Towards security in multi-agency clinical information services*

Salem Aljareh
Computing Science Department
University of Newcastle upon Tyne
Newcastle upon Tyne, NE1 7RU, UK
+44 191 222 6053
s.s.aljareh@ncl.ac.uk

Nick Rossiter
Computing Science Department
University of Newcastle upon Tyne
Newcastle upon Tyne, NE1 7RU, UK
+44 191 222 7946
b.n.rossiter@ncl.ac.uk

ABSTRACT

There is no doubt that the privacy of people, the confidentiality of their information, the integrity of transaction holding, and the availability of service systems are all essential and any threats in any one of these aspects will be costly and could lead to disaster. Thereby securing computer services has been considered as a core part of any new development one of which is the clinical information systems. In this paper we will discus a security policy model for a clinical information system and investigate whether logical languages can represent the principles of this kind of model. We have used three security logical languages: the Authorization Specification Language (ASL), a Language for Security Constrains on Objects (LaSCO) and Ponder: a Language for Specifying Security and Management Policies for Distributed Systems. We will also study whether these principles are sufficient to deal with the case of multi-agency services and sharing information with the different agencies such as social services, police and education authority.

Keywords
Security policies, security models, multi-agency services, clinical information systems.

INTRODUCTION
For economy of expression and to make it easy for readers to link with Anderson's model, we will assume that the clinician is female and the patient male.

A security model for clinical information systems has been proposed by Ross Anderson [1,2] to allow the British Medical Association (BMA) to meet security requirements of the electronic patient record (EPR) and to be the base of any proposed system claims to operate the EPR. Anderson's model is composed of a set of principles based on a statement found in the Good Medical Practice booklet issued by the General Medical Council (GMC), which says:

> *Patients have a right to expect that you will not pass on any personal information, which you learn in the course of your professional duties, unless they agree.*

* In proceedings of workshop on dependability in healthcare Informatics Edinburgh, 22nd-23rd March 2001

This raises the following questions. Is a patient/patient's guardian qualified enough to give consent and be aware about all consequences that could accrue in future including security threats? For instance some patients consider their clinician's commands as a part of their treatments and must be obeyed. Are ordinary patients usually familiar with how their information could be used in future? Despite his guardian's help, the patient may permit an action that may consequently lead to security threats. For example patients with a little knowledge about the abuse of clinical information will assume that to reject giving consent is safer. Who is responsible in case a patient died as a result of rejected consent? Nevertheless a patient may find himself forced to give a consent that authorizes other agencies to gain access to his medical record in order to obtain a particular service: for instance [8] to claim for a reimbursement for the cost of a visit to the doctor in US. There are also many examples in the UK. For instance, from your insurance company, you need to complete a form that includes the following part:

> *I authorize any physician, hospital, or other medical related facility, insurance company, or other organization, institution or person, that has any record or knowledge of me or my dependents, or our health to disclose, when ever requested to do so by CAN or its representatives, any and all such information. A photocopy of this authorization shall be considered as effective and valid as the original.*

THE PRINCIPLES

Nine principles were defined by Anderson [1,2]. These are given below with comments on obvious difficulties in their implementation.

Principle 1: *Each identifiable clinical record shall be marked with an access control list naming the people or groups of people who may read it and append data to it. The system shall prevent anyone not on the access control list from accessing the record in any way.*

Principle 2: *A clinician may open a record with herself and the patient on the access control list. Where a patient has been referred, she may open a record with herself, the patient and the referring clinician(s) on the access control list.*

The clinician-may-open-record clause in principle 2 makes this principle difficult to be logically represented. It can be only understood as the following: a clinician **must** open a new clinical record associated with a new access control list for a patient if the new information appear to be hidden from users who already have access to this patient's clinical record unless this principle is not needed to be formally implemented. There should be, at least a measurement/mechanism to find out whether a clinician was right by opening/not opening a new medical record for a patient according to her judgment about the security level of the new information.

Principle 3: *One of the clinicians on the access control list must be marked as being responsible. Only she may alter the access control list, and only she may add other health care professionals to it.*

Let us consider the case where the responsible clinician is the only authorized user to alter the access control list for a certain clinical record. For instance she forgets her password or deletes her record from the access control list, which means losing access to the access control list of that clinical record. Who is going to make the access control list available again? We may assume that a new access control list has to be created for that clinical record to replace the inaccessible one and/or that there will be another higher level of security such as system administrator. However this assumption is against the goal of principle 3, which is based on the responsible clinician being the highest security level for a certain clinical record unless the super authorization (e.g. system administrator) might be made as an exception comparable to the accident and emergency staff authorization. The second part of the confusion concerns the technical experience of the position of the responsible clinician at network level? Is it operating system level, middleware level or application level? Definitely it is going to be impracticable for a clinician to manage control on all these levels, so her control will be at one level, which very likely will be at the application level. Logically the levels beneath compromise security mechanisms at any higher level. For instance the security mechanism from the application level can be bypassed using power control of the operating system.

