FINAL EVALUATION REPORT
Gemini Computers, Incorporated
Gemini Trusted Network Processor
Version 1.01

NATIONAL
COMPUTER SECURITY CENTER

9800 Savage Road
Fort George G. Meade
Maryland 20755-6000

28 June 1995

Report No. 34-94
Library No. S-241,887

FOREWORD

This publication, the Final Evaluation Report for the Gemini Trusted Network Processor, is being issued by the National Computer Security Center under the authority of and in accordance with DoD Directive 5215.1, "Computer Security Evaluation Center." The purpose of this report is to document the results of the formal evaluation of the Gemini Trusted Network Processor (GTNP) from Gemini Computers, Incorporated. The requirements stated in this report are taken from the *Department of Defense Trusted Computer System Evaluation Criteria* [136] dated December 1985 as interpreted by the *Department of Defense Trusted Network Interpretation (TNI) of the Trusted Computer System Evaluation Criteria* [137] dated 31 July 1987.

Approved:

John C. Davis
Director,
National Computer Security Center

28 June 1995

This page intentionally left blank

ACKNOWLEDGMENTS

The formal evaluation team for the Gemini Trusted Network Processor consisted of the following individuals, who were provided by the indicated organizations:[1]

Daniel P. Faigin†‡
Anne M. Lindell†
Phillip A. Porras†
Daniel R. Tapper†

The Aerospace Corporation
El Segundo, CA 90245

Jonathan K. Millen†

The MITRE Corporation
Bedford, MA 01730

The team would also like to thank the following individuals who previously served on this evaluation team:

James Arnold†, Carl Burney, Myron M. Coplin†, Deborah Downs, Jeff Glass, Paul Hager‡, Cornelius J. Haley‡, John Janeri, Jon King†, Sue Landauer, Paul A. Olson†, Jeffrey C. Thomas†‡, Terry Vickers-Benzel, Lynn Vidmar‡, Jim Wallner, and Todd Wittbold.

The team would like to particularly highlight the support provided during the formal evaluation phase by James Arnold (architecture study coordinator), Anne Lindell (team testing coordinator) and Dan Tapper and Jon Millen (formal verification coordinators).

The team would also like to thank Mike Rogson of The Aerospace Corporation for his help in preparing this report.

The team would also like to commend the support of the vendor during this process. In particular, the team would like to acknowledge the extra technical support provided by Albert Tao and Dan Warren of Gemini Computers, Incorporated. The team is also grateful for its interactions with Dr. Roger Schell and Tim Levin during their tenures at Gemini Computers, Incorporated.

Lastly, the team would like to acknowledge the efforts of the Trusted Products and Network Support Division work center chiefs, Mike Hale and Janine Pedersen, in support of this evaluation.

[1] The following notations apply: † indicates a contributor to this document; ‡ indicates individuals who served as team leader for this evaluation.

This page intentionally left blank

TABLE OF CONTENTS

FIGURES

This page intentionally left blank

TABLES

This page intentionally left blank

EXECUTIVE SUMMARY

The security protection provided by the Gemini Trusted Network Processor (GTNP) has been examined by the National Computer Security Center (NCSC). The security features of the GTNP were examined against the requirements specified by the *Trusted Network Interpretation (TNI) of the Trusted Computer System Evaluation Criteria* [137], dated 31 July 1987.

The NCSC evaluation team has determined that the highest rating at which the GTNP satisfies all the specified requirements of the TNI, when configured according to the manner described in the Trusted Facility Manual, is that of a class A1 Mandatory-Only Network Component (M-Component).

A system that has been rated as being an A division system is characterized by the use of formal specification verification methods to assure that the controls provided by the system can effectively protect classified or other sensitive information stored or processed by the system. The system architecture is that of a minimized security reference monitor. Extensive documentation is required to demonstrate that the Trusted Computing Base (TCB)[2] meets the security requirements in all aspects of design, development, and implementation.

M-Components are components that provide network support of the Mandatory Access Control (MAC) Policy as specified in the TNI. M-Components do not necessarily include the mechanisms necessary to completely support any of the other network policies (Discretionary Access Control, Identification and Authentication, and Audit) as defined in the interpretation.

The GTNP product consists of software that operates on either a Gemini Trusted Base with up to eight processors (80286, i386, or i486), or on a Gemini-adapted PC-AT platform (which includes various IBM PC/AT models as well as the Zenith Z-248 compatibles, each with a single 80286 processor). These systems include device interfaces (e.g., Serial Input/Output (I/O), Ethernet, and High-level Data Link Control (HDLC)) to support communication with other components within an overall Network Security Architecture. Software to manage user-level communication through one of these devices in a specific network would run outside the NTCB partition as an application on the GTNP. Such application software is not part of the GTNP product, and is not evaluated here, other than to note that its use (assuming the software is in a single-level process) would not affect the evaluation rating given to this product.

The GTNP software provides a general-purpose mandatory security reference monitor upon which other applications can be developed. These applications include network-oriented applications such as network components, guards, terminal access controllers, gateways, and packet-switches, as well as general purpose secure operating systems. Security mechanisms provided by the kernel include a strict hierarchical ring mechanism, a multilevel segment naming hierarchy, and controlled access to devices. Support is also provided for encryption of critical system structures and user data using the Data Encryption Standard (DES).

[2] In the case of network component evaluations, this demonstration is for the Network TCB (NTCB) component partition.

This page intentionally left blank

Section 1

Introduction

In 1985, the National Computer Security Center (NCSC) began working with Gemini Computers, Incorporated to evaluate the Gemini Multiprocessing Secure Operating System (GEMSOS). In July 1989, this evaluation was refocused on the Gemini Trusted Network Processor (GTNP), the high-assurance M-component base upon which GEMSOS was built. This evaluation entered the formal stage in February 1992. It should be noted that the Gemini Trusted Network Processor (GTNP) Network Trusted Computing Base (NTCB) partition shares many components with GEMSOS, including the GEMSOS Security Kernel, many of the layers of the GEMSOS Network Processor System Parent (NPSP), and many of the system generation tools.

