Policy and Trust in Open Multi-Operator Networks

Pekka Nikander, Lasse Metso
Helsinki University of Technology

Abstract: In the future telecommunications network, more and more services are based on open protocols and architectures. In such an environment, there is a clear need for controlling the access of users and other operators to the network services. If the network is based on internetworked facilities, traditional address based access control may not be sufficient due to the possibility of address spoofing attacks. Thus, the usage of strong cryptography is often the only possibility for providing authenticity and integrity. However, in such a setting both key management and trust management become challenging problems.

In this paper we present an architecture, called TeSSA, for managing service level access control and other security aspects in open telecom networks, and our experiences with an early prototype version of the architecture. The architecture is based on digitally signed certificates and explicitly presented permissions. The architecture itself is policy free, thereby allowing any reasonable security policy to be applied. Furthermore, the architecture fully supports both centralized and decentralized administrative models, thereby being especially suitable for internets where the networks of several operators are interconnected. In addition to security policies, the architecture can be extended to represent any types of policies. Using the TeSSA facilities, the operators do not need fully trust each other, but can explicitly vary the level and structure of trust.

Keywords: Security Management, Access Control, Security Policy, Decentralized Authorization

1. INTRODUCTION

Managing large networks is a complex and challenging task. Through a series of development steps, quite a lot of effort in the current management systems is circulating around the concept of *policy.* However, it is by far no clear what different authors exactly mean with policy; in this paper, we start with the IETF Policy Working Group definition [1]. Thus, according to this definition, a policy is a collection of rules, consisting of conditions and actions, that are used as a basis of configuring various network elements and controlling of their functionality. However, managing a collection of inter–

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35522-1_37

H. R. van As (ed.), *Telecommunication Network Intelligence*

connected, more or less independent networks, is even more complex than managing a single network. In this case, managing and co-ordinating individual policies does not suffice, but the agreements and relationships between the inter-operating companies should also be handled. This effect becomes especially evident when considering security policies, where the individual protection needs and the desire to co-operate often conflict. In this paper, we extend the IETF policy definition with these security aspects.

Whenever two organizations agree to co-operate, the common understanding of the terms are recorded in an agreement. Based on the agreement and the backing legislation, the organizations can now expect each other to act according the terms. On the other hand, the actual nature of the common operations usually mirror the actual *trust* between the managers and employees of the organizations. Thus, the organizational behaviour in such a setting is characterized by both the terms of the agreement and the actual personal and social level trust relationships. From this point of view, even the terms of the agreement can be considered as a measure or expression of trust, as their fulfilment usually requires assumptions, or beliefs about the other party. Such assumptions or beliefs may be considered to involve trust, and all these trust aspects must be considered when defining policy.

Now, when we transfer this organization level relationship to the cyberspace, a number of difficulties arises. Based on the common agreement, the individual networks of the organizations should be configured in such a manner that they remain secure while controlled co-operation becomes possible. For example, one of the organizations may provide bandwidth to the other within a specified Service Level Agreement (SLA). Typically, such a configuration is handled through policy definitions.

In this paper, we present a framework for such policy definitions. The framework provides facilities for automated handling of the policy definitions, thereby making it possible to use policy-based network management in inter-organizational networks, e.g. in extranets or multi-operator networks. Furthermore, we consider the implicit security and trust requirements involved in the usage of such policies, and consider how services may be controlled between more than two organizations. The basic method for achieving this goal is to explicitly express the trust assumptions and stated agreements involved in the operations.

1.1 Trust

All human operation involves trust. Most of this trust is so inherent to the social nature of us human beings that we seldom think about it; consequently, inability to trust is considered to be abnormal. Usually, trust involves social or legal control to some degree. If the social control fails, or the legal system collapses, our basic security is fractured. The purpose of the social and legal system is to provide a solid foundation for trust. In a computerized system, as opposed to a physical system, trust relationships should be represented explicitly, and preferably in a way that their legal binding can be later non-repudiably proven in a court, if needed. However, in this paper we do not directly cover non-repudiation; instead, our framework is based on certificates and the assumption that the certificate's validity can be proven on need.

Thus, in the architecture that we later describe, *trust* in a service is a belief that the service, when asked to perform an action, will act according to a predefined description. In particular, this belief implies the belief that the service, or the operator behind, will not attempt to harm the requester independently of the way it fulfils the request. Thus, trust is always expressed in relation to a service provider *and* to an action assumed to be performed. Furthermore, trust is not necessarily transitive. Trusting someone for recommendation is different from trusting someone for direct action.

This definition limits the concept of trust within the distributed system itself. It does not, however, limit us from discussing other aspects of trust when needed. For example, some people consider trust only as inherently human behaviour. On the other hand, it is quite natural to speak about trust relationships between services, and therefore we keep speaking about trust.

