

AD-A099 206

MITRE CORP BEDFORD MA F/G 9/2  
A SECURE MESSAGE SYSTEM FOR THE SECURE DISTRIBUTED PROCESSING P---ETC(U)  
MAR 81 P T WITHINGTON, D J SOLOMON F19628-80-C-0001  
NTR-3955 RADG-TR-81-18 NL

UNCLASSIFIED

1 of 1  
AD A  
000 206


END
DATE FILMED
6-81
DTIC

**LEVEL II** (12)

fw

**RADC-TR-81-18**  
Final Technical Report  
March 1981



# **A SECURE MESSAGE SYSTEM FOR THE SECURE DISTRIBUTED PROCESSING PROJECT**

The MITRE Corporation

P. Tucker Withington  
Daniel J. Solomon

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

**SECRET**  
MAY 21 1981  
**A**

**ROME AIR DEVELOPMENT CENTER**  
Air Force Systems Command  
Griffiss Air Force Base, New York 13441

81 5 21 024

AD A099206

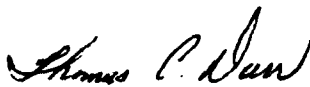
DTIC FILE COPY

X

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

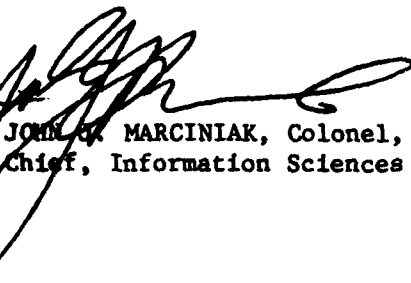
RADC-TR-81-18 has been reviewed and is approved for publication.

APPROVED:



THOMAS C. DARR, Major, USAF  
Project Engineer

APPROVED:



JOHN M. MARCINIAK, Colonel, USAF  
Chief, Information Sciences Division

FOR THE COMMANDER:



JOHN P. HUSS  
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISCP) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

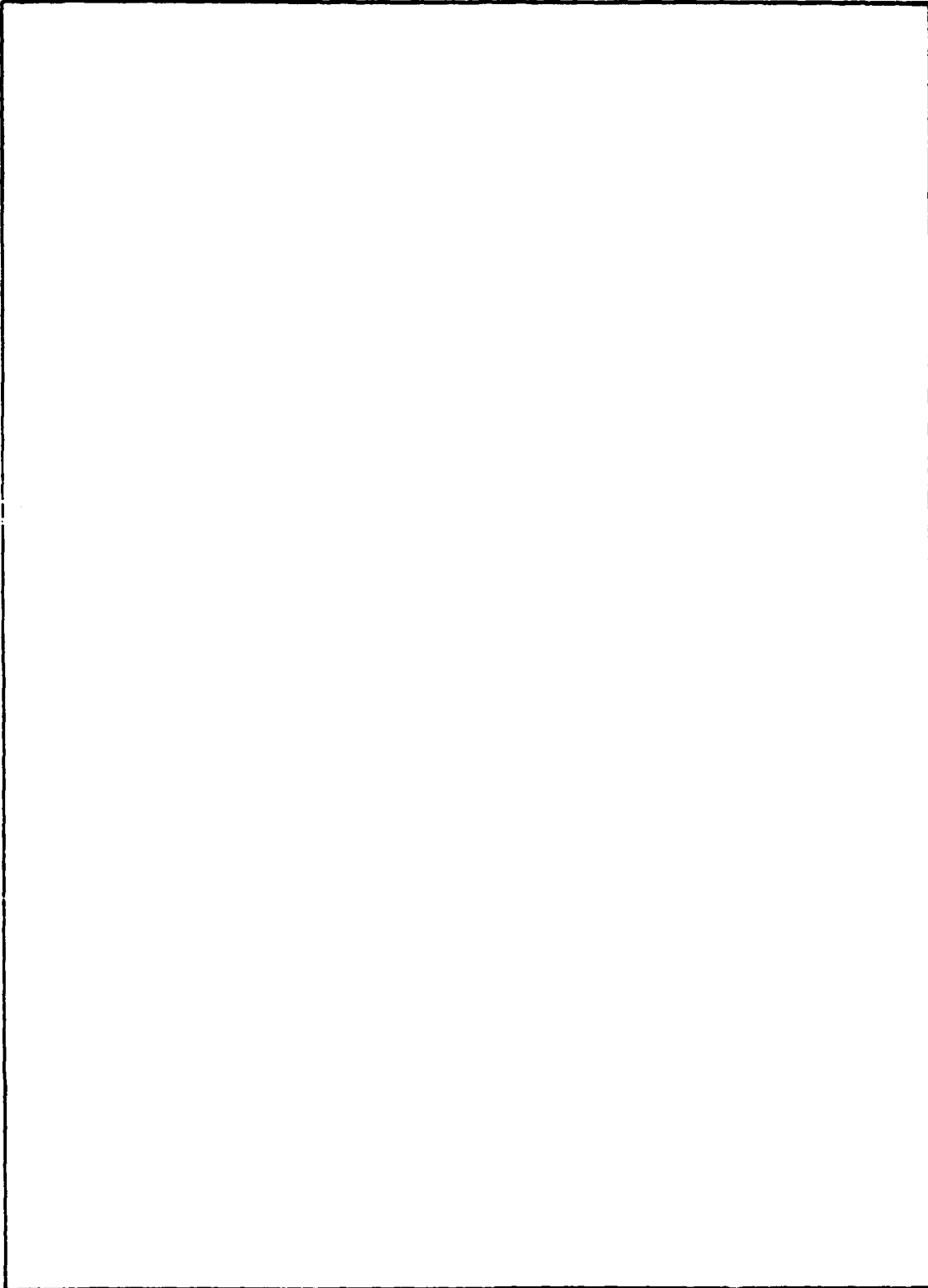
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(19) REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER RADC-TR-81-18 ✓	2. GOVT ACCESSION NO. AD-A099206	3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) A SECURE MESSAGE SYSTEM FOR THE SECURE DISTRIBUTED PROCESSING PROJECT.	(9)	5. TYPE OF REPORT & PERIOD COVERED Final Technical Report, 1 Oct 79 - 30 Sep 80	
7. AUTHOR(s) P. Tucker/Withington Daniel J. Solomon	(14)	6. PERFORMING ORG. REPORT NUMBER MTR-3955	
9. PERFORMING ORGANIZATION NAME AND ADDRESS The MITRE Corporation Bedford MA 01730	(15)	8. CONTRACT OR GRANT NUMBER(s) F19628-80-C-0001 ✓	
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (ISCP) Griffiss AFB NY 13441	(11)	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS MITRE MOIE 8100 0005	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same	(12) 311	12. REPORT DATE Mar 1981	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		13. NUMBER OF PAGES 28	
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED	
18. SUPPLEMENTARY NOTES RADC Project Engineer: Major Thomas C. Darr (ISCP)		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Computer Security            Local Network Security Kernel                UNIX KSOS                                MITRENET Message Processing			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report documents work performed under a MITRE MOIE project. The report describes the design of a multilevel-secure message system based on the RAND MH message system running under the UNIX operating system. A color terminal is used for control and display of security classifications. Major issues in the user interface, and in interfacing the message system with KSOS are discussed.			

