A Comparison of the Akenti and PERMIS Authorization Infrastructures

Authors: Sassa Otenko, David Chadwick
Information Systems Security Research Group
University of Salford

Document History

Version	Date	Comments
0.1	24.10.2002	First draft
0.2	3.12.2002	Pre-release draft
0.3	11.12.2002	Almost ready!
0.4	17.01.2003	Done?
1.0	22 Jan. 03	Initial public release
1.1	17 April	Minor eds. Added 2 advantages of Akenti
2.0	12 June 2003	Version 2 Draft. Added Ease of Use and Performance statistics and updated Initial release due to increased knowledge
2.1	30 July 2003	Final Version, incorporating feedback from LBL

This report is the deliverable of the JISC funded project “A Comparison of the AKENTI and PERMIS Authorisation Infrastructures”

Table of Contents

Glossary of Terms

ACI – Access Control Information (from ISO 10181-3). Any information used for access control purposes, including contextual information.

ADF – Access control Decision Function (from ISO 10181-3). A specialized function that makes access control decisions by applying access control policy rules to an access request, ADI (of users, targets, access requests, or that retained from prior decisions), and the context in which the access request is made.

ADI – Access control Decision Information (from ISO 10181-3). The portion (possibly all) of the ACI made available to the ADF in making a particular access control decision.

AEF – Access control Enforcement Function (from ISO 10181-3). A specialized function that is part of the access path between a user and a target on each access request and enforces the decision made by the ADF.

Client – the entity making a decision request to the ADF (it could be the target, the user, or a proxy acting on behalf of the user)

Contextual information – Information about or derived from the context in which an access request is made (e.g. time of day).

Environmental parameters – same as contextual information.

User – An entity (e.g. human user or computer-based entity) that attempts to access other entities (from ISO 10181-3).

Privilege – An attribute or property assigned to a user by an authority

Target – An entity, usually a resource, to which access may be attempted (from ISO 10181-3).

1. Introduction

This report describes the similarities and differences between the Akenti and PERMIS authorisation infrastructures. It describes the features, ease of use and performance statistics of both authorisation infrastructures. This report has been produced from a desk comparison of the available published documentation, by talking to the authors of both infrastructures, and by building both infrastructures along with a test application. The performance statistics are limited to some extent, in that it was not possible to build multiple arbitrarily complex policies in the time available, and also in order to perform a fair comparison between the two, we did not run Akenti as a stand alone server.

2. Overview

Akenti [Akenti] is an authorisation infrastructure from the Lawrence Berkeley National Laboratory in the USA. PERMIS is an authorisation infrastructure from the EC funded PrivilEge and Role Management Infrastructure Standards validation (PERMIS) project [Permis]. Both the Akenti and PERMIS Authorisation Infrastructures are trust management infrastructures according to the definition of Blaze [Blaze], and have the 5 components necessary for this, which are:

i) A language for describing `actions', which are operations with security consequences that are to be controlled by the system.

ii) A mechanism for identifying `principals', which are entities that can be authorized to perform actions.

iii) A language for specifying application `policies', which govern the actions that principals are authorized to perform.

iv) A language for specifying `credentials', which allow principals to delegate authorization to other principals.

v) A `compliance checker', which provides a service to applications for determining how an action requested by principals should be handled, given a policy and a set of credentials.

Both infrastructures have similar architectures [Johnston] [Chadwick]. This comprises the compliance checker, called the Akenti server by Akenti [Thompson], and the Access Control Decision Function (ADF) by PERMIS (after the ISO Access Control Framework [ISO]). Both have a gateway controlling user access to resources, called the Resource Gateway by Akenti and the Application Gateway by PERMIS. Both of them write their policies in XML, and store their policies in certificates. Both of them store their user credentials as certificates in LDAP directories. Hence on the face of it, the Akenti and PERMIS authorisation infrastructures seem to be almost identical.

However at the implementation level Akenti and PERMIS are very different. Akenti is written in C++, Permis in Java. The Akenti compliance checker can be called either via a function call in the gateway or as a standalone server via TCP/IP, whereas the PERMIS compliance checker is invoked as a Java object in the gateway. PERMIS credentials are built according to the latest X.509 standard [X509], whereas Akenti credentials are built in a proprietary format using XML syntax [Akenti]. Akenti requires the user to be PKI enabled and to present an X.509 public key certificate at authentication time, whereas PERMIS is authentication agnostic and leaves it up to the application to determine what type of authentication to use. Whilst both PERMIS and Akenti policies are written in XML, their DTDs are very different [Thompson] [IFIP]. PERMIS policies are held in one policy X.509 Attribute Certificate, whereas Akenti policies are hierarchical and distributed between proprietary Policy Certificates and Use-Condition Certificates. Akenti has concentrated on classical access control lists (discretionary access controls) whereas PERMIS has implemented role based access controls. Therefore at a practical level there are a significant number of differences between the two infrastructures, and it is these differences that are described in more depth below.

3. Policies

The Akenti policy is distributed and hierarchical. It comprises two components: Use-Condition certificates and Policy Certificates. A Use-Condition certificate places requirements on the attribute certificates that users must have in order to gain access to a resource. A Policy Certificate states the overall policy for controlling access to a resource, and holds the trusted CAs and Stakeholders, and pointers (URLs[1]) for searching for Use-Condition certificates that are applicable.

Policy Certificates comprise a root policy certificate, and optionally subordinate policy certificates that inherit from the root policy. Akenti sees the target as a tree of resources, e.g. a filesystem with subdirectories. Each of the branches (subdirectories) can have a policy of its own, but in addition to that the policy of the superior branch (directory) is inherited. Each of the policies can be issued by a different Stakeholder.

A stakeholder is a special kind of authority that is trusted to issue Use-Condition certificates. Each stakeholder can impose his own access control requirements independently of other stakeholders. One of the stakeholders signs the Policy Certificate. A stakeholder in Akenti is equivalent to a Source of Authority (SOA) in PERMIS.

Use-Condition certificates contain the name of a target resource, a condition (which can be a constraint), a critical flag[2], the authorities (CAs for X509 attributes, AAs for Akenti attributes) of the certificates with the attributes to be matched against these conditions, plus a list of rights/privileges that are granted. Conditions may include identity attributes that users must have (e.g. CN), role or group memberships (e.g. groupX) and environmental parameters (e.g. time of day). Rights are comma separated lists of actions

[1] Page: 3
http:, https:, ldap: and file: protocols are possible, thus enabling storage of the certificates in Web directories, LDAP directories and filestores. However, Akenti does not specify the LDAP schema for storing their UCCs and Attribute Certificates, which thus makes LDAP effectively unusable.

[2] Page: 3
If the UCC is Critical (flag set to true), the UCC must be satisfied, in the sense that if the condition fails to be satisfied, no access can be granted at all, irrespective of what other UCCs allow. If the UCC is Non-critical (flag set to false), the UCC only defines rules for one access control condition and it does not affect the decisions made by other UCCs.

on targets. Action names have to be unique for the whole domain of resources, irrespective of the target type. The AA trusted to issue each attribute value must be specified exactly (no ordering of values is provided to allow for implied permissions to issue certificates with subordinate values), but each Use-Condition can include different AAs for each attribute. By way of example, a stakeholder can specify that in order to read or execute a process on a target the user has to possess an attribute named CN with a value matching one from a given list (thus modelling DAC), signed by CA A. A detailed description of a Use-Condition certificates can be found in [UCC] [AkentiCerts].

The attributes issued to the users in attribute certificates, are independent of each other, and cannot form a role hierarchy (i.e, in which superior roles inherit the privileges of subordinate roles). Akenti supports the distributed management of attribute certificates, and an external AA may assign attributes to users if the Use-Condition certificate lists the AA under the relevant attribute value.

Contents of an Akenti Policy Certificate:

?? Name of the resource to which this policy applies
?? List of information about trusted CAs including:
    ?? Distinguished Name
    ?? public key certificate (can be self signed).
    ?? list of places to search for Identity Certificates issued by this CA (optional)
    ?? list of locations where CA stores its CRLs (optional)
?? list of Use-Condition issuers (defines the resource stakeholders)
?? list of URLs to search for Use-Condition certificates (could name a single Use-Condition, a directory containing hash-named Use-Conditions, or could be a search script)
?? optionally URLs to search for user attribute certificates
?? maximum time in seconds that any certificates that are used in satisfying conditions for this resource may be cached.

The whole policy is signed by one of the stakeholders and must be stored securely to stop it being switched for another one (or deleted altogether). Similarly, policy hierarchies are not specified in a secure way, since there are no pointers from superior to subordinate policies or vice versa. Consequently the policies must be stored in secure directory hierarchies, and the directory hierarchy determines the policy hierarchy.

A full description of the Akenti policy can be found in [AkentiPolicy] [AkentiCerts]

The PERMIS policy is one object, and is stored in an LDAP directory as a policy attribute certificate. It supports classical hierarchical RBAC, in which roles are allocated to users and privileges to roles. Superior roles inherit the privileges of subordinate roles in the hierarchy. Multiple disjoint role hierarchies can be specified. PERMIS has a very loose definition of a role; a role may be any attribute assigned to a user, not just a conventional organisational role. PERMIS supports the distributed management of attribute certificates, and multiple external SOAs can be trusted to issue roles/attributes.

Thus users can be certified externally to the domain the attribute certificates will be used in. The policies are kept in the LDAP entry of the policy issuer, and different policies are distinguished by their unique object identifiers (OIDs). There is no need for the policies to be kept securely, since the PERMIS engine validates the policy at run time to ensure it is the correct one (i.e. has the correct OID and is signed by the SOA). However, the name of the SOA has to be securely configured into the PERMIS application at start up. Policy hierarchies are not supported by PERMIS. These can either be enforced organisationally by management, or by the application instantiating several PERMIS decision engines, one per level of the hierarchy, and ensuring that each level grants permission.

The PERMIS Policy components comprise:

?? Policy OID, so the policy can be distinguished among others stored in the SOA's entry
?? Subject domains, specified as LDAP subtrees, which are the subjects who can assert roles
?? Target domains, specified as LDAP subtrees, which are the targets governed by the Target Access Policy
?? List of the distinguished names of trusted external attribute certificate issuing authorities (SOAs) which are treated as roots of the delegation trees
?? Role hierarchy specification (lists the roles as ordered attribute values)
?? Role assignment policy, telling which attribute authorities are trusted to issue which roles to which subject domains, and whether delegation is supported or not
?? Action policy, saying what the actions and their parameters are, so they can be referenced in the Target Access Policy
?? Target Access Policy, which specifies the set of roles/attributes required to perform a particular action along with any conditions.

A full description of the PERMIS policy can be found in [IFIP].

4. Policy Conditions

In PERMIS conditions are placed on which attribute certificates can be trusted (in the Role Assignment Policy) and on which attributes have which privileges and when (in the Target Access Policy).

In Akenti conditions are placed on which attributes certificates can be trusted and on which attributes have which privileges and when (in the Use-Condition certificates).

PERMIS therefore contains a level of indirection, in that principals are assigned attributes, and attributes are given privileges (i.e. RBAC). Akenti however can support DAC and $RBAC_0$, in that principals can be given privileges or group membership, and group attributes can be given privileges.