Earlier theoretical studies of trust in a distributed setting have been conducted by Raphael Yahalom et al. [2] and later, independently, by Audun Jøsang [3]. In these studies, the goal has been to develop a calculus for trust. That is, the aim has been to create a generic calculus that shows how new trust relationships may be based on existing trust relationships and recommendations. In our work, we are more interested in what are the necessary trust relationships when applying security policies in the usage of services in open distributed networks. Thus, instead of presenting a metric, language, or calculus of trust, we present a software framework that allows trust to be expressed in a machine readable form. Indeed, we believe that the types and levels of trust are application specific, and should not be limited by the management framework. [4]

1.2 Organization of this paper

The rest of this paper is organized as follows. In Sect. 2. we describe the basic security functions in open distributed networks. Next, in Sect. 3. we describe the basics of policy based management. Sect. 4. describes the *Telecommunications Software Security Architecture*, or TeSSA, the top of which our present work is based on. The next section, Sect. 5., shows how the ideas of the TeSSA architecture can be applied to inter-organizational policy-based network management. Finally, Sect. 6. contains our conclusions of this work.

2. SECURITY AND ACCESS CONTROL IN OPEN NETWORKS

While confidentiality, integrity and availability are usually defined as the primary goals of any security system, *authentication, access control* and *authorization* are the usual means used to achieve those goals. In this section, we describe the traditional view to authentication and access control, and why the underlying implicit assumptions behind these terms may not be the right ones for open distributed systems.

2.1 Authentication

In most literature the term *authentication* is used as a synonym for *identity authentication.* However, a thorough analysis reveals that there is a number of fundamental difficulties in this definition. First, the concept of *identity* is very problematic when considering a large, multi-organizational distributed system, and second, limiting authentication to be used in connection with identity reveals to be too restricting.

Let us consider the problem with identity first. The term *identity,* stemming from Latin *identidem,* originally means sameness or oneness. For example, when we meet a previously unknown person for the first time, we cannot really *identify* that person, since there is nothing to identify with. This problems gets really evident in large networks, where we meet people without knowing anything about them, not even their face.

Taking a slightly different point of view, it has been argued that in a distributed digital system the only real “identity”, with which anything can be later related to, is a private cryptographic key [4]. That is, when we for the first time meet someone in the digital world, we may be able to learn a public cryptographic key that corresponds to the private key possessed by our new acquaintance. Later on, we can really *identify* a future communicating peer with a known one by being able to convince us that the new peer possesses the same private key as the old one. Thus, we may say that only cryptographic keys should be considered suitable items to function as indications of identity.

Now, it becomes evident that the concept of identity authentication, as usually understood, is at least too limited if not outright misleading. Therefore, it is better to enlarge the term authentication to denote the act of proving the authenticity of any object or piece of information [5] instead of restricting it to denote the act of proving the authenticity of, e.g., the identity of a communicating peer or message originator, as it has been traditionally understood in the literature. Thus, we may speak about authentication of authorization information, or authentication of the possession of a cryptographic key.

2.1.1 Authentication protocols

An *authentication protocol* is a communication protocol whose purpose is to enhance the collection of evidence available to the communicating parties so that one or more of them can *believe* that a given piece of information is *authentic*. That is, it is a common practice to use cryptographic means in establishing new evidence. Typical belief goals include the belief that a (symmetric) cryptographic key is held by a communication peer, the belief that a cryptographic key or other random number has been generated during the protocol run, and the belief that the communication peer holds either of the former beliefs.

2.2 Access Control

Access control includes the means and methods with which the users and other active entities, such as processes and threads, are limited in their ability to manipulate objects within a computer system. In a communications system, it is usually used to denote facilities that limit the possibilities for opening connections or receiving messages. However, even in a communication system the term can be extended to control access to services in a more specific way. The purpose of an access control system is to maintain confidentiality, integrity and availability by making it impossible (or impractically hard) for unauthorized parties to read, modify or consume information or resources.

Capabilities are one possibility to represent access control information. A *capability* is a security token associated with a subject that lists a number of permissions. Each permission defines one or more objects, and an action or a set of actions that the subject may perform on the object(s). In this study, our interest is in explicitly *signed capabilities*, sometimes also called *credentials*, which are capabilities that are cryptographically bound to a specific subject. In the system to be presented, these signed capabilities are represented as *authorization certificates*. Thus, we define that a *signed capability*, or *authorization certificate*, is a digitally signed piece of information that assigns a subject, usually represented in the form of a cryptographic public key, one or more *permissions*, which allow the subject to perform specified actions on one or more specified objects in a target system.

What is probably interesting in this definition is the inclusion of a *target system*. By including this, we want to emphasize the local nature of capabilities in a distributed system. That is, a single capability should be valid only at a specific single system, the target system, or possibly at a (small) number of interrelated systems, e.g. a clustered server, which as a group can be considered to form a single target system.

2.3 Authorization and Delegation

Authorization, in general, denotes sanctioning or empowering someone, i.e., to make it valid, legal, binding, or official for a person to perform certain actions in the future. Delegation, on the other hand, denotes appointing someone to act as a representative, e.g., by means of a legal proxy.

