A Trust Model for Capability Delegation in Federated Policy Systems

Kevin Feeney
FAME and Knowledge & Data Engineering Group,
Intelligent Systems Lab,
Trinity College Dublin, Ireland
kevin.feeney@cs.tcd.ie

Simon N. Foley
FAME and Department of Computer Science,
University College Cork,
Cork, Ireland
s.foley@cs.ucc.ie

Rob Brennan
FAME and Knowledge & Data Engineering Group,
Intelligent Systems Lab,
Trinity College Dublin, Ireland
rob.brennan@cs.tcd.ie

***Abstract*—Federated policy systems are required to support the emergent complexity and organizational heterogeneity of modern Internet service delivery. This paper presents a distributed policy management approach which utilizes a flexible, tree-based capability authority model to partition and delegate federated capabilities or services. A trust management model and a delegation logic is defined which supports secure decentralized policy reasoning and addresses performance overheads due to distributed rule evaluation, threats from malformed or malicious federated principals and allows flexibility with respect to delegation chain reduction or capability authority re-partitioning. The system is evaluated through a security analysis and a prototype implementation of a federated policy engineering framework based on this logic is described. This framework is based on public key certificates and an extension to the Keynote Trust Management language. It provides practical management services such as key discovery and certificate revocation in addition to the core capability delegation function.**

Keywords: Service Management; Trust Management; Policy Based Management; Federated Management; Security;

I. INTRODUCTION

In today's Internet, service-oriented architectures and APIs enable, at a code-integration level, flexible mash-ups or service pipelines that create composite services from a number of autonomously provided components. The shift in emphasis from network connectivity to application or service layer integration has widened the developer pool, lowered the barriers to service provision and led to an explosive growth in the number of applications that draw on data and services from multiple domains. However, many organisations cannot justify the cost and risk of exposing their APIs, data or services to third parties on the Internet, despite the business benefits. This is due to the difficulty of securely managing flexible access to data or services in the open Internet.

There are a wide range of solutions (REST, SOAP, BPEL, XML-RPC, etc) which can be readily applied to facilitate syntactic integration between data and services residing in autonomous domains and Internet-accessible APIs based on such technologies are widely deployed. Although the existence of competing standards and diverging implementations means that syntactic integration of services is rarely straightforward. Today there are few purely technical barriers to the creation of complex, multi-domain Internet services.

While there may be some level of standardization at a syntactic level, there is little or none at a semantic level. Different APIs use different information models, messaging sequences, naming conventions, etc, and these can change over time. Thus, integrating Internet services provided by third parties into composite services remains a difficult and time-consuming task.

In order to facilitate the semantic interoperability of Internet data and services, the W3C has defined a number of standards. RDF and OWL are standards for machine processable, self-describing data and OWL-S and SAWSDL are standards that define service semantics. Once largely restricted to the research community, the recent success of pragmatic initiatives such as Linked Open Data has seen the proliferation of semantic data on the web. Semantic descriptions facilitate inter-domain data and service integration on the Internet by directly modeling resource semantics and enabling semi-automated integration to take place through ontology mapping processes, even when the providers reside in different domains and hence reference locally developed information models or schemata [1].

Although semantic technologies may come to facilitate the seamless integration of Internet services composed from across multiple domains [1], there is a lack of tool support or even access to appropriate information flows to effectively manage services in such multi-domain, composite contexts. The lack of federated mechanisms for managing security and access control remains a significant stumbling block that limits the flexibility and dynamism of service compositions. When interoperating services are managed in isolation from one another, maintaining consistent security policies requires a significant amount of customized integration and the maintenance of complex, distributed security sensitive state information that is complex, expensive and risky to manage.

Much of the research on cross-organisational management has focused on the specification of contracts and agreements between organisations, which then must be monitored and enforced by both parties, in particular focusing on service level agreements (SLAs)[2],[3],[4]. Traditionally, defining an SLA is a slow and tedious process

requiring complex legal agreement and protracted negotiation. Attempts to support federated (i.e. multi-organisational) management through policy languages and frameworks such as OrBAC [5] and X-federate [6] assume that the participating organisations will be closely aligned, adopt a common management architecture, policy language and information-models. These are thus of limited use in the flexible and dynamic world of Internet service provision. Virtual Organisation (VO) approaches [7] address this problem but the management of these VOs and the maintenance of the mappings between the VOs and their constituent organisations introduce significant overheads and are thus not well suited to lightweight, transient federations

Administrators need mechanisms which allow them to flexibly update their access control and security mechanisms to reflect the changing contexts and service compositions that their organisations participate in. In this environment, where service ownership is highly decentralized and providers compete with one another for customers, centralized management solutions are not viable. Organisations that make information services available on the Internet are accustomed to retaining ultimate control over them and are unlikely to give up their management autonomy by submitting to a centralized authority. Furthermore, given the heterogeneity of the services and data made available on the Internet, solutions that rely upon the widespread adoption of standard fine-grained information models are of limited utility.