5. Infrastructures

Both PERMIS and Akenti have a Source of Authority (SOA) or equivalent entity that creates a Policy. PERMIS uses the X.509 terminology SOA, Akenti calls it a Stakeholder. In Akenti terminology there can be multiple stakeholders participating in administering the resource, in PERMIS there can only be one SOA for the target resource (although external SOAs can be trusted to issue ACs). In Akenti all the stakeholders can issue Use-Condition certificates, in PERMIS only the SOA can issue the equivalent functionality as part of the overall Policy. In PERMIS only the SOA creates the Policy and digitally signs it. In Akenti, one of the stakeholder peers creates and signs the root policy and should places it in a secure store. Any stakeholder named in the root policy can sign the subordinate policies. Both infrastructures must be configured with the CA (authentication) roots of trust. In PERMIS it is an application dependent matter how this is configured into the system. In Akenti it is part of the policy.

Both PERMIS and Akenti recognise separate hierarchies for authentication (CAs) and authorisation (SOAs or stakeholders). The Akenti authorisation hierarchy leads from the policy signing stakeholder to other stakeholders, and via Use-Condition certificates to subordinate Attribute Authorities (so it is more a mesh than a hierarchy).

Fig. 1 below shows the Akenti PMI. In it there is a Policy object, specifying trusted stakeholders, who can issue Use-Condition Certificates (UCC) to the resource (the Target domain). The Use-Condition certificates specify the trusted AAs. Each of the AAs can issue Attribute Certificates (AC) to the users of the system (the Origin domain).

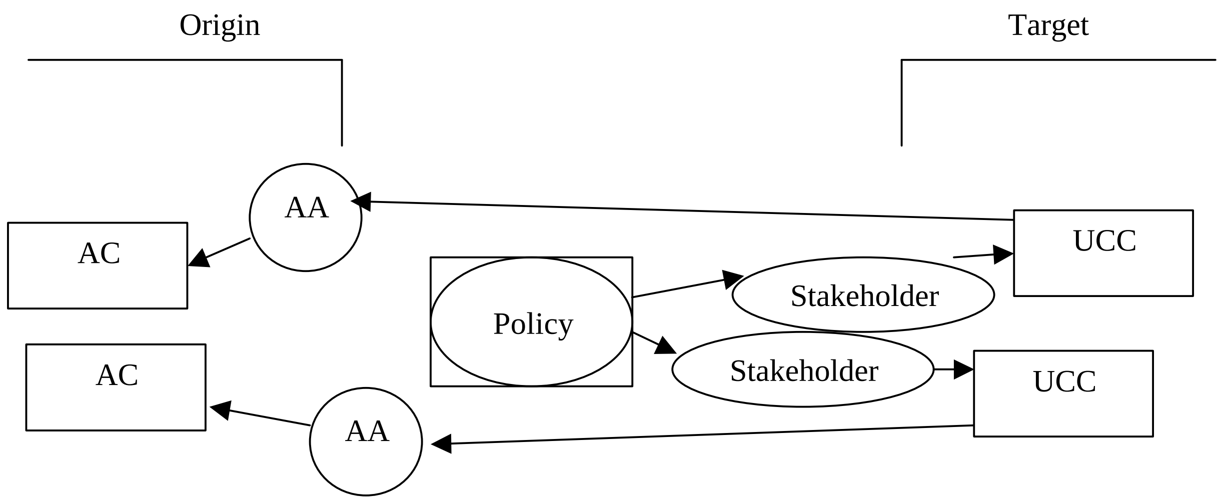

Figure 1. Akenti Privilege Management Infrastructure

In the PERMIS infrastructure the AAs form a hierarchy. Fig. 2 displays the PERMIS PMI. In it the Policy is issued by the root SOA and specifies a set of trusted foreign SOAs (each being the root of a foreign PMI). These SOAs (and their subordinate Attribute Authorities if allowed by the policy) can issue Attribute Certificates to the users. (Note that the current release does not interpret AAs).

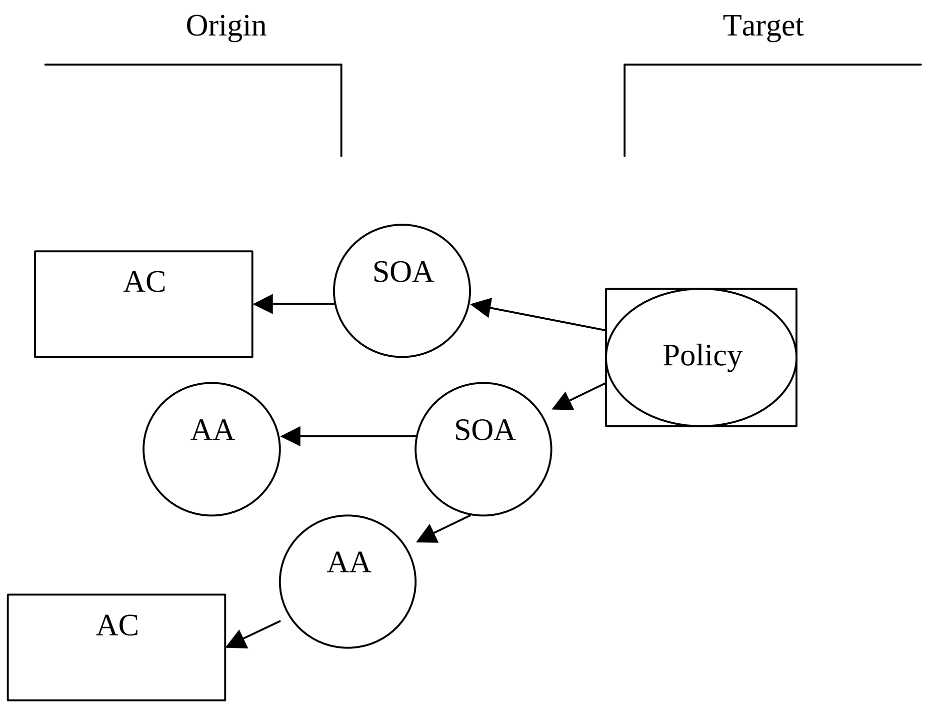

Figure 2. PERMIS Privilege Management Infrastructure

6. Trust Chains

In PERMIS, the target trusts itself and is securely configured with its SOA name (authorisation root of trust). The policy is signed by this SOA so the policy can be trusted. The policy contains the names of remote SOAs who are also trusted to issue attribute certificates. Attribute certificates must be signed by one of the trusted SOAs or their subordinates (and conform to the Role Assignment Policy) or they will be discarded. Attribute certificates contain the distinguished names of their holders (users). In PERMIS, the root of trust in authentication is the responsibility of the application, and PERMIS trusts the application to properly authenticate the users and to validate the digital signatures on attribute certificates. A user must authenticate himself to the application to prove that he is identified by a given DN, and then PERMIS can trust that attribute certificates containing this DN belong to the user.

In Akenti, the root policy is signed by a trusted stakeholder and must be securely configured into Akenti. The root policy lists the other stakeholders who are trusted to issue subordinate policies and Use-Condition certificates. Use-condition certificates state which AAs are trusted to issue which attribute certificates. Users who present attribute certificates must eventually digitally sign something to prove that they are the holder of the private key corresponding to the public key held in both the returned Capability Certificate and in the PKC referred to in the attribute certificate (the AC holder is referred to by his DN and the DN of his CA issuing his PKC). Akenti can then trust that this attribute certificate belongs to this user. The attribute certificate is further checked to ensure that it is signed by a trusted AA, and that it conforms to a trusted Use-Condition certificate. Finally, Akenti issues a Capability Certificate to the client and inserts the user's public key into this.

7. Attribute Certificates

In both systems, attribute certificates are issued to users, and hold their privileges (either directly or indirectly via an attribute/role).

Akenti uses XML certificates in their own proprietary format. (Note that this format has changed between releases, so that V1.1 wont work with V1.2). The certificates can be stored in LDAP, HTTP or a file repository (but since no LDAP schema is defined this effectively rules out the use of LDAP). A user is identified via his LDAP DN and the DN of the CA issuing his public key certificate (and he has to prove ownership of the private key corresponding to this public key certificate). Attributes comprise a type and value.

PERMIS uses BER/DER-encoded attribute certificates in X.509 standard format. A user is identified by his globally unique X.500/LDAP distinguished name, and a user has to be authenticated against that name. Attributes have a type and value, and attributes can form arbitrarily complex role hierarchies. The certificates are stored in LDAP repositories, using standard LDAP/X.500 schema. The base code can be extended to support other repositories with LDAP-based naming conventions.

8. Decision Making

PERMIS operates in multi-step decision making mode. In step 1, *getCreds*, the user's credentials are obtained and validated, and roles conforming to the policy are passed back to the calling application for caching. This typically takes place during user login. In step 2, *decision*, the requested action and target are passed, along with the user's validated roles, and a simple Boolean decision is returned, either granting or denying access. Step 2 can be repeated as often and as many times as required for different targets and different actions, as the user attempts to perform different tasks.

Akenti only operates in single step decision making mode, but is able to make different types of decisions. The client can ask "What can a user do?", as well as the traditional "Can this user perform this action on this target?".

Akenti always embeds its response in a Capability Certificate for export back to the client. The Capability Certificate comprises the public key of the user, the DN of the user and his CA's DN, the name of the resource, and the privileges that the user enjoys, optionally with a list of conditions attached to each of them. The Capability Certificate is then signed by Akenti and given to the client. The user can subsequently present this to a gatekeeper for improved performance. The gatekeeper, which holds the Akenti public key, merely needs to check the signature on the capability, then ask the user to sign a challenge, before granting (or denying) the user access to the resource.

PERMIS has no ability to return Capability Certificates.

9. Software Architecture

Akenti: C++ classes and dynamic link/shared libraries. The API classes can be used by C++ programs. The standalone authorisation module can receive authorisation requests via the network over an insecure connection or SSL.

PERMIS: Java classes to be used by the authorisation Java program.

Akenti has a set of modules. Some of them are standalone, some represent the API implementation that can be embedded into the application directly. The Akenti web-security module can be attached to an Apache web-server for Unix platforms.

PERMIS is an API implementation only. There is no standalone authorisation module at the moment.

10. Authorisation engine user requirements

The user sending a request to the Akenti PMI must be PKI enabled. User authentication is done via signature verification. Akenti does not require the user to sign anything, but his public key will be used for authentication by the gatekeeper receiving the user's Capability Certificate. The root of trust that issues the policy and stakeholders issuing the use conditions must be PKI-enabled. So must the AAs issuing ACs.

The user sending a request to the PERMIS PMI does not have to be PKI enabled. Any type of authentication can be used. PERMIS is authentication-agnostic. The only requirement is that the AEF authenticates the user, and maps the authenticated identity into the user's DN in the attribute certificate (in many cases they will be the same). However, the AAs and the SOAs have to be PKI-enabled, because the signatures on the attribute certificates have to be verified.

11. Applicability

The Akenti standalone module can be applied in any system with a TCP connection to the network. The Akenti Web-server authorisation module can be embedded into the Apache web-server.