Principle 4: *The responsible clinician must notify the patient of the names on his record's access control list when it is opened, of all subsequent additions, and whenever responsibility is transferred. His consent must also be obtained, except in emergency or in the case of statutory exemptions.*

Principle 5: *No one shall have the ability to delete clinical information until the appropriate time period has expired.*

Principle 6: *All accesses to clinical records shall be marked on the record with the subject's name, as well as the date and time. An audit trail must also be kept of all deletions.*

Principle 7: *Information derived from record A may be appended to record B if and only if B's access control list is contained in A's.*

Principle 8: *There shall be effective measures to prevent the aggregation of personal health information. In particular, patients must receive special notification if any person whom it is proposed to add to their access control list already has access to personal health information on a large number of people.*

Principle 9: *Computer systems that handle personal health information shall have a subsystem that enforces the above principles in an effective way. Its effectiveness shall be subject to evaluation by independent experts.*

The apparent manual nature of principle 9 makes it unsuitable to be represented by the logical languages. For this reason we will assume that it has no part to play in the following sections.

USING SECURITY LOGICAL LANGUAGES TO REPRESENT ANDERSON'S PRINCIPLES:

Three languages are assessed in turn for their effectiveness in handling Anderson's principles: ASL, LaSCO and Ponder.

General Definitions:

Some important definitions need to be stated before we start using a logical language to represent a security policy:

Closed Policies (positive Authorizations):

An access is granted if there is an authorization stating that the user can access the object.

Open policies (negative Authorizations):

A user can access any object unless it has been explicitly denied.

An access control policy is a set of rules defining what is authorized.

An access control mechanism is a policy implementation to ensure that all accesses are in accordance with the underlying policy.

THE AUTHORIZATION SPECIFICATION LANGUAGE (ASL):

Definitions and an overview:

ASL [3] is a language for expressing the authorization according to access control policies. ASL supports a model based on two elements, an object (o) which could be a file or directory in a operating system or table in relational database, and an authorized entity who could be a user (U), group (G) or roles (R). An authorization policy in ASL is a mapping that maps 4-tuples (*o, u, R, a*) to the set {+, -}, where *o* is an object, *u* is a user, *R* is a role and *a* is an action, while + means authorized and - means denied.

ASL is designed principally to express the following rules:

Authorization Rules: used by the System Security Officer (SSO) to allow or deny accesses to objects explicitly in the following form:

$$cando(o, s, <sign>a) \leftarrow L1\& \ldots. \&Ln$$

This predicate symbol states that a subject *s* can (*Positive authorization sign = "+")* or cannot (*negative authorization sign = "-")* perform the action *a* on the object *o* under the conditions specified by *L1& ..&Ln. L 1 ,…, Ln* could be one of the following literals: in, dirin or typeof. Principle 1 in the following section is an example of this rule.

Derivation Rules: used to derive implicit authorizations from explicit authorizations and determine the authorization policy. Indeed it is for expressing propagation of authorization along a subject's hierarchies. In addition derivation rules can express some kinds of implication relationships such as the derivation of an authorization in the base of the presence or the absence of other authorizations. The derivation rule has the following form:

$$dercando(o, s, <sign>a) \leftarrow L1\& \ldots. \&Ln$$

The right hand side of this rule derives a positive or negative authorization The outcome is determined by <*sign*>) for a subject *s* to perform the action *a* on the object *o* according to another authorization in the right hand side (*L1& …&Ln*). *L1 ,…, Ln* could be one of the following literals: cando, dercando, done, do, in, dirin or typeof.

Resolution Rules: used to regulate how to resolve any conflict could accrue between authorizations specified by the authorization rules *cando* and *dercando* as in the following form:

$$do(o, s, <sign>a) \leftarrow L1\& \ldots. \&Ln$$

This form states the enforcement of exercising (if *sign* = +) or forbidding (if *sign* = -) an access on an object by a subject *s* in the case of a conflict in the Authorization rules (*cando* or *dercando*) in the right hand side.

Access Control Rules: to be used to regulate access control decisions on the basis of authorization specified by the authorization rules.