255 484

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

ABSTRACT

The purpose of the Secure Distributed Processing project is to investigate the construction and use of a network of kernelized secure processors. A review of the experimental hardware base is given, and a design for a potential application is presented.

Distribution For	
General	<input checked="" type="checkbox"/>
TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Classification	
Distribution/Availability Codes	
Avail and/or Special	
A	

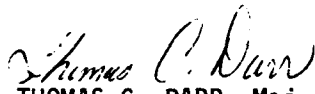
## EVALUATION

Over the last 10 years, the DOD has been sponsoring research and development in Computer Security Technology. The major emphasis of this research has been the development of general purpose, multi-user operating systems whose internal controls are correct, reliable and unsubvertable. The technology for implementing such systems has been termed "security kernel technology". Several prototype operating systems based on this technology are now being introduced into DOD applications.

The engineering of applications upon kernel-based trusted operating systems is a major thrust of RADC's computer security program. The interface of the user to the application, and to the kernel involves issues of security, functionality, and performance tradeoffs.

The MITRE Corp, under a Mission Oriented Investigation and Experimentation (MOIE) project, designed and prototyped a multilevel-secure message processing system for potential integration with a trusted version of the UNIX operating system. A unique aspect of the system was the use of a multicolor terminal to display security levels of information.

The results of their work are documented in this report, and will serve as a basis for future programs which implement multilevel applications on kernel-based operating systems.

  
THOMAS C. DARR, Maj, USAF  
Project Engineer

## TABLE OF CONTENTS

<u>Section</u>		<u>Page</u>
	LIST OF ILLUSTRATIONS	vii
1	INTRODUCTION	1
	OBJECTIVE	1
	BACKGROUND	1
	Prototype	2
	FOCUS	2
	Kernel-Network	2
	Application-Kernel	2
	User-Application	3
	OVERVIEW	3
	Synopsis	4
2	INITIAL CHOICES	5
	THE RAND MESSAGE HANDLER	5
	WINDOWS	6
	COLOR	6
	MESSAGE STRUCTURE	7
	THEME	8
	Additional Benefits	9
3	DESIGN PARTICULARS	10
	WINDOWS AND PANES	10
	MH OVERVIEW	10
	COMMAND DESCRIPTIONS	11
	SHOW	11
	SCAN	12
	COMPOSE	12
	SELECT	13
	IMPLEMENTATION	14



TABLE OF CONTENTS (Concluded)

		<u>Page</u>
4	TRUSTED TERMINAL MANAGER	15
	MULTI-LEVEL OBJECTS	15
	Problems	15
	PROTECTED NAMING	16
	MEMORYLESS PROCESSES	17
	TRUSTED FUNCTIONS	17
	DESIGN	18
	PROGNOSIS	19
5	ISSUES	20
	COMMAND ISSUES	20
	Compose	20
	Select	20
	GRANULARITY MISMATCH	21
	UNFORESEEN ISSUES	21
6	CONCLUSION	23
	REFERENCES	25
	DISTRIBUTION LIST	27

LIST OF ILLUSTRATIONS

<u>Figure Number</u>		<u>Page</u>
1	Overview of System Layers	8
2	Example of the SHOW Command	11
3	Example of the SCAN Command	12
4	Example of the COMPOSE Command	13
5	Extracting Viewed Data While Composing	14
6	Trusted Terminal Manager	18

## SECTION 1

### INTRODUCTION

The Secure Distributed Processing Project was formed to investigate the problems associated with, and the benefits deriving from, interconnecting a number of kernelized, single-location operating systems over a secured network. The method chosen for exploration was to design and implement a potential application on an experimental hardware base. Because the hardware base represents the theme of the project, the project has become known colloquially as KNet (kay-net) for Kernel Network.

#### OBJECTIVE

The objective of KNet is to integrate the technologies of Computer Security, Communications Security, and Distributed Processing or Networking. This integration is seen as the next logical step for providing secure processing power in an efficient and reliable manner for military and other sensitive data management tasks.

KNet represents one of the few existing projects to investigate this complex area. The approach taken by the KNet project has been to investigate the requirements of a Secure Distributed Message System as a potential use for the integrated technologies.

In this paper we review the status of the hardware and propose a design for a message system as an application.

#### BACKGROUND

The initial effort in KNet was to set up a two-node network and implement a rudimentary message handling capability on it. The use of a color terminal as an interface to the application system was investigated and a demonstration system developed.