In the context of a (distributed) computer system, both authorization and delegation can be basically defined as acts that change access control rights. In most current systems, local operating system usually assigns a number of rights to a process when the process creates a new object. More generally, any principal can create objects on the behalf of other objects, but this also means that the creating principal will be responsible for controlling the access to the created object [6]. This is important since it allows us to understand the fundamental issues that the concept of a *certificate loop* (see Sect. 5.1) is based on.

Thus, by definition, when a computer or communications node is installed, the node operating system is given implicit access permissions to all the physical and logical objects that are stored in the node. When a software based service is created, on the other hand, it is only given access to the objects that are parts of that service. In fact, these are the only implicit access rights; all other rights are delegated. Thus, we can say that a principal that has a permission to control another principal or access an object may *delegate*, on its will, this permission to a third principal, unless explicitly prohibited [6]. When delegating, some permissions assigned to the delegating principal are copied to the delegate.

Examples of existing distributed authorization and delegation systems include the Digital Systems Security Architecture (DSSA) [7], Kerberos [8], our TeSSA [9] (see Sect. 4). All of these are pretty similar in many respects. However, there are a number of differences as well. The most important differences between TeSSA and the other two can be summarized as follows.

- First, our system explicitly models more types of trust than either DSSA or Kerberos; especially, we model types that are not related to access control but to generic security conditions that must be met.
- The DSSA architecture is based on names and local access control lists, while our architecture uses signed credentials and thereby allows anonymous operation.
- The Kerberos architecture is based on symmetric cryptography and centralized key distribution centres. Our architecture is based on public key cryptography and is fully decentralized.
- Our architecture explicitly contains the concept of policy, and make it possible to represent various kinds of policies in addition to security policies.

Other existing prototype authorization systems that have had influence on our system include the PolicyMaker [10] and the SDSI/SPKI proposal [4][11][12]. Our system is based on the same (but independently developed) ideas. [6]

3. POLICY BASED MANAGEMENT

Policy-Based Networking (PBN) is gaining a wider acceptance in the management of open, IP based network, mainly due to it makes possible more unified control and management in complex IP networks [13]. In this section, we describe the related concepts and terminology, as defined by the IETF [1]. Later, in Sect. 5., we briefly describe how these definitions must be extended in order to take care of the various security aspects involved when applying policies in multi-organizational setting.

3.1 Policies, Rules, Conditions, and Actions

A *policy* is the combination of *rules* and *services*, where rules define the criteria for resource access and usage. Each policy rule is comprised of a set of *conditions* and a corresponding set of *actions*. The conditions define when the policy rule is applicable. Once a policy rule is so activated, one or more actions contained by that policy rule may then be executed.

Policies can contain other policies; this allows one to build complex policies from a set of simpler policies, so they are easier to manage. The inclusion possibility also enables the reuse previously built policy blocks.

3.1.1 Policy rules

Policy rules can be classified by their purpose; the following classification has been proposed by Moore et al. [14].

- *Service policies* describe services available in the network. For example, QoS classes are made by using service policies.
- *Usage policies* describe how to allocate the services defined by service policies. Usage policies control the selection and configuration of entities based on specific usage data.
- *Security policies* identify client, permit or deny access to resources, select and apply appropriate authentication mechanisms, and perform accounting and audit of resources.
- *Motivational policies* describe how a policy's goal is accomplished. For example the scheduling of file backup based on activity of writing onto disk is a kind of motivational policies.
- *Configuration policies* define the default setup of a managed entity, e.g., the setup of a network forwarding service or a network-hosted print queue.
- *Installation policies* define what can be put on the system, as well as the configuration of the mechanisms that perform the installation.
- *Error and event policies,* for example, ask the user to call the system administrator, if a device fails between 8am and 5pm.

3.1.2 Policy conditions and actions

A policy condition consist of two parts, which are policy condition *type* and policy condition *element.* A policy condition type is a set of predefined conditions, which all network vendors can implement. They can be attached to a policy rule for evaluation. A policy condition element is a policy condition type instance that is being evaluated. Policy condition elements are related together to form a Boolean expression.

A policy action is the changing of the configuration of one or more network elements in order to achieve a desired state; the state provides one or more behaviours. Like a policy condition, a policy action is comprised of two parts, a policy action type and a policy action element.

3.2 Policy Decision, Behaviour and State

A policy *decision* is the abstraction of activating and evaluating one or more policy rules. Each policy rule is interpreted in the context of a specific request for accessing and using resources. It connotes taking pre-determined actions based on whether or not a given set of policy conditions were satisfied.

A policy *behaviour* controls how traffic is treated, what network resources must be utilized, and what network services are provided. A policy *mechanism* is a set of vendor-specific commands that configures a network element to put a policy rule into effect. Policy behaviours define one or more mechanisms that are used to implement the policy. Therefore, different devices can carry out the same policy using different behaviours.