The PERMIS API does not have any "shell" that would receive requests in any particular manner. Bologna Municipality have developed an authorisation servlet for their web-server.

12. Administration: Allocating Privileges and Setting Policies

In Akenti there is a special command line tool for creating the Policy, Use-Condition and Attribute Certificates. A GUI tool can also be used, and if a Resource Definition Server is running this will ensure that the administrator conforms to the policy when issuing ACs.

In PERMIS, there is a GUI application, the PA (Privilege Allocator) that is used to create and sign policy ACs and user ACs and store them in an LDAP directory. There is also a programmable API that can used for the bulk creation of ACs. (The piloting partners from Barcelona have also produced a programmable PAT (Privilege Allocating Tool) but we are unsure about its release conditions). The PA and API will issue any type of AC, and it is the administrator's responsibility to ensure that the contents are correct.

13. Ease of Installation and Use

13.1. Installing and Using PERMIS

(text provided by Mary Thompson of LBL)

13.1.1 Obtaining the download

I downloaded onto a Solaris 5.7 machine, so I did not have the Word instruction file open at the time. Thus it took me some time to find the pbs-sample.zip file. But looking at your Web page today, I see that you have the instructions in html, so that would no longer be a problem.

13.1.2 Setting up your environment

All I had to change were the pathnames in the sample1.cfg \ -> / .

13.1.3 Installing the software

This was very straightforward. I did not need to compile anything and already had the Xerces xalan software installed. Running the pab example worked just as advertised.

13.1.4 Using the software

The pab example was very simple to use. Editing the existing TEST1.xml was fairly obvious. I gather that there are tools to at least sign a policy and maybe to help you edit one. But all I could find was the class documentation, which did not make it clear which classes to invoke or what the arguments might be. It looks like the kernel_app.bat will get me some sort of GUI, but it is late and I am running the Java remotely which is never very satisfactory.

13.1.5 Creating a basic policy

I created the "medium" ALS policy in about 2-3 hours using the two example policies, the DTD and the "RBAC POLICIES IN XML FOR X.509 BASED PRIVILEGE MANAGEMENT" document [IFIP].

13.1.6 Creating attribute certificates

I did not try this. I did not see any examples of text based certificates or find any tools to create the ASN.1 certificates. The BER viewer *{from Aram Perez }* does not work on my Linux, Solaris systems.

13.1.7 Comments on documentation

I downloaded the pa_doc021218.zip file which promised to contain a user manual but

seemed to be only the Java class documentation. If there is a user manual it would be good to link to it from your tutorial page.
(Editor's Note. This has now been done)

13.2. Installing and Using Akenti

(Text provided by Sassa Otenko of University of Salford)

13.2.1 Obtaining the download

This was very straightforward.

13.2.2 Setting up your environment

No specifications about Linux versions were given. I had problems with trying to compile the code on Linux Red Hat 8.0. I also had problems with running the JRE that they provide on Linux Red Hat 8.0, and therefore could not create the certificates.

The instructions are quite detailed, but not all of them reflect what the code actually does. E.g. the configuration file for the UCC and Attribute Certificate creation tool looks for the *.htauthority* file only, although there is a configuration parameter to specify an alternative file name.

There were no instructions about the Apache configuration. However, the Akenti engine refused to work with Apache 1.3.24 running several Virtual Hosts and could not retrieve remotely stored certificates from it.

There were no instructions about the naming convention for the certificates. After talking to the developers I have been given imprecise instructions, which led to delays in setting up the environment: Akenti refused to pick up seemingly correct certificates.

13.2.3 Installing the software

Installation was straightforward, because it consisted of unpacking the TAR-GZip archive. No extra libraries were required. Compilation was necessary to get a working standalone Akenti server and the performance tester. Except for the aforementioned problems with the compilation of Akenti on Linux Red Hat 8.0 with GCC3.2 compiler, there were no problems with compiling both modules and they worked straightaway.

There were no problems connected with configuring the server or performance tester.

Installation of JVM was required. The JRE provided with the binary distribution failed to work on Linux Red Hat 8.0. There were no instructions as to what JVM is preferred, but it appeared that the IBM JVM does not provide the (optional) security provider the code uses. I had to install the Sun JVM.

13.2.4 Using the software

There are examples and test programs provided. Modification of the example program is straightforward.

If using the standalone Akenti server, there would be more code required than when using a built-in Akenti authorisation module accessed via API. In the latter case all the coding required is only for collecting user request parameters and enforcing the decision (the same as for PERMIS). Note that in the case where the returned Capability Certificate/decision contains a conditional permission, more code would be needed to evaluate the condition (i.e. additional code as compared to PERMIS, as PERMIS always gives the final decision).

There were no instructions on how to compile your own program with the embedded Akenti module, so I have copied the example program and updated the way the API calls were made. The resulting performance tester cannot be moved from <akenti distribution directory>/src/exampleTools, which is a bit limiting.

The documentation does not specify what libraries are required for the Akenti standalone server (or any program using the embedded Akenti decision making module). This makes it impossible to move the program to a different location on the PC.

13.2.5 Creating a basic policy

There are a command-line tool and a GUI tool for creating a policy. Writing a basic policy is easy, but there were several problems with using the program itself, which caused a delay of over two weeks and forced the Akenti team to release several updates of their binary distribution.

I did not use the command-line tool. The problems with running the GUI tool are:

- the JRE provided with the binary distribution failed to run on Linux Red Hat 8.0; I had to change the script running the GUI tool (some knowledge of the scripting language was required)
- it ignores some of the configuration parameters, like the name of the policy file, when creating Use Condition Certificates
- it failed to sign the policy and certificates when running the IBM JVM instead of the Sun JVM - no security provider was installed; the documentation does not say anything about this
- there were several problems with the GUI itself:
    - ?? inconvenient LDAP search dialogs (they assume that the DN will always contain an OU component in it)
    - ?? the file open dialog for the policy file does not display anything (when picking up the policy for creating ACs and UCCs)
    - ?? many Cancel buttons do not work (Akenti developers comment that this is a usual problem for various Linux versions)
    - ?? the initial version of Akenti tools (downloaded after April 25) generated a wrong policy (generated 'AND' instead of '&&', so the engine failed to understand the policy)
    - ?? the initial version of Akenti tools (downloaded after April 25) did not generate the XML of the certificate, though this was expected, as per their documentation

?? the subsequent versions of Akenti tools do put the XML into certificates, but it is completely ignored by the decision-making engine (therefore it can be tampered with and be of incorrect syntax altogether); having the XML policies and certificates has no benefit, because they could rather be expressed in English; indeed, having XML inserted in the certificates only deteriorates performance, as the whole certificate has to be transmitted via the network

13.2.6 Creating attribute certificates

Same as above. The problems encountered were connected with running JVM and the interface was not especially convenient in certain respects (LDAP search facility, file open dialog, cancel buttons). It is important that the GUI contains hints displayed at the top of the window, which makes the process of certificate creation straightforward. Note however, that the interface differs from what the documentation shows.

13.2.7 Documentation

From a user prospective the documentation does not fully correspond to what the code does, or how the GUI looks. From an administrator prospective the documentation dedicated to particular certificate types did not contain detailed enough information. There is a document with all the XML tags used in Akenti described in much detail, but it is difficult to use, because I don't know what tags to look for in a particular kind of certificate.

The Akenti documentation seems to allow UCCs and other certificates to be kept in LDAP, but they do not provide the schema for that. The documentation contains old names of the scripts to run.

13.3. Code Sizes

We compared the sizes of the test programs, assuming they provide the same functionality. The PERMIS testing program with the PBA engine is a JAR file of approximately 200KB, which makes it about 900KB with all the necessary libraries (XML parsing, cryptography etc.).

We must also add the size of the JRE, which is about 130MB for IBM’s JVM, but this can be significantly reduced by removing all graphics related modules, if only the PERMIS engine is used on the computer. Note that Sun’s JVM is at least twice as small, but we did not run tests on that JVM. There are even smaller versions of JVM, which are sufficient to run PERMIS (i.e. Sun JVM 1.2.2 for Windows is about 27MB in size).

The Akenti executable file is about 16MB (compiled for Linux), plus it requires additional shared libraries, the total size of which amounts to another 6MB. The Akenti developers report that the engine is even bigger on Solaris computers and may exceed 50MB [Private Correspondence with LBL].

Note that Akenti also needs a JRE for certificate signing.

14. Performance Testing the Akenti and PERMIS APIs

14.1. Test Environment

This is an outline of the implementation of the test environment for the Akenti-PERMIS performance testing. It describes the PKI structure, LDAP directory structure and the number of certificates issued for each kind of test.

There will be two kinds of test: Basic and Medium, the level being defined by the complexity of the policy and the number of users participating in it[3]. For each of them a number of access requests will be issued and the following resource consumption measured:

- time it takes to collect and verify the subject's credentials (Akenti ACs, PERMIS ACs – a call to *getCreds*). PERMIS will also perform time measurement for a call to the *decision* method, which will be output as a separate set of measurements to compare how that affects the decision-making process. The Akenti engine does not have an equivalent two-step decision process so this measurement cannot be taken for it.
- Memory used during the above processes
- CPU usage statistics

The requests will be fed into two specialised programs embedding the respective APIs. The requests will be formed as a text file containing text input for the programs, i.e. User DN, Target Name, Action Request, etc. The programs will process the files to the end and output the statistics onto the standard output, which can be redirected into an appropriate file.

The measurements will be output in text format, which can be parsed for processing using MS Excel. Appropriate graphs can be built as a result.

During the tests there will be the default cryptographic configuration for the Akenti API (i.e. Public Key Cryptography). The PERMIS API will be tested in two modes: a) no signature verification on the ACs; b) full signature verification on the ACs, using the same PKI as Akenti. Users will be authenticated with no cryptography involved (i.e. the users will simply provide their names), because the PERMIS API is user authentication agnostic and it would be unfair to require Akenti to use cryptography for this purpose. Therefore the Akenti test program will also be provided with a user name, and, as the engine prescribes, the CA name as well. The CA "authenticating" the user will be set to Akenti Root CA (see PKI section).

14.2. Hardware and Software configuration

Hardware: PII 500MHz, 256MB RAM, 20GB SCASI hard drive.
Operating System: Linux Red Hat 8.0, kernel 2.4.18.
Software: JRE 1.4.0 IBM.

3 Note that we had initially wanted to perform three tests, including an Advanced one, but this was frustrated by not being able to find an exemplar advanced policy to implement.

14.3. PKI and PMI

Both authorisation engines will be configured with the same PKI and similar PMIs.

The PKI consists of the Root CA that directly certifies all entities participating in the authorisation process. These are: the root authorisation authority (Stakeholder in Akenti, SOA in PERMIS), other authorisation Attribute Authorities (Stakeholders and AAs in Akenti, multiple SOAs in PERMIS). The PKI entities will be issued with an RSA 1024-bit key pair, using OpenSSL free software.

Root CA: cn=Root CA, o=PERMIS, c=GB

Other subjects and authorities will be created as required by the policies and their PKCs will be kept in the LDAP entries of their holders. In both cases the PKCs will have to be retrieved from the remote site for a fair comparison, even though the retrieval algorithms may differ (PERMIS will use a *DefaultSecurity* as a signature verification object, which will retrieve a PKC each time a signature has to be verified; the Akenti algorithm of retrieval of UCCs is unknown to us).