The initial configuration of the KNet network consisted of a PDP-11/45 connected to a PDP-11/70 using the MITRE Cable Bus Interface Units (BIU's) [1]. Each PDP-11 has an Intecolor color terminal to interface to the message system [2]. The PDP-11's run the Unix†

---

†Unix is a trade/service mark of Bell Laboratories.

operating system [3]. The message system used is the MH (Message Handler) system of RAND [4]. Network communication is supported by the Transmission Control Protocol (TCP) [5].

The major effort required to support the initial demonstration was to become familiar with the various hardware and software elements, to integrate them, and finally debug them. The initial demonstration used only unsecured versions of all software and was, essentially, to prove that the hardware configuration could work.

### Prototype

The proposed hardware base for the prototype KNet effort is two kernelized machines networked using upgraded BIU's. The kernelized machines will run the Kernelized Secure Operating System — KSOS [6]. In addition, an unclassified machine is to be available on the network to provide a gateway to the ArpaNet. Presently, the delivery of the kernel software and the unclassified ArpaNet connection has been delayed. Fortunately, our work is not significantly impacted by these delays because a non-secure version of KSOS (the Unix operating system) is available for work to proceed on.

### FOCUS

Having successfully established a working hardware base, the KNet project moved on to focus on the issues of secure, distributed processing. The issues have a common point: interfacing independent security domains.

### Kernel-Network

At the lowest level, the network represents an interface between the separate security domains of independent kernelized operating systems. Work of a theoretical nature has been done in a related project to investigate the problems of supporting this interface [7]. The main conclusion of this work was that a solution to the problem centers around providing a way to communicate security information from one domain to another in a trustworthy manner.

The KNet project has not investigated this lowest level interface, but rather has been concerned with two others.

### Application-Kernel

A security kernel provides a stark interface for implementing the complex functions of a distributed data management system. An important issue that KNet would like to investigate is the

reasonability of the kernel interface for the task chosen. This investigation has been delayed pending the receipt of the kernel software.

### User-Application

The last security domain interface in the KNet investigation is the interface between the user and the application, embodied in the terminal. It is this interface that is presently being addressed by KNet research and is the primary subject of this paper.

While the user interface has been dealt with somewhat in previous kernel designs, KNet represents one of the first efforts to implement a secure transaction-like system and can thus present significant new problems. It is our feeling that the correct design of this user interface can lead to discovery and display of some of the more subtle security issues hidden in lower layers of security design. It is for this reason that the terminal interface design was chosen to be addressed first in the KNet project.

### OVERVIEW

The design of the terminal interface is obviously directed to a large degree by the message system it is to interface. Three simplifying choices have been made that, to a large extent, dictate many other decisions.

First, we have decided to use the KSOS-provided unit of protection (files and processes). This decision was made to speed the initial investigation. At the same time, it does not preclude extrapolation of results to other possibilities.

Second, we have assumed many of the functions provided by current message systems are appropriate to secure usage. While this decision allows us to use an existing message system design, it may lead to complications if security cannot be easily imposed upon the existing design.

Our third assumption is that color is an appropriate output medium for communicating security information to the user. This assumption may be the most risky, but is made reasonable by monetary considerations. It is our belief that many of the conclusions derived from our interface investigations could apply to other choices of output media.

## Synopsis

The remainder of the paper elaborates on the user interface and then discusses our proposal for a secure terminal, based on a trusted terminal manager that can interact with processes at several levels to provide a multiply-classified display. The terminal manager represents a significant technological effort as a first attempt to support "multi-level objects" outside of a security kernel.

## SECTION 2

### INITIAL CHOICES

#### THE RAND MESSAGE HANDLER

Since the ultimate goal in this investigation is to isolate and clarify security issues in an interactive environment (such as a message system), we have chosen to use existing tools for as much of the system as possible. Given the computing facilities of a PDP 11/45 and the Unix<sup>6</sup> operating system (of which KSOS is a kernelized secure version), it was necessary to choose a system that would be suitable to modification and experimentation. The Rand Message Handler (MH) has several features in its favor. It is designed to utilize the characteristics of the Unix<sup>6</sup> operating system (hierarchical file structure, and multiple small processes), and possibly reduces the protection granularity problem that other message systems encountered with implementing security controls.

The protection granularity problem concerns the environment provided by the kernel for processing multilevel information. Although the kernel does provide multilevel data objects (in the form of file directories), processes are restricted to run at a single protection level. This necessitates using multiple process communication techniques to coordinate multilevel tasks. Since many message system commands are multilevel in nature (e.g., scanning a folder of messages), [8] a simple task for an insecure message system can turn into a complicated exercise in interprocess communication.

MH is potentially well suited to this multiple process environment since each MH command is a separate (and small) process. In Unix<sup>6</sup> it is commonplace to have a process create sub-processes to do concurrent tasks. If a single command is to provide a multilevel interface, a separate instance of the command might be created to process the information at each different protection level. This technique would maintain conformity with the KSOS restriction of a process operating at a single level, yet it can appear to the user that multiple levels of information are being processed by a single command. The coordination of these sub-processes is discussed in greater detail in section 4.

## WINDOWS

A desirable feature for a message system, though not currently part of MH, is window partitioning of the user's terminal. Separating a physical terminal into several logical terminals allows a user to simultaneously perform multiple message functions. A typical use of this feature might be to view a received message while composing a reply.

The window design does not functionally alter the MH system, it simply permits the user to interact with several separate tasks at once, and is analogous to using a separate physical terminal per task.

## COLOR

An issue that warrants special attention in a message system interface design is the communication of security information to the user in a manner that is:

- (a) Easily noticeable;
- (b) Distinguishable from textual information;
- (c) Believable (kernel enforced).

The most effective previous solution to this issue that we have observed was the SIGMA system's use of colored lights attached to the display. [8] In practice, however, the lights proved to be inadequate in many regards. They are small and dim. They are distinct from, but physically too far from the textual information, requiring that security labels be written along with the text on the screen. And, it is unclear whether to believe the lights or the text labels, since there are occasions when they differ!