A policy *state* is a description of the settings of one or more network elements. These settings correspond to providing the set of services to be provided by the network. For example, a Service Level Agreement (SLA) can describe services contracted for by subscribers, this corresponds to a state that various network elements must be put into in order to provide those services.

3.3 Policy Functions

Policy is comprised of three functions: *decision-making (evaluation), enforcement* and *monitoring.* Policy *evaluation* is the determination of whether or not the network is in a desired policy state. This is usually determined by processing static or dynamic data against policy rules, the key point being that the decision is made by comparing definitional data stored in the policy repository with current data from the network that does not have to be stored in the policy repository. If it is found that the network elements are not in the desired policy state, then one or more policy actions will be taken to move the network elements from their current state to the desired state. This is called policy enforcement.

Policy *enforcement* is the action of placing the network in a desired state using a set of management commands. When this definition is applied to network elements, these management commands change the configuration of the devices using one or more mechanisms. Enforcement is carried out in the context of a policy rule.

Policy *monitoring* is an on-going active or passive examination of the network and its constituent devices for checking network health, whether policies are being satisfied, and whether clients are taking unfair advantage of network services. This is done for following reasons: to ensure that clients are receiving the services that they have contracted for, to monitor network statistics as part of checking network health, to monitor network statistics as part of checking whether policies that are currently in use are being satisfied, or to ensure that clients of a given set of policies are not abusing their privileges.

3.4 Security Policies

In strict contrast to the formal policy model presented above, security policy is *usually* defined to be the set of more or less informal rules that specify how the integrity, confidentiality and availability of the stored information and available resources are to be protected within an information processing system. The rules often involve manual operations and management conventions that are expected to be followed by humans.

Our point of view in this work, however, is somewhat different from the usual one. As the focus is on how to make such distributed system secure that span several organizations, our interest lies on how to express a security policy in a form that allows network nodes and services to enforce it. Furthermore, we want to apply security policies to all forms of trust involved in a distributed system, not just the access control type of type of trust usually considered. Based on this, we now attempt to define security policy, trying to collect the most important aspects together.

A (formal) *security policy* is a collection of policy rules (as defined above) that describe how a network node shall modify its behaviour when considering statements made by trusted third parties (TTPs), recommendations given by the TTPs, and statements considering claimed authority. The actions defined in a policy rule may involve access control, usage of cryptographic keys, or some other aspect of total system security. The purpose of the policy is to minimize the risk of losing the integrity, confidentiality, or availability of the protected information and resources.

In an open distributed system, the ability to publish a policy, and to base actions on the published policy of other principals, may prove to be an important feature. For example, it would allow software applications that work for a user to make policy decisions even when the user is off-line. [6]

4. THE TELECOMMUNICATIONS SOFTWARE SECURITY ARCHITECTURE

In this Section, we describe the *Telecommunication Software Security Architecture* (TeSSA) that we have developed in our research group. First, a conceptual view is presented, followed by architectural, functional, and implementation level views.

4.1 Conceptual overview

The basic layered structure of the TeSSA architecture, on the conceptual level, is shown in Fig. 1. The top component is the *trust and policy management infrastructure,* sometimes also called the public key infrastructure (PKI). It allows the users and administrators to explicitly define, for example, which network nodes are trusted for which operations, how users are allowed to access resources, and what kind of protection is required from the connections between different applications. In practice, the data present in the management infrastructure is represented in the form of certificates. The certificates are stored in a distributed *certificate repository,* which allows convenient storage and easy access to the certificates.

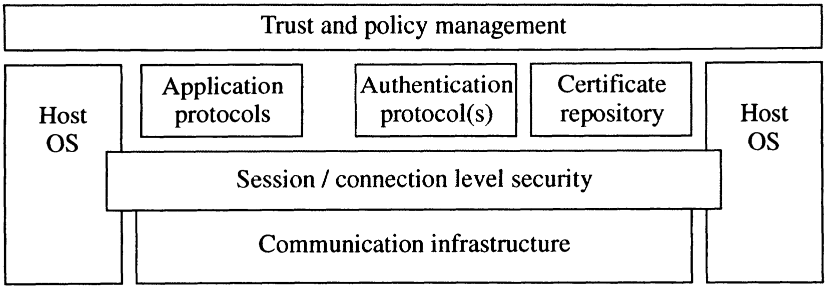
Trust and policy management
Host OS
Application protocols
Authentication protocol(s)
Certificate repository
Host OS
Session / connection level security
Communication infrastructure

Fig. 1. The conceptual building blocks of the TeSSA security architecture

The certificates are used, among other things, to control and facilitate the operation of the authentication protocols. That is, the authentication protocols need the public keys of nodes and applications, along with semantic information about what the keys are good for. This information is available from the trust and policy management infrastructure.

