An American National Standard

Standard Guide for Privilege Management Infrastructure[1]

This standard is issued under the fixed designation E2595; the number immediately following the designation indicates the year of original adoption or, in the case of revision, the year of last revision. A number in parentheses indicates the year of last reapproval. A superscript epsilon (ε) indicates an editorial change since the last revision or reapproval.

INTRODUCTION

This guide arises from the ongoing development and implementation of privilege management infrastructures (PMIs) within the healthcare environment. The healthcare environment supported by this guide is enterprise-wide and extends beyond traditional borders to include external providers, suppliers, and other healthcare partners. This guide supports privilege management within distributed computing as well as service-oriented architecture environments. This guide supports a distributed security environment in which security is also a distributed service.

The healthcare sector is continually improving the delivery of care by leveraging technical advances in computer-based applications. Health professionals are increasingly accessing multiple applications to schedule, diagnose, and administer patient care. These disparate applications are typically connected to a common network infrastructure that typically supports patient, business, and nonbusiness services, communications, and protocols. Because increased access is made possible through a common network infrastructure, secure access to these distributed, and often loosely coupled applications, is even more important than when these applications were accessed as stand-alone devices.

Secure access to legacy computer-based healthcare applications typically involves authentication of the user to the application using single-factor identification, such as a password, or multifactor identification, such as a password combined with a token or biometric devices. After authentication, the application determines the authority that user may have to use aspects of the application. Determining the level of authority a user has is typically done, if at all, by each application. The application may restrict operations (such as read, write, modify, or delete) to an application-specific group or role affiliation. Authenticated users are frequently associated with groups or roles using a local database or flat file under the control of an application administrator.

The use of a local mechanism for authorization creates a patchwork of approaches difficult to administer centrally across the breadth of a healthcare enterprise. That is, the software logic determining authorization is distinctive to each application. In some cases, applications can be adapted to use a network database that contains a trusted source of name-value pairs. This information allows applications to determine the user's group or role affiliation. This approach permits centralized control over a shared user base. However, the resulting granularity of control over user authorization is coarse and shall be interpreted by each application specialist. Granularity of user authority can only be improved by increasing the number of application-specific groups or roles in the shared database. Storing information specific to each application causes exponential growth of roles per user and results in provisioning difficulties. The better solution is to associate industry standard permissions to users. Each application can examine the permissions listed for a user and determine their level of authorization regardless of their group affiliation within the healthcare organization.

The resulting system is a PMI. By the nature of the problem, the privileges shall be defined in an industry standard way. This guide will discuss various aspects of identifying a PMI standard to vendors providing healthcare applications to the contemporary healthcare enterprise.

1. Scope

1.1 This guide defines interoperable mechanisms to manage privileges in a distributed environment. This guide is oriented towards support of a distributed or service-oriented architecture (SOA) in which security services are themselves distributed and applications are consumers of distributed services.

1.2 This guide incorporates privilege management mechanisms alluded to in a number of existing standards (for

Copyright © ASTM International, 100 Barr Harbor Drive, PO Box C700, West Conshohocken, PA 19428-2959. United States

example, Guide E1986 and Specification E2084). The privilege mechanisms in this guide support policy-based access control (including role-, entity-, and contextual-based access control) including the application of policy constraints, patient-requested restrictions, and delegation. Finally, this guide supports hierarchical, enterprise-wide privilege management.

1.3 The mechanisms defined in this guide may be used to support a privilege management infrastructure (PMI) using existing public key infrastructure (PKI) technology.

1.4 This guide does not specifically support mechanisms based on secret-key cryptography. Mechanisms involving privilege credentials are specified in ISO 9594-8:2000 (attribute certificates) and Organization for the Advancement of Structured Information Standards (OASIS) Security Assertion Markup Language (SAML) (attribute assertions); however, this guide does not mandate or assume the use of such standards.

1.5 Many current systems require only local privilege management functionality (on a single computer system). Such systems frequently use proprietary mechanisms. This guide does not address this type of functionality; rather, it addresses an environment in which privileges and capabilities (authorizations) shall be managed between computer systems across the enterprise and with business partners.

1.6 *This standard does not purport to address all of the safety concerns, if any, associated with its use. It is the responsibility of the user of this standard to establish appropriate safety and health practices and determine the applicability of regulatory limitations prior to use.*

2. Referenced Documents

2.1 *ASTM Standards:*[2]

E1762 Guide for Electronic Authentication of Health Care Information

E1985 Guide for User Authentication and Authorization

E1986 Guide for Information Access Privileges to Health Information

E2084 Specification for Authentication of Healthcare Information Using Digital Signatures (Withdrawn 2009)[3]

E2212 Practice for Healthcare Certificate Policy

2.2 *ANSI Standards:*[4]

X9.45 Enhanced Management Controls Using Digital Signatures and Attribute Certificates

INCITS 359 Role-Based Access Control

2.3 *HL7 Standard:*[5]

Health Level 7 Context Management "CCOW" (Clinical Context Object Workgroup) Standard, Version 1.5

2.4 *IETF Standards:*[6]

RFC 3198 Terminology for Policy-Based Management

RFC 3280 Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

RFC 3881 Security Audit and Access Accountability Message XML Data Definitions for Healthcare Applications

2.5 *ISO Standards:*[7]

ISO 9594-8 The Directory: Public-Key and Attribute Certificate Frameworks; also available as ITU-T X.509: 2000

ISO 10181-3-00 Security Frameworks for Open Systems: Access Control Framework; also available as ITU-T X.812: 1995

ISO/TS 21298 Functional and Structure Roles

ISO/TS 22600-2:2006 Health Informatics—Privilege Management and Access Control—Part 2: Formal Models

2.6 *OASIS Standards:*[8]

Security Assertion Markup Language (SAML) v2.0

SAML 2.0 Profile of XACML

Security Provisioning Markup Language (SPML) v2.0, (OASIS)

Web Services Business Process Execution Language (WS-BPEL v2)

WS-Trust (WS-Trust 1.3)

eXtensible Access Control Markup Language (XACML) v2.0

XACML Profile for Role Based Access Control (RBAC): Committee Draft 01

XACML Profile for Web Services (WS-XACML)

2.7 *NIST Standards:*

NIST Special Publication 800-33 Underlying Technical Models for Information Technology Security, (Stoneburner), December 2001

NIST Special Publication 800-95 (Draft) Guide to Secure Web Services, (Singhal, et al), September 2006

NIST Special Publication 800-100 Information Security Handbook: A Guide for Managers, (Bowen, et al), October 2006

FIPS PUB 66 Standard Industrial Classification (SIC) Codes[9]

3. Terminology

3.1 *Definitions:*

3.1.1 *access control decision function (ADF), n*—specialized function that makes access control decisions by applying access control policy rules to a requested action; see *policy decision point.*

[1] This guide is under the jurisdiction of ASTM Committee E31 on Healthcare Informatics and is the direct responsibility of Subcommittee E31.25 on Healthcare Data Management, Security, Confidentiality, and Privacy.

Current edition approved March 1, 2013. Published March 2013. Originally approved in 2007. Last previous edition approved in 2007 as E2595–07. DOI: 10.1520/E2595-07R13.

[2] For referenced ASTM standards, visit the ASTM website, www.astm.org, or contact ASTM Customer Service at service@astm.org. For *Annual Book of ASTM Standards* volume information, refer to the standard's Document Summary page on the ASTM website.

[3] The last approved version of this historical standard is referenced on www.astm.org.

[4] Available from American National Standards Institute (ANSI), 25 W. 43rd St., 4th Floor, New York, NY 10036, http://www.ansi.org.

[5] Available from Health Level Seven, Inc., 3300 Washtenaw Ave., Suite 227, Ann Arbor, MI 48104.

[6] Available from Internet Engineering Task Force, www.ietf.org/rfc.html.

[7] Available from International Organization for Standardization (ISO), 1 rue de Varembé, Case postale 56, CH-1211, Geneva 20, Switzerland, http://www.iso.ch.

[8] Available from www.oasis-open.org/specs/index.php.

[9] Withdrawn Feb. 8, 2005.

3.1.2 *access control enforcement function (AEF), n*—specialized function that is part of the access path between a requestor and a protected resource that enforces the decisions made by the ADF; see *policy enforcement point*.

3.1.3 *access control information (ACI), n*—any information used for access control purposes, including contextual information.

3.1.4 *attribute certificate (AC), n*—data structure that includes some attribute values and identification information about the owner of the attribute certificate, all digitally signed by an attribute authority (this includes certificates that an authority issues to itself) and this authority's signature serves as the guarantee of the binding between the attributes and their owner.

3.1.4.1 *Discussion*—Types: role specification and role assignment described in Ref (**1**).[10]

3.1.5 *attribute authority (AA), n*—authority, trusted by the verifier to delegate privilege, that issues attribute certificates.

3.1.6 *attribute authority revocation list (AARL), n*—revocation list containing attribute certificates issued to attribute authorities that are no longer considered valid by the certificate issuer.

3.1.7 *attribute certificate revocation list (ACRL), n*—revocation list containing attribute certificates issued to claimants that are no longer considered valid by the certificate issuer.

3.1.8 *authority, n*—entity responsible for the issuance of certificates.

3.1.8.1 *Discussion*—Two types are defined in this guide: certificate authority that issues public-key certificates and attribute authority that issues attribute certificates.

3.1.9 *authorization, n*—granting of rights that includes the granting of access based on access rights.

3.1.10 *authorization credential, n*—signed assertion of a user's permission attributes.

3.1.11 *authority revocation list (ARL), n*—revocation list containing public-key certificates issued to authorities that are no longer considered valid by the certificate issuer.

3.1.12 *authority certificate, n*—certificate issued to an authority (for example, either to a certification authority or to an attribute authority).

3.1.13 *business partner agreement, n*—document used to demarcate the legal, ethical, and practical responsibilities between subscribers to a privilege management infrastructure (PMI) and between cooperating PMI implementations.

3.1.14 *certificate revocation list (CRL), n*—signed list indicating a set of certificates that are no longer considered valid by the certificate issuer.

3.1.14.1 *Discussion*—CRLs may be used to identify revoked public-key certificates or attribute certificates and may represent revocation of certificates issued to authorities or to users. The term CRL is also commonly used as a generic term applying to all the different types of revocation lists, including CRLs, ARLs, ACRLs, and so forth.

3.1.15 *certificate validation, n*—process of ensuring that a certificate is valid, including possibly the construction and processing of a certification path, and ensuring that all certificates in that path have not expired or been revoked.

3.1.16 *claimant, n*—entity requesting that a sensitive service be performed or provided by a verifier based on the claimant's privileges as identified in its proffered attribute assertion, attribute certificate, or subject directory attributes extension of their public-key certificate.

3.1.17 *credential, n*—information describing the security attributes (identity or privileges or both) of a user or other principal.

3.1.17.1 *Discussion*—Credentials are claimed through authentication or delegation and used by access control.

3.1.18 *delegation, n*—conveyance of privilege from one entity that holds such privilege to another entity.

3.1.19 *delegation path, n*—ordered sequence of credentials that can be processed to verify the authenticity of a claimant's privilege.

3.1.20 *domain, n*—set of objects that a subject is allowed to access.

3.1.21 *environmental variables, n*—those aspects of policy required for an authorization decision that are not contained within structural structures but are available through some local means to a verifier (for example, time of day or current account balance).

3.1.22 *functional role, n*—job function within the context of an organization whose permissions are defined by operations on tasks, scenarios, aggregations, or data objects.

3.1.22.1 *Discussion*—Functional roles provide detailed permissions defining what a user can do within the context of an application. Examples include permissions to create an order, permission to sign a check, permission to read a database row, and so forth. A functional role applies to a workflow's individual process tasks.

3.1.23 *interoperable role, n*—as defined by HL7, a job function within the context of two or more organizations representing the lowest common level of interoperable permissions defined by a standardized vocabulary.

3.1.24 *owner, n*—entity to whom some privilege has been delegated either directly from the source of authority or indirectly through another attribute authority.

3.1.24.1 *Discussion*—An owner asserts its claim to that privilege by presenting authoritative credentials to a verifier and acting as a claimant for its privilege.

3.1.25 *permission, n*—approval to perform an operation on one or more protected objects.

3.1.26 *permission attributes, n*—operations and objects that define a permission.

3.1.27 *policy, n*—a set of rules, and an identifier for the rule-combining algorithm and (optionally) a set of obligations (OASIS XACML).

[10] The boldface numbers in parentheses refer to the list of references at the end of this standard.

3.1.28 *policy decision point (PDP), n*—system entity that evaluates applicable policy and renders an authorization decision.

3.1.28.1 *Discussion*—This term is defined in a joint effort by the IETF Policy Framework Working Group and the Distributed Management Task Force (DMTF)/Common Information Model (CIM) in RFC 3198. This term corresponds to access decision function (ADF) in ISO 10181-3-00.

3.1.29 *policy enforcement point (PEP), n*—system entity that performs access control by making decision requests and enforcing authorization decisions.

3.1.29.1 *Discussion*—This term is defined in a joint effort by the IETF Policy Framework Working Group and the Distributed Management Task Force (DMTF)/Common Information Model (CIM) in RFC 3198. This term corresponds to access enforcement function (AEF) in ISO 10181-3-00.

3.1.30 *privilege, n*—capacity assigned to an entity by an authority.

3.1.31 *privilege management infrastructure (PMI), n*—complete set of processes required to provide an authorization service.

3.1.32 *privilege policy, n*—policy that outlines conditions for verifiers to provide/perform sensitive services to/for qualified claimants.

3.1.32.1 *Discussion*—Privilege policy relates attributes associated with the service as well as attributes associated with claimants.

3.1.33 *public key infrastructure (PKI), n*—complete set of processes required to provide encryption and digital signature services.

3.1.34 *relying party, n*—user or agent that relies on data in an attribute assertion or certificate to make an access control decision.

3.1.35 *role, n*—job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role.

3.1.36 *role certificate, n*—certificate that assigns privileges to a role rather than directly to individuals.

3.1.36.1 *Discussion*—Individuals assigned to that role, through an attribute certificate or public-key certificate with a subject directory attributes extension containing that assignment, are indirectly assigned the privileges contained in the role certificate. Some PMIs use two types of X.509 attribute certificates: *(1)* role-specification AC held by the role containing privileges held by the role, and *(2)* role-assignment AC held by the user containing roles assigned to the user.

3.1.37 *sensitivity, n*—characteristic of a resource that implies its value or importance.

3.1.38 *source of authority (SoA), n*—special type of attribute authority upon which a verifier endows unlimited privilege.

3.1.38.1 *Discussion*—The verifier trusts the SoA to delegate that privilege to owners, some of whom may further delegate that privilege to other owners.

3.1.39 *security domain, n*—set of subjects, their information objects, and a common security policy (NIST Special Publication 800-33).

3.1.40 *security policy, n*—statement of required protection of the information objects.

3.1.41 *structural role, n*—job function within the context of an organization whose permissions are defined by operations on workflow objects.

3.1.41.1 *Discussion*—Structural roles provide authorizations on objects at a global level without regard to internal details. Examples include authorization to participate in a session, connect authorization to a database, authorization to participate in an order workflow, or connection to a protected uniform resource locator (URL). A structural role applies to the business process task as a group.

3.1.42 *target, n*—resource being accessed by a claimant.

3.1.43 *verifier, n*—entity responsible for performing or providing a sensitive service for/to qualified claimants.

3.1.43.1 *Discussion*—The verifier enforces the privilege policy. When validating certification paths, a verifier is a type of relying party.

3.1.44 *workflow, n*—representation of an organizational or business process in which documents, information, or tasks are passed from one participant to another in a way that is governed by rules or procedures; a workflow separates the various activities of a given organizational process into a set of well-defined tasks.