A common solution for marking information in general on a CRT is to use screen attributes (e.g., inverse video; blinking characters). The most flexible attribute currently available is color, which offers a wide variety of homogeneous, yet distinct ways to mark displayed data. Color is:

- (a) Very visible and in fact very hard to ignore (yet careful color choices can retain legibility);
- (b) Distinct from, but physically located with the text;



(c) Believable, because it can be controlled independently from text.

Using color, the granularity of the protection information can be as fine as a single character, although that much flexibility requires careful control in the interface design to preserve the user's sanity. Color divisions will in practice be limited to a small number of contiguous sections of the screen.

#### MESSAGE STRUCTURE

The structure of a message in many ways determines the organization and appearance of the functions that deal with them. The design decisions made here are in consideration of the available tools for implementation, and propagate the limitations of these tools.

The KSOS restriction of a file as the smallest protectable object discouraged us from considering features such as multilevel message text. This is unfortunate, since the marking of paragraphs at different security levels within a single message is essential in the military message environment. With these considerations, our design aims at attacking the class of issues that might arise with multiply classified text, but in a simplified fashion.

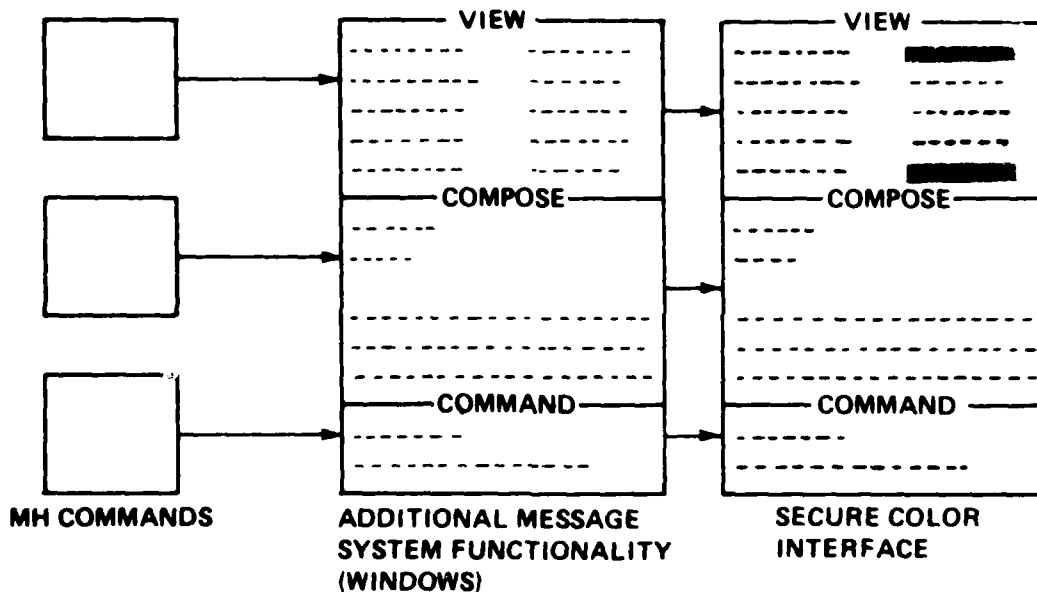
In our prototype design, a message consists of three classifiable parts, a "header", a "subject", and a "body".

- 1) The header contains origin, destination, and date-time information. It is required to be (and makes sense to be) unclassified.
- 2) The subject is the title of the message and may either be unclassified, or classified at the level of the message body.
- 3) The body contains the text of the message and may be classified at any single protection level.

It should be noted that definitions such as this message structure are the subject of experimentation, and as the system is developed will evolve accordingly.

**THEME**

The underlying idea throughout the design of this message system, is to maintain the integrity of each "layer" of the system by keeping them as independent as possible. Each layer is superimposed on the previous one to add its particular feature, while not imposing on the function or structure of the underlying layers (figure 1).



**Figure 1. OVERVIEW OF SYSTEM LAYERS**

The three layers in figure 1 are the Rand Message Handler, the window concept of terminal division, and color encoding to indicate protection level. There are several motives for choosing this "layered" approach. It minimizes the modifications required to adapt imported software (MH) in realizing the proposed design. In addition, tools developed at each layer are applicable to user utilities other than the message system. This should prove to be invaluable on a new operating system such as KSOS.

### Additional Benefits

By imposing security mechanisms as a filter on the message system, we get a security display mechanism for the entire Unix system as well. This filter concept is particularly effective in Unix where redirection of I/O is both commonplace and trivial. If the windowing mechanism can be implemented as a filter as well, it will share this system wide applicability.

In the next section we provide examples of specific message system command appearance and usage in a secure environment. The few commands described are of particular interest in that they exemplify many of the issues involved in the user-security interface.

## SECTION 3

### DESIGN PARTICULARS

#### WINDOWS AND PANES

We will define the terms "window" and "pane" as they are used in the remainder of the paper. A window is a screen partition organizing commands for simultaneous display on the terminal. Information displayed in a window may be scrolled within the window boundaries to a depth of several pages. There are three, possibly concurrent, windows:

**COMMAND WINDOW** — This is a window ranging in size from one to four lines located at the bottom of the screen, and is for the purpose of entering commands to Unix (MH commands are a subset of all Unix commands). Scrolling of this window could permit the editing and resubmission of previously entered commands.

**VIEW WINDOW** — The view window displays the results of commands entered in the command window. The screen area occupied by the view window attempts to balance the amount of information to be displayed with the requirements of other windows presently on the screen. This window is "read only", permitting information to be read and extracted by the user, but not modified.

**COMPOSE WINDOW** — The compose window is located between the view and command windows, and provides the user with a full screen editor for composing messages. When the compose window is created, the other windows present on the screen adjust in size to accommodate the new window.

These windows have boundaries that are visible to the user only with regard to scrolling. Cursor movement may freely cross window boundaries, placing the user actively in the process occupying (controlling) the new window.

Window "panes" are sections of windows at a single protection level (and thus of a single color). They are used in describing aspects of commands involving multilevel screen displays.