The Akenti PMI will consist of one Stakeholder for the Basic tests, and of multiple Stakeholders for the Medium tests. This will measure how the number of stakeholders affects performance. There will be as many Attribute Authorities as required for the respective policy. The Akenti Policy will always contain only one CA, the root CA.

The PERMIS PMI will consist of one root SOA. There will be as many additional SOAs as required for the respective policy.

Akenti is given a signing key (RSA 1024-bit) and a PKC, issued to:
Akenti DN: cn=Akenti Server, o=PERMIS,c=GB

14.4. LDAP and Web

The PERMIS ACs and PKCs and the Akenti PKCs were stored in an LDAP server. It was not possible to store the Akenti ACs in an LDAP server as we were unable to determine the schema required for this. Instead, the Akenti certificates were stored on a Web server as separate files in a web directory.

The LDAP and Web servers ran on the same Linux machine - a P166MHz with FPU computer with 128MB RAM, 2GB+10GB IDE hard drives, 10MB Ethernet card. It had Linux Red Hat 7.2 installed on it. The Web server was Apache Web-server v1.3.24. The LDAP server was OpenLDAP version 2.0.23. The Linux machine was in the same segment of the network (under the same router) as the PC running the test programs.

The performance measurements were taken at different times during the day to account for the varying amount of local traffic, which may affect the LDAP and Web servers performance.

The LDAP schema is given in Appendix A.

The user entities of the tests will be entries with the pmiUser objectClass. Their Attribute Certificates will be kept in the attributeCertificateAttribute for PERMIS. It was not possible to store Akenti ACs in LDAP.

The SOAs, Stakeholders and Attribute Authorities will also be entries with pkiUser objectClass and will have their PKCs kept in the userCertificate attribute.

14.5. Simple Policy

The simple policy is for controlling access to one server in a department (e.g. a print server).
All staff in the department can write files to laser printer x, Jim the administrator can write files, delete any files from the print queue, pause the printing, and resume the printing at the laser printer x. No-one else is allowed access to the printer.

The policy has been encoded to govern the ou=Venables,o=permis,c=gb department. The SOA is cn=SOA,o=permis,c=gb. Any users with a DN ending with ou=Venables,o=permis,c=gb are allowed to print on the cn=printer,ou=Venables,o=permis,c=gb printer. Any users from the ou=Venables,o=permis,c=gb department with administrator role can also pause, resume print jobs, and delete jobs from the queue.

Jim: cn=Jim,ou=Venables,o=permis,c=gb is issued with an AC with the permisRole attribute with value “administrator”. The AC is signed by the SOA.

The Policy encoding for PERMIS and Akenti are given in Appendix B.

14.6. Medium Policy

The medium policy was provided by the Akenti project at LBL. It is the policy for controlling access to the Advanced Light source at LBL.
The director of the lab wants to make sure that citizens of Iraq, Iran and North Korea can't touch the Advanced Light source, no matter what. The director of the facility wants to be sure that everyone who touches it has taken and passed the lab's X-ray safety training course. The PI for the project wants to allow access for his group members between the hours of 8am-8pm PST and for his colleague's group between 8pm and 8am PST. The role of leader is allowed to control, operate and observe experiments, the role of experimenter is allowed to operate and observe experiments and students are allowed to observe experiments.

The policy encoding is as follows:
The signer of the policy is cn=SOA,o=permis,c=gb. It authorises other SOAs to issue ACs.

Jim: cn=Jim,ou=Venables,o=permis,c=gb is issued with an AC with “experimenter” role by the Director SOA (boss), “Jones” group by PI SOA (Joe Jones), “lbnl-xray-101” by X SOA (smith).

Bob: cn=Bob,o=lbl,c=us is issued with an AC with “student” role by the Director SOA (boss), “Doe” group by Colleague SOA (Jane Doe).

Judy: cn=Judy,o=lbl,c=us is issued with an AC with “student” role by the Director SOA (boss), “Doe” group by Colleague SOA (Jane Doe), “Jones” group by PI SOA (Joe Jones), “lbnl-xray-101” by X SOA (smith).

Sharon: cn=Sharon,o=lbl,c=us is issued with an AC with “leader” role by the Director SOA (boss), “Jones” group by PI SOA (Joe Jones), “lbnl-xray-101” by X SOA (smith).

The Policy encoding for PERMIS is given in Appendix B.

14.7. Overview of Test Results

There were significant problems when trying to use Akenti. The problem was aggravated with the distance between our testers and the developers, which caused big time delays between questions and answers because of the time difference (note that the testers were located in UK, GMT+1, whilst the Akenti developers were located in USA, California, GMT-7).

The testing programs are available at [Raw].

The statistics were collected for various configurations for the PERMIS and Akenti engine. 120 access requests were issued for the Simple Policy and Medium Policy[4].

The same sequence of requests has been used for testing the PERMIS engine in four modes, thus producing four samples:

1. L+C, communicating to LDAP and performing cryptographic checks using *issrg.security.DefaultSecurity*
2. L-C, communicating to LDAP and performing no cryptographic checks on the ACs; this can be compared to sample 1 to see how much of the time is devoted to cryptography
3. C-L, performing cryptographic checks on the ACs, but the ACs are stored in memory (*issrg.repository.VirtualRepository*) and are loaded at initialisation time; this can be compared to sample 1 to see how LDAP operations affect performance of the system
4. –C-L, performing no cryptographic checks on the ACs, and ACs are stored in memory, like in 3; this can be compared to sample 2 to see how LDAP affects

[4] Due to the problems with installing Akenti we have only succeeded to collect Simple Policy test results for Akenti. Further, Akenti does not evaluate any SYSTEM attributes, like time, and it does not have plug-in mechanism yet, and so is unable to give a simple granted/denied response for the Medium Policy.

productivity of the system, and can be compared to sample 3 to see how cryptography affects the PERMIS system.

The times were taken in four modes for *getCreds* only[5]. The time spent on *decision* is not affected by the mode of operation, and the average for it is calculated over all samples. The Excel workbooks with graphs and raw data are provided at our web site [Raw]. We only provide the average times and standard deviations, with no other manipulation of the raw data.

The same sequence of requests were issued to the Akenti engine (but only the Simple Policy mode was tested)[6]. The engine was tested in the following modes:

1. L+C, when the identity certificates (X.509 PKCs) are stored in LDAP, but the rest of the certificates are stored on a web-server, and the returned Capability Certificates are signed
2. R+C, when all certificates (including identity certificates) are stored on a web-server, and the returned Capability Certificates are signed
3. C-L, when all certificates are stored on a local hard drive, and the returned Capability Certificates are signed
4. L-C, the same as 1, but the returned Capability Certificates are not signed (setting in the configuration file)
5. R-C, the same as 2, but the returned Capability Certificates are not signed
6. -C-L, the same as 3, but the returned Capability Certificates are not signed

It is not possible to completely switch cryptography off in Akenti without editing the C code and recompiling. The sequence of these tests is repeated with caching switched on and off.

Note that switching the cryptography off in Akenti is not the same as switching cryptography off in PERMIS. In PERMIS it means that no signature verification on any attribute certificates is done. In Akenti it only affects the creation of the capability certificates sent as the reply of the Akenti engine. If cryptography is on, the capability certificate returned by the Akenti engine is signed prior to returning to the client. If cryptography is off, the capability certificate returned by the Akenti engine is unsigned, and the signatures on the capability certificates are not checked. This setting also affects the signing and signature verification of cached certificates.

It was noticed that performance of the Akenti engine can be affected by the way the URLs are specified for the certificate stores. The engine prefers to issue a request to the

[5] In fact, raw data output from the performance tester contains time and memory measurements for both *getCreds* and *decision*. However, statistical analysis of the time was performed in the four modes only for *getCreds*, since the values for *decision* were not affected by the mode.

[6] Note that there is a slight difference in configuration. The PERMIS engine retrieves and parses all the ACs in the LDAP server, for both the Simple Policy and Medium Policy modes (i.e. retrieves more data than is actually needed) and invalidates the certificates from the Policy not under test. However, the Akenti engine retrieves only the certificates for the Simple Policy test (i.e. does not retrieve or parse any redundant data), because no certificates have been created for the Medium Policy test.

Web server requesting the directory with certificates as is, and if that fails with error code 301 (Moved Permanently), it makes another attempt with '/' appended to the URL[7]. Therefore two separate sets of test have been run. One set had all URLs with the trailing slash present for all Web certificate stores (referenced as Full URL in the tables below). The other set had all URLs without the trailing slash present for such stores (referenced as Chopped URL in the tables below).

The measurements of PERMIS and Akenti that can be meaningfully compared are:

1. **Single Decision Making**. PERMIS L+C to Akenti L-C and R-C with caching off In this case PERMIS retrieves all of its certificates from a remote LDAP site, performs all cryptographic checks, and returns the internal representation of the credentials. Akenti in mode L-C and R-C is doing similar tasks only fetching certificates from an LDAP or Web server. Switching cryptography off in Akenti means that the engine will not sign the returned Capability Certificate, which is approximately the same as a PERMIS return.

2. **Multiple Decision Making**. PERMIS *decision* to Akenti -C-L with caching on (minus the time for the first call)
   In this case PERMIS has validated the credentials (via *getCreds*) and returned validated roles to the application for caching. The application can then call *decision* multiple times for the same user. Akenti has validated the credentials the first time they are used, then caches them internally, and uses the cache for subsequent decision making for the same user.

The behaviour of the engines in the rest of the tests is incomparable. The measurements for these tests are provided to illustrate how the engines behave under various circumstances.

14.8. The Test Results

The raw unprocessed data can be downloaded from [Raw]. The tables below simply provide the average and standard deviation values.

14.8.1 PERMIS Results

In the Simple Policy, the average number of certificates processed per request is 1.6 ACs (48 requests for Jim with 4 ACs, 3 of which are redundant; 66 requests for Adam with no ACs, 6 requests for Sarah with no LDAP operations)

[7] The Akenti developer's rationale for this is that the URL could point to either a file (without a slash) or a directory (with a slash). By adding a slash to URLs without them allows Akenti to be user friendly in cases where the stakeholder entered the wrong syntax into the policy. Akenti will still return a Capability Certificate to the user, at the expense of worse performance. The alternative would have been to return an error to the user, when the user was not at fault, but because the policy was badly configured.

Table 1. PERMIS Simple Policy Test

	L+C	L-C	C-L	-C-L
Initialisation time(seconds)	1.521	0.833	1.047	0.788
Time for getCreds, average (ms)	142.217	73.070	59.834	25.954
Time for getCreds (ms), std deviation	289.842	189.559	157.875	71.319
Memory used, average (system units)	2621.77	1775.17	2511.16	1627.73
Memory used (system units), stdev	2.51	6.80	4.69	5.08
Ratio of System to User CPU time	0.025	0.014	0.020	0.013

Notes: system unit for measuring memory is a Page of memory. A memory page size was 4KB (which can vary on different systems).