The purpose of this report is to provide evidence and analysis of the security features and assurances provided by the Gemini Trusted Network Processor. This report documents the evaluation team's understanding of the product's security design and appraises its functionality and integrity against the interpretation of the requirements for an A Division M-component as defined in the *Trusted Network Interpretation (TNI) of the Trusted Computer System Evaluation Criteria (TCSEC)* [137].

This evaluation applies to the versions of the GTNP System available from Gemini Computers, Incorporated, defined in appendix A and appendix B.

Material for this report was gathered by the NCSC GEMSOS/GTNP evaluation team through analysis of documentation, testing, and interaction with system developers.

1.1 Evaluation Process Overview

The Department of Defense Computer Security Center (DODCSC) was established in January 1981 to encourage the widespread availability of trusted computer systems for use by facilities processing classified or other sensitive information. In August 1985 the name of the organization was changed to the National Computer Security Center. In order to assist in assessing the degree of trust one could place in a given computer system, the DoD Trusted Computer System Evaluation Criteria (TCSEC) was written. The TCSEC establishes specific requirements that a computer system must meet in order to achieve a predefined level of trustworthiness. The TCSEC levels are arranged hierarchically into four major divisions of protection, each with certain security-relevant characteristics. These divisions are in turn subdivided into classes. To determine the division and class at which all requirements are met by a system, the system must be evaluated against the TCSEC by an NCSC evaluation team. Since the TCSEC was written, additional interpretations have been published, such as the Computer Security Subsystem Interpretation [138], the Trusted Network Interpretation [137], and the Trusted Database Interpretation [139]. These interpretations are used in the evaluation of subsystems, networks and network components, and database systems.

The NCSC supports the creation of secure computer products in varying stages of development from initial design to those that are commercially available. Preliminary to an evaluation, products must go through

the Proposal Review Phase. This phase includes an assessment of the vendor's capability to create a secure system and complete the evaluation process. To support this assessment a Preliminary Technical Review (PTR) of the system is done by the NCSC. This consists of a quick review of the current state of the system by a small but expert team and the creation of a short report on the state of the system. If a vendor passes the Proposal Review Phase they will enter a support phase preliminary to evaluation. This support phase has two steps, the Vendor Assistance Phase (VAP) and the Design Analysis Phase (DAP). During VAP, the newly assigned team reviews design specifications and answers technical questions that the vendor may have about the ability of the design to meet the requirements. A product will stay in VAP until the vendor's design, design documentation, and other required evidence for the target TCSEC (and any relevant interpretation) class are complete and the vendor is well into implementation. At that time, the support moves into DAP.

The primary thrust of DAP is an in-depth examination of a manufacturer's design for either a new trusted product or for security enhancements to an existing product. DAP is based on design documentation and information supplied by the industry source. It involves little "hands on" use of the system, but during this phase the vendor should virtually complete implementation of the product. DAP results in the production of an Initial Product Assessment Report (IPAR) by the NCSC evaluation team. The IPAR documents the team's understanding of the system based on the information presented by the vendor. Because the IPAR may contain proprietary information and represents only a preliminary analysis by the NCSC, distribution is restricted to the vendor and the NCSC.

Products that have completed the support phase with the successful creation of the IPAR, enter formal evaluation. Products entering formal evaluation must be complete security systems. In addition, the release being evaluated must not undergo any additional development. The formal evaluation is an analysis of the hardware and software components of a system, all system documentation, and a mapping of the security features and assurances to the TCSEC and the relevant interpretations. The analysis performed during the formal evaluation requires "hands on" testing (i.e., functional testing and, if applicable, penetration testing). The formal evaluation results in the production of a final report and an Evaluated Products List (EPL) entry. The final report is a summary of the evaluation and includes the EPL rating which indicates the final class at which the product satisfies all TCSEC and relevant interpretation requirements in terms of both features and assurances. The final report and EPL entry are made public.

1.2 Document Organization

This report consists of a main body, that includes ten chapters and the first two appendices; and supplemental information, contained in the remaining appendices. The main body describes the system and how it meets the requirements; the appendices included with the main body detail the specific hardware and software configuration that was evaluated. The remaining appendices provide additional reference material for the convenience of the reader.

The basic purpose of each of the chapters and appendices is as follows:

- Section 1 is this introduction.
- Section 2 provides a high-level overview of the system and its policy, as well as its hardware and software architecture.
- Section 3 describes the hardware architecture of the GTNP.

- Section 4 describes the software architecture of the GTNP.
- Section 5 discusses the resources that are protected by the GTNP NTCB partition.
- Section 6 focuses on the mechanisms present in the GTNP Trusted Computing Base (TCB) that protect the resources discussed in section 5.
- Section 7 presents the various assurances that increase the trust in the GTNP.
- Section 8 summarizes the requirements specified in the TNI; for each, it describes how the GTNP satisfies the requirement.
- Section 9 evaluates the GTNP with respect to the additional qualitative features and functionality discussed in part II of the TNI.
- Section 10 presents any additional comments from the evaluation team that concern either positive or negative characteristics of the system that have not been covered elsewhere in the report.
- Appendix A summarizes the various hardware configurations that are covered by this evaluation.
- Appendix B summarizes the various software configurations that are covered by this evaluation.
- Appendix C presents the EPL entry for the GTNP.
- Appendices D, E, and F present detailed overviews of the processor bases (80286, i386, and i486, respectively) covered by the evaluation. These are generic descriptions of the processor capabilities and are independent of GTNP specifics.