An *authentication protocol* is used to create, modify and delete dynamic security associations and contexts between any two principals. The PKI layer above provides the initial security contexts on the base of which new contexts may be created. In practice, the protocol allows creation of session level keying material combined with authentication of the public key of the peer, the negotiation parameters, and the keying material itself.

Finally, the session or connection level security components take care of protecting the user data in transit. They allow user data to be protected from eavesdropping, modification, replay, and other active and passive attacks. Moreover, it is important to notice that each secure connection has a policy associated with it. That is, each secure connection is annotated with knowledge about the permissions of the peer object. This allows the local access control system to determine when a remotely initiated operation is authorized and when not.

4.2 Definition of architectural concepts

To give a more precise meaning to the various components of the conceptual architecture, the following definitions are given.

- A *security context* is a collection of security related variables, such as asymmetric or symmetric keys, policy rules and credentials, shared by two or more parties. When creating a new security context, authenticity and integrity of the information must be ensured. Typically, also some or all of the information is confidential as well. Technically, new security contexts may only be created on existing trusted security contexts. In the architecture, the trust and policy management infrastructure provides the initial security contexts, using which new contexts may be created. The initial security contexts are represented in the form of certificates. Creation of such contexts is based on management decisions external to the presented architecture.
- The *session / connection level security components* take care of the actual (cryptographic) protection of user data in transit. Typically, the layer consists of a number of cryptographic protocols and associated policy management functionality. The keying material and policy rules pertaining to a particular secure session are defined in a corresponding security context.
- The *authentication protocol* component of the architecture denotes that or those implementation level protocols that are capable of *creating, modifying and deleting* new security contexts, based on existing security contexts. Thus, the main function of the authentication protocol is the establishment and management of security contexts. A notable issue here is that, in addition to the key authentication and negotiation, the protocol takes care of managing the associated policy information as well.
- The *certificate repository* is a distributed database that facilitates on-line storing and retrieving of certificates. Ideally, the repository should be efficient, fault tolerant and transparent to applications.

4.3 Functional concepts

Changing the point of view from the layered one to a more functional one, the architecture can be depicted as in Fig. 2. First, there is a *reference monitor*, which is a functional concept within the local operating system that takes care of protecting local resources. Whenever a piece of software (e.g. a client program) attempts to access a protected resource, including services provided by other applications, the request is routed through the reference monitor. The reference monitor decides, using the local policy rules and the permissions the application has, whether the request is authorized or not. An unauthorized request is aborted.

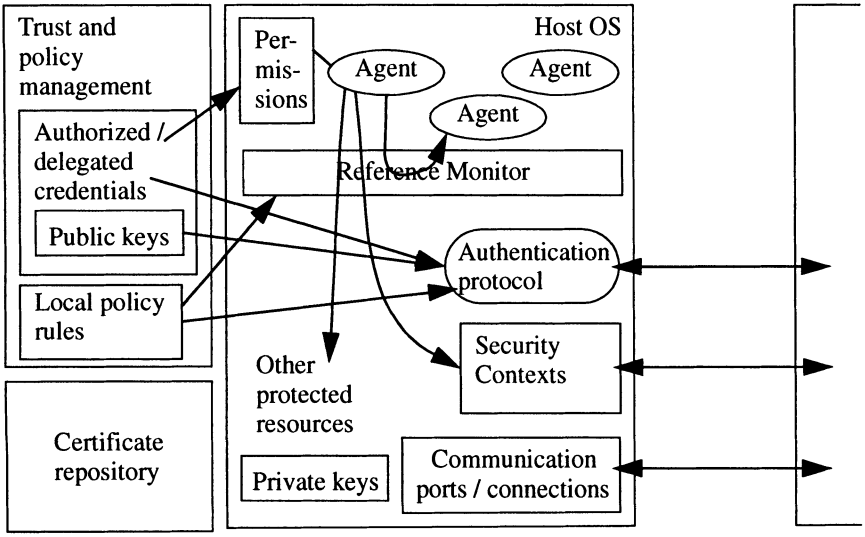

Fig. 2. A functional view to the TeSSA architecture

The protected resources include, among other things, *security contexts* and communication ports or connections. In order to have secure connections, and to enforce local security policy with respect to opening of connections and to having quality of protection on connections, both of these resources must be locally protected. When a software module wants to have a secure connection with a remote server, a security association and a connection (or a number of connections) that use the association must be established.

When a new security context is needed, an *authentication protocol* is used to negotiate one. The authentication protocol gets any needed public keys, local policy rules and credentials from the trust and policy management infrastructure. These allow it to perform the negotiation towards its peer, and to establish the local policy conditions that apply to the newly created association.

Thus, the *trust and policy management infrastructure*, or PKI, can be seen to provide control and management information to the various functions, including creation of security context, access control decisions made by the reference monitor, and control of quality of protection on connections. The PKI manages these resources by representing credentials of various parties.

