

Dynamic Cryptographic Context Management (DCCM)¹

Report #2, Version 2

Cryptographic Context Negotiation Template

February 24, 1999

Mr. David M. Balenson, PI
Dr. Dennis K. Branstad²
Dr. David A. McGrew³
Mr. Jay W. Turner⁴
Mr. Michael Heyman

Cryptographic Technologies Group
TIS Labs at Network Associates, Inc.
3060 Washington Road (Rt. 97)
Glenwood, MD 21738

¹ Supported by the Defense Advanced Research Projects Agency (DARPA) under Rome Laboratory Contract No. F30602-97-C-0277.

² Previously an employee of Trusted Information Systems and a Principal Investigator; presently a part-time, temporary employee of TIS Labs at Network Associates, Inc.

³ Previously an employee of Trusted Information Systems; presently the manager of the Cryptographic Software Development Group at Cisco Systems.

⁴ Guest researcher from the National Security Agency (NSA), under the NSA Industry Partnership Program during the period May 1998 to August 1998.

Abstract

The Cryptographic Context Negotiation Template (CCNT) report, Version 2, is a revised version of the second publication [Bal98b] resulting from the Dynamic Cryptographic Context Management (DCCM) Project. This project is sponsored by the Defense Advanced Research Projects Agency (DARPA) and is being performed by TIS Labs at Network Associates, Inc. (previously Trusted Information Systems, Inc.) This report describes an abstract template for use in negotiating a cryptographic context among a very large number of participants from various organizations. The Cryptographic Context Negotiation Template, or CCNT, is used to establish a cryptographic association among the group of participants and to enforce the cryptographic provisions of the security policy imposed on a sensitive computer application or project.

The DCCM project assumes that a number of organizations, perhaps with differing security policies, have agreed to work cooperatively on a specific project under the control of a designated project initiator. The CCNT is designed to accommodate negotiating a single cryptographic-based security policy based on the desires of the project initiator and the constraints of the organizations and individuals participating in the application.

This report has been revised to reflect the changes in the template now in use with the Cryptographic Context Negotiation Protocol described in Report Three of the DCCM project.

The significant results of the project contained in this report include:

1. An overview of the policies that can be encoded in a CCNT; and
2. A revised specification of the CCNT in the BNF (Backus-Naur Form) syntax specification language.

Keywords

Access control; authentication; authorization; confidentiality; cryptographic context; cryptography; group keys; integrity; group-key management; large-group management; multicast; multi-party security; peer-peer security protocols; policy negotiation; security policy.

Contents

1. INTRODUCTION	1
1.1 Project Overview	1
1.1.1 Project Goals	2
1.1.2 Project Scope	2
1.2 DCCM Approach.....	3
1.3 Report Overview	5
1.4 Revisions	5
2. SYSTEM POLICY	7
2.1 Security Policy.....	9
2.2 System Policy for DCCM.....	10
3. CRYPTOGRAPHIC CONTEXT NEGOTIATION TEMPLATE.....	13
3.1 SPL: A Grammar for Security Policies and Contexts	14
3.2 Semantics.....	15
3.3 The SPL Grammar.....	18
3.4 Examples	19
3.4.1 Example Policies	19
3.4.2 Example Contexts.....	19
3.5 Implementation.....	19
4. CONCLUSIONS.....	21
ACKNOWLEDGMENTS	22
REFERENCES	23
APPENDIX A: GENERAL SYSTEM SECURITY POLICY	24
A.1 Prevention.....	24
A.2 Detection.....	27
A.3 Response.....	27
A.4 Group Policy.....	28
A.4.1 Group Dynamics.....	28
A.4.2 Group Keying	29
APPENDIX B: JAVACC FILE FOR SPL BNF.....	31

List of Figures

Figure 1: Venn Diagram of policy inputs to context creation.....8
Figure 2: Relationship between policy, context and templates.....13

List of Tables

Table 1: Revised DCCM terminology6
Table 2: Template categories and values14
Table 3: Syntax tokens.....16
Table 4: SPL Axes and point tokens.....18

1. Introduction

1.1 Project Overview

The Dynamic Cryptographic Context Management (DCCM) project is addressing the problem of efficiently providing security for sensitive information technology applications involving large, dynamically-changing groups of participants. For example, command and control of tactical military forces from different armed forces units and perhaps from different countries and working together under one commanding officer for a period of time or for a specific military exercise will require a variety of security services. By “large” we mean groups with a number of members ranging from approximately 1,000 to 100,000. By “dynamic,” we mean that new members may be added to the group at any time and existing members may be evicted from the group (e.g., a position may be overrun by enemy forces), thereby requiring immediate changes to some of the security provisions. Members need not be humans; they can include a variety of communicating entities such as sensors, mobile client workstations, server workstations, or network nodes. Participants in a project (i.e., members of the group authorized to participate actively in the project) can be organized in several ways, ranging from one large uniform group under a single management to a complex organizational structure with several layers of management.

In the DCCM Architecture and System Design report[Bal98a], we laid a framework for:

- Categorizing large-group applications and their security requirements;
- Defining large-group management models and authorization needs;
- Identifying candidate security context management solutions; and
- Evaluating candidate solutions with regard to the requirements of particular applications.

In this report, we present:

- An overview of the policies that can be encoded in a Cryptographic Context Negotiation Template (CCNT); and
- A revised specification of a CCNT in the BNF (Backus-Naur Form) syntax specification language.

1.1.1 Project Goals

The goals of the DCCM project are to:

- Design a management system which identifies, authenticates, authorizes, and manages members of sensitive large-group computer applications;
- Create a security policy language for a project initiator to easily define the security policy to be invoked for the application;
- Create a translator for translating the selected security policy into a preferred security context using a cryptographic context template;
- Develop a large-group security context negotiation protocol that derives a group cryptographic context to be used for an application;
- Develop a testbed to demonstrate all aspects of dynamic cryptographic context management;
- Implement software in the testbed to demonstrate a manager workstation and a number of client workstations managing and participating in a large, dynamic group application; and
- Transfer the technology and the software to other researchers and developers working on large-group sensitive applications.

This report addresses the second and third goals above.

1.1.2 Project Scope

The scope of the DCCM project includes all aspects of managing the security of a large-group information technology application. Specifically, it includes: security policy definition, security context specification, cryptography-based security context negotiation, group authorization management, group authentication management, large-group keying algorithms and protocols, and efficient re-keying following dynamic changes of the group. The scope of this report includes security context specification in a formal specification language by defining a flexible, abstract template that supports a wide range of security policies and the negotiation of a single policy for any sensitive application. The negotiated policy must satisfy the desires of the project's manager while satisfying the constraints of the organizations participating in the application to the greatest extent possible.

1.2 DCCM Approach

While conducting the background investigation for this report, the project team looked at policy from different perspectives at various levels. One perspective is obtained by reviewing policy by organization and by application, obtaining a view of a hierarchy of policies (i.e., layered policies are common in many organizations) that must be taken into account while negotiating a policy for a new application or project. A second perspective can be obtained by focusing on security issues, group issues and the cryptographic sub-issues that pertain to the DCCM effort.

As we focused on the security categories and negotiation options that should be included in the cryptographic context negotiation template, the team examined the broader issues of system security policy and its current treatment in the literature. We found this to be a most useful exercise. The team consisted of individuals with a wide background in information security and each was encouraged to share his or her perspective on multi-organizational policy negotiation and how it can best be structured. In particular, one person of the group with extensive experience within the Department of Defense (DoD) shared his ideas on unstructured group dynamics, establishment of a group leader among an unstructured group of strong leaders, and negotiation. The following paragraphs summarize his thoughts and responses of other members of the team. Attempts to coalesce these thoughts into a static approach to template specification (i.e., a table) caused a change of approach to DCCM policy support (i.e., an extensible, flexible formal language).