#### MH OVERVIEW

MH commands are used in a similar fashion to Unix commands — each MH command is an individual Unix program. Arguments to these

commands are references to messages or folders (of messages), or options to modify the actions of a command.

Each message is stored as a separate Unix<sup>®</sup> file, and is referred to by a file name assigned by MH. A "folder" is any collection of messages the user wishes to group together. One particularly useful (predefined) folder is "inbox" for unread messages. These folders are actually directories of the files (messages). Using the Unix<sup>®</sup> file system structure to organize messages greatly simplifies MH since manipulative tools already exist as part of the operating system.

The following section describes the commands that are of special interest to our security design.

#### COMMAND DESCRIPTIONS

##### SHOW

The SHOW command displays a message in the view window with the header in one window pane, and the body in another pane directly below the header (figure 2). The subject might be located in the header or the body pane depending on its classification.

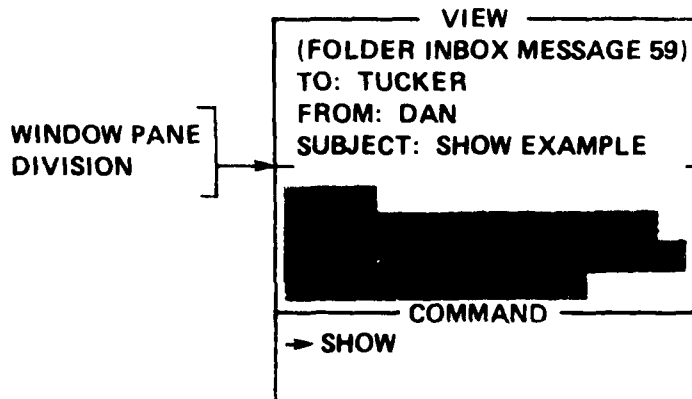


Figure 2. EXAMPLE OF THE SHOW COMMAND

## SCAN

The SCAN command displays the contents of a folder in order of entry. For each message in the folder, the header, subject, and as much of the body as can fit on one line are displayed in their correct colors. Undoubtedly the multilevel nature of this command will be the source of many security interface design issues.

VIEW			
#	DATE	FROM	SUBJECT    [ <<BODY ]
1	2-23	PTW	MONTHLY REPORT <<CAN YOU
5	2-23	BNSW	[REDACTED]
11	3-1	PTW	KNET MTR <<I HAVE HEARD F
		•	
		•	
		•	
		•	
COMMAND			
→ SCAN			

Figure 3. EXAMPLE OF THE SCAN COMMAND

## COMPOSE

COMPOSE is a command that requires special attention, since large amounts of information must be available to a user while composing. It is necessary to provide facilities for viewing old messages or folders, and conveniently extract information from these to include in the message being composed.

When a compose is initiated, a window will appear on the screen with a header template for the user to complete (figure 4). The body initially has an isolated, user private classification called "draft". The user may classify the message at any time during or after composition, but the message may not be sent or accessed by another user while it remains classified "draft". This feature allows the user to postpone classifying a draft while deliberating what the correct classification should be. If, however, the user includes already classified data in the draft, this action classifies the draft at the level of the data being copied.

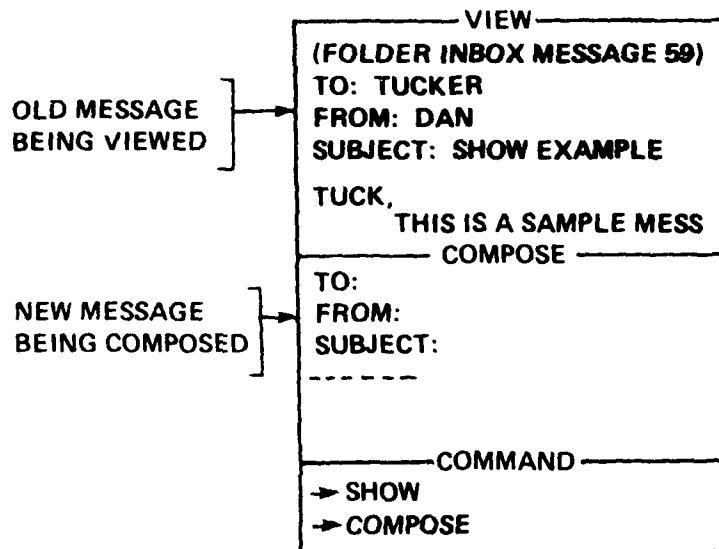


Figure 4. EXAMPLE OF THE COMPOSE COMMAND

The mechanism for obtaining parts of old messages to use in a draft, is to allow the user to access any information in the view window of the screen. The user moves the cursor from the compose window into the view window, and defines an area of the screen (by cursor movement) containing the desired information (figure 5). As soon as this area is defined, the information inside the area has a "mono-chromatic" filter applied to it, and becomes the color (and classification) of the draft message. This filtering is necessary, since we chose to have the message body single-level (this, and other implementation related problems are detailed in section 5).

Chromatically modifying the displayed data makes the user immediately aware that if the information is copied to the draft message, it will have the new classification (as indicated by its modified color). When the cursor is returned to the compose window, the view window resumes its original colors.

