User/Team + UserRole, with negative TPs | 800(?) + additional processing | 12 | Additional processing costs for RuleF processing. Override TP reached. |
| T3, TP14, TP15 | User/Team + UserRole with negative TPs | 800(?) + additional processing | 13 | Additional processing costs for RuleF processing. Permit TP and Deny TP reached. |

A further ABAC rule selection scheme determines the order of matching attributes based on the number of attribute values[24]. All attributes appearing in rules are assigned binary identifiers, as are attributes in incoming requests. Rules satisfying a request are determined without using string matching by comparing attribute identifiers. Then a decision tree structure is created using attribute values from the request, and is used to identify matching rules. The number of comparisons is minimised by selecting paths referencing attributes with decreasing numbers of values. Again, there is no accommodation for negative permissions, hierarchical attribute value structuring, and levels of break-glass security.

# 8. Evaluation

The ICO funded a project to explore the setting of restrictions on accessing medical records by professionals, which was based on an implementation of the concepts described in this paper, including break glass emergency access[10]; the legal context was extensively addressed. (Mulrine et al.[27] give details of a related project).

The following concepts were directly evaluated in this project:

Hierarchically-structured classifier (i.e., attribute) values. A hierarchy of NHS roles understandable by the general public was implemented, and also a team hierarchy to represent healthcare professionals working in teams. Extensive

explanations and help functionality were provided. The hierarchical role/team structuring was able to be used by focus group members. Work since has concentrated on efficient searching of TPs based on the ordering described in Section 6.2.

The cases for authorising access for teams, and emergency access were established. A different type of break glass access was identified—one in which opinions, requirements and additional information were to be read prior to invoking access; this was implemented by associating messages with TPs prior to activating TPs granting override. An override button was not displayed if the user did not match a Permit TP, thereby preventing the user from even knowing that higher security level data existed. We were able to demonstrate how TCM2/ABAC could support the privacy and confidentiality aspirations of focus groups consisting of members of the public. This work was carried out in collaboration with Newcastle University Medical School, with many discussions with practising healthcare professionals.

In the presence of architect-specified classifier ordering and hierarchical classifier values, the conceptual TPs matching approach described in Section 6 is optimal in that it reduces the search space for TPs to a minimum. It is difficult to compare performance with other implementations, as no other implementation we are aware of includes hierarchical structures, negative TPs, and overrides to increasing levels of protection. For a large range of user interactions in our earlier project[10], we determined that our sorted TPs searching scheme offered in excess of 90% saving when compared with searching the full TP policy set. Implementation additionally requires further careful choices of data structures and indexes.

# 9. Related Work

An authorisation system is said to implement a particular authorisation model: a comprehensive definition is given in NIST Information Technology Laboratory Computer Security Resource Center[28]. To date, the most widely used authorisation model has been Role Based Access Control, or RBAC[29], 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 an important way forward for authorisation model development[15]. It is regarded as one of the most generalized, scalable, and reliable access control models[30] with many applications. XACML is an established ABAC standard[16], has many applications, and forms the basis of ABAC middleware systems, e.g., Axiomatics[17]. XACML defines an architecture which includes Policy Enforcement Points (PEPs) and Policy Decision Points (PDPs), which have their equivalents in other ABAC architectures[31]. Authorising functionality for identities/individuals for decentralised access control is an active research area[19]. Also, using separately-specified negative permissions (sometimes called negative authorisation) is often preferred over rule models with complemented conditions, particularly for large applications[20].

Environment conditions such as time of day and location are an important feature in ABAC models. A comprehensive description of ABAC is given in NIST SP 800-162[32]. An implementation of ABAC for an object-oriented database is described by Gupta et al.[33]; this includes the use of Attribute Authorities to securely provide current attribute values.

An industrial scale implementation of ABAC involves attribute engineering, attribute assignment to the user and objects, policy engineering, etc.[34]. There are potential difficulties for permissions review/risk exposure—potentially large numbers of rules, and their processing, must be considered. Huang et al.[35] 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 objects, in addition to users; this separation facilitates permissions review.