A *credential* is a signed statement of authority, claiming that a principal has been authorized (possibly via delegation) to access a protected resource. In our architecture, the credentials are represented in the form of SPKI certificates. When a credential is accepted by a local execution environment, it is usually transformed into a local representation denoting permission to access some concrete resource. Credentials are almost always acquired through delegation. This means that a final credential does not alone qualify for creating permissions (or directly authorizing operations). Instead, a chain of authorization certificates is typically needed. Acceptance of certificate chains is controlled by the local policy rules.

A *local policy rule* is a locally understood instruction to the reference monitor that allows it to make decisions. Specifically, local policy rules may exclude credentials created by certain principals, limit the lengths of certificate chains, specify that stronger than default credentials are needed for certain operations, etc.

The local policy rules, similarly to the credentials, are represented as certificates in our architecture. This has a number of benefits. First, since they are signed, their management can be distributed. Only rules having a locally meaningful structure and a trusted issuer will be adhered to. Second, they need not necessarily be stored locally, but can be retrieved from the certificate repository only when needed.

4.4 Implementation

In addition to the conceptual model, we have also built a partial prototype of the architecture in our project. Taking the actual implementation components, the layered structure of the architecture is shown in Fig. 3. As one can see, the trust and policy management infrastructure is implemented using the IETF Simple Public Key Infrastructure (SPKI). Furthermore, the authentication protocol(s) are based on the Internet Security Association and Key Management Protocol (ISAKMP) infrastructure, the certificate repository is implemented within the Domain Name System (DNS), and connection security is taken care of with the IPSEC protocols.

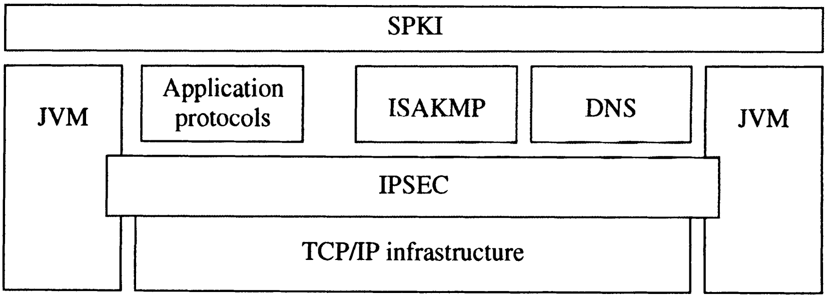

Fig. 3. Implementation architecture

All of the implementation is based on the TCP/IP protocol suite and the Java Virtual Machine version 1.2. The TCP/IP infrastructure provides ubiquitous communications and a multitude of application protocols. The JVM 1.2 provides an Object-Oriented operating environment with fine grained access control. The details of the implementation are beyond the scope of this paper, and described elsewhere, see e.g. [6][9][15][16][17][18].

5. APPLYING POLICIES BETWEEN ORGANIZATIONS

A crucial issue in Policy-Based Networking is the question of how to apply policies in a network with several operators that may have conflicting goals. If each of the operators just use PBN to manage their own network, much of the benefits of the policies are lost. That is, the operators would need to carefully co-ordinate their policies, and this co-ordination would be manual.

In this section, we show how the ideas of TeSSA based distributed security management can be applied to generic policy based networking in an inter-organizational setting.

5.1 Trust requirements at a node level

To start, let us consider a simple situation of two network nodes, one providing services and the other using the services provided by the first one. We assume that the nodes are managed by different organizations, and that the organizations do not fully trust each other. Furthermore, we don't assume anything about the security of the interconnecting link (other than that it transfers packets more or less reliably); there may be malevolent parties intercepting the link. In this situation, both nodes have a number of trust requirements.

- First, the node requesting services must determine that the target node is trusted to provide the services. That is, the requesting node must consider the target node known to run the requested services, and trustworthy enough to provide the services for the purposes of the requesting. This definition includes the idea that some nodes may be considered trustworthy enough only for some services.
- Second, the target node must be able to determine that the requesting node is indeed allowed to request the services if asks for. This includes, among other things, the permission that the new service may use resources such as network connections.

Both of these trust requirements may be represented with certificates [18]. In the first case, when the requesting node been started, it has been configured to trust a number of authoritative parties for determining trustworthy server

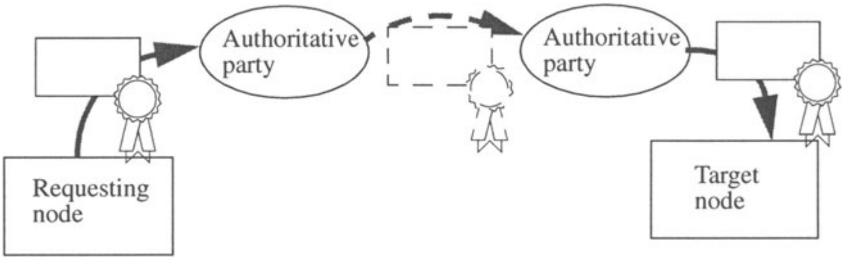

Fig. 4. Chain of certificates from the requesting node to the target node

nodes. One or more of these authoritative parties, on their behalf, have then certified the trustworthiness of the target node. This is illustrated in Fig. 4.