The remaining two appendices present a list of acronyms and a bibliography.

1.3 Conventions

The following conventions are used throughout this document:

- Unless otherwise clarified, the term *TCB* refers only to the Network TCB M-component partition implemented by the GTNP.
- A distinction is made in this document between the terms *level*, *class*, and *label*. With the exception of the terms *single-level* or *multilevel* with regards to processes and devices, the term *level* is used exclusively to refer to the hierarchical portion of an access label. The term *class* is used to refer to the combination of the hierarchical level and the non-hierarchical categories. More commonly, the term *label* is used. This refers to the class combined with information such as ring brackets. Note that this usage of the term *label* is non-standard.

  Throughout this document, the term "single-level" is used with respect to that portion of the Mandatory Access Control (MAC) policy allocated to the GTNP M-Component NTCB partition. If, for example, the MAC policy allocated to the GTNP Mandatory Network Trusted Computing Base (M-NTCB) partition recognized secrecy levels but not secrecy categories, then a resource having a single secrecy level but a range of categories would be a single-level subject.

Related to the term *label* is the concept of a *write label* and a *read label*. Gemini, in their documentation, does not use the terms *minimum* and *maximum*; instead, the *write label* corresponds to the minimum label, and the *read label* corresponds to the maximum label. These terms were chosen by Gemini because these labels control the ability to read and write, and the use of minimum and maximum in the context where a label includes both secrecy and integrity is confusing. In the GTNP, *read label* is used during dominance comparisons for reading (i.e., the label of the object must be dominated by the maximum label of the subject), while the *write label* is used during dominance comparisons for writing (i.e., the label of the object being written must dominate the minimum label of the subject). The team also follows this convention, and uses the terms *read label* and *write label*.

- In this document, the term *security relevant* is used. This term is synonymous with *protection relevant*.
- A notational convention is followed to distinguish between two types of similarly named objects used to synchronize within the kernel. This convention is discussed in section 4.2.1.4, page 68.
- The design of the GTNP incorporates both software abstractions of processors, called *virtual processors*, and the processors in hardware. The terms *physical processor* and *Central Processing Unit (CPU)* will be used interchangeably to refer to the processors in hardware.
- This system makes a distinction between the software actively used in the operational system, and software used prior to secure initial state in either an offline or pre-operational mode. The latter software is called *System Generation Software* and cannot be accessed or used while the system is operational. A majority of this software is isolated to a distinct volume, and is used exclusively for GTNP configuration.

  System Generation Software is subject to fewer requirements than TCB software. System generation software must be maintained under configuration management, and have design and test documentation for any security mechanisms that it provides. Those security mechanisms are also covered by the security test suite. Directions for the use of this software must be provided in the Trusted Facility Manual. However, system generation software is not required to address the modularity concerns of the System Architecture requirement.
- It should be noted that this system has no users *per se* (there are users of the system generation software, but this user database is not part of the operational system). From the point of view of this report, the untrusted entities accessing the system are *application processes* running unspecified code.
- The following typographical conventions are used in this document:
  - A **Bold** type font is used to indicate hardware registers.
  - A Sans-Serif type font is used to indicate commands and instructions.
  - An *Italic* type font is used to indicate when a term is being defined.
  - A `Typewriter` type font is used to indicate gate calls, processes, specific filenames, and Formal Top-Level Specification transform names.
  - A SMALL CAP type font is used to indicate signal names and state names.
  - Specific names, and references to logical operations (as opposed to specific kernel gates) are enclosed in quotation marks.

Section 2

System Overview

This chapter presents an overview of the GTNP product. It begins with a review of the history of the product and then a high-level overview of the product. Following this is a discussion of how the product fits into a network architecture. The section concludes with a summarization of the evaluated configuration.

2.1 Product History

In order to understand the history of the Gemini Trusted Network Processor (GTNP), one must first understand the history of Gemini Computers, Incorporated, and the Gemini Multiprocessing Secure Operating System (GEMSOS). This is because the GTNP product has been developed by Gemini as an extension of the GEMSOS security kernel (not the complete GEMSOS system), such that the functions of the security kernel (i.e., mandatory access control) might be evaluated and available as an M-component, in compliance with the Trusted Network Interpretation (TNI). Therefore, the early history of the GTNP is rooted in the history of the GEMSOS security kernel.

Gemini was founded in 1981 by Dr. Tien Fan Tao of the Naval Postgraduate School and Dr. Roger Schell, former Deputy Director of the Department of Defense Computer Security Center (now called the National Computer Security Center (NCSC)). The goal was to develop a family of high-assurance systems supporting multilevel processing on a multiple microcomputer system.

The developmental approach of the GEMSOS security kernel was to design and build a high-performance, high-assurance commercial product using Commercial "Off-The-Shelf" (COTS) hardware components and well-developed operating system principles and methods. The fundamental principles and methods utilized in the GEMSOS security kernel development include the internal layering and two-level scheduling described by Reed [146], the modularization described by Jansen [127], the information hiding techniques of Parnas [144, 143], the synchronization methods of Reed and Kanodia [147], the formal verification techniques described by Price [145], and the architectural approach of the Multics operating system [153].

The GEMSOS multiprocessor architecture has been designed to provide computing capacity and throughput suitable for commercial applications [151]. The ongoing progress of the Intel $80x86$ processor family, on which the GEMSOS hardware is based, as well as its multiprocessor architecture, helped the GEMSOS hardware base keep pace with performance advancements.

Early reports on the GEMSOS security kernel project [150, 151] describe the schema of the kernel interface. Since these reports have been published, the unevaluated GEMSOS security kernel has been used as the basis for several operational Multilevel Secure (MLS) systems that have been delivered to government users, all of whom use the GEMSOS security kernel product as part of their trusted computing base.