Trust Management [8],[9],[10] is an approach to constructing and interpreting trust relationships between public keys that are used to mediate security critical actions. Cryptographic certificates are used to specify delegation of authority among public keys, and are used to determine whether a signed request complies with a local authorisation policy. Trust management provides a basis for secure decentralized policy based management. In using unforgeable cryptographic certificates to embed policy rules, authorization delegation does not require reference to a central authority, thus supporting truly distributed mechanisms that mediate based on decentralized policies.

This paper presents a trust model of secure delegation of capabilities in federated policy systems. This trust model forms the basis of for the implementation of the Federal Relationship Manager (FRM). The FRM is a policy engineering framework whose goal is to allow administrators to flexibly and securely control access to their Internet-visible services in contexts where these services may ultimately be consumed by third parties with whom the provider has no direct relationship. In the FRM schema capability delegation and sub-delegation is used to share fine-grained access rights to data and services. This allows the creation of complex federated services in a thoroughly decentralized manner, where each participating domain maintains control over their own resources, and can constrain how it is used without requiring any central control or mutually trusted third party.

Section 2 of this paper gives a high-level overview of the relationship and capability authority models which the FRM framework is based upon. Section 3 presents the trust management logic that underlies the secure distribution and efficient invocation of shared capabilities. Section 4 describes the FRM architecture and the proof of concept prototype implementation of the trust model described in this paper. It explains how it facilitates public key discovery and permission revocation and describes in brief the use of the keynote trust management language as a convenient way of wrapping up the capability relationships and policy delegations within credentials so that they can be safely distributed and reasoned over. Finally, section 5 provides conclusions.

II. FRM: Relationship & Capability Models & Delegation

In traditional operating systems, a capability consists of a reference to an object and a set of actions that can be performed with that object. In our schema, which views the Internet as the operating system, a capability is a reference, identified by a URL, to a node in a capability model of the underlying object or resource that is being made accessible as a service. The owners of resources share access with third parties by delegating capability URLs to them. These capability URLs allows the recipient to invoke the services described in the capability model which resides at that URL. Delegated capabilities also provide access to semantic models of the shared services and can be sub-divided into lesser capabilities and re-delegated. Capability delegation provides a flexible and expressive means of applying access control to capability sharing in relationships that cross management domains without requiring all parties to support common policy or information models.

Conceptually, an FRM instance consists of a *domain relationship model* and a *capability authority model*. Each FRM manages the relationships of a set of domains. Each domain can be related with other domains either managed locally or by a remote FRM. Domains are considered to be the principals of the security system in that they are the entities that own resources, are identified by public keys, and are allocated access privileges in the system. Domain relationships are established through bi-lateral agreement of domain administrators. The domain relationship model is a globally distributed model with portions of the domain map being managed autonomously.

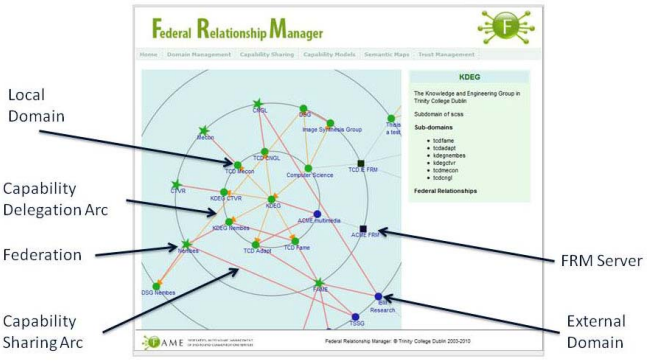

Figure 1. Annotated screenshot from FRM Implementation showing domain relationship map.

An FRM's capability authority model, defines the capabilities provided by its domains and how authority to invoke and manage these capabilities is distributed to domains through the *ownership* relation and capability delegation. A capability is owned by a local domain and domains can share capabilities with related domains through delegation of *capability authorities.* Capability authorities are references to nodes on a capability-authority tree, and this tree is implemented as a service which can be deployed by the owner of any resource that is to be shared. Capability authorities are higher-level constructs than permissions, which are the standard unit of most access control and policy management systems. They allow, for example, service providers to grant a capability covering all of their customers (or whatever subset they require) via a single delegation.

Figure 2 shows an example of a capability authority tree that might be used by provider of a communications service. The construction of the tree and the choice and positioning of the nodes is purely a function of how the owner of the underlying services wishes to make them available to third parties – this allows arbitrary bundles of authority to be created and distributed. The capability authority tree represents a model-centric way of incorporating policy rules into delegation without any need for a distinct rule language – a simple tree-based node comparison substitutes for policy evaluation.

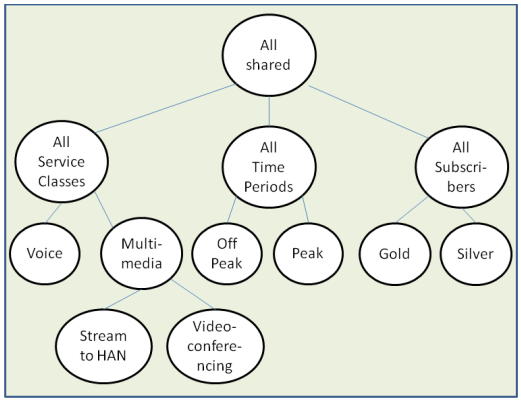

Figure 2. An example of a simple capability authority model. Nodes in the tree can be delegated to third parties, representing authority to invoke those capabilities: e.g. a delegation of [voice, off-peak, gold] allows voice services for Gold-level customers to be invoked in off-peak times.