3.2 *Acronyms:*

3.2.1 *AA*—attribute authority

3.2.2 *AARL*—attribute authority revocation list

3.2.3 *ACRL*—attribute certificate revocation list

3.2.4 *ADF*—access control decision function

3.2.5 *AEF*—access control enforcement function

3.2.6 *AC*—attribute certificate

3.2.7 *ACI*—access control information

3.2.8 *ADF*—access decision function

3.2.9 *ADI*—access control decision information

3.2.10 *AEF*—access enforcement function

3.2.11 *ANSI*—American National Standards Institute

3.2.12 *ARL*—authority revocation list

3.2.13 *BPEL*—business process execution language

3.2.14 *CA*—certification authority

3.2.15 *CCOW*—clinical context object workgroup

3.2.16 *CIM*—common information model

3.2.17 *CORBA*—common object request broker architecture

3.2.18 *COTS*—commercial off the shelf

3.2.19 *CPU*—central processing unit

3.2.20 *CRL*—certificate revocation list

3.2.21 *DEA*—Drug Enforcement Agency

3.2.22 *DHHS*—Department of Health and Human Services

3.2.23 *DMTF*—Distributed Management Task Force

3.2.24 *DSA*—digital signature algorithm

3.2.25 *DTD*—document-type definition

3.2.26 *EHR*—electronic health record

3.2.27 *EIS*—entity identification service

3.2.28 *ER*—emergency room

3.2.29 *FIPS*—Federal Information Processing Standards

3.2.30 *HL7*—Health Level Seven

3.2.31 *ID*—identification

3.2.32 *IdM*—identity management

3.2.33 *IDS*—intrusion detection system

3.2.34 *IETF*—Internet Engineering Task Force

3.2.35 *INCITS*—International Committee for Information Technology Standards

3.2.36 *I/O*—input/output

3.2.37 *ISO*—International Organization for Standardization

3.2.38 *LDAP*—lightweight directory access protocol

3.2.39 *MDS*—multiple document signatures

3.2.40 *MOU*—memorandum of understanding

3.2.41 *NIST*—National Institute of Standards and Technology

3.2.42 *OASIS*—Organization for the Advancement of Structured Information Standards

3.2.43 *OCSP*—online certificate status protocol

3.2.44 *OMG*—object management group

3.2.45 *PA*—privilege allocators

3.2.46 *PAP*—policy administration point

3.2.47 *PDP*—policy decision point

3.2.48 *PEP*—policy enforcement point

3.2.49 *PHR*—personal health records

3.2.50 *PII*—personally identifiable information

3.2.51 *PKI*—public-key infrastructure

3.2.52 *PMI*—privilege management infrastructure

3.2.53 *PPS*—permission policy set

3.2.54 *RBAC*—role-based access control

3.2.55 *RIM*—reference implementation model

3.2.56 *RPS*—role policy set

3.2.57 *RuBAC*—rule-based access control

3.2.58 *SAML*—security assertion markup language

3.2.59 *S/MIME*—secure/multipurpose internet mail extensions

3.2.60 *SOA*—service-oriented architecture

3.2.61 *SoA*—source of authority

3.2.62 *SOAP*—simple object access protocol

3.2.63 *SPML*—security provisioning markup language

3.2.64 *UDDI*—universal description, discovery, and integration

3.2.65 *UHID*—universal healthcare identifier

3.2.66 *UML*—unified modeling language

3.2.67 *URI*—uniform resource identifier

3.2.68 *URL*—uniform resource locator

3.2.69 *WS-XACML*—XACML profile for web services

3.2.70 *XACML*—eXtensible access control markup language

3.2.71 *XML*—eXtensible markup language

3.3 *Terminology Comparison*—See Table 1 for terminology comparison.

4. Significance and Use

4.1 Motivation for the PMI comes from several organizational and application areas. For example:

4.1.1 Supporting a distributed heterogeneous application architecture with a homogeneous distributed security infrastructure leveraged across the enterprise; providing user and service identities and propagation; and providing a common, consistent security authorization and access control infrastructure.

4.1.2 Providing mechanisms to describe and enforce enterprise security policy systematically throughout the organization for consistency, maintenance, and ease of modification and to demonstrate compliance to applicable regulation and law.

4.1.3 Providing support for distributed/service-oriented architectures in which enterprise-wide services and authoritative sources are protected by providing security services that themselves are also distributed using common interfaces and communication protocols.

4.1.4 Providing "economies of scale" where it is desired to change the approach of individually managing the configuration of each point of enforcement to one that establishes a consolidated view of the safeguards in effect throughout the enterprise.

4.1.5 Providing centralized control, management, and visibility to security policy across the enterprise and when connecting to other organizations. This allows for additional key features such as delegated administration, centralized policy analysis, and consolidated reporting.

TABLE 1 Terminology Comparison

ISO	Non ISO
Example: (ISO) 10181-3-00	Example: OASIS XACML
Access Control Enforcement Function (AEF)	Policy Enforcement Point (PEP)
Access Control Decision Function (ADF)	Policy Decision Point (PDP)
Access Control Decision Information (ADI)	Request Context
Initiators	Access Requestor
Target	Resource
ADI Element (format not specified)	Attribute (XML Format)
Initiator ADI (format not specified)	Subject (XML Format)
Attribute Certificate	(not specified)
Rule Element (format not specified)	Condition
Claimant (not in 10181-3-00)	Access Requestor
Context ACI	Environment
Access Control Policy (format not specified)	Policy (XML Format)
Target	Target
(not in 10181-3-00)	Role Policy Set
(not in 10181-3-00)	Permission Policy Set
Role Specification (not in 10181-3-00)	
Role Assignment (not in 10181-3-00)	

4.1.6 Providing a distributed computing security architecture allowing for synchronized security services that are efficiently maintained across the enterprise while also allowing for centralized policy control and distributed policy decision-making/enforcement. Ensuring proper security controls are enacted for each service and when used in combination.

4.1.7 Provisioning incremental updates to policy and configuration data simultaneously across all distributed decision/enforcement points. Establishing and enforcing new policies not envisioned when individual applications were fielded and adapting to new requirements and threats. Managing identity and security implemented in a diverse mix of new and old technologies.

4.1.8 Permitting an organization to grant, suspend, or revoke centrally any or all ability to connect to or access enterprise resources either individually or collectively and with the capability to enforce these policies at run-time.

4.1.9 Supporting access decisions that are sensitive to a user's credentials in addition to identity. For example, the user may have to be a licensed healthcare professional to access a medical record.

4.1.10 *Supporting Delegation*—A user might delegate access for a resource to another user (for example, a physician might delegate access to his patient's records to a specialist). This shows the need for a delegation capability for some applications.

4.1.11 *Supporting Sender Verification*—When a user receives a signed document, he shall be sure the sender was, in some sense, authorized to sign and send the document. A simple example would be a prescription that shall be signed by a doctor. A simple identity certificate is insufficient, as it does not indicate the sender's credentials (that is, that he is a doctor).

4.1.12 *Supporting Document Cosigning*—Multiple examples exist in which more than one signature is required on a document (2). For example, a transcriptionist transcribes and signs a document, but it is not a valid part of the record until it is reviewed and signed by the primary care physician. Similar mechanisms can be used to provide cosignature controls when processing claims transactions. These types of applications require the ability to convey user authorizations (in assertions, credentials, authorization certificates, or possibly as extensions in identity certificates), to label documents and other objects with their security attributes (or to extract such attributes from the document), and to express authorization rules in machine-readable form.

4.2 Existing standards, including ANSI X9.45, ISO 9594-8, IETFRFC 3280 X.509, OASIS SPML, SAML, WS-*, and XACML, define a number of mechanisms that can be used to construct a healthcare-specific PMI specification. This would include the following features:

4.2.1 Privileges needed to access a target are conveyed in a claimant's authorization credential. The claimant's authorization credential may be an authorization certificate compliant with ISO 9594-8 (a particular form of attribute certificate) or a policy set description compliant with XACML or other referenced authorization standards.

4.2.2 The sensitivity or other properties of the target being accessed may be held in a local database or in a signed data structure. This guide does not define a standard way to represent this information, since this is a local matter. It does provide guidance on how such information might be represented and manipulated using common mechanisms such as ASN.1 and XML. For a given target object, there may be multiple operations that may be performed; each such operation may have a different set of sensitivity attributes.

4.2.3 The privilege policy may be held centrally, locally, or may be conveyed as a signed data structure. Different operations on a target may be subject to different privilege policies. This guide defines several standard policies, and applications may define additional policies.

4.2.4 In the document authorization paradigm, cosignature requirements may be associated with a user or document, such that the signed document is considered authorized only if all necessary signatures are attached.

4.2.5 Users may delegate privileges to other users.

4.2.6 Users may be assigned to roles that convey permissions.

4.2.7 Some authorizations may be sufficiently dynamic that it is not feasible to place them in an enterprise authorization infrastructure (that is, the cost of maintenance is too high given the short lifetime or rapid frequency of change of the privileges or constraints). Such authorizations may be kept in a local authorization server's database and accessed as environmental variables.

4.3 The remaining sections of this guide discuss mechanisms to convey privilege, sensitivity, and policy information in a distributed PMI.

5. Models

5.1 *General:*

5.1.1 Privilege management and authorization may be assigned to individual actors or to groups of individual actors playing the same role. Actors interacting with system components are called principals and can be a human user, system, device, application, component, or even an object.

5.1.2 Privilege management and access control management models typically contain entities and acts (3). Examples of which include:

5.1.2.1 *Entities:*

(1) Principals,
(2) Policies,
(3) Documents, and
(4) Roles.

5.1.2.2 *Acts:*

(1) Policy management,
(2) Principal management,
(3) Privilege management,
(4) Authentication,
(5) Authorization,
(6) Access control management, and
(7) Audit.

5.1.3 To obtain the above described structure and functionality, a number of models, mechanisms, processes, objects, and so forth, are needed. This section considers the following security models:

5.1.3.1 Domain model,

5.1.3.2 Control model,
5.1.3.3 Delegation model,
5.1.3.4 Document model,
5.1.3.5 Policy model,
5.1.3.6 Role model,
5.1.3.7 Information distance model, and
5.1.3.8 Authorization model.

5.1.4 For enabling future-proof electronic health record (EHR) systems, all specifications made must be kept open, platform-independent, portable, and scalable. Therefore, the models provided will be described at the meta-model level and at the model level, keeping the instance level out of consideration.

5.2 *Domain Model:*

5.2.1 To manage and operate complex information systems that support shared care, principal-related components of the system are grouped into domains by common organizational, logical, and technical properties. Following Object Management Group's (OMG) definition, this could be done for common policies (policy domains), for common environments (environment domains), or common technology (technology domains). Any kind of interoperability internal to a domain is called an intradomain communication and cooperation, whereas interoperability between domains is called an interdomain communication and cooperation. For example, communication could be between departments of a hospital internally to the domain hospital (intradomain communication) but externally to the domain of a special department (interdomain communication). Real-world systems are most likely composed of multidomain information objects that cut across different information contexts.

5.2.2 As used here, a security domain is a set of subjects, their information objects, and a common security policy (NIST Special Publication 800-33). A domain is characterized by a domain identifier, domain name, domain authority, and domain qualifier (ISO/TS 22600-2:2006).

5.2.3 Within a security domain, all information objects exist at the same level of sensitivity. Members of a domain may have different security attributes, such as read, write, or execute permissions on information objects. Security domains are not bound by systems or networks of systems. A security domain's objects may reside in multiple systems.

5.2.4 A policy describes the legal framework including rules and regulations, the organizational and administrative framework, functionalities, claims and objectives, the principals involved, agreements, rights, duties, and penalties defined, as well as the technological solution implemented for collecting, recording, processing, and communicating data in information systems. For describing policies, methods such as policy templates or formal policy modeling might be deployed (4). For example, W3C WS-Policy provides a general purpose model and syntax to describe and communicate the policies of a web service. It specifies a set of common message policy assertions within a policy and attachment mechanisms for using policy expressions with existing XML service technologies.

5.2.5 In the domain model, a domain may consist of subdomains or participation in superdomains. Subdomains will inherit policies from their parent domain. The domain may be extended by chaining subdomains to superdomains, forming a common domain of communication and cooperation that is characterized by establishing an agreed upon security policy. Such transaction-concrete policy has to be negotiated between the communicating and cooperating principals, which is also called policy bridging.

5.2.6 Users may perceive a collection of objects from different security domains as a single object. This compound object is referred to as a multidomain information object. In compound security domains, additional policies shall be written that apply to the newly created multidomain information objects. The multidomain information security policy shall state the privileges that a user must have to view, print, create, delete, or transfer multidomain information objects between information systems. It cannot be assumed that the compound domain policies are simply inherited from the subdomains.

5.2.7 The general purpose of communication is the provision of services to a client requesting these services. Most of the services have to be provided by the functionality of the healthcare information system often combined with human users' interactions. Such application services are end-system services.

5.2.8 Middleware concepts are being increasingly introduced into newer versions of healthcare information systems. In this model, not only both principals, but also the middleware, can provide requested functionality and application security services. Such an architecture can be represented by chains of different domains.

5.2.9 From the security point of view, a domain ensuring intradomain communication according to its own policy is commonly considered to need protection only at its boundary to external domains, with their specific policies (or even the policy-free domain of the internet). This is done by firewalls, proxy servers, and so forth. Regarding the external environment, a domain is therefore often considered closed. The internal domain is mistakenly assumed to be secure, often neglecting internal threats and attacks that are the majority among all security attacks (NIST Special Publication 800-33).

5.2.10 Regarding the specific requirements and conditions of healthcare, the underlying security model shall consider the whole spectrum of security services and mechanisms that can be accomplished by secure microdomains.

5.2.11 *Interaction between Security Domains*—Separate security domains in the domain model can exchange privilege information by agreement of the parties. This interaction between security domains shall be coordinated on both a technical and documentary level. The creating and exchange of privilege sets should take into consideration organizational structure.

5.2.11.1 *Technical Basis*—Exchanges of privilege information shall be examined to ensure the meaning of privileges is consistent between security domains. This can be accomplished by creating a standard set of privileges. The standard set of privileges may include a mutually defined mapping of equivalent privileges between the domains. The equivalence of the exchanged privileges shall be reviewed on a technical basis to ensure the intended security implications are achieved.

5.2.11.2 *Administrative Basis:*

(1) Privilege information exchanged between security domains may involve separate administrative entities (for example, distinct business partners or companies). An agreement as to the exchange of privileges and their use shall be documented, typically in a "business partner agreement." The use of a business partner agreement is required to distinguish the legal, ethical, and practical responsibilities between business partners and that may extend between other cooperating PMI implementations. An equivalent procedure is performed in a PKI through the use of a certificate practices statement and certificate policies. An alternative procedure uses "policy assertions" in WS-Policy.

(2) Multiple security domains may exist within a single company or organization. An agreement documenting responsibilities between such domains should also be set forth in a business partner agreement or a memorandum of understanding (MOU). The document should be periodically reviewed to ensure privileges extended across security domains exist only as long as required to meet the needs of the enterprise.

5.2.11.3 *Organizational Considerations*—Privilege information exchanged between security domains should be structured to reflect organizational considerations. Establishing a security domain that encompasses an organizational objective (for example, accounting or human resources) provides a coherent approach. The resulting standard set of privileges suitable for interdomain exchange is, as a result, highly cohesive. That is, the privilege set provides privileges to a subset of the organization (for example, accounting) without extending privileges required in an unrelated subset of the organization (for example, human resources). In addition, a cohesive privilege set provides all privileges that are required to meet a specific objective.

5.3 *Control Model:*

5.3.1 The control model illustrates how control is exerted over access to operations on protected objects. There are five components in the model: claimant, verifier, target, access control information, and access control policy. The claimant has privilege attributes assigned by enterprise domain authorities (contained in various forms such as an attribute certificate, SAML assertion, and so forth), pushed or pulled to the verifier (at run-time or through provisioning). The target gets various other access control information (for example, sensitivity attributes, environmental variables) that may be contained in a security label (separating policy between different information domains), attribute certificate, or in a local database. The model described here enables the verifier, who may be the owner of the target or an independent authority, to control access to the target by the claimant, in accordance with the control policy and other access control conditions (see Fig. 1).

5.3.2 To control access, the verifier must be able to do two things. First, it must be able to get all applicable control policies for the target. Then it shall compare these with the privileges known for the claimant. Access is granted if and only if the privileges owned by the claimant dominate the control and sensitivity policies of the target consistent with all other applicable access control information. In practice, the verifier consists of access control decision functions and access control enforcement functions that may also be combined or separated. If separated, the claimant's privileges may be available to the verifier in a wide variety of ways depending upon specific strategies.

5.3.3 The generality of this model makes the names of its parts appear somewhat abstract; however, with suitable interpretation, it can be applied to all the situations introduced in this guide.

5.3.4 *Use of Push or Pull:*

5.3.4.1 The control model uses a verifier that acquires access control information to make an access control decision. The claimant can provide the information (for example, in a token) along with the request to the verifier (push) or the verifier can get the required information from a trusted source (pull). In deciding whether to use a push or pull model, several factors should be considered:

(1) Push:

(a) Tokens should have a short time to live,

(b) Tokens shall be validated against an authentication service,

(c) Token delivery should be encrypted, and

(d) Tokens should include a nonce or a unique key within the encrypted token.

(2) Pull:

(a) Authentication services shall be accessible,

(b) Authentication service ACI repository shall be accessible,

(c) A trusted communication path to the ACI repository is required, and

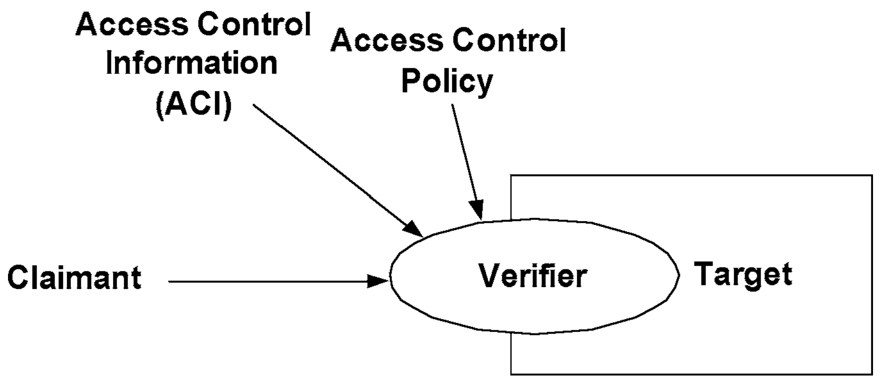

FIG. 1 Control Model

(d) Authentication information should be current with ACI.

5.3.4.2 Tokens holding authentication information shall be sufficiently secure for the environment to guard against the possibility of replay attacks.

5.4 *Delegation Model:*

5.4.1 In addition to the control model, there is a need for a delegation model. There are three components of the delegation model: the verifier, the source of authority (SoA), and the claimant (see Fig. 2).