A related area of security is Attribute Based Encryption (ABE)[36], in which users encrypt and decrypt data based on user attributes. The ciphertext-policy attribute-based encryption (CP-ABE) is the most widely used version. Data is encrypted with an ABE public key with a policy in terms of attributes. Users are issued with a secret key containing their attribute, which must match the policy to enable decryption. The ASCLEPIOS project[37] combines ABE and ABAC in a medical records application. Firstly, an ABAC component produces a permit/deny decision, and in the case of permit, an ABE policy is evaluated before recovering the resource's

symmetric decryption key.

A further area to which ABAC has been applied is to control access to information stored securely in distributed Blockchain ledgers. ABAC permissions authorise access to the transactions, which are immutable and processed using cryptographic and consensus mechanisms. These can involve high overhead costs. A comprehensive survey, with references to many recent projects, is given by Ullah et al.[38]. A further study by Wu et al.[39].

Recently, Relationship-Based Access Control or ReBAC[40], which defines access decisions based on the relationships between subjects or objects, has found applications in social networks. Graph systems implement hierarchies and inheritance. These systems naturally support hierarchical and inherited relationships. Inheritance is modelled explicitly via graph traversal. An example is Google Zanzibar[41].

The OASIS standard (XACML) and its associated developer-friendly language ALFA do not natively include hierarchical structures with inheritance like traditional RBAC systems do. The objective of Hierarchical ABAC models is to naturally represent structure, reduce rule sets and facilitate explanations. These models have used hierarchical attributes with inheritance, e.g., sensor → temperature sensor → industrial sensor[42,43]. We have used hierarchical object structures in a different but similar way to indicate relative strengths for the PO_Type classifier values in determining authorisation during searching sorted lists of T Permissions.

Hierarchical ABAC models have been achieved through extensions, hybrids (RBAC + ABAC), and graph-based models. Within RBAC+ABAC hybrid systems, roles exist and can be hierarchical (RBAC-style). ABAC policies use role attributes and Inheritance happens externally or before ABAC evaluation. Hierarchical processing is not part of the ABAC system. An example is the NIST ABAC model extensions[44].

Some ABAC engines allow dynamic attribute derivation, which enables inheritance-like behaviour. Axiomatics (XACML-based) requires explicit modelling. "Derived attributes" are implemented through Policy Information Points (PIPs) that dynamically retrieve, combine, or compute attributes from multiple sources during policy evaluation. This system also supports complemented (negative) conditions within rules, which provide similar functionality to our negative permissions[44].

The application of AI to access control is becoming increasingly prominent. It is being used to derive ABAC policies from access logs, RBAC roles, natural language documents and system configurations[45,46]. Additionally, to detect inconsistencies, identify unreachable rules, predict policy conflicts, and automatically generate test cases. This helps to support difficult and time-consuming tasks.

Machine-learning models have been used to estimate missing or dynamic attributes—such as device trust levels, user risk scores, or behavioural profiles—based on observed activity. This enables risk-adaptive ABAC, where access decisions incorporate real-time predictions rather than relying solely on static attribute values[47,48].

# 10. Conclusions

We have described a scheme which extends ABAC authorisation to support permission (rule) selection for authorising access to complex applications. The scheme offers additional control through classifier (attribute) importance ratings, and can provide substantial performance advantages through maintaining a sorted permissions base. Our scheme supports negative permissions, particularly useful for authorisations for hierarchical structures, and break glass overrides to data permitted at higher security levels. Similar although reduced functionality has been provided by other applications, mostly through programming external to the ABAC system. Where such functionality is part of the ABAC language and system, our approach is simpler to use. We described in conceptual terms how our scheme can be implemented. An early implementation of our ABAC techniques was previously used to explore patients' preferences for accessing their health data, and we have extended this work to include teams accessing sensitive data according to location. Future work will investigate the tailoring of T Permissions matching according to the results of mining the acceptability of authorisation outcomes, and the use and attempted use of overrides.

# Author Contributions

Both authors contributed equally to the conception, design, data collection, analysis, and writing of this study. Both authors have read and agreed to the published version of the manuscript.

## Funding

This work received no external funding.

## Institutional Review Board Statement

Not applicable.

## Informed Consent Statement

Not applicable.

## Data Availability Statement

All data generated or analyzed during this project are published in this article.

## Conflicts of Interest

The authors declare no conflict of interest.

## AI Use Statement

The authors declare that no artificial intelligence (AI) tools were used in the preparation of this manuscript

## References

[1] Kashani, M.H., Madanipour, M., Nikravan, M., et al., 2021. A systematic review of IoT in healthcare: Applications, techniques, and trends. Journal of Network and Computer Applications. 192, 103164.