The capability authority model is instantiated as an information service which compares two capability authorities and answers the question as to whether the first capability authority encapsulates the second according to the capability tree. This allows capability authorities to be constructed that represent arbitrary aggregations of specific permissions and distributed between domains representing collaborating organisations. Whenever a third-party wishes to invoke a capability of a partner, the partner only needs to determine if the capability being invoked is encapsulated by a capability authority that has been issued to that third party. Note that the third party does not need to understand the capability authority or the policies that it encapsulates, just how to invoke the capabilities that it implies authority for, which helps to minimise the requirement for common information models.

The only requirement for common technical infrastructure, beyond the core FRM functions, is that each party must provide a capability authority model of the capabilities that they wish to make available to third parties and, when they wish to invoke third-party capabilities, they must present a capability authority that they have been delegated by the organization which owns the underlying resource. If the owner's capability model indicates that the delegated capability authority *implies authority for* the capability being invoked, the invocation is allowed.

Capability authorities and lesser sub-capabilities can be delegated repeatedly across multiple domains. The FRM can operate in trusted or non-trusted mode. In the non-trusted mode, each FRM maintains a centralized list of which local capabilities have been delegated to which remotely managed domains. The FRM schema dictates that, in this mode, the delegation chain must be traversed in order to verify each individual delegation. In large complex systems such as multi-provider Internet applications, where there may be significant latency constraints on policy evaluations of invocations, the requirement to re-traverse the delegation chain in order to verify every invocation of a shared capability can become a significant performance bottleneck.

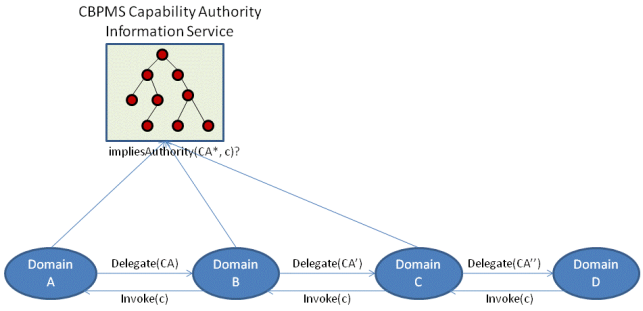

Figure 3. Capability authority delegation in the FRM. Capability authorities are delegated between domains and the FRM CAIS service ensures that delegated capability authorities are implied by possessed capability authorities and that invoked capabilities are implied by possessed capability authorities by reference to the capability's authority tree.

In order to circumvent this practical limitation, a trust management model of capability delegation has been developed, a model that ensures the security and confidentiality of shared capability authorities, but also allows capabilities to be directly invoked on the owner by any party that has been delegated a capability authority in a secure way. The following section describes that trust model.

III. TRUST MODEL OF CAPABILITY DELEGATION

Applying trust management to the FRM approach of distributed capability authorities overcomes the limitations of the non-trusted FRM schema identified in the last section and in addition allows individual capability authorities to be safely split and re-combined without the participation of the original CA delegator. This loose coupling is highly

desirable in a distributed, federated system where local decompositions of CAs are naturally more amenable for integration with local policy sets.

A. Capability Authorities

A policy S managed by a Capability Authority (CA) defines a tree-based partial ordering $\leq_S$ over a set of permissions denoted αS with tree root element denoted Ψ_S. The permission ordering $p \leq_S q$ has the usual interpretation that a principal authorized for permission q is also authorized for permission p, and that $p \leq_S \Psi_S$ for all permissions $p : \alpha S$. Since S defines a tree, then there exists a lowest upper bound operator $\oplus_S$ that provides permission conjuction.

A policy S is used by a CA information service to answer authorization queries of the form $p \leq_S q$ for $p, q : \alpha S$, and the alphabet αS is the set of permissions defined by the policy S. The policy managed by the CA is not static and can be extended by the addition of new permissions and ordering relationships. An extension is considered to be *safe* if authorizations (orderings) permitted by the new policy are consistent with the original policy.

Safe CA Policy Extension. The policy S of a capability authority may be safely extended to policy S' (denoted *extends(S,S')*) if the new policy S' does not contradict the orderings of the original policy. Define:

$$extends(S,S') \equiv \alpha S \subseteq \alpha S' \wedge \Psi_S = \Psi_{S'} \wedge \forall p,q : \alpha S \bullet p \leq_{S'} q \Leftrightarrow p \leq_S q$$

Safe CA policy extension ensures that that authorizations permitted by S are permitted by S'.

It can also be desirable to replace an existing policy S by a more restrictive policy S' in the sense that authorizations permitted in S may not necessarily be permitted in the replacement policy S'. This replacement policy is safe in the sense that using the replacement will not result in an access that would violate the original policy.