Similarly, there must exists a chain of certificates from the target node, through the security administrator of the node, to the requesting node. This chain must prove that the node is authorized to request the service, and thereby use the resource of the target node. This chain is illustrated in Fig. 5.

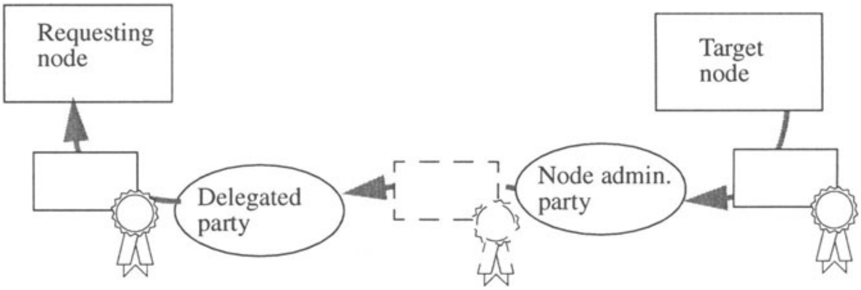

Fig. 5. Chain of certificates from the target node to the requesting party

When the requesting node contacts the target node, an authentication protocol is run. During the course of the protocol, the target node proves that it has its own private key in its possession. Similarly, the requesting node proves the possession of its key. These authenticating demonstrations may be considered to function as "virtual certificates" between the requesting node and the target node. In a way, these "virtual certificates" delegate back to the verifier the authority that the authenticating party was received through the chain of actual certificates. Using the local policy rules, the verifier can now determine if the peer actually does have the claimed authority by checking all of the certificates and seeing if the certificates really imply the claimed authority. These checks close the certificate chain into a loop, as illustrated in Fig. 6. [18]

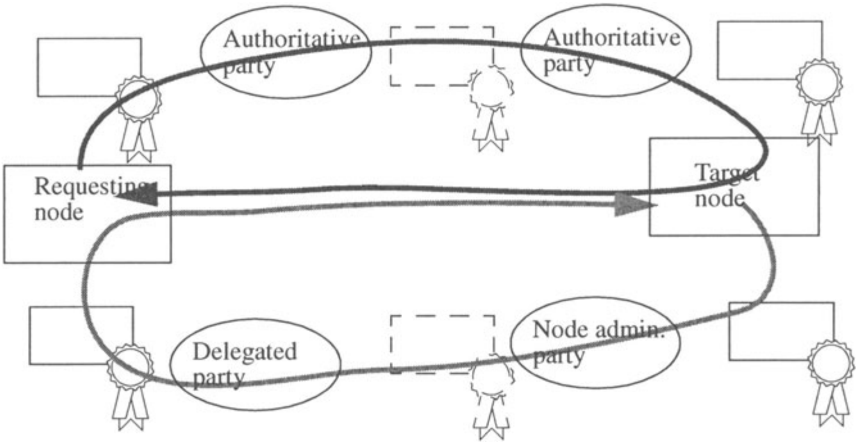

Fig. 6. Certificate chains are closed into loops by an authentication protocol

5.2 Organization level security policies

In Fig. 6, the authoritative parties, the node administrative party, and the delegated party are examples of Trusted Third Parties (TTP). Thus, architecturally, they belong to the trust and policy management layer of the architecture (see Fig. 1). Since these parties typically belong to different organizations, they, in fact, do represent organization level security policies.

So, on the simple example of Sect. 5.1., the *node administrative party* actually controls the behaviour of the target node in respect to who it allows to access the service. Accordingly, it *represents* the security policy regarding the target node as a set of certificates that control the access to the target node. More specifically, that particular certificate (or set of certificates) may apply simultaneously to several servers, thereby actually representing *organizational* level policy instead of a node specific policy.

In the same way, if we consider the trust mediated by the *authoritative parties*, the certificates provided by the parties control which servers the requesting node may use and which it must not use. Thus, they represent policies that control the security restrictions on the behaviour of the requesting node. Again, as the authoritative parties may be hosted by several organizations, they represent the security policies of their hosting organizations.

Possible conflicts in the expressed policies of the various companies are resolved in two distinct levels. First, at the certificate level, policies are represented in the certificates as sets of permissions. These sets may be easily intersected. Second, the actual node resolving the certificate chains may use additional local policy rules in order to resolve possible conflicts. We have discussed both of these aspects in more detail in [6].

5.3 Extending security policy representation to other kinds of policies

In the SPKI certificate standard [4][11], the so called tag field of the SPKI certificates are used to represent *sets of permissions.* However, the SPKI architecture allows the tag field to have local semantics that depend on the actual node interpreting the tag. Thus, it is possible to specify different kinds of semantics for the tag.