The DoD representative described the following scenario. Suppose a group of senior managers and senior technical experts is created to work on a sensitive mission. Several hidden personal agendas often exist within such a group. Some group members will be acquainted with other members and know the potential de facto leaders that will surface during group interactions. Other factors that affect consensus building and group structure include: mission “owner”; budget “owner”; seniority; professional experience; professional risk. In the military, many things are ranked or hierarchically structured. However, in a new group being established many people are of equal rank and a new leader cannot be easily determined. For example, if several generals of the same rank are included in the group, an informal ranking must be established by using factors such as the command they lead, their past assignments, their next assignment if known, and whose territory is most affected. Thus, policy negotiation among such a group must make provisions for both informal organizational structures as well as formal structures.

In spite of strong leaders always wanting to be the leader of a new group, participants must accept a subordinate role for the good of the mission and permit equality among inputs (at least during the first phase of negotiation). Progress cannot be made without quickly establishing leadership roles. Technical people usually defer to their managers. To negotiate group leadership roles and security policies in a flexible, effective, and efficient manner, the entire group must be represented by supporting managers.

The DoD team member remarked that the negotiation phase usually culminates in a set of initial agreements so that the group participants can disperse to initiate their assigned tasks. However, effective groups must have the ability to revisit the negotiation process in the near future where progress is reviewed and the negotiated agreements may be reopened to further discussion and negotiation.

The DCCM lesson learned from this example was that the mediator should act as a facilitator during initial negotiations, giving all parties equal weight initially (subject to “senior officials” who are more “equal” than others). Open negotiation should be allowed to continue until it is time for the mediator to bring the process to closure.

In the DoD world, there are legal and political reasons for not sharing certain information with certain current or potential “peers.” This “need to know” policy can also be destructive or at least a deterrent to success. If all participants in a group demonstrate trust and cooperation and have other methods of “protecting turf”, negotiations proceed more smoothly. The negotiation process should proceed in a manner that precludes anyone asking the question, “Who won?”

This discussion lead to a multi-tier negotiation process for DCCM. Policy can be divided into different levels by specificity or by the level of the policy maker in the organization. High level expressions and low level expressions of policy can be established. Policy can be addressed first but can also proceed immediately to detailed specifications.

Another team member stated the need to address the large issues of multi-organizational policies, hierarchical policies, inconsistent policies, incompatible mechanisms, and negotiation among multiple organizations with multiple goals. He noted that the team might not solve all the problems in these areas but that they need to be addressed. Flexible templates, negotiation strategies, and dynamically re-configurable participant systems can then be created. He noted that one company had the capability of providing the software needed to run in his system in order to be compatible with their system. JAVA addresses the same goal. Other security issues such as trust of downloaded software are involved, of course, but the approach should be considered. He recommended that the project team not duck these issues because one solution cannot be picked. He further noted that if the research group had difficulty arriving on a single approach then many people from many organizations will have greater difficulty in agreeing with any single solution that is picked. He recommended that approach similar to the evolution of programming languages should be taken in which syntax is formalized in a manner that can easily be extended (even by the programmer) and semantics is left up to an interpreter that is provided with the compiled program.

Based on discussions and inputs such as these, a more flexible approach to policy and context negotiation has been taken. Binding of decisions is being delayed as much as feasible. Structure is being formalized while interpretation is being generalized.

1.3 Report Overview

This report focuses on the template used to negotiate a cryptographic context among a large set of potential participants in a sensitive computer application (e.g., military exercises, cooperative research projects, televised instruction courses, multi-organizational logistical systems). The initial approach (as was suggested in the proposal) involved a static template like a blank form that would be filled in by the project initiator and transmitted to each potential participant. Such electronic forms are commonplace in electronic data interchange, electronic funds transfer, and similar structured informational systems. The template was envisioned to have a large number of sections with several alternatives that could be selected in each section. Each section would represent a policy statement or area (e.g., the need for information confidentiality of a certain level, the requirement for specific authentication and authorization methods) and the alternatives would represent the mechanisms (e.g., data encryption, passwords, authorization certificates) that would suffice for meeting the policy. Each participant would then simply electronically select the subset of alternatives that could be supported at the network entity (host, workstation, client facility) being used by the potential participant. This subset of the multiple-choice context template would be returned to the project initiator who would select the set of compatible and interoperable mechanisms best satisfying the policy and supported by the optimal (e.g., maximum, most productive) set of potential participants.

The report presents a short summary of some of the inputs received on cryptographic association establishment and management and the discussions held while trying to resolve and accommodate the inputs. It was decided to abandon the static electronic form of an information template and adopt the formal syntactic language specification form developed when specifying the ALGOrithmic Language (ALGOL) in 1960. John Backus and Peter Naur participated on the committee that specified ALGOL-60 and are credited with developing the Backus-Naur form of syntax (i.e., structure) notation [Backus59, Naur63]. A variant of the original notation (developed for punch-card systems) is used in this report.

The report also includes the specifications of the Security Policy Language (SPL) which were input to the JavaCC computer program. This program is a parser generator, which takes as input syntax specifications and builds parser code that can be used to “recognize” the defined language. This program will be used in developing the cryptographic context negotiation protocol.

1.4 Revisions

Version 2 of the CCNT represents the maturation of the DCCM project during the development of the Cryptographic Context Negotiation Protocol (CCNP) as documented in Report Three. This version is designed to be used with Report Three.

The language used to describe the template in this version is more concise than the previous version. This provides a language that is easier to code, maintain, and debug. In addition, the new language allows extensions without recoding any of the software the processes the language.

Several terms used in the earlier reports have been changed for terms that are more tightly focused or better representative of the system evolution. The changes are detailed in Table 1: Revised DCCM terminology.

Term in Report 2	Term in Report 2v2	Notes
Application	Project	The term application was intended to be broad e.g. command and control, but was often confused with the narrow computer program definition, e.g. Microsoft Word. A Project is the scope for a cryptographic context and may cover several computer applications.
Application Manager	Project Manager/Initiator	
Organization Supervisor	Security Representative	The actions expected in the DCCM architecture of the supervisor are better delegated to a role focused on security. This also describes the role for either hierarchical or non-hierarchical environments.

Table 1: Revised DCCM terminology

2. System Policy

Within the DCCM architecture, the primary information to be exchanged to establish secure group communications involves policy regarding cryptographic security and group operating procedures. Group managers and members seek to establish secure associations by agreeing on acceptable cryptographic mechanisms, algorithms and parameters specifying how the ensuing communications are to be protected. The DCCM architecture also supports establishing rules governing dynamic group membership and the ways a security association should accommodate those rules. It presents a framework for establishing policy governing the management of, and communications within, multi-organizational groups. The framework encompasses all categories of project policy and organizational policy necessary to establish an acceptable cryptographic context for any arbitrary group.

This report the result of our effort to define a negotiation template for DCCM. Our analysis was conducted looking towards the next step of defining the policy and context negotiation protocol. The template is a tool used in performing negotiation, so the two cannot be separated. Policy and context negotiation use inputs from different sources, including the architects and implementers of DCCM. The inputs include goals and constraints of both people and organizations. Negotiation also depends greatly on the availability of cryptographic mechanisms and support services for a project. In order to facilitate this negotiation, the template must allow for a wide variety of inputs described at various levels of specificity. While a static template was the working model during architecture development, this approach has been found to be unsatisfactory during this present template design phase. We may find it necessary to modify the current approach again during system development. We anticipate extending the abstract template specification as we proceed with the negotiation work, but we will continue to strive for completeness and flexibility.

Within the DCCM model, very large dynamic groups are created, maintained and managed. In the initial DCCM model, each project group must agree upon an overriding policy that applies to all group communications for the duration of the project. This includes all sessions established under that project. Hence, the DCCM system policy must cover both traditional systems security issues as well as group dynamic actions.