Safe CA Policy Replacement. The CA policy S may be safely replaced by policy S' (denoted *replaces(S,S')*) if S' enforces the permission ordering restrictions of S, that is,

$$replaces(S,S') \equiv \alpha S \subseteq \alpha S' \wedge \Psi_S = \Psi_{S'} \wedge \forall p,q : \alpha S \bullet p \leq_{S'} q \quad p \leq_S q$$

Proposition 1. It follows from their definition that any safe extension of a policy is also a safe replacement of the policy.

A CA policy R is considered to be a simple sub-policy of a policy S if it is a sub-tree contained within S. In the next section we consider how a CA managing S can delegate authority for managing the sub-tree R of S to another CA. In practice, the recipient may in turn decide to extend and/or replace the sub-tree R. So long as R (or its safe replacements) can be regarded to be a sub-tree of some safe extension of S, then S can be considered to retain authority over any (safe) extension/replacement of R. Thus we define the following ordering.

CA Partial Authority Ordering. The CA policy R has partial authority on (a portion of) policy S, (denoted $R \trianglelefteq S$) where,

$$R \trianglelefteq S \equiv \exists S' \bullet (extends(S,S') \wedge \Psi_R \in \alpha S' \wedge (\forall p,q : \alpha R \bullet p \leq_R q \Rightarrow p \leq_{S'} q))$$

and we have the following simplification

$$R \trianglelefteq S \Leftrightarrow (\forall p,q : \alpha R \cap \alpha S \bullet p \leq_R q \Rightarrow p \leq_S q)$$

Proposition 2. The safe replacement by R' of a policy R that has partial authority on policy S has partial authority on S:

$$R \trianglelefteq S \wedge replaces(R,R') \Rightarrow R' \trianglelefteq S$$

Proposition 3. It follows from its definition that the partial authority ordering is a partial order.

The partial authority ordering is intended to capture a semantics for safe policy delegation: a CA managing policy S can delegate authority for policy R to another CA if $R \trianglelefteq S$ holds. However, having delegated R, a subsequent and safe extension of the policy S to S' may conflict with policy R, that is, $R \trianglelefteq S'$ does not hold. For example, given $p,q \notin \alpha S$ and $p,q \in \alpha R$ with $(p \leq_R q)$, extending S such that $p,q \in \alpha S'$ and $\neg(p \leq_S q)$ implies that $(p \leq_R q \quad p \leq_{S'} q)$ does not hold for $p,q \in \alpha R \cap \alpha S'$ given the definition of $R \trianglelefteq S'$. If the extension of S is limited to permissions that are *not* members of the alphabet of R, then it follows that $R \trianglelefteq S'$ holds. This result can be generalized to include safe replacements of R, as follows.

Proposition 4. Partial authority ordering is preserved by safe policy extension on the condition that the extensions to the policies are non-intersecting. Given policies R and S then:

$$(R \trianglelefteq S \wedge replaces(R,R') \wedge extends(S,S') \wedge (\alpha R \cap \alpha R') = (\alpha S \cap \alpha S')) \Rightarrow R' \trianglelefteq S'$$

Capability Authority Information Service. A Capability Authority with policy S provides an information service by answering queries about permissions and policies.

B. Trusted CA Information Services

Each CA information service A owns a public-private key pair (K_A, K_A^{-1}). The public key K_A is assumed to uniquely identify the CA and any message M signed by the private key (denoted $\{M\}_{sK_A}$) is considered to have originated from K_A. Let $pol(K_A)$ define the policy that is managed by the CA with public key K_A.

A CA information service with public key K_A may delegate partial authority over its policy S to another service K_B by signing a SPKI-style [9] cryptographic certificate $\{K_B, S, D, V\}_{sK_A}$. D denotes the delegation bit (0/1: whether the recipient may further delegate the authority), and V is the validity-period (start/finish) specification for the certificate.

For ease of exposition, we do not consider the delegation bit nor the validity-period, and represent delegation as $K_A \;\; {}_S \; K_B$ which is interpreted to mean that the CA information service K_A trusts the CA information service K_B to mediate queries related to the policy *S*.

Certificate Reduction Rules. A CA delegation certificate defines delegation of policy (and orderings) rather than permission. However, since the policy ordering $R \trianglelefteq S$ defines a partial order between the delegation attributes then the usual inference rules follow for policy delegation: given certificate $\{K_B, S, D, V\}_{sK_A}$ and $R \trianglelefteq S$ then there is an implicit delegation $K_A \;\; {}_R \; K_B$, that is, we have the inference rules:

$$\frac{\{K_A, S, D, V\}_{sK_A}}{K_A \;\; {}_S \; K_B} \qquad \frac{K_A \Rightarrow_S K_B ; R \trianglelefteq S}{K_A \Rightarrow_R K_B}$$

If delegation is regarded as transitive and if K_A delegates policy *S* to K_B and K_B delegates this authority to K_C then it follows that K_A implicitly delegates authority for *S* to K_C. In general,

$$\frac{K_A \Rightarrow_S K_B ; K_B \Rightarrow_R K_C}{K_A \Rightarrow_R K_C} [R \trianglelefteq S]$$

Note the inference condition $R \trianglelefteq S$: K_B may delegate any policy R for which it holds partial authority from K_A. This is more restrictive than the usual SPKI reduction rule which does not have this condition, but infers delegation as the greatest lower bound (intersection) of attributes *R* and *S*. A FRM policy is currently implemented as tree-based partial ordering. Generalizing these policies to a lattice-based structure (with a greatest lower bound operation) would provide the basis for supporting the more general reduction rule and is a topic for future research.