[2] Opinion Editor, 2022. Why hospitals can no longer afford to ignore IoT. Available from: https://www.htworld.co.uk/news/why-hospitals-can-no-longer-afford-to-ignore-iot/ (cited 6 February 2026).

[3] CenTrak, 2024. The Power of IoT Location Technology in Hospitals. Available from: https://www.iotforall.com/the-power-of-iot-location-technology-in-hospitals (cited 6 February 2026).

[4] Natgunanathan, I., Mehmood, A., Xiang, Y., et al., 2018. Location Privacy Protection in Smart Health Care System. IEEE Internet of Things Journal. 6(2), 3055–3069.

[5] Yang, X., Gao, L., Zheng, J., et al., 2020. Location Privacy Preservation Mechanism for Location-Based Service with Incomplete Location Data. IEEE Access. 8, 95843–95854.

[6] Abu Rrub, J., Al-Jabi, J., El-Khatib, K., 2012. Security model for real time tracking system (RTLS) in the healthcare sector. In Proceedings of the 2012 International Conference on Communications and Information Technology (ICCIT), Hammamet, Tunisia, 26–28 June 2012; pp. 369–373. DOI: https://doi.org/10.1109/ICCITechnol.2012.6285828

[7] Pulkkis, G., Karlsson, J., Westerlund, M., et al., 2017. Secure and Reliable Internet of Things Systems for Healthcare. In Proceedings of the 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), Prague, Czech, 21–23 August 2017; pp. 169–176.

[8] Kumar, A., Jain, A.K., Dua, M., 2021. A comprehensive taxonomy of security and privacy issues in RFID. Complex & Intelligent Systems. 7(3), 1327–1347. DOI: https://doi.org/10.1007/s40747-021-00280-6

[9] Osman, M.S., Azizan, A., Hassan, K.N., et al., 2021. BLE-based Real-time Location System Integration with Hospital Information System to Reduce Patient Waiting Time. In Proceedings of the 2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), Kuala Lumpur, Malaysia, 12–13 June 2021; pp. 1–6. DOI: https://doi.org/10.1109/ICECCE52056.2021.9514248

[10] Longstaff, J., 2019. On Citizens Controlling Access to Their Health and Social Care Data for Direct Care. Teesside University: Middlesbrough, UK. Available from: https://research.tees.ac.uk/en/publications/on-citizens-controlling-access-to-their-health-and-social-care-da

[11] Blackstock, G., 2023. Families raise fears over vulnerable patients leaving wards. Available from: https://www.bbc.co.uk/news/uk-scotland-65855353 (cited 6 February 2026).

[12] Holt, A., 2022. Police investigate alleged killing of grandmother at care home. Available from: https://www.bbc.co.uk/news/uk-63471059 (cited 6 February 2026).

[13] BBC, 2022. Hundreds of patients dying due to A&E delays, doctors say. Available from: https://www.bbc.co.uk/news/uk-scotland-60865799 (cited 6 February 2026).

[14] McCracken, N., 2023. Northern Ireland medical negligence costs double in a year. Available from: https://www.bbc.co.uk/news/uk-northern-ireland-65352193 (cited 6 February 2026).

[15] Hu, V.C., Ferraiolo, D.F., Chandramouli, R., et al., 2017. Attribute-Based Access Control. Artech House Publishers: Norwood, MA, USA.

[16] Rissanen, E. (Ed.), 2017. [XACML-V3.0-Errata01] eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01. OASIS eXtensible Access Control Markup Language (XACML) TC: Burlington, MA, USA.