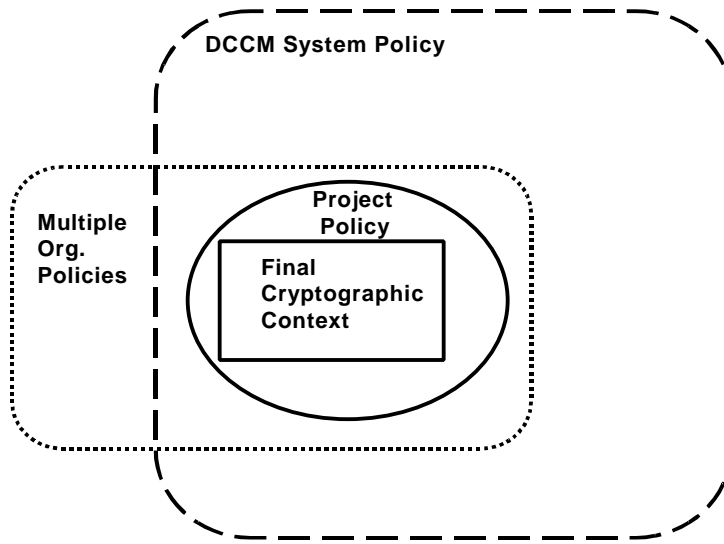


Figure 1: Venn Diagram of policy inputs to context creation

Figure 1 shows a logical intersection DCCM system policy, multiple organizational policies, and a project policy that results in a cryptographic context being created for an application.

Policy exists at different layers of an organization. Policy is normally broad when established at the top of an organization and is more detailed at the lower layers. Cryptographic policy is a part of security policy that is a part of information technology policy that is a part of corporation or government policy. Higher levels of policy from various organizations influence the security policy of a project. A cryptographic context is a very specific instantiation of the specific policy negotiated among the participating organizations for a particular large-group sensitive project.

The **DCCM system policy** must be broad and flexible enough to include the union of many **project policies** and to accommodate many diverse **organizational policies**. This is denoted as:

$$\text{DCCM System Policy} \hat{=} \mathbf{E}(\text{Project Policies}) + \mathbf{F}(\text{Organizational Policies})$$

As we shall see later, organizational policies and project policies overlap, especially in the area of key management. This is not surprising, since the DCCM System begins with today’s best network security practices and builds on their principles in securing large dynamic group communications over the network.

2.1 Security Policy

Ignoring DCCM for the moment, it is instructive to discuss the broader aspects of computer security policy. We strive for completeness by focusing on generally accepted policy principles that are pertinent to any system. We broaden previous definitions for this discussion as follows.

By **security policy** we mean the layered set of objectives, rules, regulations, principles and practices which specify secure system operation. Security policy includes, but is not limited to, the traditional cryptographic security services of confidentiality, authentication and data integrity. It also covers diverse areas such as key management, access control, availability assurance, intrusion detection, system health issues, and other factors relating to the physical and electronic well being of the network. This is consistent with RFC 2196, The Site Security Handbook, which succinctly states “A security policy is a formal statement of the rules by which people who are given access to an organization’s technology and information assets must abide”.

Mechanisms are objects and functions that are used to enforce rules or deliver services. They are therefore policy enablers. Cryptographic algorithms, which encrypt, hash and digitally sign data, are the mechanisms that facilitate cryptographic security services. Passwords, hardware tokens and biometric procedures are mechanisms associated with the policy category of personal identification. Firewalls and guards are mechanisms that address and enforce access control.

A **context** is a set of protocols and mechanisms, along with their associated parameters that have been selected to carry out some specified security policy. A **cryptographic context** is a set of cryptographic protocols and algorithms, along with their modes of operation and their associated parameters, the collection of which, is intended to address all appropriate categories identified by the security policy. Relative to each of these categories, the policy will also specify the level of security that should be attained. The algorithms, their modes, and their parameters singled out in the cryptographic context must be consistent with these prescribed security levels.

Hence, policy issues reside at a higher level than that addressed in a cryptographic context. The context is a more detailed version of policy or in other words it is an instantiation or translation of policy with finer granularity. Moreover, mechanisms enforce a rule (or deliver a service) at some prescribed level of security as determined by a group of experts. This translation from policy to mechanism is not static, since the expert’s confidence in the robustness of a particular mechanism may change over time. Nonetheless, this **mapping** from specified policies to sets of mechanisms is essentially automatic and hence can be performed by an automated, (e.g., expert) system. DCCM uses an **expert mapping** system to transform policy specifications into a detailed context.

Attacks and countermeasures drive security policy. Security policy is intended to counter attacks, whether intentional or unintentional, against an information system. Ideally, a security policy for an information system is designed to protect against:

- Unauthorized disclosure of information;
- Unauthorized modification of information; and
- Imitation (e.g., impersonation) of authorized parties and processes by unauthorized entities.

At a very high level, the rules of a strong security policy must focus on preventing these fundamental infractions. In addition, the rules must also address the detection of such infractions in the event they do occur. Finally, when infractions are detected or perhaps just suspected, an organization should have in place another set of rules and procedures to respond to this condition. Hence,

Security Policy \hat{U} Prevention + Detection + Response.

Note that the rules for responding to incidents (real or suspected) may include making adjustments to the prevention and detection procedures. In addition, technical advances in prevention, detection and response methods frequently signify that an organization's security policy may require updating. Security policies should therefore be considered to be dynamic. Appendix A presents a comprehensive discussion of computer security policy including those aspects well beyond the scope of DCCM and cryptographic context negotiation templates. It is included in this document to show the breadth of security policy that a project initiator must consider but then refine to the cryptographic related issues that are within the scope of the DCCM system.

2.2 System Policy for DCCM

In terms of security services, the scope of the DCCM program focuses on access control, authentication, confidentiality and integrity. Neither availability nor non-repudiation is addressed. Within these categories, DCCM policy issues will be concerned with cryptographic mechanisms that facilitate those services. Naturally, the policy will also focus on key management features that support those services. Security policy categories within the DCCM system will pay particular attention to the level of security delivered by each mechanism and to the layer within the network model at which the service is delivered.

Group aspects of DCCM system policy for the most part coincide with the generic issues catalogued in the previous section. We will adopt the four basic group operations: initialize group, add entity, delete entity and dissolve group. We will also fold the group

key management issues into those previously identified in the security policy section. An outline of the categories, which drive policy decisions in DCCM, follows.

- 1) Access Control
 - a) Mechanisms
 - i) Passwords
 - ii) Tokens (e. g. FORTEZZA, etc.)
 - iii) Credentials (e. g. certificate, etc.)
- 2) Authentication
 - a) Communications layer
 - b) Robustness required (i. e. high, medium, low security)
 - c) Mechanisms
 - i) Digital signature (e. g. RSA, etc.)
 - ii) Message Authentication Code (e. g. HMAC-SHA1, etc.)
- 3) Confidentiality (= Encryption)
 - a) Data in transit
 - i) Communications layer
 - ii) Robustness required
 - iii) Mechanisms
 - (1) Public Key ciphers
 - (2) Block ciphers
 - (3) Stream ciphers
 - b) Data at rest
 - i) Robustness required
 - ii) Mechanisms
- 4) Integrity
 - a) Data in transit
 - i) Communications layer
 - ii) Robustness required
 - iii) Mechanisms (e. g. SHA1, etc.)
 - b) Data at rest
 - i) Robustness required
 - ii) Mechanisms
- 5) Group operations
 - a) Dynamics (i. e. initialize, add, delete, dissolve)
 - b) Authorizations
 - i) DCCM system manager
 - ii) Organizational supervisor
 - iii) Project initiator
 - iv) Participants (i. e. members and processes)

- 6) Key management
 - a) Point-to-point key exchange
 - i) Communications layer
 - ii) Robustness
 - iii) Mechanism
 - b) Group keying scheme
 - i) Mechanism (e. g. OFT, etc.)
 - ii) Key generation (Optional – depends on mechanism)
 - (1) Contributory
 - (2) Centralized
 - iii) Key distribution (Optional – depends on mechanism)
 - (1) Delivery procedure (i. e. how encrypted, hashed and signed)
 - (a) Communication layer
 - (b) Robustness
 - (c) Mechanisms
 - c) Supporting issues
 - i) Key period
 - ii) Compromise recovery
 - iii) Data protection (i. e. forward/backward secrecy)

Assembling a specific **DCCM system policy** equates to selecting one of the available options corresponding to each line in this outline. A set of options can be translated by way of a **DCCM expert mapping** into a set of algorithms, characteristics and parameters, which collectively specify a cryptographic context. The DCCM cryptographic context negotiation template, presented in the next section, is predicated on this outline of system policy.

