Security as a Dimension of Quality of Service in Active Service Environments

Cynthia Irvine, Tim Levin, Evie Spyropoulou and Bruce Allen
Center for INFOSEC Studies and Research
Naval Postgraduate School
[irvine | levin | eviespy | ballen] @cs.nps.navy.mil

1. Introduction

The introduction of metacomputing and distributed resource management mechanisms to the Internet and World Wide Web will make available to users and applications a diverse set of previously unavailable network and computing resources. Middleware resource management systems (RMSs) will use geographically distributed, heterogeneous resources to support applications with a wide range of computation needs [1][2][3][4].

The RMS in such an environment is responsible for: efficiently scheduling multiple simultaneous tasks onto specific network resources; supporting user requirements for performance and security; and providing support for tasks to adapt to changing resource availability. This is accomplished by balancing costs for various services against their benefits, where the benefits can be to individual users or may be associated with the system as a whole, e.g. total throughput.

The notion of security variability has been discussed before. A Quality of Protection parameter is provided in the GSS-API specification [5]. This parameter is intended to manage the level of protection provided to a message communication stream by an underlying security mechanism (or service). Another early reference to a variable security service is that of Schneck and Schwan [6], which discusses variable packet authentication rates with respect to the management of system performance.

In previous work we have discussed fundamental Quality of Security Service (QoSS) concepts [7]. In [8] we defined a preliminary security service taxonomy defining the range of security services a RMS may need to manage. We addressed the problem of how users and administrators can understand and easily interact with the wide range of security services and mechanisms, by providing methods for translation of a simplified user abstraction of security to detailed underlying mechanisms [9].

We present our Quality of Security Service (QoSS) concepts in terms of variant security mechanisms and dynamic security policies. We also describe briefly our QoSS costing framework and demonstration, which illustrate how costs associated with network security services can be calculated and supplied to a RMS. Finally we talk about our experiments on linking QoSS conditions to an underlying security mechanism, such as IPsec. Our aim is to demonstrate an approach through which security could be treated as a QoS dimension.

2. Overview of Quality of Security Service

Relative to traditional QoS attributes (e.g. jitter, deadline, latency) security has been handled rather statically and indirectly. We have developed a theory of Quality of Security Service and a related security-costing framework that supports extension of QoSS functionality to embrace existing and emerging security technologies [10] [11]. Our goals have been to leverage existing security mechanisms to improve system availability, predictability, and efficiency, while maintaining, if not increasing the security of the distributed system.

Variability in *user and application* security requirements allows the underlying control system to be more adaptable in responding to requests for resources, and variability in *system and resource* security requirements allows the distributed system, e.g., through quality of service (QoS) middleware, to offer security choices to users or applications. The availability of user security choices along with support for management of security resources in response to user requests enables quality of security service (QoSS). We have found that many existing mechanisms and policies allow for security variance. For example MAC and DAC allow for complete solutions via their "dominance" and set inclusion relationships, and many so-called *fixed* requirements can be seen to actually define only minimums, allowing for a range of solutions. Some examples of security service attributes that provide ranges are the choice of cryptographic algorithm, number of rounds or key length, assurance level or strength of boundary control in a remote environment, or even the

capability level of the environment's security administrators.

QoS can be seen as the modulation of resources to deliver requested services to users, which depends on the control and variability of resources. Similarly, QoSS involves the modulation of *security* resources, and depends on the control and variability of those security resources. In a typical distributed system, the security restrictions and requirements confronted by a user emanate from many layers, components and services. How can QoS or resource management middleware make sense out of this apparent chaos in attempting to manage the system efficiently? Our approach involves several abstractions: the first is to view all security restrictions as service attributes. The second is to view all security restrictions as a range that defines a set of partially ordered possibilities, where some values are "more secure" than others. Of course, in some cases the range is degenerate, meaning the related service can be used in only one way.

To understand how these ranges can be used in a layered distributed architecture, consider how a request for execution of a task is passed between different layers, and security services are provided in response to these requests. As this *task sequence* is processed, there are both choices and limits regarding each security restriction or requirement. A *choice* is the security range request passed to the next layer. A *limit* defines which requests from previous layers are acceptable. In the end, if the task is realized by the system, meaning that the various choices and limits have been successfully resolved, the user's expectations for quality of security service will have been met. Additionally, the QoS middleware will have had additional latitude, by way of variant security requirements, in fulfilling user and system-wide goals, thereby potentially increasing the availability, predictability, and efficiency of the system.