Access control rules have the following form:

$$grant(o, u, rs, <sign>a) \leftarrow L1\& \ldots. \&Ln$$

This form states that a request submitted by a user *u* with active roles *R* to perform the action *a* will be allowed (*sign* = +) or forbidden (*sign* = -) based on an authorization condition on the right hand side *L1& …. &Ln*. *L1& …. &Ln* are either cando, dercando, done, do, in, dirin, or typeof

Integrity Rules: used to express different kinds of constraints on the specifications and the use of authorizations. An integrity rule is of the form:

$$error() \leftarrow L1\& \ldots. \&Ln$$

where *L1& …. &Ln* are either cando, dercando, done, do, in, dirin, or typeof This rule derives an error every time the conditions in the right hand side of the rules are satisfied.

Using the authorization language (ASL) for specifying the clinical security principles:

Principle 1:

A subject *s* can read from and write on a clinical record *clincal_record* if and only if she is in the access control list *Clinical_Record_ACL* (here specified as a role) of that record.

The following is simply an authorization rule. The left hand side part (*cando(clincal_record, s, +read/write)*) is the authorization that is to be given and the right hand side part (*in(s, Clinical_Record_ACL)*) is specifying conditions that must be verified for the authorization to be granted.

$$\leftarrow cando(clincal\_record, s, +read/write) \leftarrow in(s, Clinical\_Record\_ACL)$$

Principles 3 and 4:

The following code states that a subject *patient* must read his access control list *Clinical_Record_ACL* if it has been appended by a subject *clinician* who is authorized to do so

← cando(Clinical_Record_ACL, clinician, +append) ← do(Clinical_Record_ACL , patient, +read)

← cando (Clinical_Record_ACL , patient, +read) ← done (Clinical_Record_ACL, clinician, append) ← cando (Clinical_Record_ACL, clinician, +append)

Observations:

ASL is a language for expressing the authorization in matter of allowing or denying an access to an object. There is no way to express consequent actions that should be carried out after some authorized access such as auditing operations (e.g. principle 6). In addition it is not clear in this language how to express authorization restricted by the number of accesses as is needed for controlling the aggregate access problem (e.g. principle 8).

A LANGUAGE FOR SECURITY CONSTRAINTS ON OBJECTS (LaSCO):

An overview:

LsSCO [4] is based on a model where a system consists of objects and events. The attributes on an event denote the specifics of the event's execution. Policies in LaSCO are stated as policy graphs which describe a specific state of the system (domain) and specific access constraints (requirements). Predicates are annotations near the nodes (objects) and the edges (events) to describe the domains (in the graph written as bold text. For example **Type="user"** and **Method="access"** in figure 1 are descriptions of domains. Requirements are written in the graph as normal text, for example $UID ∈ ACL. LaSCO uses variables called *policy variables.* A policy variable represents a value of an attribute and relates attribute values associated with different objects and events. Variables may appear as operands in domains (e.g. in figure 1 **ID=$UID**) and in requirement predicates (e.g. in figure 1 $UID ∈ ACL). They are denoted by a "$" prefix.

Using LaSCO to describe the clinical security principles:

Principle 1:

The policy graph in figure 1 indicates that a user/subject needs to have his/her/it ID represented by the policy variable $UID included in the access control list of the clinical record in order to have an access to it, that is $UID ∈ ACL.

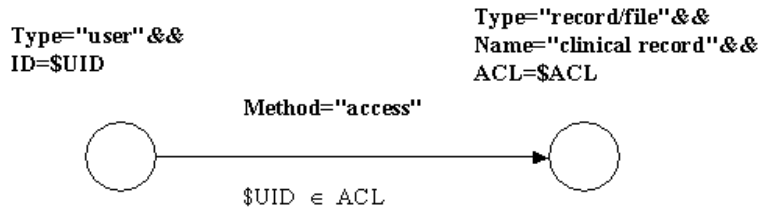

Figure 1: a security policy graph to represent principle 1 of the clinical security policy.

Principle 2:

If the user's security level/clearance, represented by the policy variable $UL, is not the same as the clinical record's security level, represented by another policy variable $FL, a clinician may create a new clinical record with the new access control list as shown in figure 2. The requirement is that the security levels are different ($UL ≠ $FL).

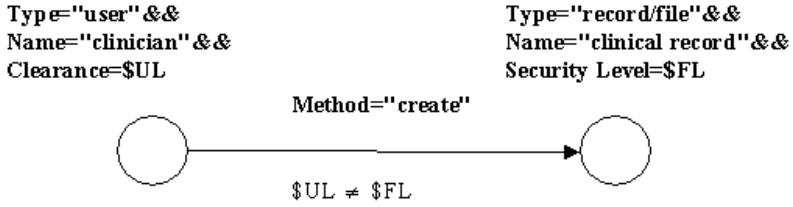

Figure 2: a security policy graph to represent principle 2 of the clinical security policy.