Table 2. PERMIS Medium Policy Test

	L+C	L-C	C-L	-C-L
Initialisation time(seconds)	1.807	1.315	1.473	1.491
Time for getCreds, average (ms)	219.124	87.062	118.342	59.529
Time for getCreds (ms), std deviation	378.320	190.200	207.662	96.375
Memory used, average (system units)	2623.63	1777.82	2522.88	1649.47
Memory used (system units), stdev	2.70	5.53	1.82	5.47
Ratio of System to User CPU time	0.021	0.013	0.017	0.012

In the Medium Policy, the average number of certificates processed per request is 3.4 ACs (9 requests for Bob with 2 ACs, 21 requests for Jim with 4 ACs, 1 of which is redundant; 40 requests for Judy with 4 ACs, 48 requests for Sharon with 3 ACs, 2 requests for Sun Yatsen with no ACs and no LDAP operations).

The number of ACs has risen 2.1 times (110%). This correlates with the rise of time for tests with ACs stored in memory (59ms=100% for C-L, 33ms=126% for -C-L). The tests with remotely stored ACs shows a smaller increase (but less than 2 times) because the system is waiting for a reply from LDAP, which remains approximately the same in both the Simple Policy and Medium Policy tests. (77ms=54% for L+C, 14ms=19% for L-C; the latter has increased less than the former, because L+C makes extra requests for PKC retrieval)

The very large standard deviations in the times for getCreds is thought to be due to Java garbage collection, which runs automatically and periodically when the JVM determines that it is necessary to do so. For example, times for Simple Policy L+C ranged from 6ms to 1.9secs, and for -C-L ranged from 0.8ms to 663ms.

The memory consumption figures are approximately the same (compare them by column) for both the Simple and Medium Policy tests because they have been measured for the whole Java process, which has its own memory management routines. The standard deviation of memory usage, however, is quite different in almost all categories. The increase of memory usage deviation in the L+C tests means that in fact these operations tend to allocate and free big chunks of memory, and that in the Medium Policy test these chunks are slightly bigger. The decrease of stdev for the L-C tests means that a lot of the memory in the L+C tests is used by the cryptographic routines, so an increase in the number of ACs does not cause an increase in the number of cryptographic operations and lessens the memory reallocations. The deviations of memory usage in the tests for locally stored ACs shows a significant amount of memory management operations in the Simple Policy test (a lot of memory is allocated temporarily), and that in the Medium Policy test memory is allocated for longer periods of time (JVM reuses the same chunks).

14.8.2 Akenti Results

Table 3. Akenti Simple Policy Test, Caching off, Full URLs

	L+C	R+C	L-C	R-C	C-L	-C-L
Initialisation time(seconds)	0.012	0.013	0.012	0.012	0.073	0.012
Time for checkAccess, average (ms)	775.6	368.2	700.5	368.8	117.6	116.1
Time for checkAccess (ms), std deviation	1024.8	40.1	1407.5	47.5	23.9	11.1
Memory used, average (system units)	894.98	879.98	894.98	879.98	846.98	846.98
Memory used (system units), stdev	0.18	0.18	0.18	0.18	0.18	0.18
Ratio of System to User CPU time	0.046	0.047	0.044	0.046	0.024	0.024

Table 4. Akenti Simple Policy Test, Caching off, Chopped URLs

	L+C	R+C	L-C	R-C	C-L	-C-L
Initialisation time(seconds)	0.012	0.012	0.012	0.013	0.025	0.026
Time for checkAccess, average (ms)	795.6	441.7	768.2	527.2	113.5	116.6
Time for checkAccess (ms), std deviation	1040.0	58.1	677.8	922.5	11.0	13.1
Memory used, average (system units)	895.98	880.98	895.98	881.54	847.98	847.98
Memory used (system units), stdev	0.18	0.18	0.18	0.55	0.18	0.18
Ratio of System to User CPU time	0.055	0.067	0.055	0.066	0.022	0.023

Notes: there is an apparent rise in time for the tests with LDAP- and HTTP- based certificate stores, when URLs are specified in a truncated form (i.e. don't contain the trailing '/', as discussed above). There is also a noticeable increase of ratio of System to User CPU time, which is caused by extra network requests (the system time increased, but the user time remained virtually the same). The times for tests with locally stored certificates have not changed at all (as expected). Strangely, the times for tests with cryptography **off** have risen more than the times for tests with cryptography **on** (signing the capability certificates is a local procedure!): L+C has risen by 20ms (2%), R+C has risen by 80ms (21%); but L-C has risen by 70ms (10%), and R-C by 160ms (43%). We would have expected the times to be the opposite of this, because in both cases Akenti would perform the same number of network operations to create a capability certificate, and would perform less cryptographic operations in the case with cryptography off. Note also that R-C (crypto **off**) is almost the same as R+C (crypto **on**) in the first case, and even ***exceeds***(!) the latter time by 80ms (18%) in the second case. Such behaviour of the Akenti engine is very unusual.

Table 5. Akenti Simple Policy Test, Caching on, Full URLs

	L+C	R+C	L-C	R-C	C-L	-C-L
Initialisation time(seconds)	0.012	0.040	0.012	0.012	0.012	0.012
Time for checkAccess, average (ms)	23.6	17.2	11.4	12.1	11.3	11.9
Time for checkAccess (ms), std deviation	85.5	51.0	22.5	25.5	23.0	26.0
Memory used, average (system units)	900.98	885.98	831.25	831.25	831.25	831.25
Memory used (system units), stdev	0.27	0.18	101.72	101.72	101.72	101.72
Ratio of System to User CPU time	0.020	0.024	0.020	0.025	0.020	0.020

Table 6. Akenti Simple Policy Test, Caching on, Chopped URLs

	L+C	R+C	L-C	R-C	C-L	-C-L
Initialisation time(seconds)	0.012	0.012	0.012	0.012	0.028	0.024
Time for checkAccess, average (ms)	23.6	19.8	11.9	12.0	11.5	9.3
Time for checkAccess (ms), std deviation	85.5	69.9	25.4	26.0	23.4	13.8
Memory used, average (system units)	900.98	886.98	831.25	831.25	853.98	785.32
Memory used (system units), stdev	0.27	0.27	101.72	101.72	0.27	83.63
Ratio of System to User CPU time	0.020	0.034	0.014	0.016	0.015	0.018

There is almost no difference in times for tests with caching on. This is because the URLs are not being used.

Note that the tests show the same strange property of the code that R-C tests are slightly slower than expected. For example, they are slower than L-C tests, whilst R+C is significantly faster than L+C, which is the expected behaviour: in cases when certificates are stored on both LDAP and HTTP servers the code might need to issue extra requests to the LDAP server.

Memory usage figures remain almost the same for all tests, and show little deviation from the mean.

Table 7. Akenti Simple Policy Test, Caching off, Optimised

	C-L (table 4)	**C-L (optimised)**
Initialisation time(seconds)	0.025	0.080
Time for checkAccess, average (ms)	113.532	61.542
Time for checkAccess (ms), std deviation	11.010	29.252
Memory used, average (system units)	847.98	833.99
Memory used (system units), stdev	0.18	0.09
Ratio of System to User CPU time	0.022	0.039

At the beginning of May we shared our preliminary performance results with the Akenti developers. They were surprised with the figures and have revised their binary distribution accordingly. They have provided a version of the engine recompiled with better optimisation. This table is intended to show how much this optimisation affects the performance. We did not have enough time to perform all the other tests and only did tests for locally stored certificates (both PKCs and Attribute Certificates). The share of User time to overall time that it takes to make one *checkAccess* call has obviously decreased (signs of optimisation) as reflected by the Ratio of System to User CPU time. The overall time it takes to make one *checkAccess* call has decreased approximately by a factor of two. However, we do not expect that network-based times will improve quite as significantly, due to the possible network and server delays.

14.9. Comparison of the Performance Results

Now, to compare the PERMIS and Akenti performance, for single decisions we should choose the L+C test for PERMIS (see Table 1) and the L-C and R-C tests with caching off for Akenti (see Tables 3 and 4).

PERMIS time is: 142ms
Akenti L-C: 700ms or 768ms.
Akenti R-C: 368ms or 527ms.

PERMIS performed 2.6 times faster than the best Akenti time in comparable categories, which is a surprising result, considering that the PERMIS API runs on a Java Virtual

Machine that interprets the code, and the Akenti engine is written in C++ and therefore is compiled directly into CPU instructions.

However, PERMIS consumes 2.9 times more memory.

For multiple decision making we choose the average time for the PERMIS *decision* method compared to Akenti -C-L with caching on (minus the time for the calls to retrieve the credentials into the cache) (Tables 5 and 6).

PERMIS *decision*: 2ms
Note. This is the average time taken over all tests. The minimum time was <0.3ms and the maximum time was 168ms[8]
Akenti (cache only): 6.2ms (excluding the calls to refresh the cache)

PERMIS *decision* with 8 *getCreds*: 7.8ms
Akenti -C-L: 11.9ms or 9.3ms (including the time for the calls to retrieve the credentials into the cache)

Note that PERMIS supports multi-step decision making, whilst Akenti does not. With multi-step decision making, PERMIS assumes that the AEF will cache the results returned by *getCreds,* and that the AEF will return these to PERMIS for it to deliver several subsequent decisions. This behaviour is comparable to Akenti with caching on (but in PERMIS the AEF does the caching, whilst in Akenti, it does the caching).

In order to make the figures truly comparable it is necessary to either add the initial time for retrieving the credentials in PERMIS (call to *getCreds*) or to subtract the times of the calls in Akenti to fetch the credentials into the cache. During the 120 tests, Akenti appeared to make 8 network calls to refresh the cache, and when these are subtracted from Tables 5 and 6, the average time for Akenti cached decision making is 6.2ms. Alternatively, the average time of a call to *getCreds* in PERMIS (C-L) (Tables 1 and 2) is 60ms or 118ms. If 8 of these calls are added to 112 *decisions,* the PERMIS average becomes 7.8ms. Thus PERMIS still outperforms Akenti when multiple decisions have to be made.

14.10. Akenti Medium Policy Considerations

When we started implementing the Medium Policy (as described in section 14.6), we found that it was very difficult to build an appropriate Policy and UCCs to implement the desired behaviour. We found that in many cases it required a UCC to be built that did not contain any actions associated with it, but only a condition that must be satisfied, for example, a UCC issued by the director that says that if the country is not Iran or Iraq the user is acceptable (but is not actually granted any access). However, the Akenti policy language appears to be deficient in this capacity and does not provide such a capability. The Akenti developers suggested a work around in which the UCC allows a "default" action (that does not actually do anything), and is marked critical. This will then deny all

[8] The large standard deviation in the times is thought to be due to garbage collection by the Java VM.

access to any user from Iran or Iraq. Another UCC for Jones' group, with role of student and time between 8am and 8pm is allowed an action of "observe". Now someone who satisfies both UCCs will be allowed actions of “default” and “observe”. The application then has to be configured to ignore "default" actions and make use of the "observe" action. Needless to say, we did not think that this was an appropriate or intuitive solution.

The developers claimed in private correspondence that their command-line tools do support the creation of UCCs without actions, and that their policy engine supports this, but we did not have time to check this and have based our conclusions on the documentation and capabilities of the GUI tools that were provided. The developers have subsequently stated that the GUI will be updated to allow for the creation of UCCs without actions.