With choices and limits applied for various security attributes at different layers in the task sequence, the question arises as to how these requirements should relate to each other. The following relationships between requirement ranges appear to be necessary for the coherent enforcement of security in a layered distributed system:

1. The maximum of each limit and choice range dominates the minimum of that range.

2. At each layer, a requirement choice range must be within the corresponding limit range.
   - This restriction reflects the protocol that a given layer will respect its own limits.

3. Each choice range must be within the previous choice range in the sequence.
   - This reflects a natural protocol to respect the choice of the previous layer, without which, the QoSS requested by the previous layers (ultimately, the user) will fail. This also indicates that security choices will constrict as the task proceeds.

4. Each choice range must be within the next limit in the sequence.
   - This restriction means that requests that are out of bounds will be rejected.

5. The limit ranges of each provider in a task sequence must all intersect.
   - This is a consequence of items 2, 3, and 4. Obviously, if two ranges in a task invocation sequence don't intersect, there does not exist a value that could satisfy both ranges; this would disallow a task from execution.

These relationships are illustrated in Figure 1. Because the choices and limits are partially ordered and consequently comparable, it is possible for a security service selection algorithm to be encoded. In the following discussion, automated mechanisms are discussed for managing security-*level* choices and network-*mode* limits.

3. Managing Costs and Variability of Security Service

If a particular security mechanism is "fixed" (i.e., always applied) then the overhead for the mechanism is part of the normal cost of running the task and the normal costing mechanism used by the QoS control mechanism will suffice. For variant security mechanisms, however, the security overhead will vary, depending on the security vector of the user's QoS request and any subsequent refinement of the user's choices due to the application or RMS. Thus, the middleware must have access to detailed information about the resource costs for each variant security mechanism.

We are working on a QoSS costing demonstration. In this approach we use a model for tasks, that incorporates the ideas of variant security services and value ranges for the security attributes. We additionally take into account an operational mode parameter, because network status could influence the security policy and security services applicable to the task: under certain conditions, a user or administrator may be willing to accept more (or less) security for a given application.

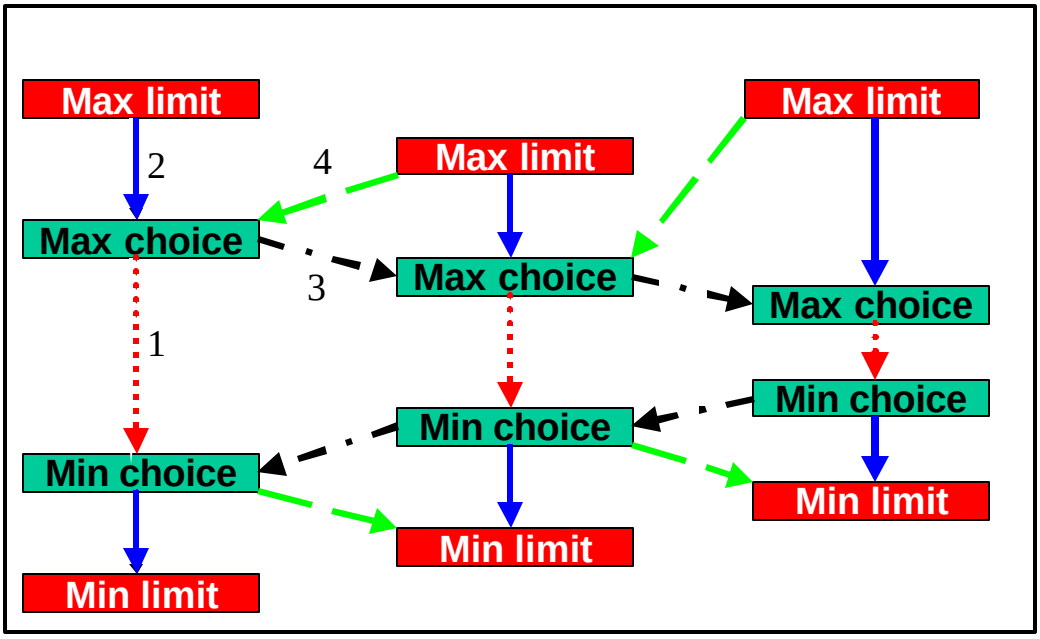

Figure 1: Requirement Range Relationships

For example, during an emergency, a military commander might decide to forgo certain security protocols in order to get some important information transmitted quickly. If such dynamic policies are created and analyzed before deployment of the computer network, the network can respond to changing environments, by having access to a predefined set of alternate security policies. In each of these cases, the effects of changes to the security mechanisms would be predefined and limited to meet the desired alternate security policy. We refer to three example modes:

- normal, which corresponds to the typical operating conditions of the system
- impacted, which represents a situation with a large amount of simultaneous requests received by the system
- emergency, which can be translated to requirements for the strongest security available (or in another interpretation it could mean completion of requested tasks as quickly as possible disregarding security).

With the modes approach, the acceptable range of values for a security variable attribute depends on the network "mode". So the selections offered to the users and applications are within the limits of the mode.

Still, the security services and underlying mechanisms may present too many variables and choices for users or applications to manage without automated support. Instead of presenting to the user all combinations of security mechanisms and parameters for the variant services, we can offer a simplified abstraction of security, in the form of security level choices, like "high", "medium", "low". These selections are mapped to detailed mechanism invocations via a translation matrix. The security administrator or system security engineer would pre-select various specific mechanisms and settings that are assigned to the security variables for each of the choices offered to the user.

3.1 An example for network modes and security levels

Let's illustrate the notion of network modes and security levels with the following example:

Assume that we work on a system which can provide data confidentiality using any of the following encryption algorithms: DES, 3DES, AES (these algorithms can be considered to be ordered by strength, measured in terms of the work factor associated with a brute force attack). This means that there is variability for the security attribute "encryption algorithm" and the set of acceptable values for it ranges from no encryption to AES. These are the limits imposed by our system's capabilities.

If we dynamically change the policy we apply, according to a network mode parameter, the limits imposed by the system can be different for each mode, e.g.:

Table 1: Translation Matrix for Security levels and Network Modes

	Security Level		
Network Mode	*Low*	*Medium*	*High*
Normal	DES	3DES	AES
Impacted	None	DES	DES
Emergency	AES	AES	AES

-in *Normal* Mode: DES, 3DES, AES

-in *Impacted* Mode: no encryption, DES

-in *Emergency* Mode: AES (in this case the range is degenerate).

Users can select any algorithm within the limits of the mode. If we supply security levels, the user will have just select "low", "medium" and "high". These levels will be mapped to a specific algorithm through a translation matrix, like Table 1. So we can see that network modes provide alternate mappings for the security levels offered to the users, since "high" security would be translated in a different way if the system is in normal or in emergency mode.

In this example the security levels are mapped to a specific value for the encryption algorithm. Security levels can also be mapped to a sub-range within the acceptable range of values for the variant security attribute. The underlying RMS would then be responsible for further modulating the request and assigning specific values to the attributes, that would be within the mapped subrange.

The relationship between system capabilities, network mode limits and security level choices can be seen in Figure 2. This scheme represents an integration among the choices and limits of the system's layers, in which both user choices and system limits are determined by network modes. Such a strong integration of modes with choices as well as system limits would be typical in a military environment, for example. In less integrated policy domains, such as the Internet, user choices and mode limits can be completely independent: that means that user choices could involve values and ranges outside the limits of the network mode (in such a case the request would be rejected, however).

3.2 Calculation of Resource Costs for Quality of Security Service

To quantify the costs related to a task's security requests, we use a costing framework (based on a security service taxonomy [8]) with cost expressions relative to every security service invoked by the task. Each service may access various resources, e.g. CPU, memory and bandwidth (other cost factors are possible, e.g. disk space, and will be added to our framework). We discriminate between start-up and streaming costs. The calculated costs can then be fed to a middleware QoS mechanism for use in its resource allocation and scheduling decisions.

Cost values of any given security selection within a security policy may be calculated by knowing the fixed and variable costs of all settings of security mechanisms. In our QoSS demo, these mechanisms are represented by variable attributes offered by applicable services. The cost for a given service is calculated using a formula dedicated to the cost type and the attributes relevant to the given service. For example, a formula might exist for calculating the CPU start-up cost associated with providing the service of integrity on the Network Wire where the variable attribute involved is the selection of the authentication algorithm.