Principle 3:

The policy graph in figure 3 states that a user, represented as an object, which is stated by a set of attributes *type* and *position* in addition to policy variable *$ID*, can only append the access control list for a clinical record if she is marked as responsible clinician. The event is represented in the policy graph as a method with value "append". The domain is represented as an object, which is stated by a set of attributes *type* and *name*. The requirement is that the ID of that user has to be the value of the attribute called *responsible_clinician.*

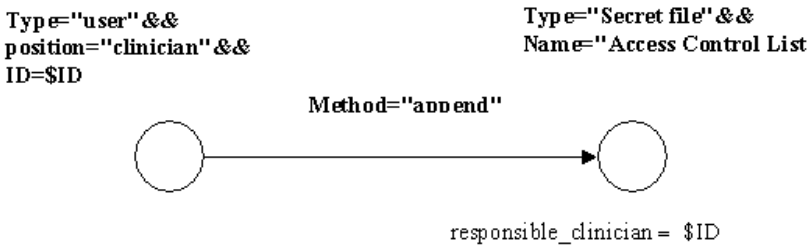

Figure 3: a security policy graph to represent principle 3 of the clinical security policy.

Principle 4:

Since principle 4 includes two events under different restrictions, we will divide it into two principles. Principle 4a considers the part that says the responsible clinician must notify the patient of all subsequent additions to the names on his record's access control list when it is opened. Principle 4b considers the part that says, whenever responsibility is transferred, the Patient's consent must also be obtained, except in emergency or in the case of statutory exemptions.

Principle 4a:

The policy graph in figure 4 specifies the first part of principle 4 by restricting the method *add* to be executed after the method *message.* Both *add* and *message* are events and the restriction is ensured by a security requirement that enforces the order of these two events (*Time > $ST*). So adding a new user to the access control list will not be allowed until a message is sent to the patient containing the name of the user who it is proposed to add to the access control list of his clinical record.

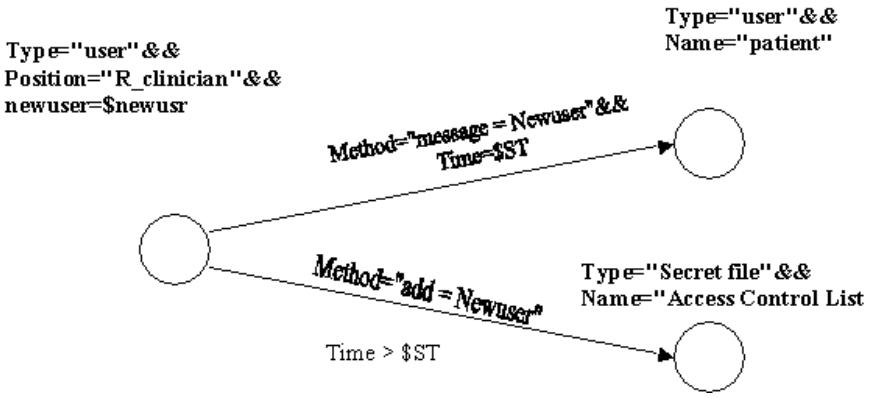

Figure 4: a security policy graph to represent the first part of principle 4 of the clinical security policy.

Principle 4b:

The policy graph in figure 5 specifies the second part of principle 4 as following: the event *change responsibility* is restricted by either the case is an emergency or the other event *Name="Consent" && Permit=$PM* has been performed and the consent has been given, that is *$PM=true || Case="emergency".*

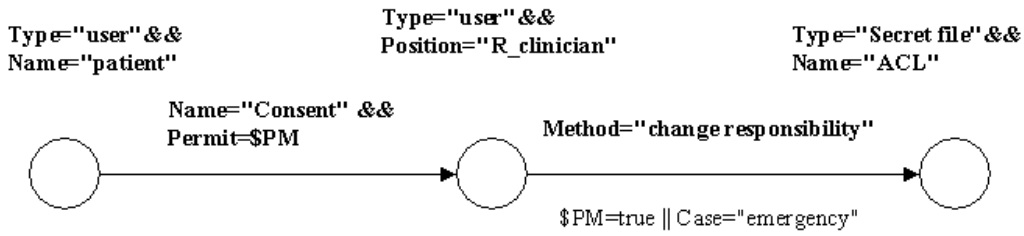

Figure 5: a security policy graph to represent the second part of principle 4 of the clinical security policy.

Principle 5:

The policy graph that is shown in figure 6 states that event *delete* can he called by a *subject* to delete a *clinical record* object if and only if the system date *$sysdate,* a policy variable, is grater than or equal to the expiry date of that clinical record *$Edate,* another Policy variable.