One approach is to use program fragments to represent policies, as suggested in [10]. However, that approach has the drawback that there must be an interpreter for the programming language. Additionally, such an approach is in direct opposition with the IETF policy definition, where the goal is to define policies in a descriptive way instead of using imperative program fragments.

Thus, we propose an approach where the policy rules are included in the tag field of the certificates. Furthermore, if the certificates are represented in XML, as suggested in [19], the policies can also easily be represented in XML, e.g., according to the approach suggested in [20]. This is very much in line with one of the original motivations for XML, namely the ability to use XML to represent configuration information. For example, the forthcoming Apple Mac OS X will use XML internally to represent configuration information.

Thus, the usage of XML allows any policies to be represented instead of security policies, and the usage of certificates allows the usage of the policies to cross organizational boundaries in a secure way. The certificates, and especially certificate chains, model the trust relationships between the organizations, and allow both mutual agreements and other trust conditions to be represented in the digital format.

6. CONCLUSIONS

In the field of Policy-Based Networking, the ongoing research has largely focused on modelling various concrete policies and applying them within a single administrative domain, or at most between domains that fully trust each other. On the other hand, various kinds of authorization certificate approaches have provided a model of how to express access control information in a fully distributed setting involving several organizations. In this paper, we have shown how mixing these two approached together with an XML based representation format can be used to represent all kinds of policies in a secure way. The certificate based security allows these policy expressions to be enforced even in a situation where there are several co-operating organizations that do not fully trust each other.

REFERENCES

1 John Strassner and Ed Ellesson, *Terminology for describing policy and services,* work in progress, IETF Internet draft, 1999, `http://www.ietf.org/internet-drafts/draft-ietf-policy-terms-00.txt`

2 Raphael Yahalom, Birgit Klein, and Thomas Beth, "Trust Relationships in Secure Systems - A Distributed Authentication Perspective", In *Proceedings of the IEEE Conference on Research in Security and Privacy,* 1993.

3 Audun Jøsang, A Model for Trust in Security Systems, in *Proceedings of the Second Nordic Workshop on Secure Computer Systems*, 1997.

4 Carl Ellison, *SPKI Certificate Theory,* RFC2693, September 1999.

5 Pekka Nikander, *Modelling of Cryptographic Protocols -- A Concurrency Perspective,* Licentiates Thesis, Helsinki University of Technology, December 1997

6 Pekka Nikander, *An Architecture for Authorization and Delegation in Distributed Object-Oriented Agent Systems*, Helsinki University of Technology, Doctoral Dissertation, 1999.

7 M. Gasser, A. Goldstein, C. Kaufman, and B. Lampson, "The Digital Distributed System Security Architecture," In *Proceedings of 1989 National Computer Security Conference.*

8 J. Kohl and C. Neuman, *The Kerberos Network Authentication Service (V5),* RFC1510, Internet Engineering Task Force, 1993.

9 Sanna Liimatainen et al., *Telecommunications Software Security Architecture*, Helsinki University of Technology, `http://www.tcm.hut.fi/Research/TeSSA`.

10 M. Blaze, J. Feigenbaum, and J. Lacy, *Decentralized Trust Management,* In Proceedings of the 1996 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, May 1996.

11 Carl Ellison, *SPKI Requirements*, RFC2692, September 1999.

12 Ronald Rivest and Butler Lampson, "SDSI - A Simple Distributed Security Infrastructure", Proceedings of the *1996 Usenix Security Symposium,* 1996.

13 David C. Blight, Takeo Hamada, Policy-Based Networking Architecture for QoS Interworking in IP management. *Proceedings of Integrated network management VI, Distributed Management for the Networked Millennium 1999, IM 98,* IEEE 1999, pp 811-826.

14 B. Moore, E. Ellesson, and J. Strassner, *Policy Framework Core Information Model,* work in progress, IETF Internet draft, 1999, `http://www.ietf.org/internet-drafts/draft-ietf-policy-core-info-model-01.txt`

15 Jonna Partanen, *Using SPKI certificates for access control in Java 1.2,* Master's Thesis, Helsinki University of Technology, August 1998.

16 Sanna Suoranta, *An Object-Oriented Implementation of an Authentication Protocol,* Master's Thesis, Helsinki University of Technology, December 1998.

17 Tero Hasu, *Storage and retrieval of SPKI certificates using the DNS*, Master's Thesis, Helsinki University of Technology, April 1999.

18 Ilari Lehti, Pekka Nikander, *Certifying Trust*, Practice and Theory in Public Key Cryptography (PKC'98), Yokohama, Japan, January 1998.

19 Juha Pääjärvi, "XML Encoding of SPKI Certificates", work in progress, Internet draft `draft-paajarvi-xml-spki-cert-00.txt`, March 2000.

20 Manfred Hauswirth, Clemens Kerer, and Roman Kurmanowytsch, *A flexible and extensible security framework for Java code,* Technical Report TUV-1841-99-14, Technical University of Vienna, March 2000.