These reduction rules are easily generalized to include the delegation bit and validity-period from the certificate.

Permissions. When defining a policy, a CA must ensure that no ambiguity can exist between the permissions it defines and permissions defined by other CAs. Any ambiguity can lead to subterfuge [11] whereby a principal misinterprets the authority associated with a permission identifier. In our current model a CA signs each permission it generates. On the basis that the public keys of CAs can be assumed to be unique, permission signing avoids subterfuge by providing globally unique permission identifiers and prevents permission forgery. Given a permission *p*, then let *orig(p)* denote the (public key of the) originator/creator of the permission and we have $p \in \alpha(pol(orig(p)))$. Permissions are signed by their originating CA in order to prevent forgery.

A CA must have *authority* over a permission if it is to answer a permission ordering query. By default, a CA with public key K_A has authority over any permission *p* that it originates, that is, given $orig(p) = K_A$ then $auth(K_A, p)$ holds. A policy delegation implicitly delegates authority on permissions (in the delegated policy). In particular, if a CA K_A has authority on permission $p \in \alpha R$ and K_A delegates a policy *R* to K_B, then K_B will have authority on *p*. This is captured by the following reduction rule.

$$\frac{p \in \alpha R; auth(K_A\, p); K_A \;\; {}_R \; K_B}{auth(K_B, p)}$$

Note that, for ease of exposition, we assume that this (implicit) delegation of authority is transitive.

C. Trusted Policy Reasoning

A CA information service (CAIS) K_A can rely on its local policy to answer ordering queries regarding any permissions *p* and *q* for which it holds authority, that is,

$$K_A.\text{impliesPermission}(p, q)$$
$$\equiv auth(K_A\, p) \wedge auth(K_A, p) \wedge p \leq_{pol(K_A)} q$$

A principal K_A can trust the answer from (signed by) CAIS K_B, concerning permission query $p \leq_S q$ if the inference $K_A \;\; {}_S \; K_B$ can be made over the available certificates and K_B has authority over *p* and *q*.

A CA K_A may answer queries about policy orderings so long as it can be considered to have authority over the policies *T,R* whose ordering is queried, that is,

$$K_A.\text{impliesAuthority}(T, R)$$
$$\equiv T \trianglelefteq pol(K_A) \wedge R \trianglelefteq pol(K_A) \wedge R \trianglelefteq T$$

If the CA does not have authority over both the policies (or permissions) in a query then it cannot answer the query (locally), in which case the query must be delegated to another CA using the existing FRM distributed query evaluation protocol [12].A series of examples are presented that illustrate how policy reasoning is done in practice.

Example 1. Suppose that the principal K_S has complete trust in a master CAIS K_M that manages a policy *M*. Principal K_A writes a policy *S* whereby $\alpha S = \{\Psi_M\}$ and delegates authority for this policy to K_M as $K_S \;\; {}_S \; K_M$; K_S effectively places all its trust in K_M.

The policy *M* managed by K_M may be extended and revised over time, and all these changes can be considered to be safe extensions of the policy *S* of K_S (Proposition 4). At some point K_S queries K_M.impliesPermission(*p,q*), and can trust the answer since, K_S can infer $K_S \;\; {}_M \; K_M$ given $M \trianglelefteq S$ and $K_S \;\; {}_S \; K_M$.

Example 2. Suppose that, in addition to extending and/or replacing its policy, K_M delegates partial policies to other CAIS's. K_S can trust the response from the query K_B.impliesPermission(*p,q*) if K_R can present a certificate chain authorizing its trust from K_M (and ultimately, from K_A). For example, if K_M had delegated authority for $R = pol(K_R)$ to K_B such that $R \trianglelefteq M \trianglelefteq S$ then we have the chain $K_S \;\; {}_S \; K_M \;\; {}_R \; K_R$ and thus $K_S \;\; {}_R \; K_R$.

Example 3. Suppose that K_S wishes to constrain how K_M can revise its policy M in as far as it impacts K_S. Rather than simply delegating the entire (safely) replaceable policy S ($\alpha S = \{\Psi_M\}$), K_S can specify any policy S. Again, the delegation chain $K_S \;\; {}_S K_M \;\; {}_R K_R$ is reducible to $K_S \;\; {}_R K_R$ only if $R \trianglelefteq M \trianglelefteq S$, which may not be the case if K_M changes its policy M in an unsafe manner (with respect to S).

Example 4. Consider again Example 2. Suppose that K_S wishes to make query K_R.impliesPermission(p,q) to the CAIS K_R that manages policy R. In this case K_R may not be an authority on the particular query (for example, when $p,q \notin \alpha R$) and K_R may wish to pass the query back to K_M, which has authority and responds to K_R who in turn responds to K_S. However, this answer from K_R is not trusted (by K_S) for the answer, in general, since $\neg M \trianglelefteq R$.