3. Cryptographic Context Negotiation Template

The previous section showed that policy issues might be expressed with varying degrees of granularity. The finer the granularity, the more definitive the rules and procedures become. A context is a natural extension of policy, when that policy is expressed in specific, finely tuned terms. This extension or translation is performed by the DCCM policy-to-context translation system. The **DCCM Cryptographic Context Negotiation Template (CCNT)** is a tool to be used by negotiators, which addresses both policy specifics and mechanism availability.

In order to derive context level expressions, the policy guidance must be specific. For instance, policy guidance that states “even the most sophisticated adversary should not be capable of reading traffic” is not specific enough to derive a cryptographic context. The guidance points to confidentiality with a requirement that the mechanism be highly robust. However, other issues such as the communications layer that will carry the service and the protocol that is to manipulate the information must be addressed before a completed context can be agreed upon.

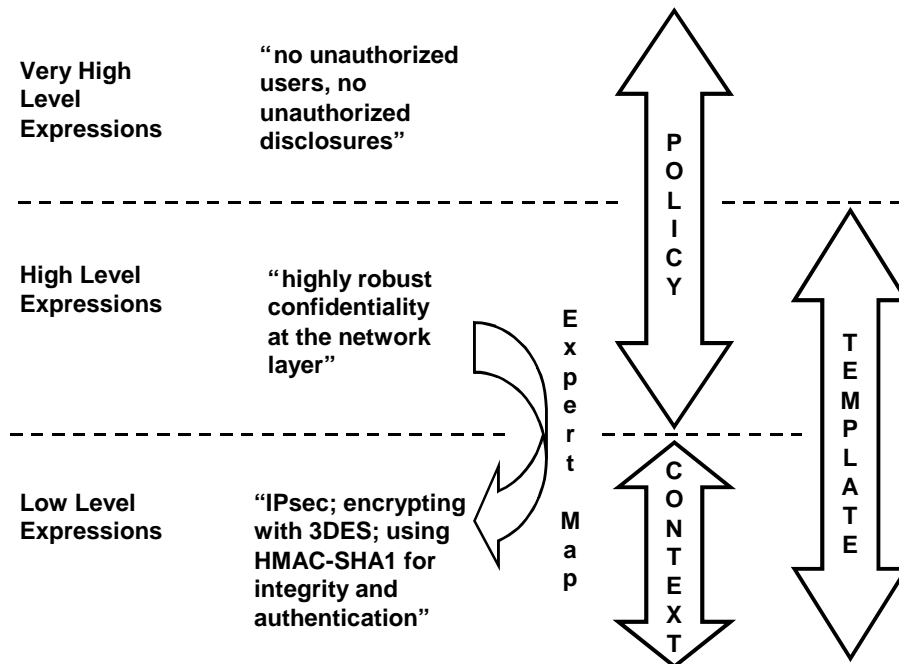


Figure 2: Relationship between policy, context and templates

Figure 2 illustrates the relationship between the policy, the context and the template. Users of the template (i.e., negotiators) will have a variety of concerns, which they will address individually. For higher level managers, concerns will focus more on the big

picture and will deal mostly with very high level expressions. The template might ask them to adjust their perspective somewhat to deal with some of the more specific policy issues at the next level of granularity. For instance, it's plausible to expect higher level managers to address the level of security. Group managers, supervisors and leaders will likely be concerned with a number of very specific policy issues. They will bring considerable knowledge of system performance and vulnerability issues into the negotiation process. It is reasonable to expect their input to the negotiation process to touch on some of the context issues as well. Representatives of the user population are expected to focus primarily on the context categories, with their input to the negotiation process. Users are driven by availability of mechanisms, convenience and efficiency.

Hence, the cryptographic context negotiation template must address policy issues at a detailed level and must also accommodate inputs that will focus on purely context level categories. Table 2 identifies some of the categories that should be delineated in any template along with a few of the values one can chose for each category.

Category	Values
Communication layers	Application, Transport, Network
Services	Access Control, Authentication, Confidentiality, Integrity, Key Management
Level of security	High, Medium, Low
Mechanism	Kerberos, SSL, IPSec, ...
Algorithm	DES, RC4, DSA, RSA, SHA-1, MD5, ...
Parameters	Operating Mode, Key size, Key period, ...
Characteristics	Forward secrecy, Backward Secrecy, ...

Table 2: Template categories and values

3.1 SPL: A Grammar for Security Policies and Contexts

Our approach to formalizing the idea of a cryptographic context negotiation template is to define a Security Policy Language (SPL) using a Backus-Naur Form (BNF) grammar. This approach enables us to represent more general, complex, and flexible policies and contexts than those that can be captured by a fixed tabular form. The SPL language is a flexible, easily extensible context template (in an abstract form).

A BNF grammar consists of a set of symbols and the rules for combining them. The rules are called *productions*, and are used to define the non-terminal symbols. Terminal symbols, called *tokens*, are the actual symbols that can appear in statements. *Pseudo-terminals* are symbols that define classes of tokens, such as integers and character strings. Each production specifies a substitution rule. By applying these rules, a parse tree can be built for any valid statement in the grammar. The leaves of this tree are tokens. In BNF productions, the symbol \rightarrow denotes 'is defined to be', and the symbol $|$ denotes the logical 'or'. See [Backus59, Naur63] for a complete description of BNF grammars.

3.2 Semantics

The grammar is set up to first lay out a multi-dimensional security space, then list specific points in that space. This multi-dimensional security space holds the context template.

The axes in the multi-dimensional space include previously discussed things like “communications layer”, “encryption”, and “error handling”. Along the “encryption” axis, there may be values like “des”, “rc5”, and “gost”. Points in the multi-dimensional space are described by the values along each axis at the intersection, such as des, SHA, PFF.

A Security Context is the complete and unambiguous specification of the cryptographic mechanisms used for a set of communications. Usually a single point in the security space will denote a security context, although some security contexts may take more than one point.

A list of specific points in the security space is the list of locally allowed mechanisms. This is also an expression of the local policy or union of local policies. If the list of points is an expression of multiple policies, then one can name the individual policies using the *category* element of SPL.

Any security context, point, or group of contexts or points can be placed into a category. Categories are higher-level mappings of contexts for local use and are not part of any negotiations. These categories can be policy names when the template holds more than one local policy. They can also be some other description. For example, they can separate mechanisms into “good” and “strong” and “type 1” (US government approved for classified) crypto. Or, a group of mechanisms could be in the category that is fast enough for real-time video. US government classified real-time video transmission could only reasonably come from the intersection of this category with the “Type 1” category.

A negotiation starts with all the mechanisms that the project initiator wishes to accommodate listed in the security space (the project initiator previously chose policies and categories to come up with this list of mechanisms). This is the Security Context Template.