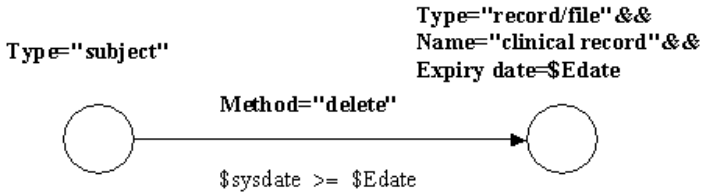

Figure 6: a security policy graph to represent principle 5 of the clinical security policy.

Principle 6:

The policy graph that is shown in figure 7 specifies principle 6 by ensuring that a log record to be created contains a subject id ($SID), the access date and time ($DT), the access id ($ADID), and the accessed clinical record ($RID) for any access to the clinical record instance.

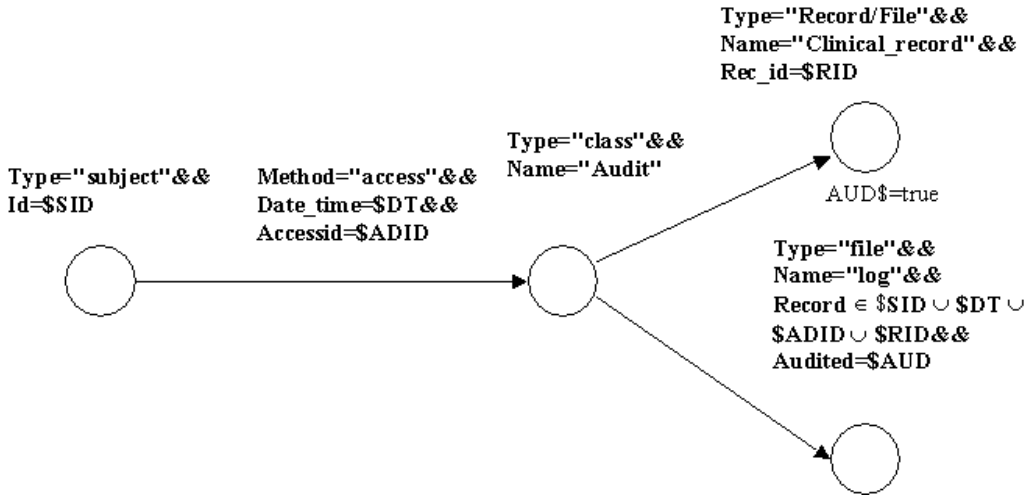

Figure7: a security policy graph to represent principle 6 of the clinical security policy.

Principle 7:

The policy graph that is shown in figure 8 represents the information flow control by ensuring that the access control list for the source record is a subset of the access control list of the destination record, that is *$ACL_B ∈ $ACL_A*.

Note that: this principle is also implicitly shown in principle 1 representation

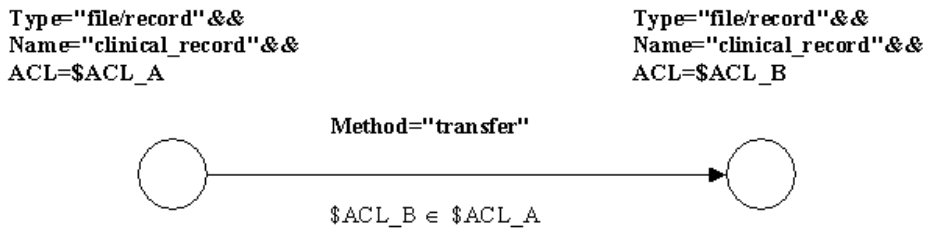

Figure 8: a security policy graph to represent principle 7 of the clinical security policy.

Principle 8:

The policy graph, shown in figure 9, states that adding a new user to the access control list of a patient's medical record is not allowed before a message is sent to the patients. This message informs them that the user who it is proposed to add to their access control list already has access to personal health information on a large number of people such that *NoA > n*. *NoA* is the number of accesses for the proposed user and *n* is a constant.

Note that the order of the two events is enforced by ensuring that the time of the add method is greater than the time of message method, that is *Time > $ST*.

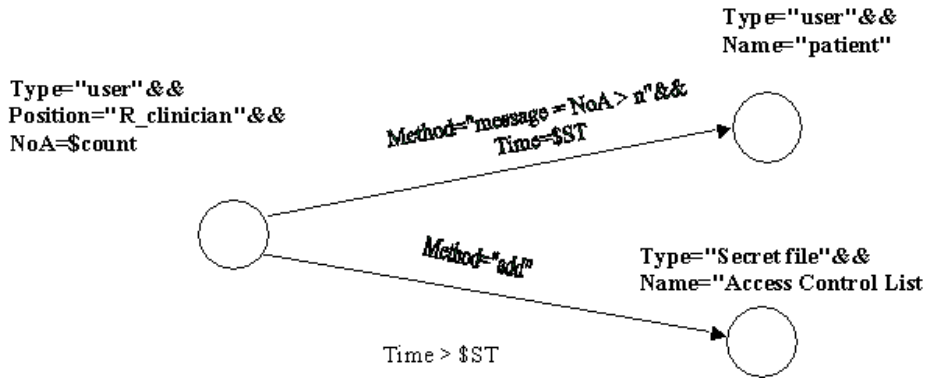

Figure 9: a security policy graph to represent principle 8 of the clinical security policy.