One of the first applications of the GEMSOS security kernel was for an MLS wide-area network called BLACKER [170], which was successfully evaluated against the Class A1 under the Trusted Computer System

Evaluation Criteria (TCSEC).[1] In the BLACKER system, the GEMSOS security kernel along with other trusted components, is used as the basis for trust for critical cryptographic and key distribution functions that maintain communications separation by cryptographic means [7].

The way in which some of the other applications have used the GEMSOS security kernel is as follows:

- As the trusted foundation for Communication Control and the Access Control modules in a complete system interlinking single-level IBM mainframes with multilevel terminals. In this design, the GEMSOS security kernel provides mandatory access control as part of an M-Component as described in the TNI [8, 167, 164].
- As a foundation for the guards[2] in a defense office automation system, supporting both X.25 wide area and Ethernet local area networks [167].
- As part of a design for the central nodes in the star topology of an overall network incorporating many of the themes enunciated in recent research results: networks of trusted systems, interface with long-haul communications systems, and the integrity lock architecture for MLS data management [152].

In addition, research problems addressed in the course of the implementation have contributed to the advancement of COTS high assurance trusted products and technologies [130, 131, 133, 163, 162].

Note that all the applications and research issues discussed above were centered around the GEMSOS security kernel, which is a portion of the planned GEMSOS system. In 1989, the evaluation of the security kernel as a distinct product, the GTNP, was initiated. The GTNP was introduced as a product at the National Computer Security Conference in 1990 [168].

2.2 GTNP Overview

The GTNP can be best thought of as a base upon which applications requiring high-assurance mandatory security can be hosted. There are two types of GTNP platforms: a PC-based platform with a single 80286 processor communicating over the PC-bus, or a Multibus-based platform (called the Trusted Base) with up to eight tightly-coupled microcomputers (80286, i386, or i486) that communicate through shared memory segments to provide high throughput over an Institute of Electrical and Electronic Engineers (IEEE) standard 796 Multibus. Bus contention is minimized through the kernel's management of local and global processor memory; data and code are kept local to the processor whenever possible. This allows a claimed effective throughput of 1/2 to 8 Million Instructions Per Second (MIPS) per system using HP processors (80286), and up to 75 MIPS using the UP processors (i486).

The multiple microcomputers in a GTNP are capable of multiprocessing as well as multiprogramming. Depending on the requirements of a particular application, the GTNP can multiplex processes onto a single processor or support their distribution among several processors. Communication between these processes is through shared segments; synchronization is achieved through the use of an eventcount and sequencer mechanism (see section 4.1.6, page 58, for more information on eventcounts and sequencers).

[1] The BLACKER development predated the TNI; hence, its evaluation (by a different branch of the National Security Agency [NSA]) was against the TCSEC. Due to the evaluation by a different NSA branch and the fact that it was not a commercial product, BLACKER does not have an Evaluated Products List (EPL) entry.

[2] Guards, in a multilevel system, restrict information flow to a single label.

A high-level picture of the hardware architecture of a Gemini system is shown in figure 2.1, page 8. The salient features of this architecture are as follows:

- IEEE Standard 796 Multibus or IBM Personal Computer/AT (PC/AT) Bus.
- Support for up to eight Standard, HP, SP, or UP processors (Gemini Trusted Bases) or an upgraded (via the addition of the the Gemini System Controller Board) single-processor PC-AT platform (various IBM PC/AT models and the Zenith Z-248), with an Intel Numeric Processor Extension (NPX).[3]
- A Gemini System Controller (GSC) board that provides the following:
  - Real-time clock with battery backup.
  - High-speed hardware data encryption device using the Data Encryption Standard (DES) algorithm.
  - Alterable and non-alterable non-volatile memory for storing various system parameters and encryption keys.
- Support for between 1 and 16 Mbytes of local Random Access Memory (RAM) per processor, depending on processor type.
- Support for 1/2 to 8 Mbytes of shared global memory per system.
- A maximum of seven secondary storage devices (two or three of which can be 5 1/4″ floppy disk drives or Quarter-Inch Cartridge (QIC) tape drives; two of which can be 5 1/4″ hard disk drives; and two of which can be 1/2″ tape drives. Note that tape drives are available only for the Trusted Base).
- A maximum of 80 RS-232 serial Input/Output (I/O) ports (up to 2 local serial I/O ports per general processor, and up to 8 extended serial I/O ports per each of the 8 potential extended serial I/O boards. Note that the processor model, system model, and/or number of other types of board may make this maximum configuration physically impossible.
- Support for X.25 level 1 and level 2 High-level Data Link Control (HDLC) – Link Access Procedure B (HDLC-LAPB). A maximum of 16 distinct HDLC devices may be present in a single system. Note that the processor model, system model, and/or number of other types of board may make this maximum configuration physically impossible.
- Support for IEEE 802.3 Ethernet connections. A maximum of 16 Ethernet devices may be present in a single system. Note that the processor model, system model, and/or number of other types of board may make this maximum configuration physically impossible.

The software itself is highly modular and is primarily written in Pascal. Use of assembly language is minimal; just under 14% of the system's source code is in assembler.[4] The design is both highly-layered and loop-free. The basic methodology used in the design of the software was Parnas modules [144].

[3] The numeric coprocessor is optional on the Standard and HP processors in the original series of the Trusted Base, and on the PC/AT platform. Models lacking an NPX do not emulate floating point.

[4] Use of assembly language is primarily for interface with processor data structures, devices, interrupt handling, and in selected areas, performance improvement.