This is addressed by K_M either signing its answer (which is then passed on to K_S by K_R) or granting temporary authority over the specific query to K_R. This latter scenario is done by K_M signing a once-off delegation certificate $\{K_R, S \cup T, N_S\}_{sK_M}$ where policy T, with $\alpha T = \{p,q\}$, captures the ordering $p \leq_M q$ and N_S is a nonce generated by K_S that is submitted as part of the request (to ensure freshness of response). In this case, the certificate chain $K_S \;\; {}_S K_M \;\; {}_T K_R$ reduces to $K_S \;\; {}_T K_R$ with $\alpha T = \{p,q\}$.

Example 5. Continuing Example 2, K_S can accept authority for mediating any policy U (under the jurisdiction of K_M) if it can infer $K_S \;\; {}_U K_S$. For example, suppose that K_R delegates authority for the policy $U \trianglelefteq R$ to K_S. In this case, by verifying $K_S \;\; {}_S K_M \;\; {}_R K_R \;\; {}_U K_S$, K_S can confirm $K_S \;\; {}_U K_S$. Note that, a principal can also declare jurisdiction over its own policy via the self-signed delegation $K_S \;\; {}_U K_S$

Example 6. Continuing Example 2, given the delegation $K_M \;\; {}_R K_R$ ($R \trianglelefteq M$), suppose that K_R safely extends its policy to *R'*. Recall that in creating new permissions K_R is unable to forge permissions from M and, therefore, cannot introduce permission orderings in *R'* that would violate M, that is, by Proposition 4, $R' \trianglelefteq M$ holds. Having modified its policy, K_R delegates authority to itself as $K_R \;\; {}_{R'} K_R$. The answer to K_B.impliesPermission(p,q) regarding permissions $p,q \in \alpha R'$ can be trusted by K_S on the basis of the policy delegation chain $K_S \;\; {}_S K_M \;\; {}_R K_R \;\; {}_{R'} K_R$.

D. Security Analysis

The FRM trust model supports secure delegation of policy decision-making between CA information services. Authentication and secure communication notwithstanding, the primary threat in this environment is a service that attempts to masquerade as a valid CAIS, answering queries related to policies over which it holds no authority. This rogue CAIS may be an attacker that is external to the FRM infrastructure or a legitimate CAIS that does not properly implement the FRM service. The latter may be a result of compromise of the server hosting the CAIS or of a legitimate but dishonest CAIS.

Principal K_S making query K_R.impliesPermission(p,q) receives an answer signed by some CAIS K_R and a certificate chain that can be used to prove that $K_S \;\; {}_R K_R$. Assuming that an appropriate public key cipher has been used to implement the signatures then the answer and cryptographic certificates cannot be forged by a rogue CAIS that does not know the private signing key.

The potential damage by a legitimate CAIS that has been compromised is limited to the policy/permissions for which the legitimate CAIS holds authority as the attacker cannot forge certificates that indicate otherwise. The delegation reduction rules imply that any modifications to the CAIS policy that are not safe mean that the existing certificates held by the compromised CAIS can no longer prove policy authority. A principle of least privilege should be followed in order to minimize the threat of a compromised CAIS: only delegate authority for that part of a policy that is needed. CA information services that manage a large policy space should be hosted on hardened systems to minimize the likelihood of compromise.

A CAIS cannot be tricked into accepting authority for mediating decisions related to a policy S that is not under the jurisdiction of a trusted CA information service K_M. CAIS K_S will not accept a policy S from rogue CAIS K_R (with policy R) unless it can prove $K_M \;\; {}_R K_R \;\; {}_S K_S$, that is, $K_M \;\; {}_S K_S$.

In our current framework permission identifiers are signed by the CAIS originating/claiming jurisdiction over the policy. This ensures global uniqueness of permission identifiers and helps to prevent delegation subterfuge attacks [11]. Globally unique permissions, along with Proposition 4 ensure that a rogue CAIS K_R cannot replace/extend its policy in a manner that could be used to contradict orderings defined by another CAIS K_S. Given $K_S \;\; {}_R K_R$, a rogue K_R cannot extend its policy with a forged permission p from $pol(K_S)$ since it cannot prove authority over the permission (*auth(*K_R*,p)*) in any query. Kr can only prove authority for new permissions it creates in *R'* or for permissions in $\alpha S \cap \alpha R'$ for which it has been delegated authority. Thus, by Proposition 4 we have $R' \trianglelefteq pol(K_S)$, a safe extension with respect to the policy S.

The proposed model of trust is monotonic in the sense that once a policy statement is trusted it cannot be contradicted *within the logic*. For example, having signed and distributed delegation certificate $K_S \;\; {}_R K_R$, any subsequent unsafe change that K_S might make to its policy (contradicting R) can be ignored by K_R simply presenting the original certificate for policy R. In conventional trust

management systems this is typically handled through revocation, whereby an issued certificate, no longer considered valid, is revoked. Revocation is typically managed by issuing only short-lived certificates [15] or by relying on third parties providing Online Certificate Status Protocol (OCSP) or Certificate Revocation Lists (CRLs) services. The FRM framework supports both of these approaches, but also supports direct, synchronous revocation of certificates through an inter-FRM protocol, as described in the following section.