The SPL syntax is flexible and extensible; new protocols, algorithms, and mechanisms can be employed at any time without affecting the negotiation process. A negotiator will not select any protocol, algorithm, or mechanism that it does not understand. So, it only makes sense to employ algorithms that others will understand. A good example is that it is likely systems that can compute AES when that gets standardized will understand a template that includes it.

It is possible to describe undefined or non-existent combinations of algorithms and protocols. For example GOST/32 on SSL (secure socket layer) could be a possible mechanism, except that though this policy may be sensible, no such cipher has been

defined for use in SSL. A sensible project initiator would not give this as an option. Sensible security representatives would not respond with this value given the choice.

In general, an SPL text will look like:

```
Axis Name ( Axis Point Name, Axis Point Name, Axis Point Name)
Axis Name ( Axis Point Name, Axis Point Name)
Axis Name ( Axis Point Name, Axis Point Name, Axis Point Name)
Axis Name ( Axis Point Name)
.
Category(index range, index range, index range, index range)
Category,Category(index range, index range, index range, index range)
                and(index range, index range, index range, index range)
```

The language tokens for SPL are described in Table 3.

Syntax Tokens	Semantics
.	Separate axes from points
()	Group axes, or points
,	Separate indices in a point
- * ~ !	For point range
And	Connect two points

Table 3: Syntax tokens

Each group of points (after the “.”) must have an “index range” for each axis. The first “index range” corresponds to the first axis, the second “index range” corresponds to the second axis, etc. An index range like: “#”, or “#-#”, or “(##,##)”, or “(##,##-#)”, or “*”, or “~”, or “!”, where “#” is a integer greater then or equal to zero, “*” represents all axis points, and “~” and “!” represents no axis points. The axis point names are referenced by their index in the axis. The first axis point name is at index 0, the second, at index 1, etc.

There is no need to have the different elements on different lines (in fact, in negotiations there may be a requirement that an SPL text be on one line). They are just placed on different lines here for clarity.

Although no axis name or axis value in the template must be predefined, it is prudent to select a common group of tokens for these names that implementations are likely to understand. Table 4 shows a list of typical values. Note, there is no reason to define any points along an axis that have no locally supported and allowed mechanisms (if you are not going to do DES encryption, there is no reason to have a “DES” point along the “Encryption” axis).

Axis	Point along Axis
Layer	<ul style="list-style-type: none"> • Application • Network • Ssh • Ssl • Ipsec • Swipe
Encryption	<ul style="list-style-type: none"> • des, cast, 3des, gost, idea (block ciphers) • Ecb, cbc, cfb, ofb, cbbp (block cipher modes) • RC4, seal (stream ciphers) • md2, md4, md5, ripemd-128, ripemd-160, sha-0, sha-1 (for update)
Key Management	<ul style="list-style-type: none"> • OFT • LKH • Diffie-Hellman • passwords • strong, weak (password type) • des-mac, md5, sha-1 (password filters)
Key Non-Ephemerality	<ul style="list-style-type: none"> • Partial Forward secrecy • No Forward secrecy • Partial Backward secrecy • No Backward secrecy
Key Recovery	<ul style="list-style-type: none"> • RSA, ElGamal (asymmetric ciphers) • PKCS1, ISO, (format)
Data Authentication	<ul style="list-style-type: none"> • DSA • RSA • PKCS1, ISO, PGP, AOEP (RSA formats)
Entity Authentication	<ul style="list-style-type: none"> • Kerberos-v4, Kerberos-v5, Radius, Weak passwords, Strong passwords, Fortezza, Secureid, Smart-card • DCCM System Manager, Supervisor
Integrity	<ul style="list-style-type: none"> • crc, md2, md4, md5, ripemd-128, ripemd-160, sha-0, sha-1
Access control	<ul style="list-style-type: none"> • see Authentication
Maximum Data Throughput	<ul style="list-style-type: none"> • <digits>(Bps KBps MBps GBps bps Kbps Mbps Gbps)
Sub-Grouping Control	<ul style="list-style-type: none"> • <digits> (levels of sub-grouping, 0 means only security representative participants allowed)
Session Launching Control	<ul style="list-style-type: none"> • <digits> (lowest level, security representative participant is level 0)
Eviction Restrictions (Who can't evict)	<ul style="list-style-type: none"> • Superior And Peers • Superior And Local Peers • Superior And Self • Self

Error handling	<ul style="list-style-type: none"> • Reject • Log • Sender Notification • Retry
----------------	---

Table 4: SPL Axes and point tokens

3.3 The SPL Grammar

Characters in **bold** are terminals.

* - ZERO OR MORE OF PRECEDING

+ - one or more of preceding

Square brackets are either part of a regular expression or they enclose an optional part of the language. (ex: in the “axis” production, “axis-label” is optional).

Policy ?	axes . point-list	<i>axes separated from points by “.”</i>
axes ?	(axis)+	<i>one or more axis</i>
axis ?	[axis-label] (point-label-list)	<i>axis: list of points with possible label</i>
axis-label ?	[^(]+	<i>all characters up to open parentheses</i>
point-label-list ?	point-label (, point-label)*	
point-label ?	([^ ,”]+ “ [^ “]* “)	
point-list ?	point point (and point)*	<i>points can be associated by “and”</i>
point ?	[categories] ((range (, range)*))	<i>range count same as axis count</i>
categories ?	category (, category)*	
category ?	([^ ,”]+ “ [^ “]* “)	
range ?	simple-range (simple-range (, simple-range)*)	<i>ex: 15 or 27-33 or (15, 27-33, 35)</i>
simple-range ?	[0-9]+ [- [0-9]+] * ~ !	<i>ex: 15 or 27-33</i>

3.4 Examples

The following is an example of how to represent three simple policies using the SPL.

Axes	{	layer(application, ipsec, swipe, ssh, ssl) encryption(des-cbc, 3des-cbc) authentication(kerberos, radius, password, fortezza, secureid, rsa/1024, hmac-sha1-96) integrity(md2, md4, md5, ripemd-128, ripemd160, sha-1, crc) access control(kerberos, radius, password, fortezza, secureid)
Mechanisms	{	• Policy1,high(0,1,~,5,~) Policy2,medium(1,0,~,2,~) Policy2,high(1,1,~,5,~) Policy3,low(0,~,2,~,~) and (1,0,~,~,~)

3.4.1 Example Policies

This shows three policies whereby:

- 1) an application can be used to do triple des and sha-1 for authentication and integrity,
- 2) ipsec can be used for all of the encryption modes listed and md5 or sha-1 for integrity
- 3) there can be password based authentication at the application layer with des encryption at the ipsec layer.

The “high”, “medium”, and “low” policies are for security levels. Note, as in “Policy2”, categories can be listed more than once. In general, it is not an error to list points or categories multiple times.

3.4.2 Example Contexts

The “Policy2,high” line means “ipsec 3des-cbc encryption using sha-1 integrity verification”.

The “Policy3” line mean “ipsec layer des-cbc encryption with application layer password based authentication”.

3.5 Implementation

The SPL grammar can be easily implemented in an interpreter. We wrote a simple syntax checking application, using the standard parser generator `JavaCC`. This application parses statements, and outputs messages indicating whether or not the statements are valid SPL statements. The source code for this parser is included in APPENDIX B: .

This application provides a basis for creating SPL parsers for policy and context evaluation and negotiation. To negotiate a context from a set of policies, a project initiator will input the policies into a SPL parser that builds a data structure to find the set intersection of the policies.

SPL statements will need to be stored by hosts and sent over the network in negotiation protocols. To minimize the size of these statements, the ASCII representation of the language can be compressed by any of the lossless compression algorithms in current use.