Formulas are defined by entering an algebraic expression and a table of cost values for each non-linear function in the expression. An algebraic expression consists of attribute variables, non-linear functions, constants, and basic mathematical operators. For example a hypothetical algebraic expression might be: "1000 + f(Authentication_Algorithm) + 2 x Key_Length". An example cost table value for the authentication algorithm attribute md5 could be f(md5) = 1200. For Key_Length=128 the resultant calculated cost for this example is 2456 cost units (where in a real expression cost units could be clocks or bytes per packet).

Our QoSS demo offers two approaches for observing calculated costs. In demo mode, the user selects the network mode, security level, and a strategy for determining settings within any selected ranges for mode and level. This strategy simulates input from a resource management system, and allows selection of minimum, middle, and maximum settings for the given mode and level. Figure 3 shows costs associated with a hypothetical task for a given mode, level and resource management strategy. This mode may be utilized for example when

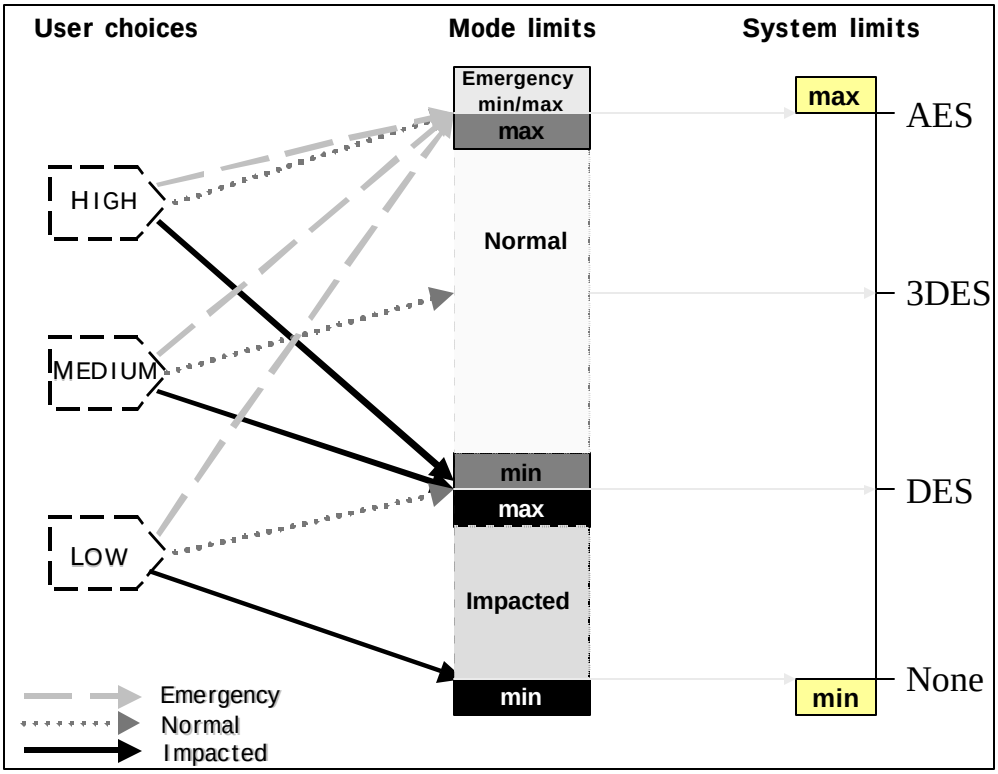

Figure 2: Security Level and Network Mode Range Relationships

planning the security requirement ranges provided to each mode and level.

In task mode, costs for executing tasks are determined based on 1) the administrator's current selection of network mode, 2) the user's current selection of security level, and 3) the current selection of the resource manager's strategy for determining specific values within the selected mode and level ranges. Costs may also be displayed in task mode, but, more significantly, task mode is able to export cost values to a QoS resource manager.

4. Modulating IPsec for provision of Quality of Security Service

For security to be a real part of QoS, security choices must be presented to users, and the QoS mechanism must be able to modulate related variables to provide predictable security service levels to those users. As a proof of concept we want to demonstrate how a specific security mechanism can be modulated to provide different levels for security, in response to QoSS requests from users.