5.4.2 The verifier endows an entity known as the SoA with unlimited privilege. The SoA assigns privilege to claimants by issuing attribute credentials (for example, attribute certificates or SAML attribute assertions). The claimant asserts its delegated privilege by demonstrating its identity. This can be done by any mechanism consistent with the domain security policy, for example, by a password/identification or, increasingly, by proving knowledge of a private key whose public counterpart is contained in a public key certificate referenced by the claimed privilege. In a security distributed/service-oriented architecture, user authentication to the verifier would be provided by an authentication service directly.

5.4.3 Since an authentication service can also provide single sign-on capability, there shall be a relationship to the authorization service to ensure that claimant privileges are available to the verifier at the time when the claimant access request is made. In the case of an executable object, it may alternatively be done by demonstrating that the digest is the same as the "owner" value of an attribute certificate which includes the claimed privilege.

5.4.4 Optionally, the claimant may delegate its privilege to another claimant (delegate).[11] The verifier shall confirm that all entities in the delegation path possess sufficient privilege to access the target requested by the direct claimant. The verifier should determine that claimants possess the level of access required to access the target. Restrictions on the delegation may be established by the SoA, claimant, or by the target and include:

5.4.4.1 *Delegation Level*—For example, the claimant in delegate role may not be allowed to further delegate privilege.

5.4.4.2 *Delegation Context*—For example, delegate my "assigned-radiologist" privilege only for a given patient identity and only for a given set of X-ray images and only for a specified period of time.

5.4.4.3 *Delegate Set*—For example, no restrictions on number of levels of delegation, but all delegates shall be from a specified set of claimants.

5.4.4.4 *Reference Restriction*—The rights to use an object under specified circumstances are passed as part of the object reference to the recipient. For example, in privilege delegation, the initiating principal's access control information (that is, its security attributes) may be delegated to further objects in the chain to give the recipient the rights to act on the initiating principal's behalf under specified circumstances.

5.4.4.5 *Improper Delegation*—Restrictions that prevent a delegate from assigning rights to inappropriate delegates, for example, a clinician assigning rights to order medications to administrative staff.

5.5 *Document Model:*

5.5.1 Processes, entity roles, and so forth shall be documented and signed expressing the particular relations between entities and processes. The combination of processes and relations leads to multiple signatures (for example, in the case of delegation).

5.5.2 The document model shall support multiple document signatures (MDS). Cryptographic message syntax supports multiple signatures on a document. Each signature is computed over the document content and optionally a set of signed attributes specific to the particular signature. These attributes may include timestamps, signature purpose, and other information.

5.6 *Policy Model:*

5.6.1 A security policy is the complex of legal, ethical, social, organizational, psychological, functional, and technical implications for assuring trustworthiness of health information

[11] Distinguish this model based on discretionary access control from "delegated administration." For example, a physician may be granted some record access privilege by a hospital. The physician may assign the office manager responsibility to extend (delegate) that privilege to various roles which do not include the office manager. In this circumstance, there is an entity in the delegation path that does not possess access privilege. This delegation administration model implemented in some Personal Health Records (PHR) implementations (cf. iMetrikus).

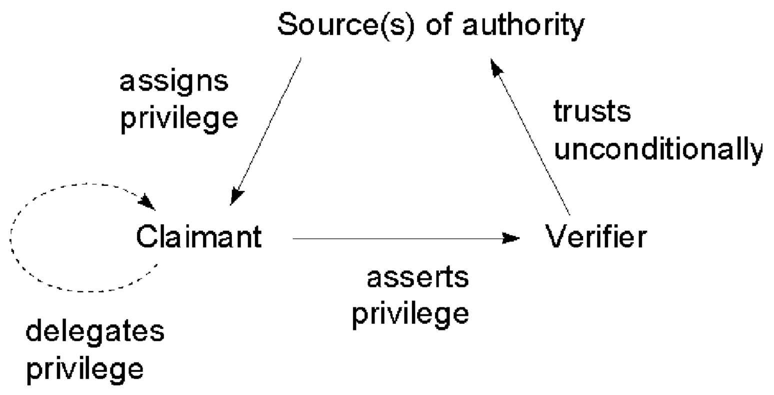

FIG. 2 Delegation Model

systems. A policy is the formulation of the concept of requirements and conditions for trustworthy creation, storage, processing and use of sensitive information. A policy can be expressed:

5.6.1.1 Verbally unstructured,

5.6.1.2 Structured using schemata or templates, or

5.6.1.3 Formally modeled, for example, using Unified Modeling Language (UML).

5.6.2 For interoperability reasons, a policy shall be formulated and encoded in a way that enables its correct interpretation and practice (**5**). Therefore, policies have to be constrained regarding syntax, semantics, vocabulary, and operation of policy documents, also called policy statements or policy agreements (agreements between the partners involved).

5.6.3 *Policy Expression*—Several approaches to expression of policy in a formal manner have been developed. Some of these are presented here.

5.6.3.1 *XML Schema:*

(1) Fig. 3 presents a simple generic XML instance for a security policy statement. One common way to express constraints is the specification of a defined XML schema. This schema should be standardized for interoperability purposes mentioned in the previous section.[12]

(2) To refer to a specific policy reliably, the policy instance shall be uniquely identified. The same is true for all the policy components such as domain, targets, operations, and their policies, which have to be named and uniquely identified too. As with any other component, policy components can be composed or decomposed according to the generic component model. A policy is therefore characterized by a policy identifier, a policy name, a policy authority, a domain identifier, a domain name, a target list, a target identifier, a target name, and a target object with its operations allowed and related policies. This guide assumes the use of XACML for expressing XML security policies.

5.6.3.2 *Ponder-Language Base-Class:*

[12] OASIS provides standardized schemata for implementing security policy in its eXtensible Access Control Markup Language (XACML) specification.

(1) The policy class can be specialized as basic policy, meta policy, and composite policy classes, as shown in Fig. 4.

(2) The specializations of the composite policy abstract class are interrelated in a complex way, which has been indicated in outlines as simple association.

5.6.3.3 *Object Management Group Security Services:*

(1) Another approach to policy decomposition has been provided by the OMG's Security Services Specification distinguishing between the following policies:

(a) Invocation access policy implementing access control policy for objects,

(b) Invocation audit policy controlling event type and criteria for audit, and

(c) Secure invocation policy specifying security policies associated with security associations and message protection.

(2) Policies regarding requirements for different object types include:

(a) Invocation delegation policy,

(b) Application access policy,

(c) Application audit policy, and

(d) Non-repudiation policy.

(3) Health information systems such as the EHR should at minimum have a Patient Policy, an Enterprise Policy, policies defined by laws and regulations, and one policy per Role.

(4) Every creation, access, or modification to an EHR component must be covered by one or more policies. The reference model of the EHR Extract includes a Policy ID attribute within the Record Component class to permit references to such policies to be made at any level of granularity within the EHR hierarchy. The policies that apply specifically to an EHR may be included within the EHR Extract, eventually including any bridged policies, as discussed in the domain model above.

5.7 *Role Model:*

5.7.1 Roles can be defined for managing relationships between principals (claimants) and objects. Roles are collections of permissions that allow principals to perform operations on protected information objects associated with work profile

```
<policy>
          <policy_name/>
          <policy_identifier/>
          <policy_authority/>
          <domain_name/>
          <domain_identifier/>
          <target_list>
                    <target_name/>
                    <target_ID/>
                    <target_object>
                              <operations/>
                              <policies/>
                    </target_object>
          </target_list>
</policy>
```

FIG. 3 Policy Template Example

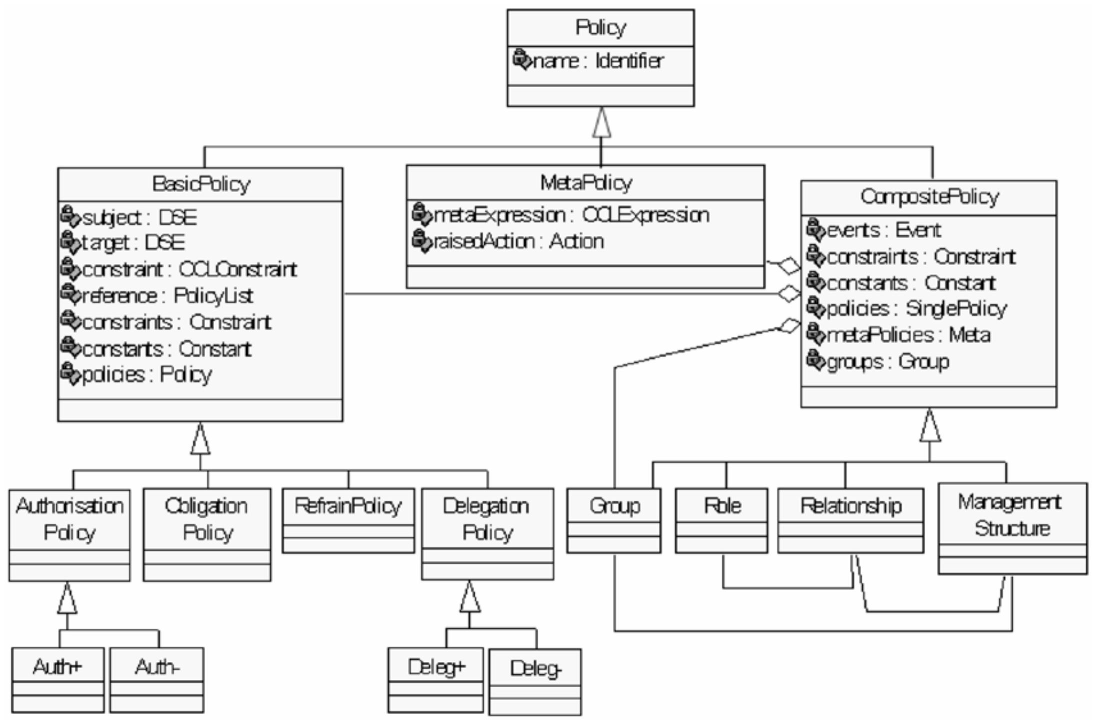

FIG. 4 PONDER Base-Class Diagram (23)

scenarios. Principals are the actors in healthcare. Roles may be assigned to any principal. Therefore, roles are associated to actors and to acts.

5.7.2 The ANSI RBAC standard defines permissions as actions on protected objects; however, it does not define specific actions or objects. Objects may exist at different logical levels. For example, concrete objects may be defined as individual data elements, tables, or aggregations of data elements and tables. More abstract objects include work profiles, tasks, scenarios, or steps.

5.7.3 Fig. 5 presents an adapted role-based access control schema from the ANSI RBAC standard. In the figure, the function, session roles, gives the roles activated by the session and the function, session users, gives the user that is associated with a session. The permissions available to the user are the permissions assigned to the roles that are currently active across all the user's sessions.

5.7.4 Each model component is defined by the subcomponents:

5.7.4.1 A set of basic element sets,

5.7.4.2 A set of RBAC relations involving those element sets (containing subsets of Cartesian products denoting valid assignments), and

5.7.4.3 A set of mapping functions that yield instances of members from one element set for a given instance from another element set.

5.8 *Information Distance Model:*

5.8.1 In considering the distance of persons to personal information, three types of person with growing distance to the information can be specified:

5.8.1.1 Originator of information (holder of data),

5.8.1.2 Producer of information (interpreter of data), and

5.8.1.3 Administrator of information (user of information). In a healthcare environment, the originator of information is normally the patient and the producer of information is the doctor. An example of an information user is a pharmacist.

Following the need to know principle, an increasing distance to information causes greater restrictions regarding privileges granted.

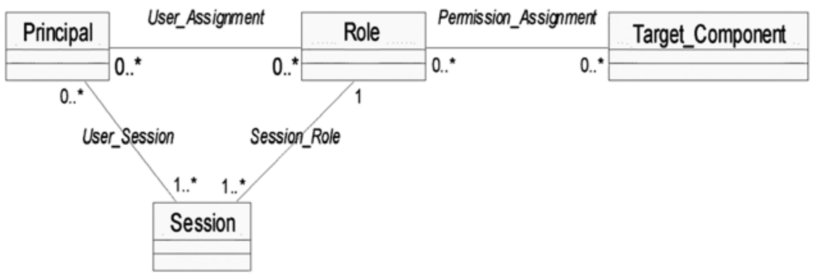

FIG. 5 Role-Based Access Control Diagram

5.9 *Authorization Model*—A variety of commonly used authorization models are discussed in 5.9.1.

5.9.1 *Role and Permission Assignments:*

5.9.1.1 Roles provide a means to assign permissions indirectly to individuals. Individuals are granted one or more roles defined by role attributes (permissions). Permissions are assigned to role-by-role specifications rather than to individuals. The indirect assignment enables the permissions assigned to a role to be updated without impacting the mechanisms that assign roles to individuals. Role credentials for a user (claimant) may be based upon SAML attribute assertions, attribute certificates, public-key certificates, entries in a directory service, XACML attributes, or other standard mechanisms.

5.9.1.2 Considerations for use:

(1) Any number of roles can be defined,

(2) The role itself and the principals assigned to the role can be defined and administered separately,

(3) Role assignment, just as any other privilege, may be delegated, and

(4) Roles and role assignment may be assigned any suitable lifetime.

5.9.1.3 For role assignment, the role attribute is contained in attribute components. A privilege asserter may present a role assignment credential to the privilege verifier demonstrating only that the privilege asserter has a particular role (for example, "manager" or "purchaser"). The privilege verifier may know a priori, or may have to discover by some other means, the permissions associated with the asserted role to accept/reject/modify a request. The role specification component can be used for this purpose.

5.9.1.4 The assignment of permissions to the role may be made within the PMI by a role specification component or outside the PMI (for example, locally configured).

5.9.1.5 The use of delegated roles within an authorization framework can increase the complexity of path processing, because such functionality essentially defines another delegation path which must be followed. The delegation path for the role assignment certificate may involve different authorities and may be independent of the authority that issued the role specification component (**6**).

5.9.1.6 The general privilege management model consists of three entities: the object, the privilege asserter and the privilege verifier. Based on database access control models, Fig. 6 shows a general privilege management model (**7**).

5.9.1.7 There are three principle decisions made in the privilege management context:

(1) Request authorized,

(2) Request denied, and

(3) Request modified.

5.9.1.8 Credentialing, privileging, and authorization are performed by connecting roles to policies.

5.9.2 *X.509 Role-Based PMI:*

5.9.2.1 The X.509 role-based PMI model uses attribute certificates (ACs) described in ISO 9594-8. An example of use of an X.509 role-based PMI is the Permis project (**1**). ACs are issued by Attribute Authorities (AAs). The AC is bound to the

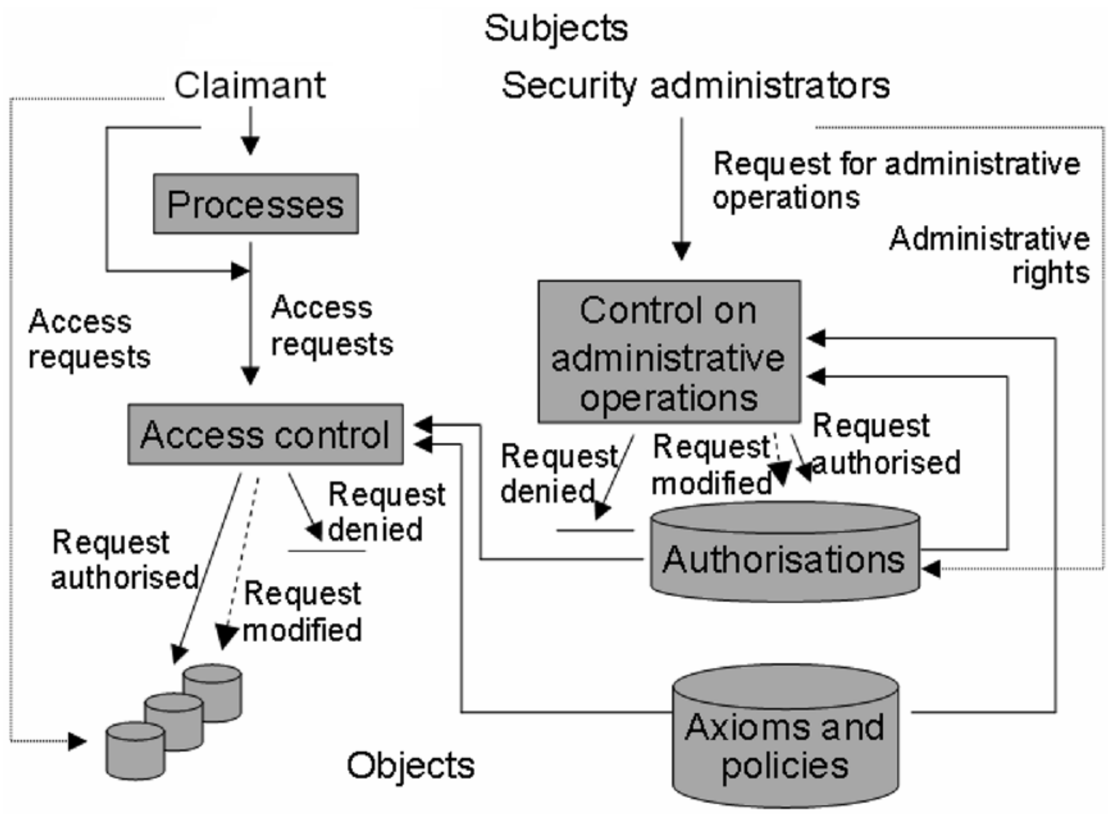

FIG. 6 Privilege Management and Access Control Model, Adapted from Ref (7)

identity using the holder field of an X.509 identity certificate. This coupling permits separate management of PMI and PKI (8). Decoupling PMI and PKI allows sensitive access control information to remain private while identity certificates can be managed by a third party. At least three types of ACs are used in an X.509 role-based PMI: role-specification ACs, role-assignment ACs, and policy ACs. All ACs are digitally signed by the AA and are therefore tamper resistant.