#### SELECT

The select command allows the user to create a new folder containing messages that meet specified criteria. The select criteria might be search keys in any of the header fields, or key words (possibly classified) in the message body. The classification of the resulting folder is not trivial to determine and is discussed

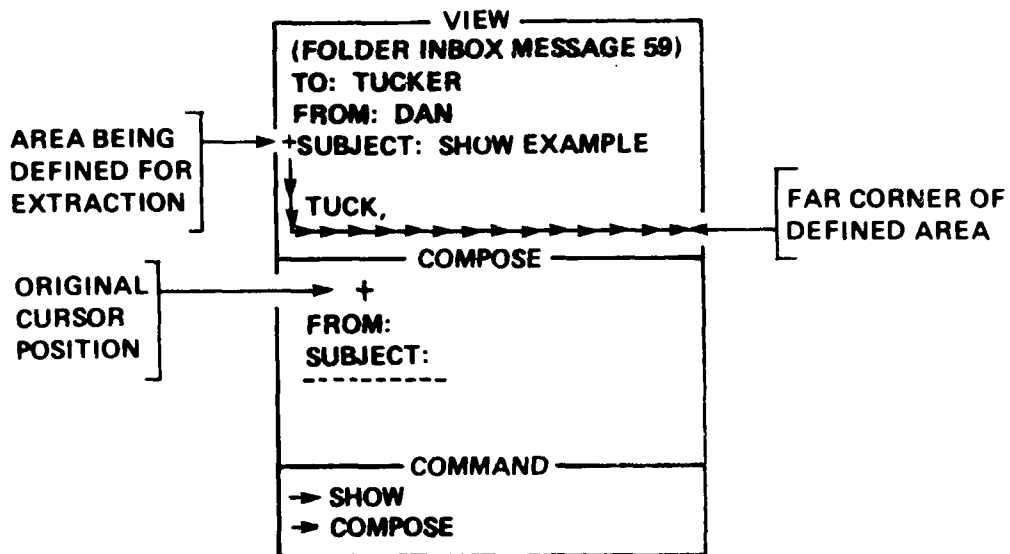


Figure 5. EXTRACTING VIEWED DATA WHILE COMPOSING

further in section 5.

#### IMPLEMENTATION

The implementation of the above commands is dependent on the construction of a trusted terminal manager that enables the user to interact with processes at different security levels, yet enforces the security model. The issues and ideas to date are discussed in the following section.



## SECTION 4

### TRUSTED TERMINAL MANAGER

As we have noted, the trusted terminal manager represents the major technological investigation of the KNet project. As such, many of the ideas we have formulated to date are still debatable. Here we outline some of the concepts we feel will be useful in attacking the problem, some issues that will have to be addressed through design or experimentation, and our present strategy.

#### MULTI-LEVEL OBJECTS

The "multi-level object" has been proposed as a potential solution to many problems that arise when applying security constraints to paper-domain tasks in a secure computer system. The multi-level object attempts to address the problem that the unit of information dealt with in the paper domain may have many sub-pieces of varying classifications (the typical document has each paragraph marked at a level equal to or lower than the overall classification of the document).

The manifestation of the problem in the computer domain is that the currently implemented computer security model does not allow any object to have more than one classification. How to support the paper requirements then? The multi-level object is an abstract concept of a set of independently classified entities collected and organized under one umbrella classification. The directory structure of a computer system can be thought of as one example. In the Military Message Experiment it was proposed that each paragraph of a multiply-classified message could be stored as an individual file and the message would be represented as a directory of these files. [9] For performance reasons this idea was deemed unacceptable; however, the notion may be of some merit.

#### Problems

The manipulation of multiply-classified entities in a computer security system seems to be impeded by four types of problems: creation, aggregation, excerption, and selection.

**Creation:** The creation problem concerns the protection of a not yet classified component of a multiply-classified entity. This component cannot be protected as unclassified because of its potential eventual classification. It should not be

overclassified because of the difficulty of downgrading in a computer system (requires trusting).

Aggregation: The aggregation problem concerns the protection of a collection of classified components. The association of the components may be classified at a higher level than the highest classified component. This higher classification cannot be derived under any known model, yet provision must be made for inputting and enforcing it.

Excerption: The excerption problem concerns the ease of removing components of low classification from a highly classified aggregation. This problem is essentially the converse of the aggregation problem. Although the aggregate cannot be viewed, individual parts should be (discretionarily) accessible.

Selection: The selection problem concerns the examination of the elements of an aggregation whose components are variously classified, to determine the components of interest. Selecting components individually is expensive (would use excerption). Selecting at the level of the aggregation, however, leads to overclassification and the coincident downgrading problem.

The trusted terminal manager will have to be able to deal with multi-level objects in the form of messages. The problem of dealing with multi-level objects requires a new approach to the protection of objects and their attributes in a secure computer system. The concept of "protected naming" may be an aid.

#### PROTECTED NAMING

Research has previously shown that it may be incorrect to assume that all attributes of an item of information are at the same classification. [10] Some unpublished research has been done to investigate, in particular, the proper classification of the name of an information object. This research is referred to as "Protected Naming".

Protected naming is an alternative interpretation of objects under access control. Protected names treat the existence of an object as a separately protectable item of information from its content information. The motivation for this interpretation comes from the observation that the creation (or deletion) of an object is independent of the content of that object.

Creation and deletion "modify" the existence of an object. If accessing an object is split into two components, first observing

its existence and second gaining access to its content, we see two distinct information channels: a "content" channel and an "existence" channel. Under conventional interpretations, the existence channel is a covert information channel; protected names legitimize the channel by making it separate and visible (in the form of a name) and by controlling the information it carries (by protecting the name at a particular level).

Isolating the channels from each other is a key to one issue of multiply-classified processing: supporting an object whose visibility is at a different classification than that of its content. The implications on a conventional kernel design are a liberalizing of certain access checks and a requirement for "memoryless" processes to access the new objects.

#### MEMORYLESS PROCESSES

Memoryless processes attempt to address the paper-domain problem of privacy — doing a computation on a set of data and giving a result based on that data without releasing the actual data. At least two proposals exist for implementing memoryless processing, both with some aid from hardware [11, 12].

The design of a security kernel provides almost exactly the required confinement of processes to make them memoryless. We intend to investigate the support of *memoryless* processes in KSOS as a potential solution to some of the problems of manipulating multi-level objects. The memoryless process could be used to manipulate an object whose content is at a different level than its existence, without revealing the content.

#### TRUSTED FUNCTIONS

Even with the use of memoryless processes, there remain several tasks in manipulating multi-level objects that technically violate the computer security model, or affect classification information in a way that is not covered by the model but can invalidate security.