4. Conclusions

This document has presented a revised approach to a cryptographic context negotiation template specification, and an initial specification of a cryptographic policy language via the context template specification. This section outlines several of the steps that may be taken in enhancing the negotiation template and developing the negotiation protocol.

The SPL syntax can be enhanced in several dimensions. Aspects of communications, environments, and organizations may be added to the specifications. For example, connectivity among subgroups of participants within an organization (e.g., protected private domains within physically secure and electronically isolated buildings) may affect the cryptographic context negotiated. While all combinations of such factors cannot be articulated in a proposed syntax, examples of how such extensions can be made are planned for the next document. Precedence of policies or priorities of policy negotiators can be added. Levels of security at various granularities can be added.

Acknowledgments

Mr. Jay W. Turner worked with the Cryptographic Technologies Group as a guest researcher from NSA during the period May 3 through August 29, 1998. We thank Mr. Turner for his significant contributions to the DCCM project and to this report. He has played a major role in the progress of this project during his tenure at TIS Labs.

Dr. Dennis K. Branstad managed the Cryptographic Technologies Group until Trusted Information Systems, Inc. was acquired by Network Associates, Inc. Subsequently, he has joined the group as a part-time employee to assist in preparing reports and technical research proposals. We thank Dr. Branstad for his initiating the DCCM project and subsequent contributions to both the overall direction and technical foundations of the project. He has played a major role in defining the architecture and components of the DCCM system.

Mr. Peter Dinsmore joined the Cryptographic Technologies Group in TIS Labs and the DCCM project in February, 1999. We thank Mr. Dinsmore for helping to edit and prepare this revision of Report Two.

References

- [Backus59] J. W. Backus, *The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference*, Proceedings of the International Conference on Information Processing, UNESCO, 1959, p. 125-123.
- [Bal98a] Balenson, David M., Dennis K. Branstad, David A. McGrew, Alan T. Sherman, *Dynamic Cryptographic Context Management (DCCM): Report 1: Architecture and System Design*, TIS Labs at Network Associates, Inc., TISR #0709, June 2, 1998.
- [Bal98b] Balenson, David M., Dennis K. Branstad, David A McGrew, Jay W. Turner, *Dynamic Cryptographic Context Management (DCCM): Report 2: Cryptographic Context Negotiation Template*, TIS Labs at Network Associates, Inc., TISR #0745, September 18, 1998.
- [Huf52] Huffman, D. A. A method for the Construction of Minimum Redundancy Codes, *Proc. IRE* 40 (1952), p. 1098-1101.
- [Iso88] International Organization for Standardization (ISO), *Information Processing Systems - Open Systems Interconnection Reference Model - Security Architecture*, ISO 7498/2 (1988).
- [Naur63] P. Naur, *Revised report on the algorithmic language ALGOL 60*, CACM 6, 1963, p. 1-17.

APPENDIX A: General System Security Policy

The body of this document addresses cryptographic-based security policy as a subset of an overall system security policy. This appendix addresses the broader goals of prevention, detection, and response of a general system security policy. This information is presented for a reader unfamiliar with the broad topic of computer security and the high-level system security policy needed to be formalized before any security system such as that provided in DCCM is considered.

A.1 Prevention

Organizations typically focus much time and energy on preventing attacks, failures, and incidents that undermine a system's unauthorized disclosure, modification and use protection. These preventive measures fall into the following broad categories.

Screening Procedures. These rules include actions commonly implemented in an organization's hiring process. Before an individual is hired, a company may request that an extensive background check be conducted on that individual. As part of their job, many new employees are authorized to use valuable company resources including automated information systems. The background check seeks to measure the prospective employee's "trustworthiness." In addition to confidence in future actions of its employees, a company seeks similar confidence with respect to its computing and network resources. Hence, prior to installation, computer software and hardware may undergo extensive acceptance testing. For instance, formal verification of specified functionality may be performed and rigorous virus scanning might be enacted on an initial and continual basis.

Limited Physical Access. Physical access control imposes human access barriers through which only authorized individuals may pass. A barrier may be some simple distinguishing personal characteristic such as a uniform to distinguish that the wearer is a member of "the club." More rigorous identification and authentication (I&A) methods might include bio-metric identification and control procedures or electronic verification using an employee badge. In addition, identification can be predicated on what someone knows. Passwords, including one-time passwords and challenge/response systems, are knowledge-based I&A mechanisms.

Other examples of physical barriers are security guards who rely on visual identification and locking mechanisms which require knowledge of a combination and or possession of a key for entry. Analogously in the electronic domain, organizations employ firewalls, guards and application gateways in order to limit access. Access control lists (ACLs) are also a frequently used mechanism within an operating system.

Access control based on privileges is a time-honored tradition, especially when performed by military organizations or governments. Generally speaking, it is customary for organizations to identify sensitive classes of activities in a system and to grant only a small portion of the user population permission to conduct them. Enforcement relies on I&A and then verifying individual authorization as established by some senior official in some database.

Education and Awareness. Education and training are vital components of a comprehensive security policy. User community awareness of their individual and group responsibilities for security helps to thwart potential attacks. Periodic training in security should be practiced in addition to initial security indoctrination. Displaying banners, posting warnings, and requiring regular training sessions reinforces an awareness of the security goals of system management.

There is also an element of detection in this category as well as prevention. Training includes more than learning techniques that may prevent attacks and acquiring knowledge of actions that might place the system in a vulnerable state. It also includes gaining an awareness of responsibilities in the event of a break-in and procedures for reporting incidents and violations.

Technical security services. This category includes features designed to protect information resident in a system or network. Common services are **authentication, confidentiality, integrity, access control, and non-repudiation**. Assuring the availability of information when and where needed is usually included in security policies and requirements. Policy provisions must also identify other services that must be included in the infrastructure in order to support these mainline security services.

- ***Access control*** means providing appropriate assurances that only specifically authorized entities (people, processes, network nodes) can electronically acquire information. This includes some of the same methods described earlier, which are based on the verifiable binding of the entity requesting access to permission for access.
- By ***authentication*** we mean a verification of the source of information. Authentication methods are usually based on binding the identity of an individual entity (could be a group of entities) to information known only by that entity or performable only by that entity. Digital signatures and keyed hash functions are common authentication mechanisms.
- ***Availability*** focuses on the capacity of network resources to support access and dissemination of information as required by authorized entities. It is perhaps the most demanding of all network security services; being vulnerable to both natural and human actions. Most attempts to make information available to the user community focus on redundancy and back up capabilities.

- **Confidentiality** refers to securing information from unauthorized disclosure. Usually we break this into data separation techniques and data protection techniques. The former includes preventing an adversary from acquiring the data or possibly even knowing of its existence. The latter covers techniques acting directly on the data that attempt to mask its content. An example of a data separation technique is labeling and routing control, where data labeled classified, for instance, is prevented from reaching undesirable areas, such as an unclassified LAN. The most widely used data protection technique is encryption. Other schemes, which have gained some publicity recently, include spread spectrum communications and steganography.
- **Integrity** refers to protection of information from unauthorized modification, where modification includes destruction, partial or total data replacement, and replay. Appending checksums or cryptographic authentication codes to data is a common practice to verify the integrity of data within a transmission or stored on some media. Applying a one-way function to a message, most commonly performed by hashing algorithms, is the most widely used forms of checksums. Keyed hashes, while addressing authentication concerns, also fit into this category. Cyclic redundancy checks (CRCs) are also quite common. Another technique to ensure some integrity is to run the data through an error detection and correction algorithm.
- By **non-repudiation** we mean preventing a transmitter from repudiating that it transmitted specific data or preventing a receiver from repudiating that it received the data. A common strategy is to require the implementation of an integrity mechanism along with the signature of the transmitter and that of the recipient. This process often requires the participation of a trusted third party (TTP) to assure that neither of the two principles can deny that the transaction was conducted.