A LANGUAGE FOR SPECIFYING SECURITY AND MANAGEMENT POLICIES FOR DISTRIBUTED SYSTEMS (PONDER):

Definitions and an overview:

Ponder [5] is a declarative and object-oriented language that includes constructs for specifying the following basic policy types:

- Authorization policies specify what activities a subject is permitted or forbidden to do. In other words specifying either positive (**auth+**) and negative (**auth+**) authorization policies is possible.

Principle 1 in the following section is represented as a positive authorization while principle 5 is an example of the negative authorization.

- Obligation policies specify what activities a subject must do. These policies are triggered by events and are usually interpreted by a manager agent. An example of this type is found in principle 2.
- Refrain policies define actions that subjects must refrain from performing.
- Delegation policies define what authorizations can be delegated and to whom.
- Composed policies are used to define a syntactic scope for specifying a set of related policies. There are four types of composed policies: groups, roles, relationships and structure management.
- Meta policies specify a permitted value for a valid policy.

The reader of this paper will note in the following section that all Anderson's principles fall into two types of security policies, authorization policies and obligation policies because these principles attempt to restrict the access control and/or enforce consequential actions such as the auditing process.

Using the Ponder language to describe the clinical security principles:

Principle 1:

A subject *s* of type *user* is authorized to read and/or append the clinical record *r* if and only if s is in the access control list of the clinical record *r.ACL* where *r.ACL* is the access control of the clinical record *r*.

Note that **type** is a type definition introducing a new user-defined policy type from which one or more policy instances of that type can be created, **auth+** is a reserved word indicating that the following is a positive authorization policy and *principle1* the name of the policy type. **subject**<user> s means that *s* is a subject of type *user*, **target** <clinicalRecord> r means that *r* is the **target** object of type *clinical Record* to be accessed by the subject *s*. **action** is a reserved word followed by the action, *read* and *append*, that is needed to be authorized. *belongs* is a user defined function to check whether the subject *s* is a member of the access control list of the record *r*. If so the positive authorisation will be allowed, that is **result =** enable;

```
type
    auth+ principle1 (subject <user>  s, target <clinicalRecord> r)
      {
        action read, append if belongs(s, r.ACL)
            {
               result = enable;
            }
       }
```

Principle 2:

In the case of new clinical information for a patient *NewInformation* appears to be in a different security level *isDifferentSecLevel*. From the existing access control list *currClinicalRecordAcl* a new access control list *newClinicalRecordAcl* has to be created

Note that **on** is a reserved word followed by the obligation condition, and *isDifferentSecLevel* is a user-defined function to compare the new information against the current security level and check whether it is a different security level. In this case the mandatory action has to be performed, that is **do** createNewACL(newClinicalRecordAcl)).

type

oblig principle2 (**subject** <responsible_clinician> s,
target <ACL> currClinicalRecordAcl, newClinicalRecordAcl,
<ClincalData> newInformation)
{
on isDifferentSecLevel (currClinicalRecordAcl, NewInformation);
do createNewACL(newClinicalRecordAcl));
}

Principle 3:

A subject *s* of type clinician is authorized to alter and/or append the access control list of a clinical record *clinicalRecordAcl* if and only if *s* is marked as a responsible user in the access control list.

type

auth+ principle3 (**subject** <clinician> s, **target** <ACL> clinicalRecordAc)
{
action alter, append
if position(s, clinicalRecordAcl) = "responsible"
{
result = enable;
}
}

Principle 4:

This principle is divided into two parts the first principle 4a concerns informing the patient about any new addition to his clinical record access control list via the responsible clinician. This part is represented as follows: in case of adding new record *addNew* to a patient's access control list of his clinical record *clinicalRecordAcl*, then that patient has to be informed through the action *informPatient*.

```
type
  oblig principle4a (subject <responsible_clinician> s,
                     target <ACL> clinicalRecordAcl, clinician newName)
        {
         on addNew (newName, clinicalRecordAcl);
         do informPatient (newName);
        }
```