IV. FRM Policy Engineering Framework Architecture & Prototype Implementation

A prototype implementation of the FRM framework has been produced which includes a proof of concept implementation of the trust model described above. The implementation incorporates interfaces for model management and support for semantic mapping and other useful tools for facilitating cross-domain service integration. They are, however, beyond the scope of this paper - this discussion will touch only upon those aspects of the implementation that are directly relevant to the trust model.

The trust model uses X.509 certificates to identify principals. Capability delegations are distributed as signed certificates containing policies. The trust framework manages the generation, discovery, distribution and validation of the keys and messages that are passed between FRMs in the background – administrators concentrate on the 'business level policies' - forming relationships and exchanging capabilities between domains, the trust management framework translates these high-level relationships into suitably secure communications.

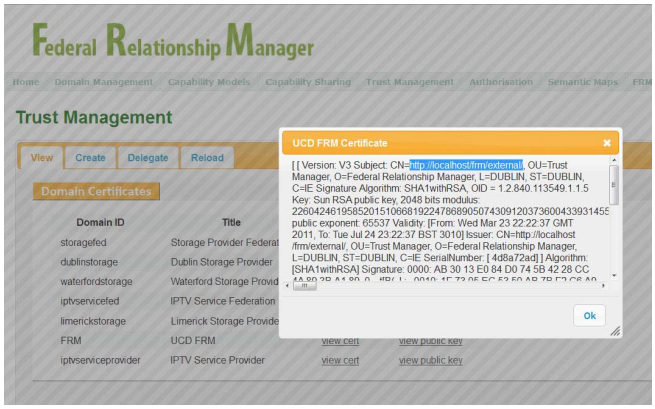

Figure 4. Screenshot of FRM Trust Management interface, with self-signed Certificate identifying FRM service highlighted.

Each FRM serves as a certificate authority and uses a self-signed X509 certificate to bind its identity to its public key. It also maintains a public and private key pair for each of the domains that it manages and signs X.509 with its self-signed certificate. Each FRM server provides an inter-FRM messaging API to support relationships and delegations between remotely managed domains. At the trust-management level, domain relationships amount to an exchange of certificates between domains, while the delegation of a capability amounts to the delegating domain sending a signed policy certificate to the recipient.

The inter-FRM API has been implemented as a RESTful HTTP interface. An FRM's identity, as indicated in the CN field of its X.500 name, is the URL at which its API can be accessed. The FRM framework requires that FRM servers must federate by exchanging certificates before inter-domain communication is permitted between them. This allows FRM administrators to define policies to govern which remote FRMs can be federated with and allows them to put in place off-line procedures to verify the provenance of the self-signed certificates. Once an FRM federation is completed and keys have been exchanged, these keys are be used to securely transmit inter-FRM messages between the peer FRM instances.

Once a trust relationship exists between peer FRMs, they can communicate securely over HTTPS with one another. The public keys contained in the certificates that they exchange are used to cryptographically secure subsequent inter-FRM communication. Domain administrators can create trust relationships between domains they control and any remote domain managed by an FRM which has a trust relationship with the local FRM. Agreement of trust relationships between domains involves the exchange of the X.509 certificates which identify the respective domains. As these certificates are signed by the FRM that manages them, and as a federal relationship between the peer FRMs is a pre-requisite to the formation of an inter-domain relationship, there is an existing trust relationship upon which the inter-domain trust relationships are based. Thus, there is no requirement for any trusted third party to manage public keys, sign certificates, define protocols or provide addressing or discovery.

Capability authority delegation between related domains is achieved through the transmission of signed policy certificates between the FRMs that manage the domains. As a proof of concept, (extended) KeyNote [8] certificates are used to encode a delegation relation $R \leq S$. This is supported by the introduction of a new *policy* datatype for KeyNote attributes whereby an FRM capability authority policy can be serialized as a value of type policy and be delgated using a KeyNote certificate. A new KeyNote condition relational comparison operator is provided for these policy values, whereby given policy values `R` and `S` then `R]=S` implements $R \leq S$. Thus, for example, if `"base64encoding:…."` is a serialization of the policy in Figure 2, then the KeyNote credential:

```
Authorizer: domainA
Licensee: domainB
Conditions:
     _POLICY ]= "base64encoding:…";
Signature: …
```

is a delegation of this policy by domainA to domainB. A KeyNote compliance check tests whether a requester (an FRM domain) can be trusted when they invoke a policy specified by `_POLICY`.

Whenever a domain administrator wishes to invoke a capability that is controlled by a federated FRM, it presents all of its potentially relevant certificates to that FRM, which translates the invocation into a capability authority URL (according to its internal model) and performs a capability comparison using its CAIS. This allows capability delegation certificates to be safely reasoned over. A desirable side-effect of this approach is that delegated policies can be specified so that some state information is evaluated locally while some is evaluated by the FRM that has issued the certificate (encoded in the capability authority).