5.9.2.2 Role-specification ACs hold the permission assignments granted to each role. Role-assignment ACs hold the roles assigned to each identity. Policy ACs indicate the root of the PMI trust and contain a pointer to a policy file as an attribute value. ACs are typically stored in a lightweight directory access protocol (LDAP)-enabled directory service. The verifier finds all role-assignment ACs granted to a user and validates the digital signatures and that a certificate has not been revoked. The verifier also finds the role-assignment ACs for each identified user role. This process can be optimized in several ways while keeping the overall strategy.

5.9.2.3 The X.509 role-based PMI can use a policy language such as Ponder, Keynote, or XACML. Use of a domain-wide authorization policy by the verifier provides a secure, centrally managed approach.

5.9.2.4 The local SoA for a security domain creates the domain-wide authorization policy. Privilege allocators (PA) use the policies signed by the SoA, possibly from a different security domain, to generate and authenticate digitally policy ACs. The SoA or the AA uses the PA to sign and publish role-assignment ACs to an LDAP directory. The resulting role-assignment ACs are used by the verifier to make access control decisions.

5.9.3 *XACML Role-Based PMI:*

5.9.3.1 The Organization for the Advancement of Structured Information Standards (OASIS) standards group developed the eXtensible Access Control Markup Language (XACML) as a language to express and evaluate access decisions. The XACML technical specification includes a profile for RBAC using XACML that complies with the ANSI RBAC standard.

5.9.3.2 For the convenience of the reader, terms found in the NIST core RBAC document (9) are compared to the terms in the XACML profile and the terms in this guide in Table 2.

5.9.3.3 The XACML RBAC profile also supports hierarchical RBAC, allowing inheritance between roles. Additional XACML policies are provided to support system and review functions described in the ANSI RBAC standard. Specifically, the Role PolicySet (RPS) associates holders of a given role attribute with a Permission PolicySet. The Permission PolicySet (PPS) describes the permissions associated with a specific role. The RPS and PPS replace the role assignment and role specification ACs in the X.509-based role model.

5.9.3.4 The XACML role-based PMI features a rich and extensible policy language integrated throughout the design. The concept of structural versus functional roles is supported using a two-tiered system comprised of role attributes. That is, users can have roles assigned to them in the request context. An entity separate from the policy decision point can use an XACML role assignment policy or PolicySet to enable attributes within the user session.

TABLE 2 RBAC Core Functionality Mapping

Core Element	XACML Profile	ASTM PMI
Users	XACML Subjects	Claimants
Roles	XACML Subject Attributes	Roles
Objects	XACML Resources	Objects
Operations	XACML Actions	Operations
Permissions	XACML Role <PolicySet> and Permission <PolicySet>	Permissions

6. Privilege Management Infrastructure Framework

6.1 The PMI framework establishes relationships between components of an abstract role system to the components of the underlying security infrastructure.

6.1.1 In Fig. 7, the control model of Fig. 1 is extended to the security distributed or service-oriented architecture. The verifier is shown with its component PDP/PEP. In the security SOA, infrastructure security authentication and authorization is provided to applications as a service. The figure abstracts the access control models of ISO 10181-3-00 (the PDP/PEP terminology follows OASIS XACML). If the external (to the target) verifier is removed, then we have the traditional control model. Various implementations are achieved by functional allocations of service between application-level and SOA levels. Access control information is used to inform the Verifier PDP of additional conditions affecting the decision. ISO ACI includes:

6.1.2 *Initiator ACI:*

6.1.2.1 Individual access control identities,

6.1.2.2 Identifier of hierarchical group in which membership is asserted, for example, organizational position,

6.1.2.3 Identifier of functional group in which membership is asserted, for example, membership of a project or task group,

6.1.2.4 Role that may be taken,

6.1.2.5 Sensitivity markings to which access is allowed,

6.1.2.6 Integrity markings to which access is allowed,

6.1.2.7 A target access control identity and the actions allowed on the target—that is, a capability,

6.1.2.8 Security attributes of delegates, and

6.1.2.9 Location, for example, sign-on workstation.

6.1.3 *Target ACI:*

6.1.3.1 Target access control identities,

6.1.3.2 Individual initiator access control identities and the actions on the target allowed or denied them,

6.1.3.3 Hierarchical group membership access control identities and the actions on the target allowed or denied them,

6.1.3.4 Functional group membership access control identities and the actions on the target allowed or denied them,

6.1.3.5 Role access control identities and the actions on the target allowed or denied them,

6.1.3.6 Authorities and the actions authorized for them,

6.1.3.7 Sensitivity markings, and

6.1.3.8 Integrity markings.

6.1.4 *Action ACI:*

6.1.4.1 ACI associated with operating zoning action (data ACI), for example:

(1) Sensitivity markings,

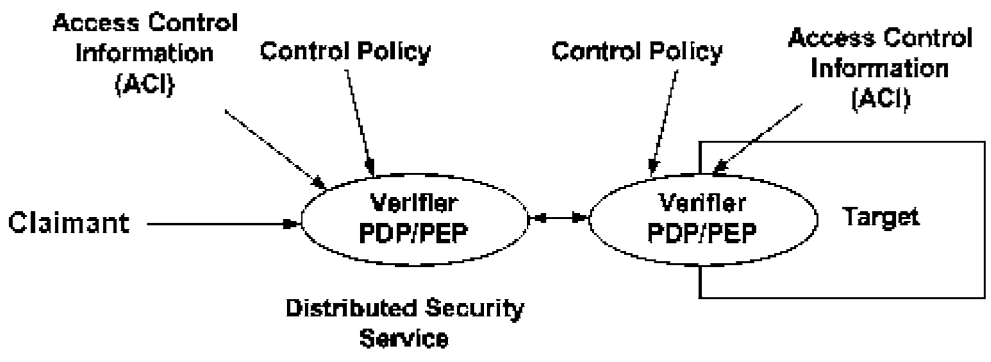
Access Control Information (ACI)
Control Policy
Control Policy
Access Control Information (ACI)
Claimant
Verifier PDP/PEP
Verifier PDP/PEP
Target
Distributed Security Service

FIG. 7 Extended Control Model

(2) Integrity markings,
(3) Originator identity, and
(4) Owner identity.

6.1.4.2 ACI associated with the action as a whole, for example:
(1) Initiator ACI,
(2) Permitted initiator and target pairs,
(3) Permitted targets,
(4) Permitted initiators (claimants),
(5) Allowed class of operations (for example, read, write), and
(6) Required integrity level.

6.1.5 *Contextual ACI:*
(1) Time periods,
(2) Route (an access may be granted only if the route being used to specific characteristics),
(3) Location (and access may be granted only two initiators as specific in-systems, workstations are terminals, or only two initiators any specific physical location),
(4) System status (and access may be granted only for a particular ACI when the system has a particular status, for example during a disaster recovery),
(5) Strength of authentication (an access may only be granted when authentication mechanisms of at least a given strength are used), and
(6) Other access currently active for this or other initiators.

6.1.5.1 There are two types of high-level healthcare role supported by the infrastructure: structural roles and functional roles. Structural roles reflect the structural aspects of relationships between entities. Structural roles describe prerequisites, feasibilities, or competencies for acts. Functional roles reflect functional aspects of relationships between entities. Functional roles are bound to the realization/performance of acts.

6.1.5.2 Possible examples for structural roles of healthcare professionals are:
(1) Medical director,
(2) Director of clinic,
(3) Head of the department,
(4) Senior physician,
(5) Resident physician,
(6) Physician,
(7) Medical assistant,
(8) Trainee,
(9) Head nurse,
(10) Nurse, and
(11) Medical student.

6.1.5.3 Possible examples for functional roles of healthcare professionals are:
(1) Caring doctor (responsible doctor),
(2) Member of diagnostic team,
(3) Member of therapeutic team,
(4) Consulting doctor,
(5) Admitting doctor,
(6) Family doctor, and
(7) Function-specific Nurse.

6.1.5.4 A detailed description of structural and functional roles is presented in the following sections.

6.2 *PMI Management:*

6.2.1 *Need for a Governance Framework:*

6.2.1.1 Proper management of the PMI requires a security governance framework that includes the establishment of chains of responsibility, authority, and communication to empower people to control the system effectively. Governance is important for the security services, as managing the security policy and implementation is vital to the integrity of the environment (NIST Special Publication 800-100). However, governance is only discussed briefly here, since the focus of this guide is on the privilege management infrastructure.

6.2.1.2 Governance in the service creation scenario involves monitoring compliance of the security services with the security policies, monitoring compliance with governance structures in place, and monitoring the overall security effectiveness of the environment.

6.2.1.3 Security compliance management measures the performance of the security implementation relative to the measures defined by the security policy. These can be realized based on reporting on system behavior using audit information and comparing that behavior to configured policies in systems. When these are viewed in the context of business-defined policies, it can provide an overarching view of where a business stands in implementation and enforcement of intended policies.

6.2.2 *Trust Management:*

6.2.2.1 From a business viewpoint, trust management includes the liability and legal aspects around the access control and authorization services. It also includes protection messages that the service provider can implement for sensitive data (**10**).

6.2.2.2 At a technology level, trust management (cf. WS-Trust) may include:

(1) The protocols for the service consumer to contact the service provider and exchange security policy and authorizations. For example, this may require a simple object access protocol (SOAP) message carried on HTTPS.

(2) The security token and its contents (credentials and authorizations) that need to be included in a WS-security message. For example, an SAML assertion carrying role-based information is required.

6.2.3 *Identity Management Services:*

6.2.3.1 The following enterprise identity management high-level needs are identified:

(1) To identify all persons of interest uniquely. This includes persons that have customer, contractual, or employee roles, or combinations thereof;

(2) To provide an enterprise identity management service/capability to support the various business lines within the enterprise; and

(3) To facilitate the sharing of information between internal lines of business and with external partners.

6.2.3.2 Identity management services within an enterprise may use several user repositories. To facilitate common views these disparate user repositories may be organized into meta-directories and virtual directories. In order to meet identity management needs, the following identity management services capabilities are identified:

(1) Enumeration—The process whereby a unique identifier is assigned to an identity, for example, UHID.

(2) Correlation—The process whereby an identifier from a line of business system is cross-referenced to an identifier determined to represent the same entity.

(3) Synchronization—The process of evaluating and applying updates to an entity's set of traits.

(4) Maintenance—The process whereby correlations or enterprise identities are maintained as valid. The processes include moving correlations, moving enterprise identities, and deprecating enterprise identities.

6.2.3.3 In identity management, provisioning policy is defined to automatically create user accounts, in the enterprise repository and in the entity identification service (EIS) system. User accounts may have differing user names but shall be capable of being correlated to the individual user.

6.2.3.4 This provisioning policy can be extended to cross the enterprise boundaries so the user is also created in the registry used for the external service consumer. This provisioning policy may include several workflow activities, for example, getting user management approvals.

6.2.3.5 As part of these provisioning policies, identifier and password policies have to be taken into account. Identifier policies define how the different attributes for the different accounts are created based on the user identity information and the company security rules.

6.2.3.6 In this example, an identifier policy may be defined to create the accounts on a system using the first letter of the first name and the letters of the family name for a user. This can also apply to other attributes, such as an e-mail address.

6.2.3.7 Password policies can be used to enforce the way passwords for the different user accounts are created and managed. For example, it can be decided to define a policy requiring a minimum length, the inclusion of numeric and special characters, and a specific expiration date.

6.2.3.8 Federation policies can be used to allow the validation, mapping, and exchange of the different security tokens that are used between the domains or systems.

6.2.3.9 Enterprises shall provide a capability that provides access on a least-privilege basis with a need to know for protected health information that is based upon users' roles in the organization and the tasks they are assigned.

6.2.3.10 Organizations therefore require an integrated approach to security that provides the infrastructure to meet the following needs:

(1) To authenticate consistently users across enterprise and extranet/federated boundaries,

(2) To authorize/grant consistently users permissions to protected information assets,

(3) To enforce robustly access by authenticated and authorized users to protected information assets,

(4) To audit access to and use of sensitive information and functions, and

(5) To meet applicable guidelines and mandates for information security.

6.2.3.11 Users may need to use a user name and password to authenticate to their portal or a strong authentication.

6.2.3.12 An SAML assertion may be required at the service components level to authenticate a user accessing the service through an external consumer, while a user name token can be enough for an internal application. The user name token provided may be different from the one used to authenticate to the local portal. Finally, on the application, the user credential is validated and mapped to the local account identity.

6.2.4 *Access Management:*

6.2.4.1 The definition of the access management policies covers the authentication services as well as the authorization and privacy services (**10**). This requires the definition within the policy infrastructure of the appropriate access control policies entitling access to authorized users. These definitions can change depending on the component performing the security enforcement, as they may not use the same access control policy format.

6.2.4.2 The service components may define a first set of authentication and authorization requirements to prevent unauthenticated users to access the service. Then finer-grained authorization can be done through role-based security.

6.3 *Engineering Security Policy:*

6.3.1 For managing access to protected entities, role-based access control provides access control information expressing security policy enforced between users and protected resources in a security domain. Among the possible types of roles, structural and functional roles are defined as key elements of role ontology.

6.3.2 Structural roles may specify broad relations between entities in the sense of competence (for example, HL7 Reference Implementation Model (RIM) roles), reflect organizational or structural relations (hierarchies), as well as provide qualifications for access to high-level organizational workflow.

6.3.3 Functional roles are bound to an act. Functional roles can be assigned to be performed during an act. They may correspond, for example, to HL7 RIM participation and provide fine-grained access control information needed to access protected objects and functions within the context of an application.

6.3.4 As an expression of security policy, roles are easily understood; however, more complex rules may not be readily stated as roles. Rule-based access control (RuBAC), as opposed to role-based access control (RBAC), allow users to access systems and information based on predetermined and configured rules (**11**). Both RBAC and RuBAC are expressible in a common engineered policy language providing for uniform enforcement and management. The following section describes RBAC (structural and functional roles) and RuBAC as elements of engineered security policy.

6.3.5 *Structural Role Framework:*

6.3.5.1 Structural roles[13] place people in the organizational hierarchy as belonging to categories of healthcare personnel warranting differing levels of access control. Similar to organizational roles, structural roles allow users to participate in the organization's workflow (for example, tasks) by job, title, or position but do not specify detailed permissions on specific information objects. Some structural role examples include: physician, pharmacist, registered nurse supervisor, and ward clerk. Structural roles may be found as noncritical certificate extensions entries to an X.509 certificate as specified in Practice E2212.

6.3.5.2 Structural roles serve as access control decision information within the PMI at a coarse-grained level by allowing authenticated users to establish a session or connect to a protected target.

6.3.5.3 To accomplish this function, the user asserts or is verified to possess, in addition to authentication information, appropriate structural roles as a prerequisite authorization to "connect" to the task or workflow containing the requested session or target. An infrastructure access enforcement function grants or denies access to the session or target based on the structural role. Structural roles would be typically managed in identity certificates (per Practice E2212) or directories. Structural roles allow an enterprise to grant, suspend, or deny access (by means of the service-oriented verifier) to any or all resources through a single point of management and control.

6.3.5.4 Guide E1986 identifies healthcare persons for whom role-based access control is warranted. These Guide E1986 person types define structural healthcare role names used within this guide.

6.3.5.5 Guide E1986 examples of structural roles for healthcare professionals include:

(1) Physician (MD/allopath, osteopath, chiropractic, naturopath, or homeopath),

(2) Advanced practice registered nurse (NP, NM, CAN, or CNS),

(3) Pharmacist (DP),

(4) Physician assistant (PA),

(5) Technician,

(6) Medical laboratory technician (MLT),

(7) Occupational therapist (OTR/L),

(8) Phlebotomist,

(9) Paramedic,

(10) Admission clerk,

(11) Transcriptionist,

(12) Students,

(13) Administrative support staff and services, and

[13] Structural roles have been introduced in several papers in the early nineties and published, for example, in *Analysis, Design and Implementation of Secure and Interoperable Distributed Health Information Systems* (**12**). They have been formally defined in ISO/TS 21298 Functional and Structure Roles. These are alternatively called static roles, basic roles, or role groups by other sources.