The second part of this principle (principle 4b) deals with the case of changing the responsibilities in the access control list of the clinical records. This part is represented as follows: a subject *s* of type responsible clinician will be authorized to change the responsibilities in the access control list of a patient clinical record if and only if he has obtained that patient's consent.

```
type
  auth+ principle4b (subject <responsible_clinician> s,  target <ACL> clinicalRecordAcl)
        {
         action changeResponsibility  if PatientConsent (clinician,  newResponsibility)=true
                    {
                        result = enable;
                    }
        }
```

Principle 5:

A subject *s* cannot delete clinical record *r* until this record has expired, that is *todayDate()* > *expiryDate(r).*

Note that **when** is called the authorization filter and is used to restrict an action by a given condition.

```
type
    auth- principle5 (subject s, target<clinicalRecord> r)
         {
          action delete;
           when todayDate() > expiryDate(r);
         }
```

Principle 6:

An audit record contains the subject identifier *s*, date *aDate* and time *aTime* of action, type of action *aType*, and the record that has been accessed *r*. All accesses on the clinical record *r* by a subject *s* must be recorded.

```
type
   oblig principle4 (subject s, target <clinicalRecord> r)
      {
       on allAccess (s, aDate, aTime, aType, r);
       do createAuditRecord (s, aDate, aTime, aType);
        }
```

Principle 7:

Transferring data from clinical record *A* to clinical record *B* is not allowed unless all records in the access control list of b *clinicalRecordAcl_B* are included in the access control list of a *clinicalRecordAcl_A*.

```
type
     auth- principle7 (clinicalRecord A, <ACL> clinicalRecordAcl_A,
                        <ACL> clinicalRecordAcl_B, target<clinicalRecord> B)
                       {
                        action transfer(a.data, b.data);
                        when List(clinicalRecordAcl_B) in List(clinicalRecordAcl_A)
                       }
```

Principle 8:

In the case of adding a new record which grants a new user access to a clinical record through a patient access control list, then the patient has to be informed of the number of records to which the new user has access.

```
type
  oblig principle8 (subject <responsible_clinician> s,
                     target <ACL> clinicalRecordAcl, clinician newName)
        {
          on addNew (newName, clinicalRecordAcl);
          do informPatient (newName, getNoAccess(newName));
        }
```

COMPARISONS:

Although all the above languages are basically targeting the specification of security polices, they focus on different aspects. For instance ASL focuses more on the access control policies and LaSCO attempts to express constraints on objects, while Ponder aims to specify security and management policies for distributed systems. Additionally they are different from the language's character point of view. Whereas ASL is based on logic and LaSCO policies are specified as logical expressions and as directed graphs, Ponder is a declarative language inheriting its syntax from the OCL "Object Constrain Language". According to the nature of each one of these languages we have found that some of Anderson's principles are not directly representable. For example principles such as those dealing with auditing operations (e.g. principle 6) and control aggregation problems (e.g. principle 8) were not representable at all by ASL and could be only indirectly expressed by LaSCO. On the other hand Ponder was more suitable for these kinds of principles since Ponder has got forms to deal with the management polices. Table 1 illustrates the comparison between these three languages according to their ability to express Anderson's clinical security principles.

Languages / Principles	ASL	LaSCO	Ponder
Principle 1	Explicitly represented	Explicitly represented	Explicitly represented
Principle 2	Not applicable	Indirectly represented	Indirectly represented
Principle 3	Indirectly represented	Explicitly represented	Explicitly represented
Principle 4	Indirectly represented	Indirectly represented	Explicitly represented
Principle 5	Not applicable	Explicitly represented	Explicitly represented
Principle 6	Not applicable	Indirectly represented	Explicitly represented
Principle 7	Not applicable	Explicitly represented	Explicitly represented
Principle 8	Not applicable	Indirectly represented	Explicitly represented
Principle 9	Not applicable	Not applicable	Not applicable

Table 1: Comparison table shows how far ASL, LaSCO and Ponder language can represent Anderson's principles.

MULTI-AGENCY SERVICES ENVIRONMENT AND COLLABORATION ISSUE

Only limited forms of cross-organizational access control were considered by Anderson [1,2]. Such aspects include principle 4 that requires informing the patient about any addition to his record access control list and principle 6 concerning the auditing aspects. In general the issue of sharing clinical information including collaboration activities with other agencies such as police, social services or the education authority were not clearly considered [7]. One possible reason could be that these principles were derived from a centralized system viewpoint at least from the responsibilities and ownership point of view.