When we investigated alternative ways of specifying the required conditions, we found a major design deficiency that does not in fact allow distributed management of the resource as claimed in the Overview of Akenti [AkentiOverview]. The deficiency is based in how the UCCs are combined to derive a decision function, and in fact requires that the Stakeholders must communicate their needs to each other in order to create UCCs with joint conditions. We therefore lose the essence of Akenti's claimed distributed management of resources – there is no point in having many issuers of the conditions, if they have to co-ordinate their wishes with a central or superior Stakeholder in order to produce their UCCs, and cannot act independently. When we put this point to the Akenti developers they replied that they have a different interpretation of distributed management of resources. They actually mean co-operating stakeholders who can each issue UCCs, but the underlying assumption is that they must co-ordinate on who is going to set policy about which attributes and who is allowed veto power over what. If multiple stakeholders start issuing policy without considering what the other stakeholders are doing, they can easily deny access to everybody. Therefore the stakeholders have to have a minimum amount of collaboration. They are only independent in the sense that they can all create some aspects of policy.

A fuller analysis of the deficiency is given in Appendix C.

Another serious lack in the system right now is a function that will display all the existing policies and UCCs for a resource, so that a stakeholder can determine the totality of the authorisation policy for a resource. The Akenti developers recognise this deficiency and plan to implement this function as a matter of urgency.

The limitations of the Akenti policy language that do not allow a stakeholder to create a no-action UCC but only with a condition that must be satisfied, and the discussed deficiency of the Akenti concept did not let us build the correct environment for the Medium Policy. Instead we attempted to build a single Use Condition Certificate that encapsulated all the conditions, issued by the Director, but even though theoretically this should have worked, there was a software problem with the UCC generator and we failed to generate such a complex UCC. This bug has been reported to the Akenti developers. In conclusion, after more than one month of trying, we failed to run any Akenti tests with the medium policy.

15. Summary Table of Features

Feature	Akenti	PERMIS	Comment
Policy Features			
Policy location	Policy certs local, Use conditions distributed at URLs	Single in LDAP	
Policy control	Multiple Stakeholders	Single Target SOA	
Support for Hierarchical RBAC	No	Yes	
User Authentication scheme	User must be PKI enabled	User can use any authentication scheme	
Powerful condition expressions	Planned	Yes	Not yet implemented in Akenti
Policy inheritance	Yes. Root policy, and subordinate policies that inherit from root	No, has to be enforced externally (e.g. offline by management)	
Policy language	Akenti normalised format (ASCII)	XML	Akenti currently ignores the XML in its policy certs
Authorisation Tokens			
Format	Akenti normalised format (ASCII)	Standard X.509 Attribute Certificates	Note. Akenti formats have changed between releases making them incompatible
Identification of holder	By LDAP DN and DN of CA issuing their PKC	By globally unique LDAP DN.	
Distributed allocation of roles/attributes	Yes	Yes	
Allocated by	Multiple federated AAs	Multiple federated SOAs	
Separation of authentication and	Yes	Yes	

authorisation trusted roots			
Supports dynamic delegation of tokens	No	Designed with delegation in view. Not implemented in current version.	
Decision Engine Features			
Application can add its own condition evaluation	Yes	Yes	Akenti returns conditions to the application for evaluation, whilst PERMIS provides a plug in capability
Environmental variables can be used in decision making	Yes	Yes	Akenti returns these conditions to the application for it to evaluate, whilst PERMIS allows them to be passed via the API for PERMIS to evaluate
User's session with ACs can be prematurely terminated and ACs re-evaluated	Policy contains max time in seconds that an AC can be cached	Application can decide during call back at decision time (application needs to provide the Java object)	
Supports extensible conditions	?. Application has to do its own evaluations	Yes, new conditions can be added via plug-ins	
Decision making	Single step, with a capability certificate (signed or unsigned) returned by the Akenti server	Multi step with Grant/Deny answer returned by PERMIS	
Instantiation	Standalone program via TCP/IP (and optionally SSL) or via C API	Java API	Akenti provide an Apache Web server module as well
Written in	C++	Java	
Management Tools			
Authorisation token creation	Command line interface and GUI. Note. GUI can run with supporting	GUI tool, the Privilege Allocator. Note that "Construct" buttons	

	servers that know users' DNs and policy, making it easy to use.	are currently not linked to policy, so SOA must know users' DNs.	
Policy creation	Generic XML tools and GUI. Command line interface for signing.	Generic XML tools only. Use the Privilege Allocator for signing	Akenti tool can be used to create Use-Condition certificates as well
Installation instructions	Yes	Yes, PA Cookbook	
Performance			
API version, one decision, no caching	Between 368-768ms	144 ms	PII 500MHz, 256MB RAM running Linux

16. References

[Akenti] see http://www-itg.lbl.gov/security/Akenti/
[AkentiCerts] see http://www-itg.lbl.gov/security/Akenti/docs/AkentiCertificates1.1.html
[AkentiOverview] see http://www-itg.lbl.gov/Akenti/docs/overview.html
[AkentiPolicy] http://www-itg.lbl.gov/Akenti/docs/PolicyCert.html
[Blaze] Blaze, M., Feigenbaum, J., Ioannidis, J. "The KeyNote Trust-Management System Version 2", RFC 2704, Sep 1999
[Chadwick] D.W.Chadwick, A. Otenko. "The PERMIS X.509 Role Based Privilege Management Infrastructure", Proc 7th ACM Symposium On Access Control Models And Technologies (SACMAT 2002), Monterey, USA, June 2002. pp135-140.
[ISO] ITU-T Rec X.812 (1995) | ISO/IEC 10181-3:1996 "Security Frameworks for open systems: Access control framework
[IFIP] D.W.Chadwick, A. Otenko. "RBAC Policies in XML for X.509 Based Privilege Management" in Security in the Information Society: Visions and Perspectives: IFIP TC11 17th Int. Conf. On Information Security (SEC2002), May 7-9, 2002, Cairo, Egypt. Ed. by M. A. Ghonaimy, M. T. El-Hadidi, H.K.Aslan, Kluwer Academic Publishers, pp 39-53.
[Johnston] Johnston, W., Mudumbai, S., Thompson, M. "Authorization and Attribute Certificates for Widely Distributed Access Control," IEEE 7th Int Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE), Stanford, CA. June, 1998. Page(s): 340 -345 (see also http://www-itg.lbl.gov/security/Akenti/)
[Permis] see http://www.permis.org
[Raw] see http://sec.isi.salford.ac.uk/akenti-permis
[Thompson] Mary R. Thompson, S. Mudumbai, A. Essiari, W. Chin. "Authorization Policy in a PKI Environment", Proceedings of the First Annual PKI Workshop, Dartmouth College, April 2002, pages 137-149. see www.cs.dartmouth.edu/~pki02
[UCC] http://www-itg.lbl.gov/Akenti/docs/UseCondition.html
[X509] ISO/ITU-T Rec. X.509(2000) The Directory: Authentication Framework

17. Appendix A. The LDAP Schema

The LDAP schema is as follows, as defined as per OpenLDAP requirements:

```
attributetype (2.5.4.58 NAME 'attributeCertificateAttribute'
                        DESC 'A binary attribute certificate'
                        EQUALITY octetStringMatch
                        SYNTAX 1.3.6.1.4.1.1466.115.121.1.8 )

attributetype   (2.5.4.59 NAME 'attributeCertificateRevocationList'
                        DESC 'A binary attribute certificate revocation list'
                        EQUALITY octetStringMatch
                        SYNTAX 1.3.6.1.4.1.1466.115.121.1.8 )

attributetype   (2.5.4.61 NAME 'aACertificate'
                        DESC 'A binary attribute authority attribute certificate'
                        EQUALITY octetStringMatch
                        SYNTAX 1.3.6.1.4.1.1466.115.121.1.8 )

attributetype   (2.5.4.63 NAME 'attributeAuthorityRevocationList'
                        DESC 'A binary attribute certificate revocation list'
                        EQUALITY octetStringMatch
                        SYNTAX 1.3.6.1.4.1.1466.115.121.1.8 )

objectclass     (2.5.6.24 NAME 'pmiUser'
                        SUP top
                        DESC 'a pmi entity that can contain X509 ACs'
                        MAY (attributeCertificate) )

objectclass     (2.5.6.25 NAME 'pmiAA'
                        SUP top
                        DESC 'a pmi entity that can contain AA ACs'
                        MAY (attributeCertificateRevocationList $
                                aACertificate $
                                attributeAuthorityRevocationList ) )

objectClass     (2.5.6.21
                        NAME 'pkiUser'
                        SUP top
                        AUXILIARY
                        MAY userCertificate )
```

18. Appendix B. The Policies

18.1. PERMIS Simple Policy

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X.509_PMI_RBAC_Policy>
<X.509_PMI_RBAC_Policy
OID="1.2.826.0.1.3344810.6.0.0.0.0.1">

  <!-- only let the people of the department in -->
  <SubjectPolicy>
    <SubjectDomainSpec ID="department">
      <Include LDAPDN="ou=Venables, o=permis, c=gb"/>
    </SubjectDomainSpec>
  </SubjectPolicy>

  <!-- there is only one role: administrator; all others
have default access -->
  <RoleHierarchyPolicy>
    <RoleSpec OID="1.2.826.0.1.3344810.1.1.14"
Type="permisRole">
      <SupRole Value="administrator"/>
    </RoleSpec>
  </RoleHierarchyPolicy>

  <!-- there is only one SOA -->
  <SOAPolicy>
    <SOASpec ID="SOA" LDAPDN="cn=SOA, o=permis, c=gb"/>
  </SOAPolicy>

  <!-- let the SOA assign the administrator role to anyone
in the department -->
  <RoleAssignmentPolicy>
    <RoleAssignment>
      <SubjectDomain ID="department"/>
      <RoleList>
        <Role Type="permisRole" Value="administrator"/>
      </RoleList>
      <Delegate Depth="0"/>
      <SOA ID="SOA"/>
      <Validity/>
    </RoleAssignment>
  </RoleAssignmentPolicy>

  <!-- define the printer domain -->
  <TargetPolicy>
```

```
    <TargetDomainSpec ID="printer">
      <Include LDAPDN="cn=printer, ou=Venables, o=permis,
c=gb"/>
    </TargetDomainSpec>
  </TargetPolicy>

  <!-- define the actions -->
  <ActionPolicy>
    <Action Name="print"/>
    <Action Name="delete"/>
    <Action Name="pause"/>
    <Action Name="resume"/>
  </ActionPolicy>

  <!-- define the target access policy -->
  <TargetAccessPolicy>

    <!-- users can only print by default -->
    <TargetAccess>
      <RoleList/>
      <TargetList>
        <Target Actions="print">
          <TargetDomain ID="printer"/>
        </Target>
      </TargetList>
    </TargetAccess>

    <!-- administrator can do anything else -->
    <TargetAccess>
      <RoleList>
        <Role Type="permisRole" Value="administrator"/>
      </RoleList>
      <TargetList>
        <Target Actions="delete pause resume">
          <TargetDomain ID="printer"/>
        </Target>
      </TargetList>
    </TargetAccess>
  </TargetAccessPolicy>
</X.509_PMI_RBAC_Policy>
```

18.2. Akenti Simple Policy and Use Condition Certificates

(for LDAP stored PKCs and HTTP-based certificate store)