These tasks are usually involved with communicating security information from one security domain to another, such as between the user and the application. Research on interfacing security domains has been done in a related project and a model has been proposed for these functions [7].

We intend to develop the software required (and in particular the trusted terminal manager) using this model as a guide, with the implication that the processes could be verified in the future.

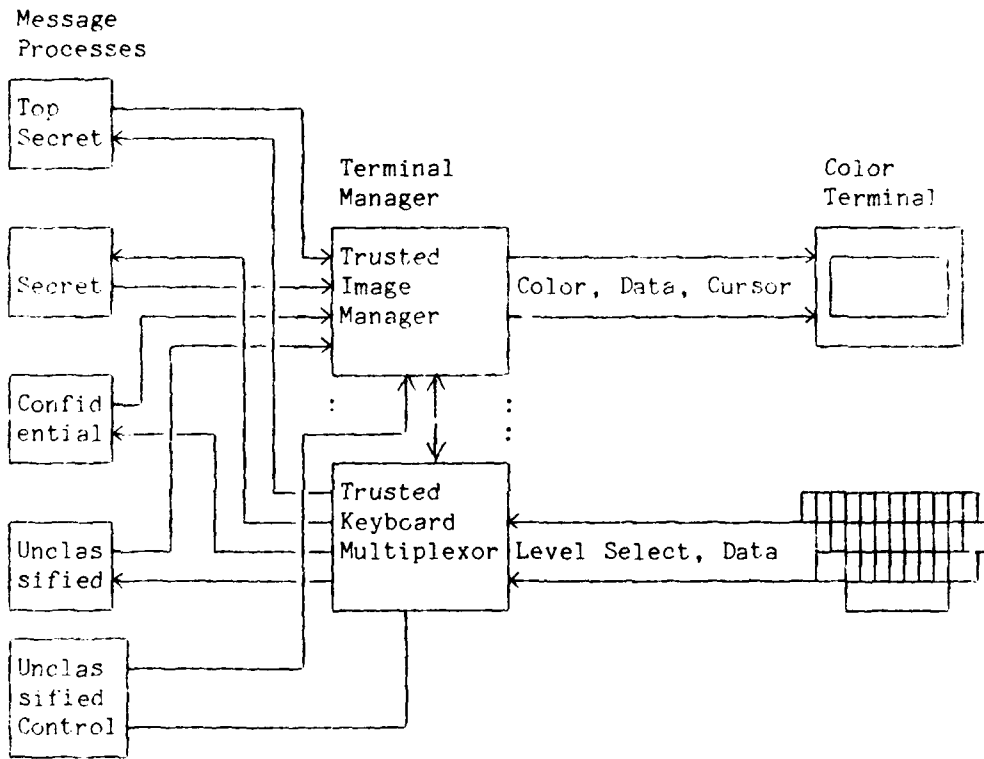


Figure 6. Trusted Terminal Manager

#### DESIGN

An abstract design of the terminal manager is given in figure 6. The terminal manager will virtualize the color terminal as four terminals, one for each classification. The terminal manager is trusted to mark each input with the appropriate color, and untrusted processes are prevented from writing any color information.

On input from the user, the terminal manager will take note of level changes by the user and direct keyboard data to the correct

classified process. The terminal manager is also responsible for sanitizing screen data that is selected for use in building new messages. It can perform this task because it maintains a complete description of the screen image and classification.

#### PROGNOSIS

The trusted terminal manager will of necessity be a trusted function, because of the nature of its work — manipulating the multi-level messages. To maintain a reasonable level of simplicity, it will rely on memoryless processes to do much of the sophisticated processing on the individual elements of each multi-level object. These memoryless processes and the elements of the multi-level object can be found and controlled for the most part by an untrusted (and unclassified) controlling process through the use of protected names, thus further reducing the size of the trusted portion of the terminal manager.

The major problem to be overcome is that all of these concepts are new, and only some of them have been demonstrated. It should be easy to accept, however, that the major technological investigation of a project should also be its primary area of risk.

The four concepts, outlined above, will be investigated in depth as potential solutions to the user/application security interface.

## SECTION 5

### ISSUES

There are conflicting considerations in any design, and this design is no exception. Security as defined by KSOS, minimal requirements of a message system, and implementation practicality have several mutually exclusive criteria that can only be resolved by compromise.

#### COMMAND ISSUES

##### Compose

Implementing the extraction feature of COMPOSE presents problems that greatly affect the user interface. If the user wants to include multilevel information as part of the body of a message, the information must be modified, since we have chosen to limit a message body to a single classification level.

To conform to this restriction the previously mentioned "monochromatic filter" is applied to the possibly multilevel data to be extracted. The information at a classification lower than the message body would be upgraded before it could be used in the message, and data classified higher than the message body would be masked from the user. This is certainly inconvenient to the user, since all security relevant distinctions are lost from the information being copied.

In the case of extracting data at a single protection level (e.g., part of another message body) the filtering is tolerable since the user can textually mark the data with the original security information, but the result of this mono-chromatic filtering on multilevel data, such as the output of a scan, could destroy much of the data's significance. This problem of copying multilevel information seems unsuitable to address with the current model used as the basis for KSOS.

##### Select

The SELECT command introduces another security related issue we are investigating in this design. The problem is determining the classification of a folder created by SELECT. In order to search all messages at once, the process that is searching folders for messages containing the selection criteria must run at the highest

level of any of the messages to be searched. If this process creates the resulting folder, the folder must be classified at the level of the process that created it.

This classification, although required by the security model, is inconvenient to the user, since the ability to read the original protection levels of the selected messages is forfeited if the folder is at the level of the highest message. This is a manifestation of the "selection" problem discussed in the previous section. Once a folder is overclassified, it can only be made useable again by declassification, which is hardly an elegant interface to the user.

From the designer's point of view the problem is a common one. Is it preferable to have the user do extra work (actively declassify information) and have the system remain mathematically secure, or to allow the computer to provide services that are potential compromise channels?