NEED-TO-KNOW PROBLEM

Need-to-know was not included in Anderson's model, as the BMA does not accept that 'need-to-know' is an acceptable basis for access control decisions. Further details may be found in [1,2]. However there might be a case where the need-to-know cannot be avoided. For instance a service provider such as social services offers its services conditioned by some information about the patient who applies for such services. There are two major problems in this case. Firstly who is authorized to decide about who needs to know in a multi-agency services environment where responsibilities are distributed. Secondly how can we resolve the conflict between the patient's consent and the need-to-know? Further investigation will take place to find out whether the mandatory need-to-know exists. Since there is no need-to-know without a task, we propose an approach based on associating the data with tasks and granting these tasks to performers rather than giving direct authorization to the secret data. The task could be in the form of an agreement between the information's owner and who needs to know (task's performer). This agreement would consist of full awareness about the task that requires the information, information size, release time, time of expiry and a guarantee to restrict the use of this information to the specified task. We shall explain this approach and need-to-know problem in detail in another report.
It is important to note that there does exist work considering a task-based access-control model in general terms [9]

CONCLUDING DISCUSSION

Anderson in his papers [1,2] argues that a security solution is an issue requiring great care to ensure that the security mechanisms work together rather than operate independently. Although some professionals in health care information technology believe that any implementation of Anderson's principles would be expensive to implement and unmanageable to maintain, others such as Denley and Smith [6] according to their experience in the implementation of these principles in three British hospitals: Conquest Hospital, Aintree Hospital and Royal Devon and Exeter Hospital, state that Anderson's principles can be applied to the electronic patient record to maximise privacy. However, the CEN (Europe Committee for Standardization) group [10] observes that Anderson's model is specified at too high a level for practical purposes and is not provable complete because it is neither precise nor exhaustive. Cohen [11] has proposed a formal model, complementary to Anderson's, which is again not accepted by the CEN group because it is not shown to be complete and because it is not based on basic security properties.

Our work contributes to the solution of the clinical information systems security problem by discussing Anderson's security model for the clinical information systems and examining logically the principles of this model by representing them using some selected languages for specifying security policies.

The ease with which the principles can be represented in a logical framework varies considerably. For example principles such as those dealing with auditing operations and control aggregation problems were not representable at all by ASL and could be only indirectly expressed by LaSCO. On the other hand Ponder was more suitable for these kinds of principles since Ponder has got forms to deal with the management polices.

Anderson's principles are mainly applicable to centralized systems. There were no precise principles in this model concerning either the multi-agency services environment or the need-to-know problem. The latter was not rated of high priority by the BMA. We consider that a task-based approach is promising in developing a need-to-know policy. Other aspects of security in multi-agency services are still being

investigated. CEN [10] also plan to broaden the model of security in healthcare to include all the potential needs of the different participants.

ACKNOWLEDGMENTS

We wish to thank John Dobson of Newcastle University for introducing us to this area and Ross Anderson of Cambridge University for his helpful responses to our queries.

REFERENCES

1. Ross Anderson. Security in clinical information systems. BMA Report, British Medical Association, Jan 1996, ISBN 0-7279-1048-5.

2. Ross Anderson. A security Policy Model for clinical information systems. In Proceedings of the IEEE Symposium on Research in Security and Privacy, Research in Security and Privacy, pp. 30–43. IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Society Press, Oakland, CA, May 1996.

3. Jajodia, Sushil, Pierangela Samarati, and V.S. Subrahmanian. A Logical Language for Expressing Authorizations. in Proceedings of the 1997 IEEE Symposium on Security and Privacy. Oakland, CA, USA: IEEE Press, 1997. p. 31-42.

4. Hoagland James, Raju Pandey, and Karl Levitt. Security Policy Specification Using a Graphical Approach. Technical report CSE-98-3, The University of California, Davis Department of Computer Science. July 1998.

5. Nicodemos Damianou, Naranker Dulay, Emil Lupu, Morris Sloman. Ponder: A Language for Specifying Security and Management Policies for Distributed Systems The Language Specification Version 2.2. 15 July, 2000, available at http://www-dse.doc.ic.ac.uk/policies

6. Ian Denley**,** Simon Weston Smith. Privacy in clinical information systems in secondary care, 15 May 1999, Available at http://www.bmj.com/cgi/content/short/318/7194/1328

7. Rory O'Conor. Commentary: Organisational and cultural aspects are also important. 15 May 1999 available at http://www.bmj.com/cgi/content/short/318/7194/1328#resp2

8. Simson Garfinkel. Database Nation - The death of privacy in 21st century. Sebastopol, CA: O'Reilly & Associates, 2000. ISBN 1-5659-2653-6

9. R. K. Thomas and R. S. Sandhu. Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management. Proceedings of the IFIP WG11.3 Workshop on Database Security, Lake Tahoe, California, August 11-13, 1997

10. European Committee for Standardization. Technical Committee for Health Informatics, available at http://www.centc251.org/

11. B. Cohen, A Formal Model of Healthcare Security Policy, City University, 1996