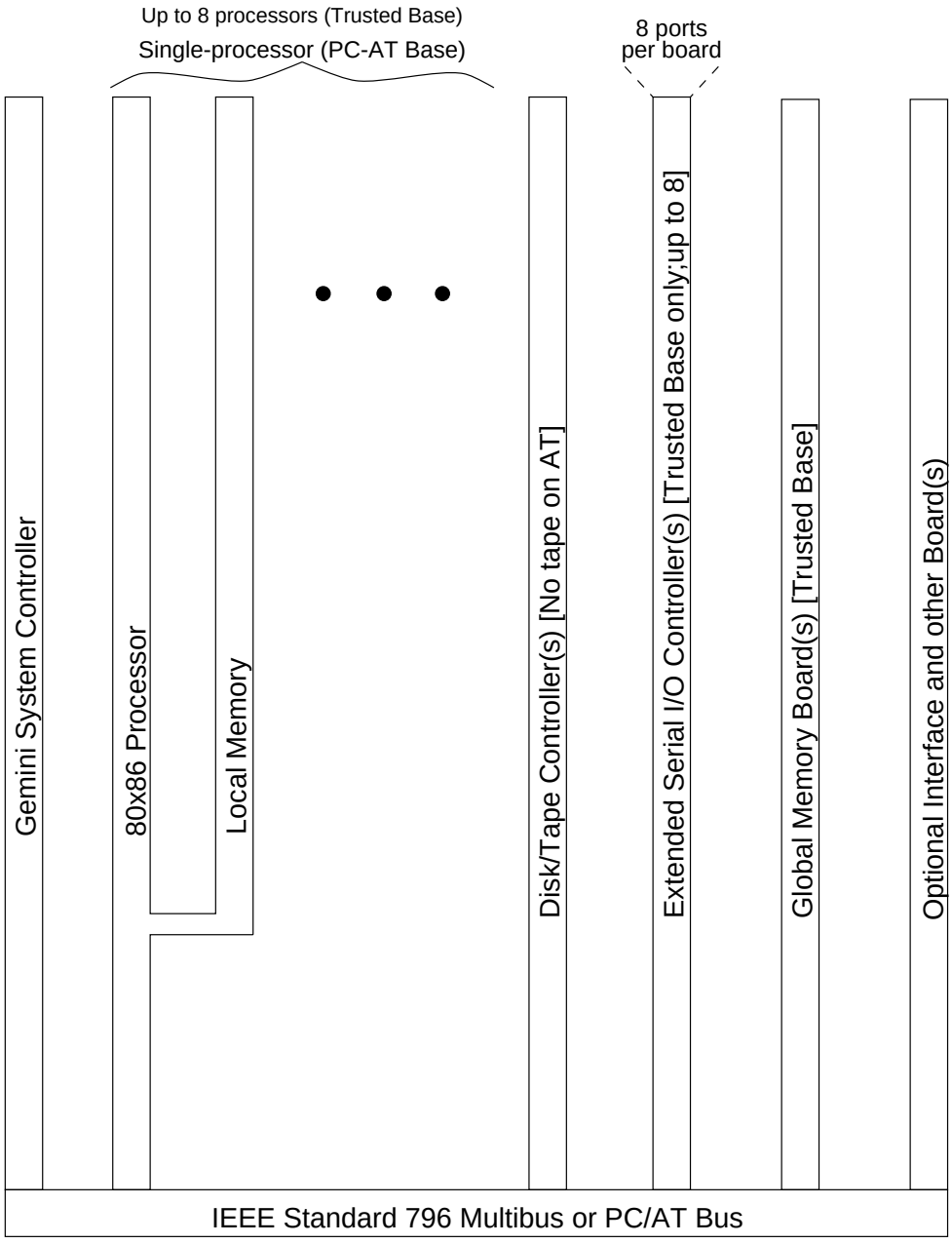


Figure 2.1. GTNP High-Level Hardware Architecture Overview

The software and the hardware together provide the security support in the system.[5] The 80286, i386, and i486 processors provide hardware support for segmentation of memory. These processors also provide an architecture with four hierarchical *privilege levels* that provide isolation of the kernel and support domain separation. The software takes full advantage of these hardware security and protection features. All information stored in a GTNP system is contained in logical objects called *segments*. The GTNP kernel maintains, for every segment, an *access label* containing the segment's *access class* and *ring brackets*. Labels in the GTNP provide two sets of hierarchical levels (16 values each) and up to 96 non-hierarchical categories. These labels are interpreted by the system to allow the implementation of both a secrecy policy and a data integrity policy. Ring brackets provide a mechanism for isolating segments to specific hierarchical rings.

In the GTNP, processes are responsible for the management of their address spaces; the kernel provides mechanisms that allow segments to be introduced to and removed from address spaces, and to be swapped to and from secondary storage. The name space of segments is not global; the only way to name segments is through a process local path that is relative to a segment already known to the process.

Additional security for data is provided through the use of the DES encryption device. Generation of cryptographic seals with this device supports trusted distribution and recovery, as well as serving as a means to detect corruption of critical system data structures on storage media, including those data structures that provide label integrity.

The three hardware privilege levels outside of the most privileged level (in which the kernel executes) are multiplexed by the GEMSOS Security Kernel into the software abstraction of eight hierarchical rings. Processes may execute in any of these rings; each ring may have a different access label range.[6] A strict view of subjects is taken by the system; each process and ring combination is considered a different subject for the purposes of access control.

The GTNP system uses a priority-based preemptive scheduling algorithm. A process in the GTNP runs until it requests a service that is currently unavailable, or until it is preempted by a higher priority process. When this happens, the scheduler chooses the next highest-priority, runnable process (on that processor) to run. A two-level scheduler is used that virtualizes the multiple processor architecture.