[17] Axiomatics, 2023. Solution Brief: Orchestrated Authorisation. Axiomatics: Stockholm, Sweden. Available

from: https://axiomatics.com/wp-content/uploads/2023/02/orchestrated-authorization-solution-brief-axiomatics-2-15-2023.pdf

[18] Talegaon, S., Batra, G., Atluri, V., et al., 2022. Contemporaneous Update and Enforcement of ABAC Policies. In Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT'22), New York, NY, USA, 8–10 June 2022; pp. 31–42. DOI: https://doi.org/10.1145/3532105.3535021

[19] Vrielynck, P.-J., Van hamme, T., Ghostin, R., et al., 2024. A Self-Sovereign Identity Approach to Decentralized Access Control with Transitive Delegations. In Proceedings of the 29th ACM Symposium on Access Control Models and Technologies (SACMAT'24), San Antonio, TX, USA, 15–17 May 2024; pp. 139–147. DOI: https://doi.org/10.1145/3649158.3657045

[20] Ruiz, J.A., Narendran, P., Masoumzadeh, A., et al., 2024. Converting Rule-Based Access Control Policies: From Complemented Conditions to Deny Rules. In Proceedings of the 29th ACM Symposium on Access Control Models and Technologies (SACMAT'24), San Antonio, TX, USA, 15–17 May 2024; pp. 159–169. DOI: https://doi.org/10.1145/3649158.3657040

[21] Longstaff, J., Noble, J., 2016. Attribute Based Access Control for Big Data applications by Query Modification. In Proceedings of the IEEE Second International Conference on Big Data Computing Service and Applications, Oxford, UK, 29 March–1 April 2016; pp. 58–65. DOI: https://doi.org/10.1109/BigDataService.2016.35

[22] Longstaff, J., Howitt, A., 2014. Tees Confidentiality Model (TCM2): Supporting dynamic authorisation and overrides in Attribute Based Access Control. In: Issac, B., Israr, N. (Eds.). Case Studies in Secure Computing: Achievements and Trends. Auerbach Publications: New York, NY, USA.

[23] Nath, R., Das, S., Sural, S., et al., 2019. PolTree: A Data Structure for Making Efficient Access Decisions in ABAC. In Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT); Toronto, ON, Canada, 3–6 June 2019; pp. 25–35. DOI: https://doi.org/10.1145/3322431.3325102

[24] Liu, M., Yang, C., Li, H., et al., 2020. An Efficient Attribute-Based Access Control (ABAC) Policy Retrieval Method Based on Attribute and Value Levels in Multimedia Networks. Sensors. 20(6), 1741. DOI: https://doi.org/10.3390/s20061741

[25] Servos, D., Osborn, S., 2017. Current Research and Open Problems in Attribute-Based Access Control. ACM Computing Surveys. 49(4), 65. DOI: https://doi.org/10.1145/3007204

[26] Xu, Z., Stoller, S.D., 2015. Mining Attribute-Based Access Control Policies. IEEE Transactions on Dependable and Secure Computing. 12(5), 533–545.

[27] Mulrine, S., Murtagh, M., Minion, J., 2018. Great North Care Record Public Engagement Report. Teesside University: Middlesbrough, UK. Available from: https://www.greatnorthcarerecord.org.uk/wp-content/uploads/2018/09/GNCR-public-engagement-report-FINAL.pdf

[28] NIST Information Technology Laboratory Computer Security Resource Center, n.d. Access Control. Available from: https://csrc.nist.gov/glossary/term/access_control (cited 6 February 2026).

[29] ANSI INCITS 359-2012 (R2022). 2012. Information Technology—Role Based Access Control.

[30] Gupta, M., Patwa, F., Sandhu, R., 2018. An Attribute-Based Access Control Model for Secure Big Data Processing in Hadoop Ecosystem. In Proceedings of the Third ACM Workshop on Attribute-Based Access Control Ecosystem (ABAC'18), Tempe, AZ, USA, 21 March 2018; pp. 13–24. DOI: https://doi.org/10.1145/3180457.3180463

[31] NIST SP 800-178. 2016. A Comparison of Attribute Based Access Control (ABAC) Standards for Data Service Applications: Extensible Access Control Markup Language (XACML) and Next Generation Access Control (NGAC).

[32] NIST SP 800-162. 2019. Guide to Attribute Based Access Control (ABAC) Definition and Considerations.

[33] Gupta, E., Sural, S., Vaidya, J., et al., 2022. Enabling Attribute-Based Access Control in NoSQL Databases. IEEE Transactions on Emerging Topics in Computing. 11(1), 208–223. DOI: https://doi.org/10.1109/TETC.2022.3193577

[34] Brossard, D., Gebel, G., Berg, M., 2017. A Systematic Approach to Implementing ABAC. In Proceedings of the CODASPY'17: Seventh ACM Conference on Data and Application Security and Privacy, Scottsdale, AZ, USA, 24 March 2017; pp. 53–59. DOI: https://doi.org/10.1145/3041048.3041051

[35] Huang, J., Nicol, D.M., Bobba, R., et al., 2012. A Framework Integrating Attribute-based Policies into Role-Based Access Control. In Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT'12), Newark, NJ, USA, 20–22 June 2012; pp. 187–196.