```
<?xml version="1.0" encoding="UTF-8"?>
<AkentiCertificate>
```

```
   <SignablePart>
      <Header CanonAlg="Ak1CanAlg" SignatureDigestAlg="RSA-
MD5" Type="Policy" Version="2">
         <UID>psyche#1693fe37#Wed Apr 30 16:27:18 BST
2003</UID>
         <Issuer>
            <UserDN>/C=GB/O=permis/CN=SOA</UserDN>
            <CADN>/C=GB/O=permis/CN=Root CA</CADN>
         </Issuer>
         <ValidityPeriod Begin="030430152654Z"
End="040429152654Z"/>
      </Header>
      <PolicyCert>
         <ResourceName>Printer</ResourceName>
         <CAInfo>
            <CADN>/C=GB/O=permis/CN=Root CA</CADN>

<X509Certificate>MIICYTCCAcqgAwIBAgIBADANBgkqhkiG9w0BAQQFAD
AwMQswCQYDVQQGEwJHQjEP
MA0GA1UEChMGcGVybWlzMRAwDgYDVQQDEwdSb290IENBMB4XDTAzMDQxNTE
zNTQ1
NFoXDTAzMDUxNTEzNTQ1NFowMDELMAkGA1UEBhMCR0IxDzANBgNVBAoTBnB
lcm1p
czEQMA4GA1UEAxMHUm9vdCBDQTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYk
CgYEA
7rDkXA8RaMZDL9Xh3NntZmL3OjZqbJIFFaBInA08nEuoMd0DhppuA33QZwI
ts+ru
Pl3Jj8d5muB3AElA5yIsEiKR7WiZIoBkHgGa/NmNathB5ZbJAfxh00/k5LK
ig1Cu
b89I7cEUjr0E+CX4vx0FbL3GgfSqAQeeuAZfZb0W+xcCAwEAAaOBijCBhzA
dBgNV
HQ4EFgQUV7s2xpIK0lDhIk0nTuQW7eysyl4wWAYDVR0jBFEwT4AUV7s2xpI
K0lDh
Ik0nTuQW7eysyl6hNKQyMDAxCzAJBgNVBAYTAkdCMQ8wDQYDVQQKEwZwZXJ
taXMx
EDAOBgNVBAMTB1Jvb3QgQ0GCAQAwDAYDVR0TBAUwAwEB/zANBgkqhkiG9w0
BAQQF
AAOBgQAOub7I0BzE5723b2CERi6Qs4mk2w3F984Ff9OnhlclyUhkdtKZzvO
zY4cY
3X/zImShtQ9rnrHJtclQTTD5xXqbyIvTmvweOB/30bzwQD6UxXWDu0C5l/t
e+DO5
H8n97wsQz7bZMlrFik6xOHVH64C6CYU8LayJYtkyoc5tRhtiig==</X509C
ertificate>
            <IdDirs>
               <URL>ldap://sec.isi.salford.ac.uk/</URL>
            </IdDirs>
         </CAInfo>
         <UseCondIssuerGroup>
            <Principal>
               <UserDN>/C=GB/O=permis/CN=SOA</UserDN>
               <CADN>/C=GB/O=permis/CN=Root CA</CADN>
            </Principal>

<URL>http://sec.isi.salford.ac.uk/download/akenticerts/</UR
L>
         </UseCondIssuerGroup>
         <AttrDirs>

<URL>http://sec.isi.salford.ac.uk/download/akenticerts/</UR
L>
         </AttrDirs>
         <CacheTime>60</CacheTime>
      </PolicyCert>
   </SignablePart>

<Signature>XN2lrClTy5LM+ZE6QbYB3ARFLOhRacagVKeAmSyyNlWh1nXW
CjDQ9gPmBVNRgZfw
BxLoK2MJDAbD1OKSXpJxvcQgVffeBqgA7ZwCnfu4/ZiynN/kniJxs7Qmcht
CzwGa
JrTge1Ju/ZwB7DgVloOHiLfejoJgyZEwnbi6vMNS2rc=</Signature>
</AkentiCertificate>
-----BEGIN AKENTI POLICY CERTIFICATE-----
UG9saWN5IFYyIHBzeWNoZSMxNjkzZmUzNyNXZWRcIEFwclwgMzBcIDE2OjI
3OjE4
XCBCU1RcIDIwMDMgL0M9R0IvTz1wZXJtaXMvQ049U09BIC9DPUdCL089cGV
ybWlz
L0NOPVJvb3RcIENBIDAgMDMwNDMwMTUyNjU0WiAwNDA0MjkxNTI2NTRaIEF
rMUNh
bkFsZyBSU0EtTUQ1IFByaW50ZXIgMSAxIDYxMyAwggJhMIIByqADAgECAgE
AMA0G
CSqGSIb3DQEBBAUAMDAxCzAJBgNVBAYTAkdCMQ8wDQYDVQQKEwZwZXJtaXM
xEDAO
BgNVBAMTB1Jvb3QgQ0EwHhcNMDMwNDE1MTM1NDU0WhcNMDMwNTE1MTM1NDU
0WjAw
MQswCQYDVQQGEwJHQjEPMA0GA1UEChMGcGVybWlzMRAwDgYDVQQDEwdSb29
0IENB
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDusORcDxFoxkMv1eHc2e1
mYvc6
NmpskgUVoEicDTycS6gx3QOGmm4DfdBnAi2z6u4+XcmPx3ma4HcASUDnIiw
SIpHt
aJkigGQeAZr82Y1q2EHllskB/GHTT+TksqKDUK5vz0jtwRSOvQT4Jfi/HQV
svcaB
9KoBB564Bl9lvRb7FwIDAQABo4GKMIGHMB0GA1UdDgQWBBRXuzbGkgrSUOE
iTSdO
```

```
5Bbt7KzKXjBYBgNVHSMEUTBPgBRXuzbGkgrSUOEiTSdO5Bbt7KzKXqE0pDI
wMDEL
MAkGA1UEBhMCR0IxDzANBgNVBAoTBnBlcm1pczEQMA4GA1UEAxMHUm9vdCB
DQYIB
ADAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBBAUAA4GBAA65vsjQHMTnvbd
vYIRG
LpCziaTbDcX3zgV/06eGVyXJSGR20pnO87Njhxjdf/MiZKG1D2uescm1yVB
NMPnF
epvIi9Oa/B44H/fRvPBAPpTFdYO7QLmX+174M7kfyf3vCxDPttkyWsWKTrE
4dUfr
gLoJhTwtrIli2TKhzm1GG2KKIDEgbGRhcDovL3NlYy5pc2kuc2FsZm9yZC5
hYy51
ay8gMCAxIDEgL0M9R0IvTz1wZXJtaXMvQ049U09BIC9DPUdCL089cGVybWl
zL0NO
PVJvb3RcIENBIDEgaHR0cDovL3NlYy5pc2kuc2FsZm9yZC5hYy51ay9kb3d
ubG9h
ZC9ha2VudGljZXJ0cy8gMSBodHRwOi8vc2VjLmlzaS5zYWxmb3JkLmFjLnV
rL2Rv
d25sb2FkL2FrZW50aWNlcnRzLyA2MCAxMjggXN2lrClTy5LM+ZE6QbYB3AR
FLOhR
acagVKeAmSyyNlWh1nXWCjDQ9gPmBVNRgZfwBxLoK2MJDAbD1OKSXpJxvcQ
gVffe
BqgA7ZwCnfu4/ZiynN/kniJxs7QmchtCzwGaJrTge1Ju/ZwB7DgVloOHiLf
ejoJg
yZEwnbi6vMNS2rc=
-----END AKENTI POLICY CERTIFICATE-----
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<AkentiCertificate>
   <SignablePart>
      <Header CanonAlg="Ak1CanAlg" SignatureDigestAlg="RSA-
MD5" Type="UseCondition" Version="2">
         <UID>psyche#1693fe37#Fri May 02 11:29:34 BST
2003</UID>
         <Issuer>
            <UserDN>/C=GB/O=permis/CN=SOA</UserDN>
            <CADN>/C=GB/O=permis/CN=Root CA</CADN>
         </Issuer>
         <ValidityPeriod Begin="030502102901Z"
End="040501102901Z"/>
      </Header>
      <UseConditionCert critical="false" scope="local">
         <ResourceName>Printer</ResourceName>
         <Condition>
            <Constraint>role = administrator</Constraint>
            <AttributeInfo type="AKENTI">
               <AttrName>role</AttrName>
               <AttrValue>administrator</AttrValue>
               <Principal>
                  <UserDN>/C=GB/O=permis/CN=SOA</UserDN>
                  <CADN>/C=GB/O=permis/CN=Root CA</CADN>
               </Principal>
               <AttrDirs>

<URL>http://sec.isi.salford.ac.uk/download/akenticerts</URL
>
               </AttrDirs>
            </AttributeInfo>
         </Condition>
         <Rights>pause,resume,delete</Rights>
      </UseConditionCert>
   </SignablePart>

<Signature>XigzmtKHrh2Z2IZQNm/GVqywPYBQ9L76CUK/d1tILGULirSn
NJMy9bNe3Kf3MSIV
Hpr+9s7UeJaHhOniTrJDGolXmsEJ8pwwf4xpsVtmK5lJJAiFnX7VVrGov4T
VOnPZ
rxQVRehj3BEjzgabrvZNmPmdWWZMmc/Cb5KNRogSDGI=</Signature>
</AkentiCertificate>
-----BEGIN AKENTI USECONDITION CERTIFICATE-----
VXNlQ29uZGl0aW9uIFYyIHBzeWNoZSMxNjkzZmUzNyNGcmlcIE1heVwgMDJ
cIDEx
OjI5OjM0XCBCU1RcIDIwMDMgL0M9R0IvTz1wZXJtaXMvQ049U09BIC9DPUd
CL089
cGVybWlzL0NOPVJvb3RcIENBIDAgMDMwNTAyMTAyOTAxWiAwNDA1MDExMDI
5MDFa
IEFrMUNhbkFsZyBSU0EtTUQ1IFByaW50ZXIgMSAwIHJvbGVcID1cIGFkbWl
uaXN0
cmF0b3IgMSAyIHJvbGUgYWRtaW5pc3RyYXRvciAxIC9DPUdCL089cGVybWl
zL0NO
PVNPQSAvQz1HQi9PPXBlcm1pcy9DTj1Sb290XCBDQSAxIGh0dHA6Ly9zZWM
uaXNp
LnNhbGZvcmQuYWMudWsvZG93bmxvYWQvYWtlbnRpY2VydHMgMyBwYXVzZSB
yZXN1
bWUgZGVsZXRlIDEyOCBeKDOa0oeuHZnYhlA2b8ZWrLA9gFD0vvoJQr93W0g
sZQuK
tKc0kzL1s17cp/cxIhUemv72ztR4loeE6eJOskMaiVeawQnynDB/jGmxW2Y
rmUkk
CIWdftVWsai/hNU6c9mvFBVF6GPcESPOBpuu9k2Y+Z1ZZkyZz8Jvko1GiBI
MYg==