Device management in the GTNP is based on the notion that each device driver is "intelligent" enough to invoke process synchronization services as necessary. Completion of device activity for asynchronous devices is signalled to a process through the use of an eventcount that is shared with the device driver.

The software portion of the Trusted Computing Base (TCB) is organized into the following three principal pieces (illustrated in figure 2.2, page 11):

- The GEMSOS Security Kernel, which serves as the reference monitor for the mandatory policy enforcement. The same kernel is used in the full GEMSOS system. It is the kernel that provides the basic features discussed above.

[5] As this is a high-level discussion, many of these concepts are left undefined at this point. For more details, please consult the following sections: *devices*: section 4.1.8, page 59; *eventcounts and sequencers*: section 4.1.6, page 58; *labels*: section 4.1.1, page 46; *privilege levels*: section 3.2.1, page 16; *ring brackets*: section 4.1.2.1.2, page 49; *rings*: section 4.1.2, page 47; *segments*: section 4.1.4, page 54; *virtual processors*: section 4.1.3, page 52; and *volumes*: section 4.1.5, page 57.

[6] Applications are restricted to single-level processes in the evaluated configuration, although multi-level processes can be used to provide a trusted application interface, which must be separately evaluated. Applications are also restricted to rings two through seven; the ability to relax this latter restriction is controlled by Gemini through license agreements. Note that doing this will invalidate the rating of the system and require re-evaluation.

- The GEMSOS Network Processor System Parent (NPSP), a trusted multilevel initial process that executes on each processor and is responsible for starting the initial application processes on the various processors. The NPSP includes the run-time intersegment linkage mechanism provided by the GEMSOS Multilevel Subjects Intersegment Linkage Tool (MIT).

- The GEMSOS Kernel Gate Library, which is linked with the NPSP to provide the NPSP with a high level language interface to the kernel. Copies of the Kernel Gate Library code may also be linked within application processes, however these copies are not considered to be part of the TCB when executed within a non-TCB subject.

Additional software components are provided to initialize the kernel, and to perform system generation, configuration, and installation/recovery. These components all execute before the establishment of secure initial state and are not part of the TCB.

2.3 GTNP Network Security Architecture

The GTNP is designed to serve as an M-component within a network. The GTNP enforces a mandatory security policy and provides a base upon which single-level applications can be run. For use in a specific network, functions supporting intercomponent communication (e.g., in support of the application-specific Network Security Architecture) must be implemented as such applications.

The GTNP is intended to support "internal subjects" and virtual machines as mentioned in Section I.3.2.2.2 of the TNI. It provides a Mandatory Access Control (MAC) Network Trusted Computing Base partition (M-NTCB) within a system, and single-level interfaces to other network components within a network. It does not, in and of itself, support any concept of subjects serving as direct surrogates for a human user. That abstraction, if present, would be implemented by application software running on the GTNP M-NTCB. The M-NTCB's function is to ensure implicit MAC separation (i.e., no action by the software above the M-NTCB is required to provide the separation).

2.3.1 The GTNP's Role in an Overall Network Architecture

Within a network security architecture (as defined by Part I of the TNI), the GTNP assumes that other non-GTNP network components are providing the additional services necessary to provide the overall network with a satisfactory Network Trusted Computing Base (NTCB). These other services include Identification and Authentication (I); Audit (A); and Discretionary Access Control (D) functions as defined in the TNI. Integration of a GTNP system into an overall network will involve the development of appropriate applications that will operate on the GTNP. These applications will not be trusted with respect to that portion of the overall network security policy allocated to the GTNP M-NTCB, but may be trusted with respect to the network-wide security policy. The subjects supported by the GTNP are intended to be internal subjects. The GTNP has no knowledge of human users or network subjects, although applications operating on the GTNP may have such knowledge.

The TNI does require that an M-component be capable of providing various audit information when composed with an A-component. Although the GTNP performs no actions that require auditing within the