We chose to experiment with IPsec. The IPsec mechanism provides services including confidentiality, integrity, authenticity, through the establishment of Security Associations (SA) among the entities that wish to communicate. The SA is a "simplex connection that affords security services to the traffic carried by it" and it essentially is "a management construct used to enforce a security policy in the IPsec environment" [12]. There is a set of parameters associated with each SA, which includes, among others: SA lifetime, encryption and/or authentication algorithms and keys, and protocol mode (tunnel/transport). The SAs can be generated manually, but that approach does not scale well. The Internet Key Exchange (IKE) along with the Internet Security Association and Key Management Protocol (ISAKMP) address the problem of establishing and maintaining SAs through the use of an automated daemon.

The IPsec protocols themselves do not include an approach for managing the policies that control which host is allowed to establish SAs with another host and what kind of characteristics the SAs should have. We are using the OpenBSD's implementation of IPsec [13]. This implementation addresses the SA management problem by including a trust management system, KeyNote, and providing an additional check in the IPsec processing: it makes sure that the SAs to be created agree with a local security policy (that can be expressed in the trust management system's language [14]).

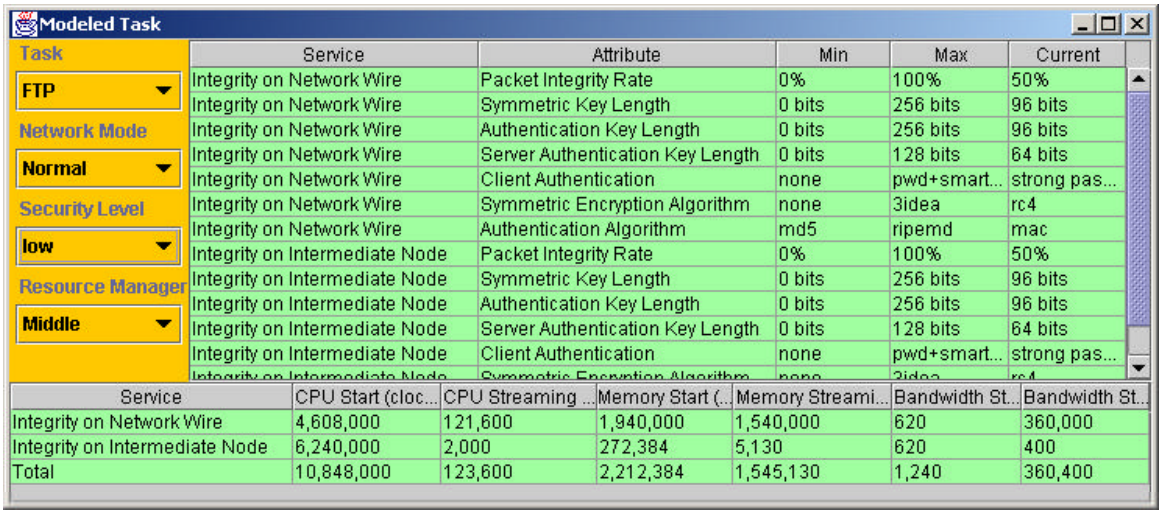

Figure 3: Task Costs for a given Mode, Level and Strategy

The IPsec protocols themselves do not include an approach for managing the policies that control which host is allowed to establish SAs with another host and what kind of characteristics the SAs should have. We are using the OpenBSD's implementation of IPsec [13]. This implementation addresses the SA management problem by including a trust management system, KeyNote, and providing an additional check in the IPsec processing: it makes sure that the SAs to be created agree with a local security policy (that can be expressed in the trust management system's language [14]).

This foundation gives us the opportunity to apply our QoSS ideas. We activate the local security policy for IPsec based on the current selection for the network mode and the security level. If the network mode changes to reflect a modification in the system status, or if we just want to execute the same application but with higher security, then we update the local security policy enforced by the trust management system. Then we signal the automated keying daemon that from now on it should use the new policy when negotiating SAs. Additionally, currently active SAs are renegotiated to conform to the current set of security requirements.

This way, we manage to provide different IPsec processing to traffic according to system settings: for example, when we are in "Normal" mode and "Low" security level, we apply no IPsec processing to finger traffic and we encrypt telnet traffic with DES. If we change security level to "High", then subsequent finger traffic is authenticated with SHA, and telnet traffic is encrypted with AES. Other things could change as a result of our selections: the set of hosts we are willing to communicate with using IPsec, the SA lifetimes, the key lengths or rounds for variable key-size / variable round algorithms.