The primary services supporting these critical security services are **key management** and **certificate management**. As mentioned previously, non-repudiation may require a trusted third party for adjudication reasons. Depending on their implementation, these support services may also require a TTP to act as judge, controller, or an impartial observer.

- **Certificate management** pertains to those security services that make use of a structured set of credentials instantiated in the form of a certificate. Access control, authentication, digital signatures, and non-repudiation often use such certificates. In cases where public key cryptography has been chosen for encryption purposes, certificate services may also apply to confidentiality. It is common practice in certificate infrastructures to use an agreed upon TTP as a signer and issuer of certificates. This attaches validity to the certificates. Certificate lifetimes and revocation in the event of compromises and violations of trust are also policy issues.

- **Key management** supports essentially all services except for availability. Policy issues focus on the generation of keys, the distribution of keys, key exchange methods, key storage, and the recovery of data or keys through key management techniques. Additional issues focus on how and when to re-key. They include key updating procedures, setting key periods, compromise recovery techniques, and forward and backward secrecy concerns.

A.2 Detection

A sound security policy goes well beyond protective measures. In spite of the most thorough protection against attack, breaches of security almost surely will occur. An organization should therefore employ detection mechanisms as well as prevention mechanisms. Effective detection mechanisms often serve to deter intentional attacks against a system. Actions designed to uncover security breaches generally fall into three categories.

Testing. Exercising safeguards and protective measures in order to: 1) verify their correctness and viability, and 2) diagnose inappropriate states is a common practice. Testing frequency is an important issue. Ideally, an organization should institute thorough testing on a periodic basis while at times invoking an element of surprise by performing an unplanned test or drill. Intrusion detection schemes exemplify regular testing, to the point of being a constant search for unauthorized invasions into the system.

Alarms. Incorporating alarms that react to detected violations is a common practice in secure system design. Alarm mechanisms should address both insider and outsider threats as well as inadvertent actions. Alarms should be tested periodically. Alarms may be “silent” so that the intruder is not warned that an activity is being monitored.

Audit. Policy in this area focuses on institution of metrics to measure deviations from accepted norms, tracking system activities, and monitoring user-initiated actions in the system. These procedures often catalogue measurements and activities. Again, auditing procedures should be testable features.

A.3 Response

While the goal is to anticipate intrusions, attacks and violations, it is difficult to put appropriate response mechanisms in place before such events do occur. Generally speaking, the response to detected or suspected incidents focuses on improving preventive measures. Responding actions fall into three categories.

Change Settings. Diagnoses of the incident should be initiated immediately and all uncovered vulnerabilities should be minimized by appropriate changes. Organizations should adopt a policy of immediate but prudent reaction. For example, compromised keys must get changed immediately, security holes must get plugged (even temporarily with

non-automated solutions until automated solutions can be implemented). A policy should be that security violators get evicted from system authorization immediately and that prosecution will be pursued.

Conduct Analysis. Following the “quick reaction” changes, organizations should initiate a concerted effort to analyze and evaluate the incident including the state and behavior of the system leading up to the incident. Ideally this process should include the organization’s initial response (i. e. the change settings phase) to the security breach.

Update Procedures. This is the lessons learned phase, where policy is perhaps radically changed to improve the security robustness of the system. A security analysis often calls for new procedures to be put into place and a strengthening of existing procedures.

A.4 Group Policy

Group policy establishes procedures for group management and group keying. Group management refers to those actions that create or dissolve groups and that modify group membership. Group keying covers strategies for generating, distributing, and managing group communication keys. Group keying policy also addresses subgroup keying and whether or not to publicly announce subsequent application sessions.

A.4.1 Group Dynamics

Policy of group dynamics and management are driven by what actions should be permitted regarding the formation of groups and modifications to their membership roles. Who is authorized to initiate these actions should also be addressed.

Common actions that are performed on groups include creation and dissolution of a group, merging groups to form larger groups, splitting groups into subgroups, adding members to groups and deleting members from groups. These group modifications can be accomplished with a few primitive commands.

- **Initialize.** This command creates the group and registers its name and characteristics. The extent of those characteristics (for example, maximum size, intended length of service, level of classification and so forth) is a matter of policy. Other policy issues include group leadership. For instance, do formal managerial/supervisory roles exist and if so, how many leaders will there be? Authorization is yet another issue. Which roles and processes are permitted to issue a group initialize command should be established in policy.
- **Add.** This action adds an entity to a group, where the entity can be either an individual member or a formal group. Again, policy issues abound. The issue of whom or what role or process is authorized to initiate this action is important. If the entity is a group, how must its characteristics compare to those of the parent group? Does the subgroup to be subsumed retain its identity and structure?

- **Delete.** This command allows for removal of members from the group as well as secession of subgroups from the parent group. This action, of course, carries with it similar conventions to those of the add command.
- **Dissolve.** This is the group termination command. In addition to determining which roles and processes can terminate groups, policy should also address historical issues. Should the group's existence be recorded and if so, what are the pertinent characteristics and actions to record? Should the system be concerned with re-creation?

Group topology may also be a key issue relative to these actions. One might ask, should participants with similar characteristics be closely tied to one another within the group? Among other things, the answer depends on the functions to be conducted by that group and whether or not such internal subgroup structure facilitates operational economies pursuant to those functions.

A.4.2 Group Keying

Choosing a Scheme. Numerous schemes for establishing communication keys for groups have been proposed in the literature. Which scheme or schemes are invoked for a particular system is a policy matter. Determining an optimal group-keying scheme can be difficult. Clearly, group size is a dominant factor. Communications characteristics also weigh heavily in the decision. Each group-keying scheme possesses a unique set of characteristics that are interdependent with the group management actions defined above. For instance, certain schemes induce a simple linear ordering on the set of group members, while others overlay a hierarchical topology onto the membership. Some require synchrony of communications between members during the manufacture of the group traffic key. Hence, choosing the keying scheme requires that its specific characteristics be tailored for consistency with system properties and other policy settings.

Key Management. Group key generation, distribution, and efficient management are the critical issues. However, there are a few characteristics of key management that are unique to group keys.

- **Key Generation.** Some schemes have a definitive process by which the traffic key is produced, while others are less well defined and have a few options. Some require the creation of point-to-point keys for private communications between the member and group authority leading up the creation of the traffic key. The size of the traffic key is obviously dictated by the mechanisms that have been negotiated to deliver confidentiality. Key periods and compromises of key must also be considered.

- **Key Distribution.** Depending on the scheme, group members may be required to have certain intermediary keys in their possession. Securing those distributions with point-to-point or other special keys is a policy consideration.

- **Data Protection.** Forward and backward secrecy issues are still of concern. Previously transmitted traffic, which might now be stored data, along with future communications should be considered in the policy.

Existence Issues. Organizations may wish to mask the mere existence of certain groups. While it may be difficult to cover all the tracks, some aspects can be handled by keying policies. For instance, confidentiality of session announcements could be accomplished by establishing a group at the next higher level, generating a key for that group and then encrypting the ensuing announcements with the higher-level group key.