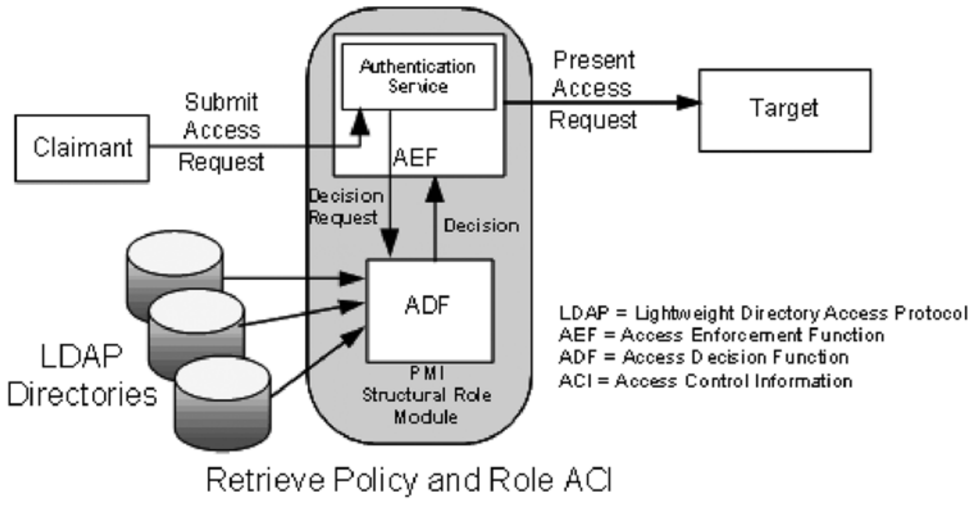
Authentication Service
Submit Access Request
Claimant
Present Access Request
Target
AEF
Decision Request
Decision
ADF
PMI Structural Role Module
LDAP Directories
Retrieve Policy and Role ACI
LDAP = Lightweight Directory Access Protocol
AEF = Access Enforcement Function
ADF = Access Decision Function
ACI = Access Control Information

FIG. 8 Use of Structural Role

(14) Executive officers.

6.3.5.6 *Structural Role Guidelines:*

(1) By assigning Guide E1986 healthcare personnel that warrant differing levels of access control to defined healthcare workflows, they can be used for engineering purposes to define structural roles. These roles provide the requisite precursor role that gives a person access to a "session" or "connection" in ANSI International Committee for Information Technology Standards (INCITS) RBAC. ASTM International structural roles are readily mapped to U.S. National Uniform Claims Committee taxonomies. Other mappings are possible.

(2) In a PMI, structural roles allow a user possessing that role to participate in a work profile. By placing the access control enforcement function for structural roles at a sufficiently high level (as part of an enterprise authentication service), structural roles can be used to grant, suspend, or revoke access to all applications across the enterprise from a single central location. This is useful for immediately revoking privileges for a user across the enterprise for some reason (for example, leaving the organization) or for providing an insulating layer between users and applications. Without the correct structural role, the user is unable to initiate sessions with any or all applications.

(3) Considerations for use:

(a) To control granting of access to one or more applications on an enterprise basis,

(b) To temporarily suspend user access to one or more or all applications for a period of time,

(c) To permanently remove user access to one or more applications on an enterprise basis,

(d) To grant access to an application data and function where the access can be simply stated (for example, URL or directory),

(e) To provide a simplified role hierarchy,

(f) To act as functional roles when the roles for an application are simple,

(g) To quickly change access rights on a global or enterprise basis,

(h) To manage authorizations at the enterprise level,

(i) As an instrument of governance,

(j) To establish broad requirements for external business partner access,

(k) To determine purpose of use, and

(l) To enable emergency access.

6.3.6 *Functional Role Framework:*

6.3.6.1 Functional roles consist of all the permissions on health information system objects needed to perform a task. Functional role names associate groups of permissions for convenience in assignment to users. A user (claimant) may be assigned one or more functional roles and thereby be assigned all of the permissions associated with a corresponding healthcare workflow. Permissions will ultimately be used to set the system operations (for example, create, read, update, delete, execute, and so forth) for data and software applications. Functional roles may be found as entries in a user attribute credential or stored in a distributed authorization directory.

6.3.6.2 Functional role activation cannot occur until the session is established, so structural role authorization/access is prerequisite to establishing a session or connection to the target. In the extended control model, what is desired is a decision on the user's authorizations to perform operations on the target's protected objects. The result of the decision information is used as an input to the verifier policy enforcement point (PEP) for the purpose of access control.

6.3.6.3 Functional roles describe the permissions that a user has available once the session is established and his/her roles are activated. Functional roles are contained in applications, directories, attribute certificates, and XACML extensions. Functional roles specifically define, in terms of permissions, what authorizations are required to access protected resource functions and data. Functional roles that allow the user to participate in the business process workflow are therefore much more granular than structural roles. Functional roles created from standards-based permissions have applicability both within and across the enterprise and with business partners. Functional roles are aligned with existing access control mechanisms by mapping their constituent standard permissions to underlying application enforcement mechanisms. As described in Fig. 9, functional roles may have both external/internal components.

6.3.6.4 The Health Level 7 (HL7) standards development organization has begun an effort to develop standard healthcare-wide permissions that can be used to define interoperable roles. The HL7 effort is based upon a defined role-engineering process broadly based upon work by Neuman and Strembeck (**13**). Currently available as a draft standard for trial use, these standard permissions allow for the expression of functional roles that are ANSI, ISO, and OASIS compliant.

6.3.6.5 *Functional Role Guidelines:*

(1) Functional roles reflect rights needed to perform the essential business functions that need to be performed once access to the application session is granted. Functional roles define what an entity can do once connected to a protected resource.

(2) Considerations for use:

(a) Provide detailed fine-grained access rights needed to specific application data and functions,

(b) Provide standards-based roles for specific communities to avoid duplication and promote functional interoperability,

(c) For the purpose of supporting dynamic roles,

(d) For the purpose of supporting rich hierarchical strategies,

(e) To provide needed granularity to support roles by location, by clinic, and so forth,

(f) To provide flexibility for expressing constraints and other detailed rules,

(g) To support service-oriented architectures and allow for separation of business and security rules,

(h) To simplify management,

(i) For interoperability and cross platform management or for proprietary, application-specific purposes,

(j) Expressions in standard policy language such as XACML,

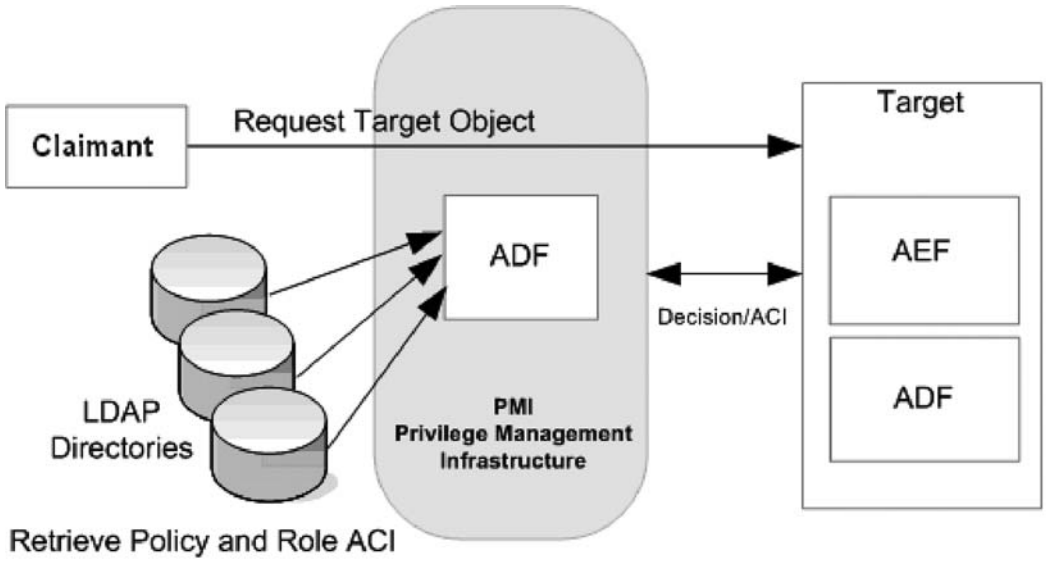

FIG. 9 Use of a Functional Role

(k) Support role provisioning,

(l) To develop functional roles by means of a documented role-engineering process,

(m) Management of roles through an enterprise level management group of healthcare professionals and subject matter experts, and

(n) To provide support for application-specific requirements analysis.

6.3.7 *Relationship of Structural and Functional Roles:*

6.3.7.1 Fig. 10 illustrates the core unified service-oriented role viewpoint describing the relationship between structural roles and functional roles based upon a role-engineering approach.

6.3.7.2 Fig. 10 places the function role into a context that includes structural roles. In this case, the structural role is a prerequisite role that a user must have before additional roles may be activated in a user session.[14] Structural (rarely changing) roles would be typically contained in identity certificates, directories, or SAML assertions.

6.3.7.3 On the right half of the figure, users granted structural roles are permitted to participate in work profiles that contextually allow access to specific enterprise applications/ databases. In a client-server context, this amounts to an authorization to establish a session. In a distributed security service scenario, this access control decision is made at the network level.

6.3.7.4 On the left-hand side of the figure, from the user point of view, he/she has been granted the permissions (aggregated into functional roles according to the principle of least privilege) that allow performing operations on protected information objects associated with the work profile scenarios.

6.3.7.5 In Table 3, several action and object pairs are displayed that may be included in permission definitions. For PMI interoperability, only the structural and functional roles are used.

[14] A structural role could also serve as a functional role, should the security policy permit.

6.3.7.6 Aggregations are composed objects. They are named and treated as complete objects without reference to their internal aggregate data structure.

6.3.7.7 Functional role names compliant with ANSI INCITS RBAC are essentially arbitrary handles on groups of permissions.

6.3.7.8 Describing permissions at the aggregation level provides a clinically relevant way of describing permissions. It replaces traditional lower-level discrete system actions on individual data elements. Furthermore, lower-level actions are typically associated with application-specific mechanisms. Accordingly, for purposes of interoperability, selecting objects at a higher level is more effective for interoperability than selecting objects at a lower level.

6.3.7.9 By analyzing workflow and decomposing actions on objects, permissions and functional roles can be defined. Neumann and Strembeck summarize their general approach in the scenario-driven role engineering process **(13)**.

6.3.7.10 Considerations for use:

(1) Identify and model usage scenarios,

(2) Derive permissions from scenarios,

(3) Identify permission constraints,

(4) Refine scenario model,

(5) Define tasks and work profiles,

(6) Derive preliminary role hierarchy, and

(7) Define RBAC model.

6.3.7.11 The scenario model, as shown in Fig. 11, illustrates the hierarchy of work profile, task, scenario, and step. Permissions are defined relative to steps (described in 6.3.7.12).

6.3.7.12 In the scenario-based role-engineering approach, each action and event within a scenario can be seen as a step that is associated with a particular access operation. Scenarios, which are applied in a particular order to reach a predefined task goal, act as sources for the derivation of permissions. The user performing a scenario shall own all permissions that are needed to complete every step of the scenario. Additional considerations in implementing the role-engineering process include:

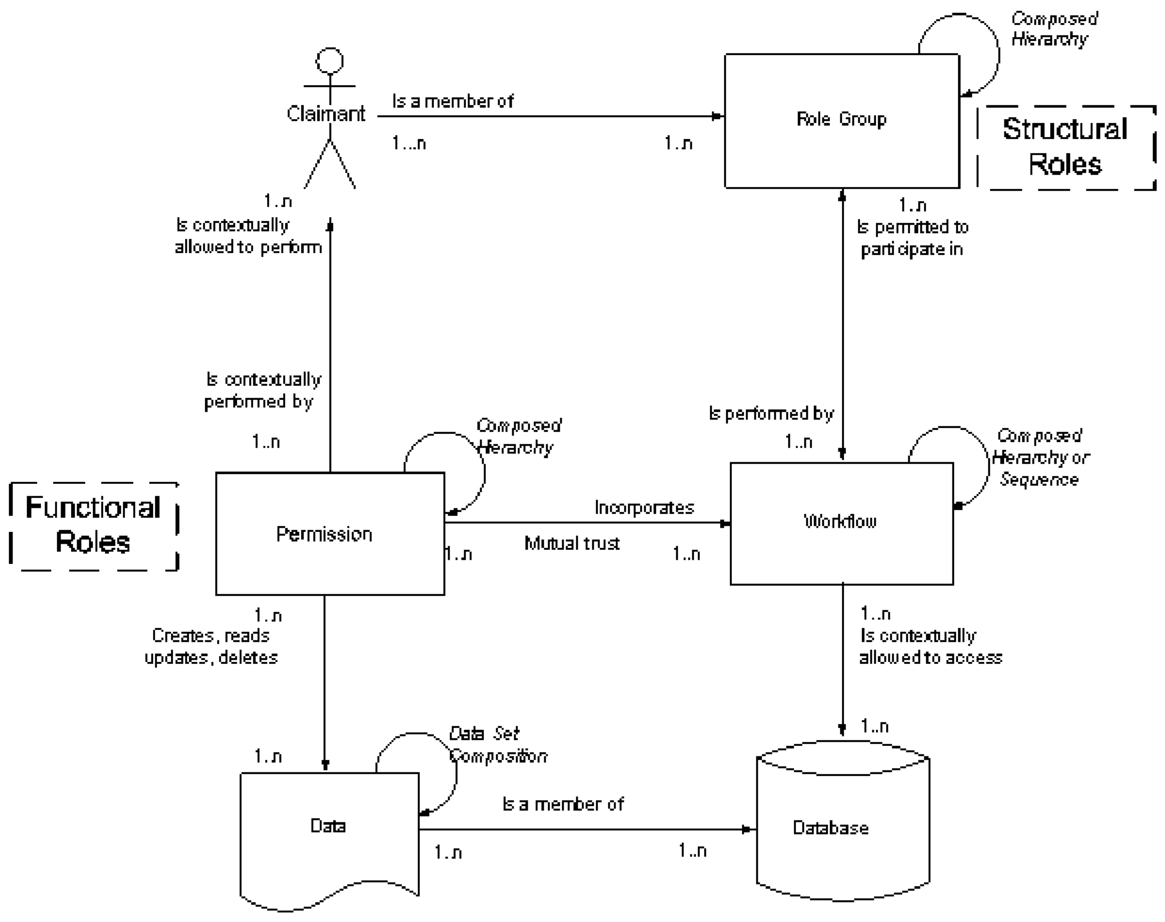

FIG. 10 Role Engineering

TABLE 3 Examples of Objects and Actions at Different Levels of a Role Engineering Hierarchy (Role Ontology)

Action	Role Engineering Object	Role
Participate	Work Profile	Structural Role
Execute	Task	
Execute	Scenario	
Perform	Step	
Create, Read, Update, Delete[A]	Aggregation	Functional Role
Execute	Function	Functional Role
Create, Read, Update, Delete	Data Table	
Create, Read, Update, Delete	Data Element	

[A] Many healthcare organizations do not "delete" objects, but instead add a new object that replaces the older one. In this case, "delete" may be effectively implemented as "addend."

(1) Create or adopt/adapt natural language scenarios that describe a healthcare workflow.

(2) From the scenario description, prepare a sequence diagram that captures actors and actions.

(3) In the diagram, identify where interactions with an information system object occur at an appropriate and consistent level of the role hierarchy (Table 3). These are called steps.

(4) Identify permissions required to perform each step. Enter the permissions into a permission catalog.

(5) In the permission catalog, combine permissions that are considered to be duplicates.

6.3.7.13 *Optional:*

(1) For each permission, refine/map it to actual low-level system access control terms (for example, create, read, update, delete, and execute).

(2) Document constraints in a constraint catalog.

6.3.7.14 *Permission-Permission Constraints:*

(1) Permission constraints are recorded as part of the role-engineering process. For example, a constrained permission occurs when only one role is allowed to perform a particular task at any given time. A constraint occurs for a permission when its definition is tied to cardinality. In this case, the constraint on the permission would be the cardinality specification. Examples of permission constraints include:

(a) Head nurse on a hospital floor (cardinality of one),

(b) Chief of staff (cardinality of one),

(c) Lab technician versus lab technician supervisor (separation of duties),

(d) Provider's access to a remote hospital that is not his/her primary workplace (location), and

(e) Physician working in a clinic (time dependency) versus physician working in the emergency room (ER) (no time dependency).

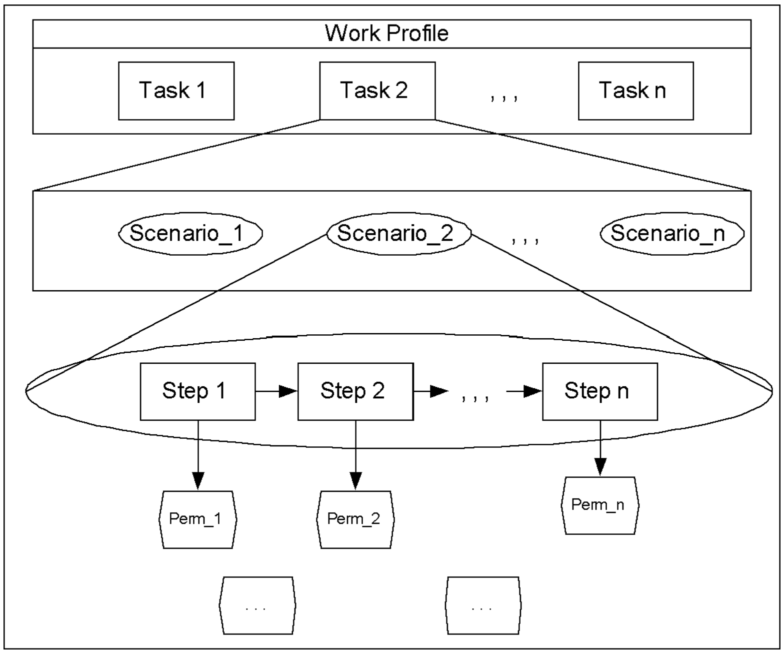

FIG. 11 Role Engineering Model from Ref (13)

(2) The discussion in 6.3.7.14*(1)* relies on the fact that the functional role name is arbitrary and only has meaning as a collection of ANSI INCITS compliant permissions. Furthermore, the defined roles are necessarily specific to the organization defining them and, hence, do not by themselves provide any interoperability with business partners other than through business partner agreement. To achieve true interoperability, organizations will require standardization of the healthcare permissions and a means to advertise and assert standard rights.