[36] Qiao, Z., Liang, S., Davis, S., et al., 2014. Survey of attribute based encryption. In Proceedings of the 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Las Vegas, NV, USA, 30 June–2 July 2014; pp. 1–6. DOI: https://doi.org/10.1109/SNPD.2014.6888687

[37] ASCLEPIOS Consortium, 2019. D3.1—ASCLEPIOS Security and Policies Model. ICCS: Athina, Greece. Available from: https://zenodo.org/records/4022334

[38] Ullah, S., Oleshchuk, V., Gardiyawasam Pussewalage, H.S., 2023. A survey on blockchain envisioned attribute based access control for internet of things:

Overview, comparative analysis, and open research challenges. Computer Networks. 235, 109994. DOI: https://doi.org/10.1016/j.comnet.2023.109994

[39] Wu, N., Xu, L., Zhu, L., 2023. A blockchain based access control scheme with hidden policy and attribute. Future Generation Computer Systems. 141, 186–196.

[40] Bui, T., Stoller, S., Le, H., 2019. Efficient and Extensible Policy Mining for Relationship-Based Access Control. In Proceedings of the SACMAT'19: The 24th ACM Symposium on Access Control Models and Technologies, Toronto, ON, Canada, 3–6 June 2019; pp. 161–172. DOI: https://doi.org/10.1145/3322431.3325106

[41] Pang, R., Cáceres, R., Burrows, M., et al., 2019. Zanzibar: Google's Consistent, Global Authorization System. In Proceedings of the 2019 USENIX Annual Technical Conference, Renton, WA, USA, 10–12 July 2019.

[42] Xiao, M., Li, H., Huang, Q., et al., 2022. Attribute-Based Hierarchical Access Control With Extendable Policy. IEEE Transactions on Information Forensics and Security. 17, 1868–1883.

[43] Servos, D., Osborn, S.L., 2015. HGABAC: Towards a Formal Model of Hierarchical Attribute-Based Access Control. In: Cuppens, F., Garcia-Alfaro, J., Zincir Heywood, N., et al. (Eds.). Foundations and Practice of Security (FPS 2014), Lecture Notes in Computer Science, Vol. 8930. Springer: Cham, Switzerland. pp. 187–204.

[44] Axiomatics, n.d. Confidently scale authorization policies across your organization with externalized architecture. Available from: https://axiomatics.com/solutions/scale-authorization-externalized-architecture (cited 6 February 2026).

[45] Xu, M., Stoller, S.D., 2013. Mining Attribute-Based Access Control (ABAC) Policies from RBAC policies. In Proceedings of the 2013 10th International Conference and Expo on Emerging Technologies for a Smarter World (CEWIT), Melville, NY, USA, 21–22 October 2013; pp. 1–6. DOI: https://doi.org/10.1109/CEWIT.2013.6713753

[46] Lawal, S., Zhao, X., Rios, A., et al., 2024. Translating Natural Language Specifications into Access Control Policies by Leveraging Large Language Models. In Proceedings of the 2024 IEEE 6th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA), Washington, DC, USA, 28–31 October 2024; pp. 361–370. DOI: https://doi.org/10.1109/TPS-ISA62245.2024.00048

[47] Tripathi, A., Rajan, K., Kumar, V., et al., 2024. Real Time Adaptive Access Control with Behavioral Analytics for Enhanced Cybersecurity in IoT and Cloud Systems. In: Solanki, V.K., Tan, T.D., Kumar, P., et al. (Eds.). Proceedings of the Ninth International Conference on Research in Intelligent Computing in Engineering. Polish Information Processing Society (PTI): Warszawa, Poland. pp. 151–155.

[48] Amour, S., Gudes, E., 2025. Predictive Enhancement of ABACPolicies Using Access Log Analytics. In Proceedings of the 30th ACM Symposium on Access Control Models and Technologies (SACMAT'25), Stony Brook, NY, USA, 8–10 July 2025; pp. 16–21. DOI: https://doi.org/10.1145/3734436.3734455