-----END AKENTI USECONDITION CERTIFICATE-----
```

18.3. PERMIS Medium Policy

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X.509_PMI_RBAC_Policy SYSTEM
"file://localhost/C:/research/projects/permis/policy10.dtd"
>
<X.509_PMI_RBAC_Policy
OID="1.2.826.0.1.3344810.6.0.0.0.1.1">
  <SubjectPolicy>
    <SubjectDomainSpec ID="dns">
      <Include LDAPDN=""/>
      <Exclude LDAPDN="c=iq"/>
      <Exclude LDAPDN="c=kp"/>
      <Exclude LDAPDN="c=ir"/>
    </SubjectDomainSpec>
  </SubjectPolicy>
  <RoleHierarchyPolicy>
    <RoleSpec OID="1.2.826.0.1.3344810.1.1.14"
Type="permisRole">
      <SupRole Value="leader">
        <SubRole Value="experimenter"/>
      </SupRole>
      <SupRole Value="experimenter">
        <SubRole Value="student"/>
      </SupRole>
      <SupRole Value="student"/>
    </RoleSpec>
    <RoleSpec OID="1.2.826.0.1.3344810.1.1.101" Type="tr-
Course">
      <SupRole Value="lbnl-xray-101"/>
    </RoleSpec>
    <RoleSpec OID="1.2.826.0.1.3344810.1.1.100"
Type="group">
      <SupRole Value="Jones"/>
      <SupRole Value="Doe"/>
    </RoleSpec>
  </RoleHierarchyPolicy>
  <SOAPolicy>
    <SOASpec ID="X" LDAPDN="cn=smith,o=lbl,c=us"/>
    <SOASpec ID="PI" LDAPDN="cn=Dr Joe Jones,o=lbl,c=us"/>
    <SOASpec ID="Colleague" LDAPDN="cn=Dr Jane
Doe,o=lbl,c=us"/>
    <SOASpec ID="Director" LDAPDN="cn=Boss,o=lbl,c=us"/>
  </SOAPolicy>
  <RoleAssignmentPolicy>
    <RoleAssignment>
      <SubjectDomain ID="dns"/>
      <RoleList>
        <Role Type="tr-Course" Value="lbnl-xray-101"/>
      </RoleList>
      <Delegate Depth="0"/>
      <SOA ID="X"/>
      <Validity/>
    </RoleAssignment>
    <RoleAssignment>
      <SubjectDomain ID="dns"/>
      <RoleList>
        <Role Type="permisRole"/>
      </RoleList>
      <Delegate Depth="0"/>
      <SOA ID="Director"/>
      <Validity/>
    </RoleAssignment>
    <RoleAssignment>
      <SubjectDomain ID="dns"/>
      <RoleList>
        <Role Type="group" Value="Jones"/>
      </RoleList>
      <Delegate Depth="0"/>
      <SOA ID="PI"/>
      <Validity/>
    </RoleAssignment>
    <RoleAssignment>
      <SubjectDomain ID="dns"/>
      <RoleList>
        <Role Type="group" Value="Doe"/>
      </RoleList>
      <Delegate Depth="0"/>
      <SOA ID="Colleague"/>
      <Validity/>
    </RoleAssignment>
  </RoleAssignmentPolicy>
  <TargetPolicy>
    <TargetDomainSpec ID="lightSource">
      <Include LDAPDN="cn=light source,o=lbl,c=us"/>
    </TargetDomainSpec>
  </TargetPolicy>
  <ActionPolicy>
    <Action Name="control"/>
    <Action Name="operate"/>
    <Action Name="observe"/>
  </ActionPolicy>
  <TargetAccessPolicy>
    <TargetAccess>
    <RoleList>
      <Role Type="permisRole" Value="leader"/>
      <Role Type="group" Value="Jones"/>
    </RoleList>
    <TargetList>
      <Target Actions="control">
        <TargetDomain ID="lightSource"/>
      </Target>
    </TargetList>
    <IF>
      <AND>
        <GE>
          <Environment Parameter="time" Type="Time"/>
          <Constant Type="Time" Value="8am"/>
        </GE>
        <LE>
          <Environment Parameter="time" Type="Time"/>
          <Constant Type="Time" Value="8pm"/>
        </LE>
      </AND>
    </IF>
  </TargetAccess>
  <TargetAccess>
    <RoleList>
      <Role Type="permisRole" Value="experimenter"/>
      <Role Type="group" Value="Jones"/>
    </RoleList>
    <TargetList>
      <Target Actions="operate">
        <TargetDomain ID="lightSource"/>
      </Target>
    </TargetList>
    <IF>
      <AND>
        <GE>
          <Environment Parameter="time" Type="Time"/>
          <Constant Type="Time" Value="8am"/>
        </GE>
        <LE>
          <Environment Parameter="time" Type="Time"/>
          <Constant Type="Time" Value="8pm"/>
        </LE>
      </AND>
    </IF>
  </TargetAccess>
  <TargetAccess>
    <RoleList>
```

```
      <Role Type="permisRole" Value="student"/>
      <Role Type="group" Value="Jones"/>
    </RoleList>
    <TargetList>
      <Target Actions="observe">
        <TargetDomain ID="lightSource"/>
      </Target>
    </TargetList>
    <IF>
      <AND>
        <GE>
          <Environment Parameter="time" Type="Time"/>
          <Constant Type="Time" Value="8am"/>
        </GE>
        <LE>
          <Environment Parameter="time" Type="Time"/>
          <Constant Type="Time" Value="8pm"/>
        </LE>
      </AND>
    </IF>
  </TargetAccess>
  <TargetAccess>
    <RoleList>
      <Role Type="group" Value="Doe"/>
      <Role Type="permisRole" Value="leader"/>
    </RoleList>
    <TargetList>
      <Target Actions="control">
        <TargetDomain ID="lightSource"/>
      </Target>
    </TargetList>
    <IF>
      <AND>
        <GE>
          <Environment Parameter="time" Type="Time"/>
          <Constant Type="Time" Value="8pm"/>
        </GE>
        <LE>
          <Environment Parameter="time" Type="Time"/>
          <Constant Type="Time" Value="8am"/>
        </LE>
      </AND>
    </IF>
  </TargetAccess>
  <TargetAccess>
    <RoleList>
      <Role Type="group" Value="Doe"/>
```

```
        <Role Type="permisRole" Value="experimenter"/>
      </RoleList>
      <TargetList>
        <Target Actions="operate">
          <TargetDomain ID="lightSource"/>
        </Target>
      </TargetList>
      <IF>
        <AND>
          <GE>
            <Environment Parameter="time" Type="Time"/>
            <Constant Type="Time" Value="8pm"/>
          </GE>
          <LE>
            <Environment Parameter="time" Type="Time"/>
            <Constant Type="Time" Value="8am"/>
          </LE>
        </AND>
      </IF>
    </TargetAccess>
    <TargetAccess>
      <RoleList>
        <Role Type="group" Value="Doe"/>
        <Role Type="permisRole" Value="student"/>
      </RoleList>
      <TargetList>
        <Target Actions="observe">
          <TargetDomain ID="lightSource"/>
        </Target>
      </TargetList>
      <IF>
        <AND>
          <GE>
            <Environment Parameter="time" Type="Time"/>
            <Constant Type="Time" Value="8pm"/>
          </GE>
          <LE>
            <Environment Parameter="time" Type="Time"/>
            <Constant Type="Time" Value="8am"/>
          </LE>
        </AND>
      </IF>
    </TargetAccess>
  </TargetAccessPolicy>
</X.509_PMI_RBAC_Policy>
```

19. Appendix C. Analysis of Akenti deficiency in Distributed Management of Resources

If we write critical UCCs (those with the Critical flag set) as C_i and Non-critical UCCs (those with Critical flag reset) as N_i (where the index represents the UCC issued by the i-th Stakeholder), then the whole set of UCCs defining the policy can be represented as follows:

C_1 & C_2 & C_3 & … & $(1 + N_1 + N_2 + …)$ (1)

(& stands for boolean AND operation, + stands for boolean OR operation)

This boolean expression is neither a Disjunctive, nor Conjunctive Normal Form. Therefore it cannot cater for all boolean functions that can be built using Akenti UCCs.

Even though the set of boolean functions that can be expressed using Akenti UCCs is not empty, we will show that it is not possible to build such a function for the Medium Policy (if each of the UCCs has to be issued by a different Stakeholder). The Policy can be expressed as

C & X & R & (G_0 & T_0 + G_1 & T_1) (2)

where C is a country restriction, X is the X-ray training course restriction, R is the role, G_i are groups and T_i are allowed times of access for the corresponding groups. Note that we assume that G_i and T_i restrictions can be created by the same Stakeholder, but they should be different Stakeholders for these two groups (Dr Joe Jones and Dr Jane Doe).

This already is not possible to represent as Akenti UCCs issued by different Stakeholders (since it does not match the general expression 1). The condition on groups and their time of access should therefore be incorporated in one UCC. Since all of the Stakeholders are equal, there is no definite decision about who must issue such Use Conditions (incorporating both G_i and their T_i), and under circumstances of ad hoc changes to the policy conditions (which is promoted by Akenti), this may not be possible. Note also that if we assume that the Stakeholders are able to communicate their needs to each other to create UCCs with joint conditions, then we lose the essence of Akenti with its distributed management of resources – there is no point in having many issuers of the conditions, since they have to co-ordinate their wishes with a central or superior Stakeholder.

Note also that R is a hierarchy of roles, which can be expressed in Akenti UCCs as $R_0+R_1+R_2$, each of these UCCs corresponding to the definitions of the access rights. R_0 would then contain “student, experimenter or leader is allowed to observe”, R_1 would contain “experimenter or leader is allowed to operate”, R_2 would contain “leader is allowed to control”. Since the R_i have an implication relation ($R_1=R_0?$ R_1, $R_2=R_1?$ R_2, meaning that R_1 cannot be true unless R_0 is and similarly for R_2 and R_1), R can be rewritten as follows: R_0 & $(1 + R_1 + R_2)$. Now Medium Policy can be transformed into the following:

$$C \,\&\, X \,\&\, (G_0 \,\&\, T_0 + G_1 \,\&\, T_1) \,\&\, R_0 \,\&\, (1 + R_1 + R_2) \qquad (3)$$

Therefore, this policy can be represented using Akenti UCCs, but if and only if the condition on groups and their times is issued as one Use Condition Certificate. It will not be possible to have C, X and this combined group-time condition as separate UCCs, since UCCs should always stipulate a list of allowed Actions (and these conditions do not define any allowed actions). Therefore all of the conditions combined with AND must be incorporated in one UCC. The definition of R_i must be done by the same Stakeholder (semantically they describe the same hierarchy). Thus there will be only one Stakeholder that defines all of the conditions.

Akenti Stakeholders have the absolute right to introduce a new law extending access (allowing more) and have the absolute right to veto *everything* (i.e. reduce access granted by *all* Stakeholders), which is the only way to *confine* access (allowing less) granted by another Stakeholder.

Since not every case can be implemented in Akenti, the engine should be redesigned or the cases for which Akenti does not cater for distributed resource management must be defined.