6.3.8 *Rule-Based Access Control Framework:*

6.3.8.1 Rule-based access control allows enterprises to create logical expressions that reflect security policy in ways that would be difficult otherwise. Rule-based access control can be used to express business rules, for example, that control access to system resources at a detailed level. By separating business rules for business process from those that control access, it is possible to create new models for access control of fine-grained decisions.

6.3.8.2 Rule-based access control may also be a promising mechanism for dealing with complex privacy rules. For example, the Health Insurance Portability and Accountability Act provides patients the opportunity to request restrictions on access to their data. Similar provisions are reflected in the laws of many countries. Rule-based access control can possibly enforce rules such as this in which there are no existing well-defined security mechanisms.

6.3.8.3 Rule-based access control implemented as a service in an enterprise authorization framework appears especially attractive. For example, privacy rules could be enforced globally without requiring recoding of each application handling personal health information. Since the rules and their enforcement occur outside of the application, it is possible to create and enforce dynamically new policies never envisioned when the applications were created, all without reworking or touching the applications themselves.

6.3.8.4 While there is currently no standard way of implementing rule-based access control, a standards-based language is one way to express and leverage policy enforcement among applications. Centralized policy management also provides for consistent policy description and enforcement. The guideline language for this standard is OASIS eXtensible Access Control Markup Language (XACML).

6.3.8.5 Considerations for use:

(1) Identifying RuBAC as a component of enterprise strategic planning, architecture, and migration approaches,

(2) Balancing RuBAC use between embedded application code and service-oriented authorization approaches,

(3) Integrating RuBAC with other mechanisms such as RBAC,

(4) Separating security logic embedded in legacy applications from business code,

(5) Balancing "wire-level" protocols in application development environments against network-level protocols such as SOAP,

(6) Availability of hardware-based accelerators to speed slower XML parsing (**14**),

(7) Availability of XACML graphical user interfaces rather than manually constructing complex XACML expressions, and

(8) Use of standards-based protocols for exchanging RuBAC policy with business partners.

6.3.9 *Separating Security from Business Logic:*

6.3.9.1 In designing a service-oriented PMI framework, there is opportunity to distinguish between enforcement of security policy from execution of business logic. To implement RuBAC as a service, it is advantageous to separate and isolate security and business application code whenever possible. This separation is essential for security engineers to obtain a clear definition of security boundaries, ensure security code is not affected by changes to business logic, and support certification.

6.3.9.2 Consideration for use:

(1) Relieve business applications developers from writing, managing, testing, certifying, and maintaining embedded security code,

(2) Create security services that are independent of application-level details and that can be managed and maintained separately from application development, and

(3) Improve applications maintainability by separating business and security layers.

6.3.9.3 Separating security from business logic can be difficult. In general, access to a resource within business logic deals with interaction upon a set of resources the user has authority to access. Rules for separating security logic can be identified using the criteria in 6.3.9.4.

6.3.9.4 Consideration for use:

(1) Involves a change of security state (for example, confidential to nonconfidential),

(2) Involves enforcing confidentiality, integrity, or availability of data,

(3) Involves enforcing concepts of least privilege, need to know, or separation of duties,

(4) Involves enforcement of security regulatory requirements through standards-based security mechanisms, and

(5) Involves the collection, parsing, and evaluation of access control information defined at the beginning of this section (initiator ACI, target ACI, action ACI, and contextual ACI).

6.3.9.5 *Privacy:*

(1) Privacy is a special case of the authorization in which access to personally identifiable information (PII) is controlled. This type of authorization is usually based on the data being retrieved. For example, it may be okay to access a customer's home address but not their home phone number. Healthcare privacy policies for other than treatment, payment, or healthcare operations can become complex. Even under these conditions, local rules may prevail. Complex privacy rules, if not carefully managed, could cause denial of service conditions and patient safety issues. Privacy generally not known to security personnel and adding privacy administrators to security poses new issues. Enforcing privacy rules in security would involve significant unplanned costs. Furthermore, security standards that could enforce privacy "authorizations" do not address privacy explicitly.

(2) Privacy protections, as commonly defined, require knowledge of the application data values and some sort of labeled security. Typical commercial off-the-shelf (COTS) security subsystems, below ISO Layer 7, are architecturally unaware of such values and do not support mandatory access controls using data labels.

(3) Even though industry expects privacy and security to be closely related, standards are lacking that define the relationships among privacy protected healthcare data values, labels, policies, and access controls.

(4) A major issue is that privacy is not a universal concept at the detail level of law, regulation, localized social custom, organizational policies, and personal sensitivities. Even if XACML were used to convey privacy policies, there is currently no standard vocabulary and grammar to express these privacy details. Since individual healthcare record entries have been described as having variable privacy issues, we also need to deal with practical matters of bandwidth and storage space allocated to data labeling. The practicality of labor or automated techniques to label the data is also a concern.

(5) Considerations for use:

(a) If privacy rules do not require the same level of assurance as security policies, consider enforcing complex privacy rules using application code. This avoids the overhead associated with security development, testing, certification, and management as a "security policy."

(b) If privacy rules can be expressed in simple name-value pairs, then use of privacy "policy point" may be an option; however, such structures can also be readily created without using security services.

(c) Security services of confidentiality, integrity, and nonrepudiation are more understood in supporting privacy. In assigning policy rules to security for enforcement of access control, be aware that most security organizations do not currently consider "privacy" as a type of security policy, nor do they have experience in implementing detailed privacy polices.

(d) Consider application-layer "break-glass" barriers to enforce privacy policies before allocating to security services.

(e) Consider the impact that privacy policies enforcement may have in implementing "emergency access" when life or threat of injury to the patient is at stake.

6.3.9.6 *Business Process Execution Language (BPEL):*

(1) As organizations implement a service-oriented approach for managing their business processes, services are becoming the fundamental elements of application development. BPEL is a standard for orchestrating these services and managing execution of business processes.

(2) Most business processes contain multiple decision points in which certain criteria are evaluated. Based on these criteria/business rules, business processes change their behavior. Thus, these business rules drive the business process. These rules are mostly embedded within the business process itself causing the following problems:

(a) Most organizations lack a central repository. Consequently, organization-wide change in policy cannot be applied across all business processes.

(b) Business rules change more often than the processes themselves. Thus, developers often have to commit lots of time to changing and managing these embedded business rules.

(c) Business processes are capable of reusing the rules. Hence, developers end up designing rules for each and every process leading inconsistency or redundancy or both.

(3) Using a rules engine eliminates the above problems by separating business processes from business rules. In this approach, rules are exposed as services and BPEL processes leverage these services by querying the engine when they reach the decision points (**15**).

6.4 *Provisioning:*

6.4.1 Provisioning refers to the process of managing attributes and accounts within the scope of a defined business process or interaction. Provisioning an account or service may involve the creation, modification, deletion, suspension, or restoration of a defined set of accounts or attributes.

6.4.2 Provisioning of user access control credentials refers to the creation, maintenance, correlation, synchronization, and deactivation of user objects and user attributes, as they exist in one or more systems, directories, or applications, in response to automated or interactive business processes. Provisioning software may include one or more of the following processes: change propagation, self-service workflow, consolidated user administration, delegated user administration, and federated change control. Provisioning is typically a subsystem or function of an identity management system that is particularly useful within organizations in which users may be represented by multiple-user objects on multiple systems.

6.4.3 Considerations for use:

6.4.3.1 Provides the ability to provision incremental updates to policy and configuration data simultaneously across all distributed decision/enforcement points,

6.4.3.2 Automates the process of user management (authorization, roles, and authentication), and

6.4.3.3 An extensible, XML-based standard language.

6.5 *Policy Enforcement Point Guidelines:*

6.5.1 The design of a PMI can influence placement of the PEP. In a distributed PMI, a PEP can be placed at the application level or the enterprise level. The following factors are important in deciding the proper placement.

6.5.2 Considerations for use:

6.5.2.1 Availability of the PEP,

6.5.2.2 Reduction of the number of PEPs,

6.5.2.3 Ability to manage centrally,

6.5.2.4 PEP lifecycle maintenance,

6.5.2.5 Physical security of the PEP,

6.5.2.6 PEP access to the network, and

6.5.2.7 PEP proximity to the application.

6.5.3 Co-location of the PEP with applications will decrease latency. In these cases, a local policy store should be available to allow the policy engine to work during extranet outages.

6.5.4 *LDAP-Enabled Directory Service Versus Policy Engines:*

6.5.4.1 The choice of claimant mechanism can have an effect on performance and complexity of the PMI framework. LDAP-based mechanisms are typically fast but provide simple data, for example, whether the claimant is a member of a specific role. Alternatively, policy engines can factor in several independent elements into the authorization decision process. Considerations are listed in the following:

(1) LDAP—Use of an LDAP-enabled directory service is suggested when speed is required and the decision can be based on possession of a certain role, especially structural roles (see 6.3.5).

(a) Considerations for use:

(1) Data retrieved consists of single strings,

(2) There is a sensitivity to processing overhead,

(3) There is a need for documented interface, and

(4) There is no need for evaluating policies.

(2) Policy Engine—Policy engines offer flexibility but reduced speed compared to LDAP lookup when implemented in general purpose software engines. Special purpose hardware-based policy engines, on the other hand, offer advantages over LDAP when evaluating complex rules, constraints, and combining rules from multiple policy points. Note that caching of decisions (within their validity period) from a policy engine keyed by a hash of the request attributes can improve performance significantly in an environment in which many accesses are being made in a given period of time and many of these involve the same request attributes.

(a) Considerations for use:

(1) Data retrieved consists of complex or multiple elements,

(2) Requirement for the evaluation of multiple elements,

(3) Need to support complex policy languages (for example, XACML), and

(4) Ability to tolerate slower response time.

6.5.4.2 Use of an LDAP-enabled directory service is suggested when speed is required and the decision can be based on possession of a certain role, especially structural roles (see 6.3.5). Note that caching of decisions (within their validity period) from a policy engine keyed by a hash of the request attributes can improve performance significantly in an environment in which many accesses are being made in a given period of time, and many of these involve the same request attributes.

6.5.4.3 Policy engines are appropriate when evaluating participation in a workflow, especially in functional roles (see 6.3.6). The privilege management infrastructure framework may also use a combination of LDAP and policy engine (**16**).

6.6 *Policy Decision Point (PDP):*

6.6.1 Policy decision points (PDPs) can be implemented in many ways depending on the model appropriate to the infrastructure (see SAML 2.0 profile of XACML). The PDP is responsible for deciding if an access control request should be allowed or denied. The decision is based on policies available to the PDP from one or more policy stores. Additional ACI can be used by the PDP in making the access control decision. The decision is returned to the requestor directly to the PEP or through an intermediary such as a context handler.

6.6.2 Considerations for use:

6.6.2.1 Place PDP such that it can be centrally managed,

6.6.2.2 Use XACML to convey access control requests and decisions,

6.6.2.3 Consider the security of the PDP and interprocess communications,

6.6.2.4 Consider if the PDP located on the application (local decisions) or network (global decisions) or both,

6.6.2.5 Consider mechanisms to provide assurance of the decision provided to the PEP, and

6.6.2.6 Place PDP on hardware assisted accelerator to improve performance.

6.6.3 Many design options are possible that increase security and flexibility of the PDP support infrastructure (**17**).

6.7 *Claimant Mechanisms:*

6.7.1 Possible approaches to management of user privileges in a service-oriented PMI architecture include:

6.7.1.1 Storage of privileges in a digitally signed credential,

6.7.1.2 Centralized storage of privileges (for example, in an LDAP-enabled service), and

6.7.1.3 Assertions by an AA.

6.7.2 Storage of privileges in a digitally signed certificate can be based on a public key certificate, an attribute certificate, or an XACML attribute. The advantages to each approach are summarized in ISO 9594-8 (2000). Public key certificates may store privileges as noncritical extensions as described in Practice E2212. The use of an identity certificate in this way tightly binds authentication to authorization and arguably provides resilience to network failure. However, there are several disadvantages in this approach.

6.7.3 Invalidating or changing any privilege owned by the certificate holder would require revoking and reissuing the certificate. Revocation of the certificate typically involves listing the certificate identification number on a certificate revocation list (CRL). As a result, the identity holder is unable to use their card (or other token) until it is reissued. Since identity certificates are typically in the possession of the holder, the reissuing process is unwieldy. Note that use of the identity certificate to store privileges does not provide resilience to network failure, since the CRL must be consulted to validate the certificate. Accordingly, identity certificates are more appropriate for "structural" roles (see 6.3.5) rather than "functional" roles (see 6.3.6).

6.7.4 One alternative is the use of a separate digitally signed certificate, called an attribute certificate, designed to hold user privileges. Additional infrastructure is required to issue and manage this second type of digitally signed certificate. However, there are several advantages in this approach.

6.7.5 The attribute certificate is issued and signed by an attribute authority separate from the certificate authority that manages identity certificates. Authorization and authentication are still tightly bound by placing the user's identity certificate serial number in the holder field of the attribute certificate. Revocation of the attribute can be accomplished using an attribute certificate revocation list (ACRL). However, there is no reason why a user should take physical possession of an attribute certificate. Thus, the attribute certificates themselves can be stored in an LDAP-enabled service. Revocation of the certificate, therefore, involves replacement of the earlier attribute certificate with the current attribute certificate. There is no penalty in frequent changes of privileges held by the attribute certificate seen in the previous mechanism.

6.7.6 There is an additional benefit to the additional infrastructure required to support attribute certificates as pointed out in Ref (**6**). Certificate authorities issuing identity certificates are typically managed by a trusted entity outside the enterprise. Adding infrastructure within the enterprise to support attribute certificates helps keep sensitive privilege information from outsiders. Attribute certificates are typically only needed by the verifier, so there is no need to expose the privileges held by a user to entities outside the enterprise.

6.7.7 *LDAP Lookup (Role Lookup):*

6.7.7.1 Role lookup can be supported by keeping role information in an LDAP structure. LDAP provides fast search and read functionality required in quickly collecting role information required in determining privileges associated with a user. The use of LDAP for role lookup is typically supported by web application servers. Alternatively, LDAP can be called from applications running independently of application servers. The preferred approach is to deploy a PDP that performs the role lookup in responding to an access control request. In any case, LDAP role lookup is beneficial because it provides a mechanism that separates the role information from the application code requesting the information.

6.7.7.2 Role information can be provided from an assertion (for example, an SAML assertion) reducing the need for LDAP lookup.

6.7.8 *Certificates*—As discussed in 6.7.7, certificates provide assurance by binding an identity to a certificate through the use of a cryptographic key. Certificates can be used by services to establish trust relationships throughput the PMI. Trusted certificates can be used to provide assurance over assertions made by trusted services of identity or privilege to a relying party. Identity certificates can provide role information; however, this information should be static in nature (that is, structural roles) because changing role information in an identity certificate necessitates cancellation and reissuance of the certificate. Attribute certificates offer a better way to pass nonstatic privilege and role (that is, functional role) information. Certificates also provide means for confidentiality and nonrepudiation.

6.7.8.1 *Attribute Certificates:*

(1) A user's attribute certificate may contain a reference to another attribute certificate which contains additional privileges. This provides an efficient mechanism for implementing privileged roles.

(2) Many environments which have authorization requirements require the use of role-based privileges (typically in conjunction with identity-based privileges) for some aspect of their operation. Thus, a claimant may present something to the verifier demonstrating only that the claimant has a particular role (for example, "licensed healthcare provider" or "file clerk"). The verifier may know a priori, or may have to discover by some other means, the privileges associated with the asserted role in order to make a pass/fail authorization decision.

(3) Considerations for use:

(a) Lack of reliable communication system,

(b) Desire not to confirm information with issuing authority,

(c) Slow changing or relatively static roles, and

(d) Appropriateness of use to support functional role.

(4) The following are all possible:

(a) Any number of roles can be defined by any AA,

(b) The role itself and the members of a role can be defined and administered separately by separate AAs,

(c) The privileges assigned to a given role may be placed into one or more attribute certificates,

(d) A member of a role may be assigned only a subset of the privileges associated with a role, if desired,

(e) Role membership may be delegated, and

(f) Roles and membership may be assigned any suitable lifetime.

(5) An entity is assigned an attribute certificate containing an attribute asserting that the entity occupies a certain role. That certificate may have an extension pointing to another

attribute certificate which defines the role (that is, this role certificate specifies the role as owner and contains a list of privileges assigned to that role). The issuer of the attribute certificate may be independent of the issuer of the role certificate and these may be administered (for example, expired, revoked, and so on) entirely separately.

(6) Not all forms of **GeneralName** are appropriate for use as role names. The most useful choices are object identifiers and distinguished names.

6.7.9 *Medical Credentials:*

6.7.9.1 One common type of privilege is the user credential. These credentials are issued by a trusted authority and include an identification string. Examples include licensing of medical professionals by state boards and assignment of Drug Enforcement Agency (DEA) numbers. A credential includes a type, an issuer name, and an identifier. Geographically structured issuer names can be useful to indicate state and other locality information. Credentials are typically matched by type (for example, "physician") or type and issuer (for example, "physician licensed in Virginia").