#### GRANULARITY MISMATCH

It might seem inappropriate that this design has situations where the user is allowed to view information at multiple protection levels, but not allowed to use the information elsewhere. The source of this discrepancy is a difference in the granularity of protection information between the display and KSOS.

The display is to be driven by the trusted terminal manager, which can display security information with a granularity as fine as a single character. Any attempt to use the multilevel information on the screen, however, requires transferring the information to a Unix file, which KSOS restricts to a single level.

This design must therefore be limited to addressing the issues surrounding the display of multiply-classified data, but cannot under present assumptions investigate the problems of creating and copying multilevel objects.

#### UNFORESEEN ISSUES

There are bound to be unforeseen problems surrounding the implementation of the trusted terminal manager. It is a focal point for several reasons. It is the only part of the proposed design that violates the existing security model. This in itself implies (correctly) that we are not basing our ideas on a formal model, but on intuitive concepts about (hopefully) legitimate tradeoffs between

security and utility. The terminal manager is also a focus for the message system routines, since it functions as the interface between the user and the rest of the system.

It is our hope that these issues will highlight a general class of problems in the current security model, and point to some possible solutions.



## SECTION 6

### CONCLUSION

We have described a proposed application for a secure distributed processing system — a secure message handler; and we have discussed some of the issues that arise when interfacing distinct security domains — in this case, the user and the application.

Two important ideas have been proposed. The first is the use of color as a medium for communicating security information to the user. Previous attempts at marking security information have encountered problems with believability (if the security information is simply additional text on the screen), and with observability (either remote lights or text). Color seems to be a workable solution to both of these problems, and is economical. In addition, color gives us the freedom to mark unrestricted areas of text and experiment with user acceptance.

The second idea is the use of a trusted process, the trusted terminal manager, to present to the user an interface that supports multilevel operations on a conventional kernel. This idea is the most risky, as it involves several new concepts, and the overall viability has yet to be demonstrated. It may be that the only result is that the secure terminal manager approach is inappropriate.

Finally, the project is presently at an interim stage with much work and many results remaining to be achieved. Nevertheless, the project is progressing well and the results to date have been promising.

#### REFERENCES

1. G. T. Hopkins, "Multimode Communications on the MITRENET", Proceedings of the Local Area Communications Symposium, Boston, Massachusetts, May 1979, pp. 169-177.
2. \_\_\_\_\_, "8001 CRT Manual," Intelligent Systems Corporation, Norcross, Georgia, May 1979.
3. D. M. Ritchie and K. Thompson, "The Unix Time-Sharing System," Communications of the ACM, volume 17, number 5, May 1974, pp. 365-375.
4. B. S. Borden, R. S. Gaines, and N. Z. Shapiro, "The MH Message Handling System: User's Manual," R-2367-AF, The RAND Corporation, Santa Monica, California, November 1979.
5. J. Postel (editor), "Transmission control Protocol (TCP) — Version 4," IEN: 81, Information Sciences Institute, Marina del Rey, California, February 1979.
6. E. J. McCauley and P. Drongowski, "KSOS: Design of a Secure Operating System," Proceedings of the 1979 NCC, volume 48, AFIPS Press, New York, New York, June 1979, pp. 345-354.
7. P. T. Withington, "The Trusted Function in Secure Decentralized Processing," MTR-3892, The MITRE Corporation, Bedford, Massachusetts, September 1979.
8. S. R. Ames, Jr., "Design of a Message Processing System for a Multilevel Secure Environment," MTR-3449, The MITRE Corporation, Bedford Massachusetts, June 1978.
9. J. R. Miller, "HERMES Security Design", report 4121, Bolt, Beranek, and Newmann Inc., Cambridge, Massachusetts, March 1979.
10. S. R. Ames, Jr., "File Attributes and Their Relationship to Computer Security," ESD-TR-76-372, Case Western Reserve University, Cleveland, Ohio, June 1974.
11. J. S. Fenton, "Memoryless Subsystems," The Computer Journal, volume 17, number 2, May 1974, pp. 143-147.

REFERENCES (Concluded)

10. I. Gat and H. J. Saal, "Memoryless Execution: A Programmer's Viewpoint," Software Practice and Experience, volume 6, 1976, pp. 463-471.

DISTRIBUTION LIST

INTERNAL

D-10

E. L. Key

D-40

F. R. Murphy

D-43

E. C. DeMone

D-44

M. Ferdman

D-50

F. Chess

D-64

V. A. DeMarines

G. T. Hopkins

J. Kowalchuk

B. P. Schanning

D-65

G. Lewitzky

D-70

W. S. Attridge

E. L. Lafferty

W. S. Melahn

D-71

M. Hazle

D-72

M. J. Corasick

A. S. Cressy

N. C. Goodwin

D. A. MacQueen

R. J. McGue

B. N. Seeber-Wagner

D-73

J. A. Clapp

D. G. Miller

D-75

S. R. Ames, Jr. (10)

D. L. Baldauf

D. J. Baughman

E. L. Burke

M. H. Cheheyl

T. L. Connors

D. L. Drake

K. B. Gasser

M. Gasser

R. A. J. Gildea

B. A. Hartman

G. A. Huff

J. Keeton-Williams

L. J. LaPadula

J. K. Millen

G. H. Nibaldi

J. J. O'Connor

S. J. Rajunas

D. P. Sidhu

D. J. Solomon (20)

J. D. Tangney

P. S. Tasker

E. T. Trotter

E. E. Wiatrowski

W. F. Wilson

J. P. L. Woodward

DISTRIBUTION LIST

INTERNAL

D-84

A. M. Dahod

W-31

S. F. Holmgren

A. P. Skelton

J. K. Summers

D. C. Wood

PROJECT

Rome Air Development Center  
Griffiss Air Force Base  
Rome, New York 13441

ASCF

Mr. T. C. Darr (10)



*MISSION*  
*of*  
*Rome Air Development Center*

*RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C<sup>3</sup>I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.*

END

DATE  
FILMED

6 - 8 - 1

DTIC