Currently we have predefined sets of alternate local security policies that describe the characteristics we want our SAs to have for each <network mode, security level> pair and we make active the proper selection through one of our programs. We are working on identifying an architecture that would allow the trust management system and/or the automated daemon to be automatically notified of changes to QoSS parameters (like network mode, security level) and to adjust properly the SA characteristics they are willing to negotiate.

Furthermore we plan to conduct experiments and measurements to help us understand the impact of QoSS on the performance of applications under various network operational modes and high level policies.

5. Conclusion

We presented here our approach for handling security as a QoS dimension and we discussed how variability in network security services and their associated costs can be managed in a middleware environment. Finally we illustrated that a security mechanism like IPsec, can be modulated to provide levels for security in harmony with QoSS requests.

6. References

[1] Foster, I. and Kesselman, C., “Globus: A Metacomputing Infrastructure Toolkit”, Intl J. Supercomputer applications, 11(2):115-128, 1997

[2] Dail, H., Obertelli, G., Berman F., Wolski, R., Grimshaw, A., “Application-Aware Scheduling of a Magnetohydrodynamics Application in the Legion Metasystem”, Proc. of the Ninth Heterogeneous Computing Workshop (HCW 2000), Cancun, Mexico, May 2000, pp. 216-228

[3] Huh, E.N., Welch, L.R., Shirazi, B.A., Cavanaugh, C.D., “Heterogeneous Resource Management for Dynamic Real-Time Systems”, Proc. of the Ninth Heterogeneous Computing Workshop (HCW 2000), Cancun, Mexico, May 2000, pp. 287-296

[4] Hensgen, D., Kidd, T., St. John. D. Schnaidt, M., Siegel, H.J., Braun, T., Maheswaran, M., Ali, S., Kim, J., Irvine, C., Levin, T., Freund, R., Kussow, M., Godfrey, M., Duman, A., Carff, P., Kidd, S., Prasanna, V., Bhat, P., Alhusaini, A., “An Overview of MSHN: The Management System for Heterogeneous Networks”, Proc. of the Eighth Heterogeneous Computing Workshop (HCW’99), San Juan, Puerto Rico, April 1999, pp. 184-198

[5] Linn, J., Generic Security Service Application Program Interface, IETF Request for Comments: 1508, September 1993

[6] Schneck, P.A. and Schwan, K, “Dynamic Authentication for High-Performance Networked Applications”, Technical Report GIT-CC-98-08, Georgia Institute of Technology, College of Computing, Atlanta, GA, 1998

[7] Irvine, C. and Levin, T., “The Effects of Security Choices and Limits in a Metacomputing Environment”, Technical Report NPS-CS-00-004, Naval Postgraduate School, Monterey, CA, January 2000

[8] Irvine, C. and Levin, T., “Toward a Taxonomy and Costing Method for Security Services”, Proc. of the Computer Security Applications Conference, Phoenix, AZ, December 1999, pp. 183-188.

[9] Irvine, C. and Levin, T., “A Note on Mapping User-Oriented Security Policies to Complex Mechanisms and Services”, Technical Report NPS-CS-99-08, Naval Postgraduate School, Monterey, CA, June 1999.

[10] Irvine, C. and Levin, T., “Quality of Security Service”, Proc. of New Security Paradigms Workshop 2000, Cork, Ireland, September 2000, pp. 91-99

[11] Spyropoulou, E., Levin, T., and Irvine, C., "Calculating Costs for Quality of Security Service", Proc. of the Computer Security Applications Conference, New Orleans, LA, December 2000, pp. 334-343.

[12] Kent. S. and Atkinson, R., "Security Architecture for the Internet Protocol", Internet RFC 2401, Internet Engineering Task Force, November 1998

[13] Blaze, M., Ioannidis, J. and Keromytis, A.D., "Trust Management for IPSec", Proc. of the Internet Society Symposium on Network and Distributed Systems Security 2001, San Diego, CA, February 2001, pp. 139-151.

[14] Blaze, M., Feigenbaum, J., Ioannidis, J. and Keromytis, A.D., "The KeyNote Trust Management System Version 2", Internet RFC 2704, Internet Engineering Task Force, September 1999.