6.7.9.2 If the credential issuer name is absent, then the issuer name from the enclosing attribute or public key certificate is used. If the certificate issuer name is absent, the credential issuer name must be present. (Note that a certificate may explicitly reflect more than one credential, from more than one issuer, to minimize the number of attribute certificate authorities (AAs) in a system.)

6.7.9.3 *Considerations for use:*

(1) Consider use of Practice E2212 descriptions of the use of noncritical X.509 fields to describe a user's medical credentials in an identity certificate,

(2) Consider use of medical credentials (current/noncurrent, location of applicability) as an additional constraint on granting authorizations to clinicians to health care information, and

(3) Alternatively, clinician medical credentials could be considered for inclusion as part of the security provisioning of user attributes in a privilege management infrastructure.

6.7.10 *SAML Assertions:*

6.7.10.1 SAML assertions can be used to transmit security information from an asserting party to a relying party. The assertion can be made on behalf of a subject during authentication or in response to a request from another SAML entity. Assertions can be constructed from privilege or role information stored in either a central store or from signed certificate information.

6.7.10.2 Considerations for use:

(1) Flexibility in federated environment,

(2) Acquire additional information on claimant,

(3) Consideration on network availability and reliability,

(4) Existence of SAML service,

(5) Support for intra-enterprise assertions,

(6) Compatibility with SOAP and WS security, and

(7) Whether one needs simultaneous support for variety of authentication mechanisms (for example, PKI Credentials, Kerberos tokens, biometrics, and so forth).

6.8 *Target Sensitivity Mechanisms*—Management of target attributes (for example, access control lists and sensitivity labels) has traditionally been done on a per-system basis, and there has been little standardization of the representation of this information. This guide does not dictate how such information is represented, but it does give suggestions based on several document syntaxes (ASN.1 or XML).

6.8.1 *Signed Data Encapsulation:*

6.8.1.1 Attributes and other sensitivity information may be bound to the digest of the target using the **SignedData** construct of Specification E2084. In particular, the use of detached signatures (with the object conveyed separately from the signature structure) would be appropriate. Sensitivity information would be carried as signed attributes, with the owner of the information being the signer.

6.8.1.2 The types of authorization information that can be attached to a target include:

(1) Access control information (ACI), as described in ISO 10181-3-00 and Guide E1985,

(2) Cosignature requirements, as described in 6.9.8, and

(3) Descriptive information about the document (for example, document type).

6.8.2 *Use of XML:*

6.8.2.1 Extensible Markup Language (XML) provides a software- and hardware-independent tool for transmitting information. It is expected that many documents will be expressed using XML. The structure for such a document is defined in a document-type definition (DTD) or an XML schema.

6.8.2.2 Privileges may be defined as XML elements in which the name of the element represents the privilege identifier. Alternatively, an XML element can be associated with one or more XML attributes that represent the privilege identifier(s).

6.8.2.3 Privileges can be grouped into useful sets using XML. For example, a set of privileges encoded into XML can be associated with a uniform resource identifier (URI) such as:

```
"urn:application_name:attribute:privilege_set_name"
```

6.8.2.4 Alternatively, privilege sets can be associated within a schema definition addressed by a unique namespace such as:

```
xmlns: privilege_set_name="http://www.astm.org/:privilege_sets/
privilege_set_name/"
```

6.8.2.5 XML privilege sets can be used by the verifier to associate the claimant to its scope of authority. Claimants can be associated with standard groups or roles before evaluation by the verifier. Alternatively, the verifier can associate the claimant to one or more privilege sets through a database or an LDAP-enabled directory service. The verifier can also look up the requestor group or role association in an external XML document using XPath.

6.8.2.6 A verifier using a privilege policy may act directly on the XML elements (for example, by comparing attributes in an authorization certificate to elements in the document). One example of an XML-based policy that can be used to verify privileges is the eXtensible Access Control Markup Language (XACML). The following sections discuss the comparison rules in detail. Generally, single-valued attributes will be compared to a single (complete) element, while multi-valued attributes will be compared to a collection of elements in a model group.

6.8.3 *Access Control Framework:*

6.8.3.1 This section defines an access control information (ACI) attribute that can be used to indicate to a recipient (or trusted third party) which entities may read the target's contents.

6.8.3.2 Access is allowed to the target if the requester matches an entry in the proper list (by name, role, group, or organizational unit) and if the requester's attribute certificate matches all of the constraints contained in the target's ACI attribute. The constraints associated with the requester's attribute certificate would be contained in the **Constraints** attribute in the requester's certificate.

6.9 *Policy Specification Mechanisms:*

6.9.1 Although this guide does not dictate how privilege policies are represented within an end system, XACML provides a standard approach to authorization policies. Several scenarios are evident.

6.9.2 Two entities may need to determine whether their authorization policies are compatible, especially in a web services environment (NIST Special Publication 800-95). If their policies are compatible, the entities need to determine specific policy variable values that are acceptable to both. In this case, an XACMLAuthzAssertion, defined in the "XACML Profile for Web Services (WS-XACML)" may be used. Such an assertion may be included in a WS-policy instance or provided as independent metadata.

6.9.3 An XACMLAuthzAssertion may also be used in a web services environment by a service provider to publish an authorization policy for retrieval by potential clients. Publishing authorization policies is not appropriate for all environments, but publishing certain aspects of an authorization policy may be useful even where publication of an entire policy would be a security problem.

6.9.4 *Transferring XACML Policies:*

6.9.4.1 Policies may need to be transferred from one entity to another in a PMI. Some of the situations in which this is required are:

(1) A PDP evaluates a policy that references other policies by name. The other policies shall be fetched from a policy administration point (PAP) when required for evaluation.

(2) A PDP may need to obtain its "root" policy from the enterprise policy administration point as part of configuration.

(3) A resource may be transferred between security domains, and the source domain may transfer a policy for protection of the resource that the destination domain is responsible for enforcing.

(4) Multiple sites may need to use common policies, even though their PDPs are local for performance reasons. These policies need to be transferred from the central PAP to each site's PDP.

6.9.4.2 While XACML defines a policy language, it is designed to be one component in an overall authorization system. It relies on other components to provide mechanisms for verifying that policy instances were issued by a trusted PAP, for protecting the integrity and confidentiality of instances of policies, and for protocols used to query for and respond with policy instances. XACML has been integrated with the OASIS Security Assertion Markup Language (SAML) Version 2.0 as one way of providing these necessary functions. SAML may be

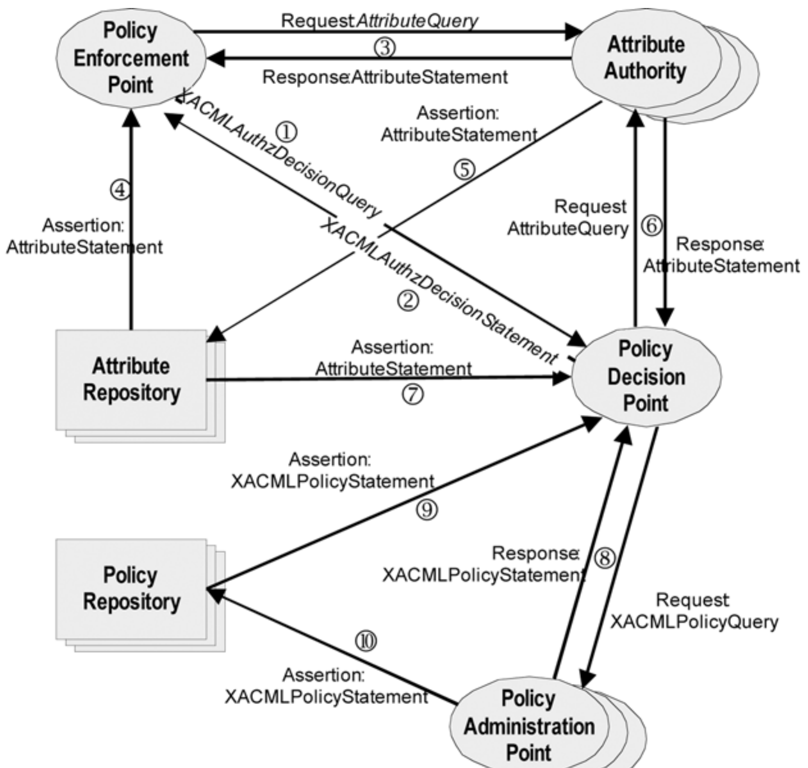
Policy Enforcement Point
Request:AttributeQuery
③
Response:AttributeStatement
Attribute Authority
XACMLAuthzDecisionQuery
①
Assertion: AttributeStatement
⑤
④
Assertion: AttributeStatement
Request AttributeQuery
⑥
Response: AttributeStatement
XACMLAuthzDecisionStatement
②
Attribute Repository
Assertion: AttributeStatement
⑦
Policy Decision Point
Assertion: XACMLPolicyStatement
⑨
Policy Repository
Response: XACMLPolicyStatement
⑧
Request XACMLPolicyQuery
⑩
Assertion: XACMLPolicyStatement
Policy Administration Point

FIG. 12 Using SAML 2.0 to Transport XACML

used with XACML to protect ACI attributes as well as policies. Fig. 12 illustrates the integration of SAML and XACML.

6.9.4.3 As shown in Fig. 12, when the enforcement point requires an authorization decision, a request is made of the PDP (1). The PDP evaluates the request against its available policies and attributes and produces an authorization decision (2) that is returned to the PEP. The PEP may obtain attributes from on-line AAs (3) or from attribute repositories (4) into which AAs have previously stored attributes (5). The PDP may obtain attributes from on-line AAs (6) or from attribute repositories (7).

6.9.4.4 The authorization decision of the PDP is based on policies returned from the PAP (8) or retrieved from the on-line policy repository (9). The policy repository serves as a cache of policies previously stored by a PAP (10).

6.9.4.5 The XACMLPolicyQuery is an SAML query defined in this profile that may be used to request policies from a PAP, either by name or by applicability to a certain request. A corresponding XACMLPolicyStatement is returned in an SAML response. The XACMLPolicyStatement may be digitally signed and may be associated with issuer and validity period information, among other things.

6.9.5 *Credential Matching:*

6.9.5.1 The credential attribute was defined in 5.4. This is issued by a trusted authority and includes an identification string. Examples include licensing of medical professionals by state boards and assignment of DEA numbers. A credential includes a type, an issuer name, and an identifier.

6.9.5.2 To match a credential policy, the claimant's certificates shall, in combination, contain a matching credential for each entry in the credential list. To match an entry, the credential shall have the same credential type, and, if the entry has an issuer name, the credential (or enclosing certificate) shall have the same issuer name.

6.9.6 *Security Label Matching*—Security label matching compares the initiator's clearance to the target's security label. All of the following must be true for authorization to be granted:

6.9.6.1 The security policy identifiers shall be identical,

6.9.6.2 The classification level of the initiator shall be greater than or equal to that of the target (that is, there shall be at least one value in the classification list of the clearance greater than or equal to the classification of the target), and

6.9.6.3 For each security category in the target label, there shall be a security category of the same type in the initiator's clearance and the initiator's classification level shall dominate that of the target.

6.9.7 *General Assertion Matching:*

6.9.7.1 A privilege policy consists of one of the following:

(1) ppPredicate—an assertion about a specific attribute;

(2) and relation—a list of constituent policies, all of which shall be true for this policy to be true;

(3) or relation—a list of simpler policies, at least one of which shall be true for this policy to be true;

(4) not function—a single policy, which shall be false for this policy to be true; or

(5) orderedPPE—a list of simpler policies, which are verified in the order specified.

6.9.7.2 Predicates may be:

(1) single value assertion—a single attribute value in a target document (or context variable) is compared to an attribute value in the assertion;

(2) set value assertion—the entire set of attribute values in a target document (or context variable) is compared to the set of values in the assertion;

(3) present—the attribute shall be present in the document;

(4) approximateMatch—the asserted value(s) match the value(s) in the document, using some locally defined matching algorithms (for example, phonetic matches or approximate arithmetic matches); or

(5) extensibleMatch—the asserted value(s) match the value(s) in the document using a matching rule defined using the X.500 MATCHING-RULE macro.

6.9.7.3 Single value assertions allow authorization based on a simple value comparison. For example, **lessOrEqual** might restrict the signer to some monetary limit. The semantics of each choice are:

(1) equality—The value in the document shall be equal to that in the assertion. This assertion can be used with any attribute type; complex attributes are compared using the DER encodings of their values.

(2) substrings—The value of the document attribute shall contain the asserted substrings in the specified order; the initial substring of the value shall match the initial component of the assertion (if present), the any components (if present) shall appear in the value in the specified order, and the final substring of the value shall match the final component of the assertion (if present). The substrings shall not overlap in the document attribute. This assertion can be used with any ASN.1 string type (for example, IA5 string, UTF8 string, and so forth).

(3) greaterOrEqual—The value in the document shall be greater than or equal to that in the assertion. This assertion may be used with integers, enumerateds, and octet strings.

(4) lessOrEqual—The value in the document shall be less than or equal to that in the assertion. This assertion may be used with integers, enumerateds, and octet strings.

(5) subordinate—The asserted value matches the leading components of the value in the document attribute; it is only valid for object identifiers and names (a sequence of relative distinguished names).

6.9.7.4 Set valued assertions involve all values of an attribute that are found in a target. For example, the standard military compartment mechanism would dictate that the set of compartments attached to a document shall be a **subsetOf** those in the signer's certificate. Similarly, a need-to-know mechanism would use a **nonNullIntersection** assertion. These attributes will be of type SEQUENCE OF or SET OF. The semantics are:

(1) subsetOf—all attribute values in the document shall appear in the assertion;

(2) supersetOf—all attribute values in the assertion shall appear in the document; and

(3) nonNullIntersection—at least one attribute value in the document shall appear in the assertion.

6.9.7.5 A predicate may contain the specific attribute values to be compared against the target, or it may reference an attribute in the claimant's attribute certificate, which holds the value to be compared against the target. In the second case, the **PrivilegeIDPair** contains two attribute types; the first refers to the target, and the second refers to the claimant.

6.9.7.6 Multiple-target attribute syntaxes are supported. Currently, these include ASN.1 and XML. Since claimant privileges are carried as ASN.1 attributes, an attribute type is required in a privilege ID; an XML link is optional (to indicate the corresponding content in the target XML document). The link is structured as defined in the XLink, XPointer, and XPath recommendations, with the additional constraint that it shall reference one or more entire, contiguous XML elements (or their attributes).

6.9.7.7 Mapping between standard ASN.1 types and XML elements is done as follows (where possible, this maps to ongoing work on XML schemas):

(1) An ASN.1 Boolean maps to an XML element content or attribute with the following (case-insensitive) values: For TRUE: true, yes, or 1. For FALSE: false, no, or 0. Only equality matching is allowed.

(2) An ASN.1 integer maps to an XML element content or attribute consisting of solely numeric characters, with an optional sign character (+ or –) in front.

(3) An ASN.1 real maps to an XML element or attribute which uses the ASN.1 value notation for a real number.

(4) ASN.1 bit strings and octet strings map to any XML element content (#PCDATA). As the XML schema work evolves, this should map to an XML binary object; such objects may be encoded in base64 for transport, with the encoding indicated either in the schema or as an attribute of the element.

(5) ASN.1 enumerated map to XML attributes of type NMTOKENS (a list of strings) in which each string is the identifier of one of the enumerated values.

6.9.7.8 AAs should ensure that policies are internally consistent (for example, the same attribute type should not appear in two logically contradictory clauses). Policies should be signed by an AA; they may be conveyed in a claimant's authorization certificate or as separate objects.

6.9.8 *Signature Requirements:*

6.9.8.1 Multiple signatures may be conveyed as multiple **SignerInfo** structures in a **SignedData** instance. Countersignatures may be attached using the (unsigned) **countersignature** attribute. Signature requirements are conveyed as a privilege policy associated with a particular target and operation.

6.9.8.2 Each cosigner entry contains either the identity of a cosigner (a role or an individual name or certificate identifier) or a list of required signature purposes or both. If a role is present, there shall be a signature on the document using that role (as a signature attribute), and the signer shall be allowed to act in the role (as indicated in the **role** attribute in the signer's authorization certificate). If a signer's name is present, there shall be a signature on the document that can be verified using one of that user's certificates. If a particular certificate is identified (by name and key ID or by issuer name and serial number), there shall be a signature on the document that can be verified using the specified certificate. If a list of signature purposes is specified, there shall be a signature on the document using one of the purposes (in the **signaturePurpose** signature attribute). If both a signer ID and signature purpose(s) are present, the specified signer shall use one of the listed purposes.

6.9.8.3 Each cosigner may optionally be assigned a weight to allow a varying number of signers. The quorum specifies the total weight required for the cosigner list to be ratified. In the common case in which all weights are one, the quorum is simply the number of cosigners needed. By assigning weights, however, one could construct a scheme in which (for example) the signature of the president, any two vice presidents, or any four directors is required for authorization. A quorum of zero indicates all list members shall sign the document. A particular placement of the cosignature (joint signature on the document or countersignature) (**2**) may be required.

6.10 *Integration with PKI:*

6.10.1 The PMI relies could be designed to rely on a public key infrastructure (PKI) for identity certificates. These certificates are used to authenticate the owner of an attribute certificate to the verifier (using digital signatures). Each attribute certificate references either the name or (more frequently) an identity certificate of its owner. This decoupling of the PKI and PMI provides several advantages already described.