APPENDIX B: JavaCC File for SPL BNF

The following code parses the SPL and loads a "MechanismSpace" class with axes and points along those axes.

```

/*
 * SPL
 */

options {
//DEBUG_TOKEN_MANAGER=true;
IGNORE_CASE=true;
LOOKAHEAD=1;
}

PARSER_BEGIN(Spl)

package com.nai.dccm;
import java.util.*;

/**
 * Parse the Security Policy Language
 */
public class Spl
{
    public static void main(String args[]) throws ParseException
    {
        Spl parser = new Spl(System.in);
        MechanismSpace mechanismSpace = new MechanismSpace();
        while (true)
        {
            System.out.print("Enter Expression: ");
            System.out.flush();
            try
            {
                switch (parser.policy(mechanismSpace))
                {
                    case -1:
                        mechanismSpace.barf();
                        System.exit(0);
                    default:
                        break;
                }
            }
            catch (ParseException x)
            {
                System.out.println("Exiting.");
                throw x;
            }
        }
    }
} // class Spl

PARSER_END(Spl)

TOKEN_MGR_DECLS :
{
    static int pointParenLevels;
}

<DEFAULT> SKIP : { " " | "\t" | "\r" | "\n" }
<DoneAxisLabel> SKIP : { " " | "\t" | "\r" | "\n" }
<WithinAxis> SKIP : { " " | "\t" | "\r" | "\n" }

```

```

<WithinPointList> SKIP : { " " | "\t" | "\r" | "\n" }
<WithinPoint>        SKIP : { " " | "\t" | "\r" | "\n" }

<DEFAULT> TOKEN :
{
  < AXISLABELSTARTQUOTE : "\""> : WithinAxisLabelQuote
  | < STARTAXIS: "(" > : WithinAxis
  | < DOT: "." > : WithinPointList
  | < AXISLABEL: ~[" ", "\t", "\n", "\r", "\"", "(", "."] ( ~["("] )* > :
DoneAxisLabel
} // <DEFAULT>

<DoneAxisLabel> TOKEN :
{
  < STARTAXISAFTERLABEL: "(" > : WithinAxis
} // <DoneAxisLabel>

<WithinAxisLabelQuote> TOKEN :
{
  <AXISLABELENDQUOTE: "\"" > : DoneAxisLabel
  | <AXISLABELTEXT:
    ( (~["\\", "\\\"] )
      | ("\\")
        ( ~["0"-7]
          | ["0"-7] ( ["0"-7] )?
          | ["0"-3] ["0"-7] ["0"-7]
        )
      )
    )+
  >
} // <WithinAxisLabel>

<WithinAxis> TOKEN :
{
  <POINTLABELSTARTQUOTE : "\""> : WithinPointLabelQuote
  | <POINTLABELSEP: ", " >
  | <STOPAXIS: ")" > : DEFAULT
  | <POINTLABEL : ~[" ", "\t", "\r", "\n", "\"", ",", "(" ,")" ] ( ~[")", ",", "]
)* >
} // <WithinAxis>

<WithinPointLabelQuote> TOKEN :
{
  <POINTLABELENDQUOTE: "\"" > : WithinAxis
  | <POINTLABELTEXT:
    ( (~["\\", "\\\"] )
      | ("\\")
        ( ~["0"-7]
          | ["0"-7] ( ["0"-7] )?
          | ["0"-3] ["0"-7] ["0"-7]
        )
      )
    )+
  >
} // <WithinPointLabelQuote>

<WithinPointList> TOKEN :
{
  < CATEGORYSTARTQUOTE : "\""> : WithinCategoryQuote
  | < CATEGORYLABEL: ~[" ", "\t", "\n", "\r", "\"", "(", ", "] ( ~["(", ", "] )* >
: DoneAxisLabel
  | < CATEGORYSEP: ", " >
  | < STARTPOINT: "(" > { pointParenLevels = 0; } : WithinPoint
  | < AND: "and" ( [" ", "\t", "\n", "\r" ] )* "(" > { pointParenLevels = 0; } :
WithinPoint
} // <WithinPointList>

```

```

<WithinCategoryQuote> TOKEN :
{
  <CATEGORYENDQUOTE: "\" > : WithinPointList
|  <CATEGORYTEXT:
    ( [ " ", "\t", "\r", "\n" ] )*
    ( (~[ "\" , "\\ " ] )
      | ( "\\ "
          ( ~[ "0"- "7" ]
            | [ "0"- "7" ] ( [ "0"- "7" ] )?
            | [ "0"- "3" ] [ "0"- "7" ] [ "0"- "7" ]
          )
        )
      )+
    )
} // <WithinCategoryQuote>

<WithinPoint> TOKEN :
{
  < STARTPOINTPAREN: "(" > { pointParenLevels++; }
|  < STAR: "*" >
|  < POINTSEP: "," >
|  < DASH: "-" >
|  < INTEGER: ( [ "0" - "9" ] )+ >
|  < STOPPOINT: ")" > { if(pointParenLevels > 0) pointParenLevels--; else
SwitchTo(WithinPointList); }
} // <WithinPoint>

int policy(MechanismSpace mechanismSpace) :
{
}
{
  axes(mechanismSpace) <DOT> point_list()
  { return 1; }
| <EOF>
  { return -1; }
} // policy

void axes(MechanismSpace mechanismSpace) :
{
}
{
  ( axis(mechanismSpace) )+
} // axes

void axis(MechanismSpace mechanismSpace) :
{
  String label;
  MechanismSpace.Axis axis;
}
{
  ( ( <AXISLABEL>
      {
        label = token.image.trim();
      }
    <STARTAXISAFTERLABEL> )
  | ( <AXISLABELSTARTQUOTE>
    <AXISLABELTEXT>
      {
        label = token.image.trim();
      }
    <AXISLABELENDQUOTE>
    <STARTAXISAFTERLABEL> )
  )
  { axis = mechanismSpace.axis(label); }
  point_label_list(axis)
  <STOPAXIS>
} // axis

//

```

```

// Comma deliniated point labels. Ends at ")" (end of axis)
//
void point_label_list(MechanismSpace.Axis axis) :
{
}
{
    point_label(axis) ( <POINTLABELSEP> point_label(axis) )*
} // point_label_list

//
// Point label ends at comma (maybe quoted if it needs to include a comma)
//
void point_label(MechanismSpace.Axis axis) :
{
}
{
    (
        <POINTLABEL>
        { axis.point(token.image.trim()); }
    |
    <POINTLABELSTARTQUOTE>
    <POINTLABELTEXT>
    { axis.point(token.image.trim()); }
    <POINTLABLENDQUOTE>
    )
} // point_label

//
// The second half of the Security Policy Language text is the list of points
// in the mechanism space
//
// Each point can belong to multiple categories
//
void point_list() :
{
    Vector categories = new Vector();
}
{
    ( [ categories = category() ] <STARTPOINT> point() ( <AND> point() )* )*
} // point_list

//
// Example: "category1",category2,category3
//
Vector category() :
{
    String categoryText;
    Vector categories = new Vector();
}
{
    categoryText = category_text()
    {
        categories.addElement(categoryText);
    }
    (
        <CATEGORYSEP>
        categoryText = category_text()
        {
            categories.addElement(categoryText);
        }
    )*
    {
        return categories;
    }
} // category

//
// Category text may or may not have surrounding quotes
//
String category_text() :
{

```

```

    String text;
}
{
    (
        <CATEGORYLABEL>
        {
            text = token.image.trim();
        }
    |
        <CATEGORYSTARTQUOTE>
        <CATEGORYTEXT>
        {
            text = token.image.trim();
        }
        <CATEGORYENDQUOTE>
    )
    {
        return text;
    }
} // category_text

//
// A point is a comma separated list of number ranges
// (ranges are just syntatic sugar for a series of points)
//
void point() :
{
}
{
    range() ( <POINTSEP> range())* <STOPPOINT>
} // point

//
// A range is a single integer, a pair of integers with a dash
// between them (to indicated both integers and all values between them),
// or a parenthesis enclosed comma separated list of ranges. (Note the
// recursive definition.)
//
void range() :
{
}
{
    LOOKAHEAD(2)
    simple_range()
    | <STARTPOINTPAREN> range() ( <POINTSEP> range() )* <STOPPOINT>
} // range

//
// an integer or pair of integers separated by a dash.
//
void simple_range() :
{
}
{
    <INTEGER> [ <DASH> <INTEGER> ]
} // simple_range

```