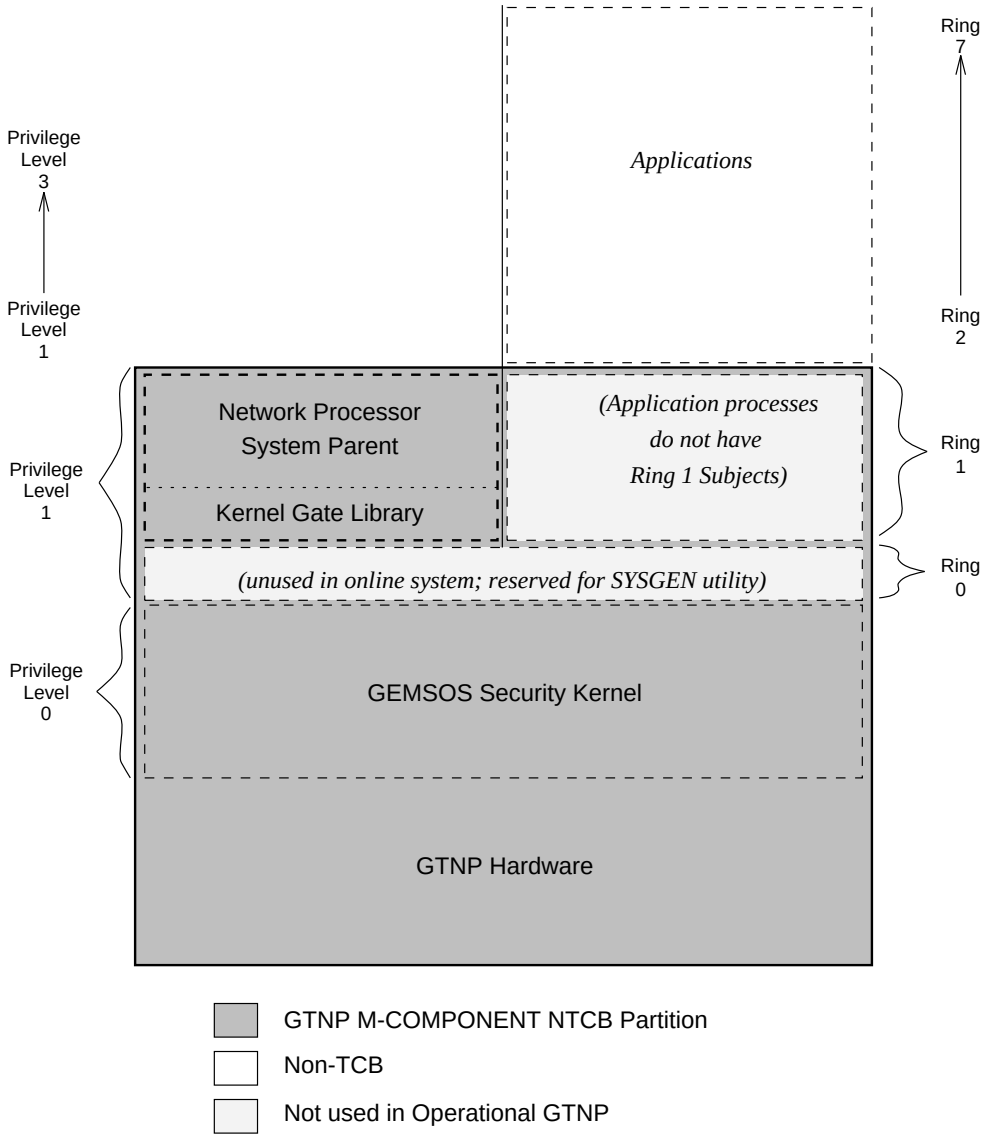


Figure 2.2. GTNP High-Level Software Architecture Overview

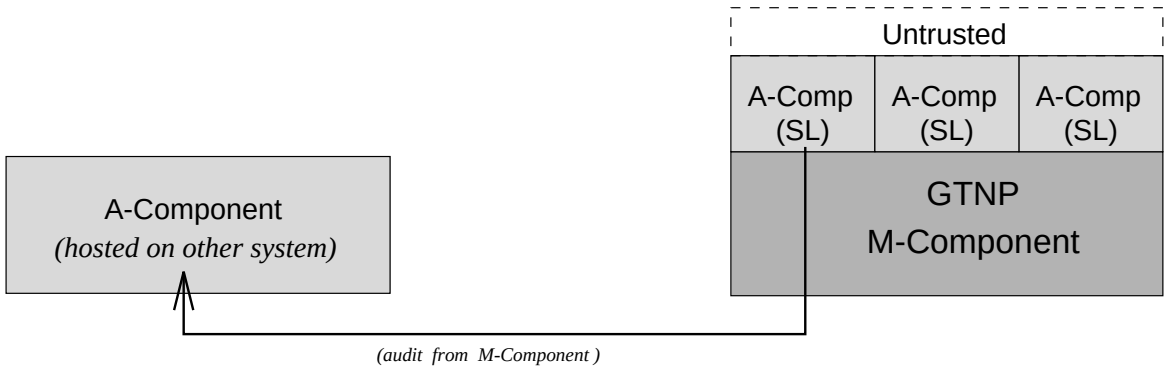


Figure 2.3. Composition of an A-Component with the GTNP M-Component

M-NTCB (under a strict reading of the M-component requirements),[7] there are certain implied audit conditions resulting from the composition (such as the denial of an action due to the mandatory policy). In these cases, the GTNP M-NTCB returns to the application program an appropriate error condition. In order for an A-Component to be composed with the M-Component, the A-Component must be implemented as a trusted virtual machine (with respect to the A-Component's NTCB) on top of the GTNP. If the overall network security architecture requires that the M-Component's activity be audited, the GTNP cannot support untrusted subjects that are not within some virtual machine implementing an A-Component. If untrusted subjects had direct access to the GTNP M-NTCB, their actions could not be audited. By implementing A-Components as virtual machines on the GTNP, the resulting architecture presents a layering in which the A-Component's NTCB partitions exist as a layer between the GTNP M-NTCB and the untrusted subjects of the virtual machines. This layering is such that all kernel calls that may return conditions that would result in audit must pass through the A-Component's NTCB partition before being handed to the virtual machine's untrusted subjects (i.e., those above the A-Component's NTCB partition). The virtual machine A-Component would then forward this audit information in an appropriate protocol format to another A-Component. This is illustrated in figure 2.3, page 12.

Other evaluated components in a network can be composed with a GTNP M-component in any of the following ways:

- Through an interface with a distinct device on the network.
- As a virtual machine on top of the Virtual Machine Monitor provided by the GTNP M-component.
- As a device at a preallocated memory address on the GTNP.

[7] Under a strict reading of the TNI requirements for an M-component, only four requirements specify a need for audit. The "Labels" requirement requires auditing for the import of non-labeled data when a user supplies the label of the data. The GTNP, in the evaluated configuration, has only single-level devices with the label being defined during offline system configuration. Under "Exportation of Labeled Information," auditing of a change in labels of a device is required. Again, this change can only be made when the system is reconfigured offline. Under "Labeling Human-Readable Output," audit is required when the marking defaults are overridden. The GTNP has only single-level devices and provides no user interface for changing labels nor printer devices for printouts. Last, the Covert Channel Guidelines imply auditing of storage channels over a particular bandwidth. Gemini's Covert Channel Analysis reports that, when configured in accordance with the Trusted Facility Manual, the GTNP has no storage channels that exceed the threshold that would require audit.

The following sections discuss these manners of composition in more detail.