6.10.2 Attribute certificates are issued by attribute authorities (AAs). These AAs may be arranged in a hierarchy, similar to a CA hierarchy. While identity certificates are requested by the subscriber, issuance of attribute certificates may be unsolicited. This would be the case in which the subscriber does not control his privileges.

6.10.3 Revocation of attribute certificates is done in the same way as identity certificates, (that is, using revocation lists). Alternatively, online protocols like online certificate status protocol (OCSP) may be used. However, there is typically no need for the user to possess their AC physically. Therefore, attribute certificates can be stored in a directory service and simply updated whenever the contents of the attribute certificate are outdated. In this model, certificate revocation is not needed, since access to the attribute certificates storage function provides the most current attribute certificates for the user.

6.10.4 Two types of delegation may be used in a PMI:

6.10.4.1 AAs delegate their own authority to subordinate AAs and end users. Thus, authority increases as one ascends the AA hierarchy. Delegation checks are done on the AA certificates.

6.10.4.2 Users request that their authorizations be delegated, and the AA issues the certificate after performing delegation checks on the delegator's certificate. The delegation hierarchy can be reconstructed using the **delegatorAttributeIdentifier** extension in the attribute certificates.

6.10.5 Integration of the PMI with an existing PKI is discussed further in ISO 9594-8 X.509.

6.11 *Identity Management Systems:*

6.11.1 Several functional areas within the PMI require a secure identity management subsystem (IdM). IdM is responsible for securely providing identity information and identity management functions. Identity management functions include:

6.11.1.1 Administration functions (user provisioning, password management),

6.11.1.2 Identity data control functions (metadata, identity content),

6.11.1.3 Access (authenticate requests, confidentiality),

6.11.1.4 Lifecycle management (configuration, patches, disaster recovery), and

6.11.1.5 Backup, audit, logging, and reporting functions.

6.11.2 IdM shall provide a secure access mechanism for the request and delivery of identity information, typically using mutual authentication. Access to the IdM can be viewed as a service accessible throughout an enterprise. Organizationally, the IdM subsystem can be deployed in various ways:

6.11.2.1 Authoritative source integral to PMI security framework,

6.11.2.2 An administrative function available to the PMI as needed, and

6.11.2.3 Integral to a special purpose IdM product solution.

6.11.3 Having an IdM as a service provides several advantages over a local special-purpose IdM. By their nature, IdM services provide a network interface using standard protocols that provide flexibility in changes to enterprise architecture. An IdM service can be centrally managed, allowing consistent enforcement of security and business policies.

6.12 *Audit:*

6.12.1 The major purpose of the audit subsystem is to provide accountability of actions taken by agents on the network. Audit is not instrumental in the use of privileges to allow or disallow access to protected resources. However, the audit subsystem should interface with the PMI so that its correct operation can be verified. For a discussion of the use of audit in healthcare applications, see RFC 3881.

6.12.2 Accountability is the concept that individual persons or entities can be held responsible for specified actions, such as obtaining informed consent or breaching confidentiality (**18**). Accountability is achieved through the implementation of a pervasive technical audit service. Audit provides a record of potential insecurities irrefutably traceable back to the originator of the action. Security audit provides not only accountability, but a means to assess damage done to a system by malicious action or accident. Security audit generated by the actions of other security services provides a check on their proper operation. In a distributed system, centralized audit collection and processing also provides a method to obtain near-real-time misuse detection and alerts.

6.12.3 A security audit trail provides a journal of security related events collected for potential use in intrusion detection or security audits or both. Audit is a pervasive function of the healthcare system providing essential accountability features. Audit also provides assurance of the correct operation of the system's security features by monitoring user and system access to data and resources. Audit is generated as a by-product of the security controls in place: authentication, access, and authorization (privileging) and upon occurrence of specific security relevant events (for example, modifying a file). Audit acts as a deterrent to (unauthorized) user activities, and as such, users should know that their actions are being monitored (usually part of a log-on banner). Audit also provides a means to assess the degree of harm caused should a break-in occur.

6.12.4 In a distributed architecture involving diverse commercial off-the-shelf (COTS) products, each product produces audit trails in a proprietary format. Even the events recorded may be different from product to product (for example, use of "grant" option makes sense in a database but not in an operating system). System audit trails may be character-based or binary. COTS audit trails often require specialized audit tools for review and processing. Audit trails may be stored in the file system or in database tables and so forth. Audit-analyzing systems shall be able to harmonize and account for these differences.

6.12.5 In distributed systems, audit is produced at multiple locations on multiple components, making review and analysis difficult. Accordingly, in such a system, it is very desirable to consolidate and forward low-level audit from various audit-producing sources to a central audit server. There the audit can be reformatted to a single-composite format and automatically processed by a tool. Several such COTS tools are available, providing for a distributed audit capability for collecting, forwarding, processing, and reporting audit events originating from diverse sources. Since the amount of audit produced may be considerable, a single centralized audit server is a practical way to manage workflow without affecting the response time of operational systems. Audit processing may be both real time and batch.

6.12.6 An automated audit tool provides the means of identifying events at different levels of security, performing automatic profiling, reporting and alerting, and a facility to store, sort, and search for potential insecurities. Automated tools manage audit collection across host- and network-based audit systems. The placement of the audit tool, agents, and components (including real-time network monitoring and intrusion detection) is considered to maximize the effectiveness of the audit system.

6.12.7 Audit records are reviewed by examination of the audit trail. Consolidation of audit records when more than one source is involved at a central "audit server" facilitates review by providing an automated means to examine the (typically) large amount of audit generated from these events. Continuous monitoring of audit records should be a part of the operation phase of the system development life cycle (**19**).

6.12.8 The security architecture should support the establishment of auditing capabilities on an application, facility, or national basis. To meet the requirement for a persistent retention capability, the audit function will include long-term archival and storage facilities. Archival and storage requirements specify the minimum length of time for which the archive shall be retained. Organizations should establish policies and procedures for log management consistent with accepted standards (**20**).

6.12.9 Patient consent can act as the trigger of this audit record. Collection of disclosures made under this requirement requires that the audit configuration for this event be "mandatory." The security architecture supports the centralized collection, processing, and reporting of disclosures of patient information. Storage of events recording certain disclosure under the provisions of the privacy act may require a longer period of storage than simple security audit.

6.12.10 Intrusion detection systems should be an integral part of the distributed architecture. Commercially available intrusion detection systems provide alarm capabilities to permit rapid notification of specified intrusions. Intrusions can be categorized into two main classes, misuse and anomaly intrusions. Misuse intrusions are well-defined attacks on known weak points of a system. They can be detected by watching for certain actions being performed on certain objects. Anomaly intrusions are based on observations of deviations from normal system usage patterns. They are detected by building up a profile of the system being monitored and detecting significant deviations from this profile.

6.12.11 Adherence to industry standards facilitates a robust audit subsystem. Industry standards groups, such as Integrating the Healthcare Enterprise (IHE) publish profiles that describe how to use established standards to share healthcare information better in the clinical setting. IHE has published the Audit Trail and Node Authentication (ATNA) Integration Profile that describes security measures that, together with the security policy and procedures, provide patient information confidentiality, data integrity, and user accountability **(21)**. The IHE ATNA profile is consistent with Dicom Supplement 95: Audit Trail Messages **(22)**.

7. Example Applications

7.1 This section presents some example applications using the mechanisms defined in Section 6. These are not presented in great detail. Specific applications will be the topics of future standards or proprietary specifications. These examples are meant to illustrate the use of privilege management mechanisms to support the types of applications discussed in Guide E1762, Guide E1986, and Specification E2084, as well as current work in the area of certificate policies and extensions.

7.2 *Credentials Application:*

7.2.1 In this application, a physician is prescribing controlled substances. The prescription is electronically signed and sent to the pharmacy using secure/multipurpose internet mail extensions (S/MIME). The pharmacist shall ensure, since the prescription is for a controlled substance, that the physician has a valid DEA number. This would be provided with a credential in either the physician's identification (ID) certificate or attribute certificate. The credential type would be "DEA number" and the issuer would be the DEA.

7.2.2 Similar mechanisms can be used to prove that an individual is a physician (credential type of "medical license"). This can be restricted to a particular state by examining the credential issuer. Note that, by using distinguished names, conventions for issuers can be established at the state level, federal (agency) level, and using Federal Information Processing Standards (FIPS) PUB 66 at the county level.

7.3 *Access Control:*

7.3.1 This application allows a verifier to control access to a target (in this case, some portion of the patient's medical record) based on the target's attributes. In this example, the claimant's role and constraints are found in his attribute certificate. The following constraints are used:

7.3.1.1 Plan registration, and

7.3.1.2 Department.

7.3.2 The attribute certificate for the claimant contains one or more roles, as well as a list of plan registrations and departments with which the claimant is associated. Separate role certificates (with attributes specific to the role) are not used in this application.

7.3.3 The target's access control information is represented using the ACI attribute defined in Specification E2084.

7.3.4 For access to be allowed to the target for this claimant:

7.3.4.1 At least one of the claimant's roles shall appear in the target's access control list,

7.3.4.2 At least one of the claimant's plan registrations shall appear in the target's constraints, and

7.3.4.3 At least one of the claimant's departments shall appear in the target's constraints.

7.4 *Signature Requirements:*

7.4.1 This application builds on the signature purpose mechanism defined in Guide E1762 and Specification E2084. For a document to be accepted as part of the medical record, it shall have one or more signatures, as specified in the privilege policy. For example, the policy might require a signature by the author or by a transcriptionist and a reviewer.

7.4.2 Each signer has an attribute certificate indicating which signature purposes he may exercise. When signing a document, the signature purpose is included as a signed attribute (see Specification E2084). The policy is represented using the **SignatureRequirements** syntax defined in 7.4.1.

7.4.3 The verifier will:

7.4.3.1 Check the attribute certificate of each signer to ensure the signature purpose is allowed, and

7.4.3.2 Ensure that all necessary signatures are present, as required by the privilege policy.

7.5 *Document Authorization:*

7.5.1 This application builds on the mechanisms and attributes defined in ANSI X9.45 and Specification E2084.

7.5.2 Claimant privileges are conveyed in authorization certificates. Claimants may also exercise multiple roles (although only one at a time) through the use of role certificates. The claimant's authorization certificate will contain an **allowableRoles** attribute indicating the roles the user may exercise.

7.5.3 Target attributes may be extracted from the document (for example, as XML elements), held in a local database, or may be embedded in a **SignedData** structure (detached signature). This structure is linked to the target object by the object's digest.

7.5.4 The privilege policy consists of signature requirements and a general assertion policy as defined in 7.3.

7.5.5 The verifier will use the authorization certificate of the claimant, along with associated role certificates, and the target attributes, as input to the general assertion policy in 7.3. If this policy is satisfied, signature requirements are checked. The current signature structure (containing one or more signatures on the document, as well as possibly countersignatures) is matched against the signature requirements policy. If these are also satisfied, the document is considered authorized.

7.5.6 Specific attributes to be included in the authorization certificate include:

7.5.6.1 Restrictions on documents that may be signed (see the document attribute list in 7.5.7),

7.5.6.2 Allowable roles, and

7.5.6.3 Allowable signature purposes.

7.5.7 Attributes associated with the document (mostly from Specification E2084) include:

7.5.7.1 Document type,

7.5.7.2 Location,

7.5.7.3 Patient ID,

7.5.7.4 Event ID,

7.5.7.5 Amendment information (pointer to document being amended),

7.5.7.6 Data type and format information,

7.5.7.7 Originating organization,

7.5.7.8 Event time,

7.5.7.9 Document creation, modification, and access times,

7.5.7.10 Monetary value,

7.5.7.11 Document identifier,

7.5.7.12 Category list, and

7.5.7.13 Owner and author information (which may also be derived from the signatures on the document).

7.5.8 Signed attributes that may be attached to the document include:

7.5.8.1 Signing time,

7.5.8.2 Signature purpose,

7.5.8.3 Role being exercised,

7.5.8.4 Signing certificate and policy ID,

7.5.8.5 Signature reason (textual description),

7.5.8.6 Annotations, and

7.5.8.7 Device identifier of signing cryptographic module.

7.5.9 A countersignature is conveyed as an unsigned attribute that signs the signature value in the **SignerInfo** structure that contains it.

8. Keywords

8.1 access control; delegation; healthcare environment; PMI; privilege management infrastructure; security

REFERENCES

(1) Chadwick, et al, "Role-Based Access Control with X.509 Attribute Certificates," IEEE Internet Computing, 2003, pp. 62-69.

(2) Fischer, A., "Electronic Document Authorization," *Proceedings of the 13th National Computer Security Conference*, 1990.

(3) *Person-Centered Health Records Toward HealthePeople*, J. Demetriades, G. Christopherson, and R. Kolodner, Eds., Spring 2005, pp. 147-168.

(4) Rust, G., "Metadata: The Right Approach—An Integrated Model for Descriptive and Rights Metadata in E-Commerce," *D-Lib Magazine*, July/Aug. 1998.

(5) Arms, W., "Implementing Policies for Access Management," *D-Lib Magazine*, Feb. 1998.

(6) Welch, V., Foster, I., Kesselman, C., Mulmo, O., Pearlman, L., et al, "X.509 Proxy Certificates for Dynamic Delegation," 3rd Annual PKI R&D Workshop, 2004.

(7) Castano, S., Fugini, M., Martella, G., and Samarati, P., *Database Security*, Addison—Wesley Publishing Company, Wokingham, MA, 1995.

(8) Dawson, et al, "A New Design of Privilege Management Infrastructure for Organizations Using Outsourced PKI," ISC 2002, LNCS 2433, 2002, pp. 136-149.

(9) Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E., "Role-Based Access Control Models," *IEEE Computer*, Vol 29, No 2, 1996, pp. 38-47.

(10) Buecher, et al, *Understanding SOA Security Design and Implementation*, IBM Redbook Series, SG24-7310-00, 2007.

(11) "Assessment of Access Control Systems," Interagency Report 7316 (NISTIR 7316), National Institute of Standards and Technology, Sept. 2006.

(12) Blobel, B., *Analysis, Design and Implementation of Secure and Interoperable Distributed Health Information Systems*, IOS Press, 2002.

(13) Neumann, G. and Strembeck, M., "A Scenario-Driven Role Engineering Process for Functional RBAC Roles," June 2002.

(14) Juric, M. B., "Comparison of Performance of Web Services, WS-Security, RMI, and RMI-SSL," *Journal of Systems and Software 79*, May 2006, pp. 689-700.

(15) Bolie, et al, *BPEL Cookbook: Best Practices for SOA-Based Integration and Composite Applications Development*, Pack Publishing, 2006.

(16) Kirschner, B. A., Hacker, T. J., Adamson, W. A., and Athey, B. D., "Walden: A Scalable Solution for Grid Account Management," Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04), 2004 , pp. 102-109.

(17) Stowe, G., "A Secure Network Node Approach to the Policy Decision Point in Distributed Access Control," Thesis, Tech Report TR2004-502, Dartmouth College Department of Computer Science, 2004.

(18) National Research Council, *Computers at Risk: Safe Computing in the Information Age*, National Academy Press, Washington, DC, 1991.

(19) Bowen, P., Hash, J., and Wilson, M., *Information Security Handbook: A Guide for Managers*, NIST Special Publication 800-100, NIST Computer Security Division, Gaithersburg, MD, Oct. 2006.

(20) Kent, et al, *Guide to Computer Security Log Management*, NIST Special Publication 800-92, NIST Computer Security Division, Gaithersburg, MD, Sept. 2006.

(21) "IHE IT Infrastructure Technical Framework, Vol 1 (ITI TF-1) Integration Profiles," Revision 3.0, ACC/HIMSS/RSNA, 2006, p. 55.

(22) "Digital Imaging and Communications in Medicine (DICOM) Supplement 95: Audit Trail Messages," Trial Standard, 2004.

(23) Damianou, N., et al, "A Language for Specifying Security and Management Policies for Distributed Systems," *The Language Specification* , Version 2.3, Imperial College Research Report DoC 2000/1, 2000.

BIBLIOGRAPHY

(1) Kearney, et al, "An Overview of Web Services Security," *BT Technology Journal*, Vol 22, No 1, 2004, pp. 27-42.

(2) *Glossary of Key Information Security Terms*, Kissel, Ed., April 25, 2006.

ASTM International takes no position respecting the validity of any patent rights asserted in connection with any item mentioned in this standard. Users of this standard are expressly advised that determination of the validity of any such patent rights, and the risk of infringement of such rights, are entirely their own responsibility.

This standard is subject to revision at any time by the responsible technical committee and must be reviewed every five years and if not revised, either reapproved or withdrawn. Your comments are invited either for revision of this standard or for additional standards and should be addressed to ASTM International Headquarters. Your comments will receive careful consideration at a meeting of the responsible technical committee, which you may attend. If you feel that your comments have not received a fair hearing you should make your views known to the ASTM Committee on Standards, at the address shown below.

This standard is copyrighted by ASTM International, 100 Barr Harbor Drive, PO Box C700, West Conshohocken, PA 19428-2959, United States. Individual reprints (single or multiple copies) of this standard may be obtained by contacting ASTM at the above address or at 610-832-9585 (phone), 610-832-9555 (fax), or service@astm.org (e-mail); or through the ASTM website (www.astm.org). Permission rights to photocopy the standard may also be secured from the ASTM website (www.astm.org/COPYRIGHT/).