The FRM API provides an interface which translates capability authority URLs into semantic descriptions of the capabilities (expressed as RDF triples). This facilitates delegation chains where domain administrators can sub-divide received capabilities into sub-capabilities and re-delegate them to related domains. The final notable feature of the FRM policy engineering architecture is its support for synchronous revocation of delegated policies and any policies derived from these policies and re-delegated. Each delegation certificate contains within it a reference to the FRM which controls that capability which allows direct revocation via the provider.

V. CONCLUSIONS AND FUTURE WORK

Trust Management systems typically focus on delegation of authority in terms of permissions, whereby there is effectively a global permission ordering defined in terms of permissions that have a global interpretation. The FRM framework extends this, by permitting federated policies of permissions and their orderings to be locally defined and delegated. This paper describes a Trust Management-based logic that can be used to manage the trust relationships resulting from federated policy delegation. While the logic is described in the context of FRM its principles are applicable to federated policy systems in general. Such federated relationships are supported today without the use of an automated system and are instead based on extensive human-centric commercial contacts, legal contracts and integration based on trial and error. It is hoped that through the deployment of truly distributed, trusted, federated policy-based systems more agile business models will be supported and costs dramatically reduced.

Systems that coordinate and enforce policy across distributed and heterogeneous coalitions can benefit from a federated policy approach [6]. Rather than relying on a centralized definition of policy, individual coalitions can create, manage and delegate their own policies. We have demonstrated that the current proposal is both sound and addresses significant challenges in the problem space. Future research will investigate how the approach proposed in this paper can be applied to dynamic coalition forming frameworks such as [13].

Capability Authorities are generated and issued by the CA that controls access to the underlying service represented by the capability. Thus CAs can restrict sub-delegations to approved FRMs and domains and can maintain a record of delegations of locally controlled capabilities – however the implementation of this mechanism remains to be completed.

Another important area for future study arising from the current work is the issue of quantifying the trade-off between distributed validation of CAs and the cryptographic processing and reasoning overheads of the trusted CA approach outlined here. It is likely that these engineering trade-offs will to a certain extent be application specific (e.g. depending on policy rule complexity, service latency requirements, etc) and details of the deployment environment for the distributed, federated system (e.g. CAIS node processing load, communications network delays). However it might be possible to develop analytic or empirical rules that would give guidance for when or how best to apply activities like certificate chain reduction.

ACKNOWLEDGMENT

This research is supported by the Science Foundation Ireland (Grant 08/SRC/I1403) as part of the Federated, Autonomic End to End Communications Services Strategic Research Cluster (www.fame.ie).

REFERENCES

[1] N. Noy "Semantic Integration: A Survey of Ontology-Based Approaches". Special Issue on Semantic Integration, SIGMOD Record, Volume 33, Issue 4, December 2004, pp. 65-70

[2] P. Bhoj, S. Singhal and S. Chutani “SLA management in federated environments Export”, Computer Networks, Vol. 35, No. 1. (January 2001), pp. 5-24.

[3] T. Nurmela and L. Kutvonen, “Service level agreement management in federated virtual organizations.” In Distributed Applications and Interoperable Systems, volume 4531 of Lecture Notes in Computer Science, pages 62-75, Paphos, Cyprus, June 2007. Springer

[4] B.S. Firozabadi, M. Sergot, A. Squicciarini, E. Bertino, "A Framework for Contractual Resource Sharing in Coalitions," policy, pp.117, Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'04),

[5] S. El-Kalam et al. “Organization based access control”. In POLICY '03: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks, Washington, DC, USA, 2003

[6] R. Bhatti, E. Bertino A. Ghafoor, "X-FEDERATE: a policy engineering framework for federated access management" Software Engineering, IEEE Transactions on. Volume 32, Issue 5, May 2006 Page(s):330 - 346 DOI: 10.1109/TSE.2006.49

[7] L.M. Camarinha-Matos, H. Afsarmanesh, M. Ollus, (Eds.), Virtual Organizations, Systems and Practices,Springer, 2005.

[8] M Blaze, J Feigenbaum, et al. “The role of Trust Management in Distributed System Security”, LNCS 1603, pages 185-210, Springer Verlag, 1999.

[9] C. Ellison, B. Frantz, B. Lampson, R.L. Rivest, B. Thomas, and R. Ylonen. “SPKI certificate theory”, IETF RFC 2693, September 1999.

[10] N. Li, J. C. Mitchell. RT: “A role-based trust-management framework”. in Proc. of the Third DARPA Information Survivability Conference and Exposition, DISCEX III, Washington, D.C., April 2003, pages 201-212

[11] K. Feeney, D. Lewis, D.O’Sullivan, “Service Oriented Policy Management for Web-Application Frameworks”, IEEE Internet Computing Magazine, Nov/Dec 2009, vol 13, issue 6, pp. 39-47

[12] S.N. Foley, H. Zhou: “Authorisation Subterfuge by Delegation in Decentralised Networks”, in Proc. of the 13th International Security Protocols Workshop, Cambridge, UK, April, 2005

[13] H. Zhou, H and S.N. Foley, “A Framework for Establishing Decentralized Secure Coalitions”, IEEE Computer Security Foundations Workshop, 2006.