2.3.1.1 Distinct Device Interface to Other Network Components

The GTNP provides a variety of single-level device interfaces for use in communication with other network components. These device interfaces include RS-232 serial lines, X.25 HDLC lines, Ethernet lines (IEEE 802.3 Media Access Control), and a Centronics parallel port. These interfaces are provided at the device level and can support only single-level communication channels; the labels are defined during system configuration. The protocols used over the lines are the low-level link layer protocols.

In order to connect the GTNP to another component via the device interface, application software must be written to operate on the GTNP to interact according to some defined protocol between the devices. The GTNP places no requirements on this protocol, other than the fact that the other component must be at the administratively-defined label of the communication channel (this is a side-effect of single-level devices).

2.3.1.2 Virtual Machine Interface

Another way to compose other network components with the GTNP so as to build a complete NTCB is to implement components as virtual machines on the GTNP. The interface between the GTNP M-Component and components implemented as virtual machines on the GTNP is the union of the set of unprivileged machine instructions and the set of kernel gates. Note that this interface is limited to single-level "connections," as all processes executing on the GTNP that are untrusted with respect to the GTNP M-component policy must be single-level. Thus, if a component is to exist at multiple labels, it must have a distinct instantiation on the GTNP at each label or be trusted with respect to the M-NTCB.

The ring mechanism (described in section 4.1.2, page 47) provides a mechanism whereby a complete ring integrity policy may be implemented. The resulting rings may be used to implement multi-state virtual machines. The NTCB partitions within components implemented as virtual machines upon the GTNP are required to use this mechanism correctly to create a subject abstraction in a domain less privileged than the domain of the component partition.

2.3.1.3 Preallocated Address Interface (PAI)

A third way of composing other network components with the GTNP is through a preallocated address interface.[8] In this approach, the other components are named and identified by a unique range of I/O and/or memory addresses, accessible only to the GTNP M-NTCB, on the GTNP bus. In order to manipulate the other component, the applications use the device interface. Thus, an appropriate device driver must be present.[9] The protocols used to communicate with the device are restricted to be single-level instantiations of the standard IEEE 796 MULTIBUS and IBM PC/AT bus protocols.

[8] This interface differs from the device interface in the manner of connection. Devices are attached to connections on the GTNP box and no violation of the physical enclosure is required. In the PAI, components are connected to the bus; as a result, they are within the physical enclosure.

[9] In fact, it must be one of the device drivers covered by this evaluation. Introduction of a new device driver for a specific network component is out of the scope of this evaluation and will invalidate any rating assigned.

2.3.2 Extensions

It should be noted that the NTCB can also be augmented through mechanisms that would require extension of the perimeter of trust provided by the GTNP M-component. This occurs when a system is configured to support multilevel subjects. Such subjects are trusted to behave properly with respect to the system MAC policy within the range to which they are confined.

The Trusted Facility Manual [87] for the evaluated configuration expressly warns that use of subjects that are multilevel with respect to the GTNP's M-NTCB's MAC policy will necessitate re-evaluation of the M-component. The evaluation of the multilevel subject would need to focus on the policy enforced by that subject, and how that subject was protected as a part of the TCB. The integrity of the existing GTNP TCB would be preserved under the addition of such a subject through the combination of both hardware privilege-level separation of the GTNP kernel from the non-kernel TCB and the ring integrity policy that would restrict access to non-kernel TCB code and data structures.

2.4 Evaluated Configuration

Throughout this report, reference is made to the "evaluated configuration" of the GTNP. This configuration is the evaluated version of the software defined by appendix B, page B-1, running on some configuration of the hardware specified in appendix A, page A-1, installed and operated in accordance with the Trusted Facility Manual.

The hardware is defined to the software through the various Programmable Read Only Memories (PROMs): the *Boot PROM* for processor-local information, the *ID PROM* for model-specific information, and the *System PROM* for device information.

To verify that the software corresponds to the evaluated system, distinct distribution keys are used to separate the evaluated from the unevaluatable system. Unevaluatable systems consist of GTNP configurations containing unevaluatable devices, or configurations that allow creation of processes with Ring one subjects, or configurations with unevaluated Ring one code.

Section 3

Hardware Architecture

3.1 Hardware Overview

The GTNP software operates on several Gemini Trusted Base configurations (Models 6, 9, 12, 15, and 26), or on a Gemini-modified[1] IBM PC/AT, or Zenith Z-248. Each Trusted Base system is an expandable system using 80286, i386, or i486 microprocessors to support concurrent computing and multilevel security as well as general purpose computing. The IBM PC/AT and Zenith Z-248 systems are single 80286 microprocessor systems that also support a wide range of computing needs.

The evaluated GTNP hardware base includes the following:

- Gemini Multibus multiple-slot chassis, including the following components:
  - Between one and eight iSBC 486 (with built-in numeric coprocessors), iSBC 386 with Intel387 Numerical Processor Extension (NPX) or iSBC 286 Single Board Computers (SBCs, with optional Intel287 NPX).
  - Between one and eight Intel CX series random-access memory (RAM) boards and/or one or more Central Data CD21/2032 EPROM/RAM boards.
  - One Gemini System Controller (GSC).
  - Optionally, up to eight Intel iSBC 188/56 Serial Input/Output (I/O) Controllers (providing eight serial I/O ports each).
  - Optionally, up to 16 Ethernet/HDLC controller boards (hosted on an iSBC 186/51 Single Board Computer, with an 82586 Local Area Network (LAN) Coprocessor for Ethernet, and a 8274 controller for High-level Data Link Control (HDLC)). The board may be configured such that any combination may be enabled; that is, Ethernet, HDLC, or both.
  - Optionally, up to two hard disk drives (with associated controller).
  - Optionally, up to two floppy disk drives (with associated controller).
  - Optionally, up to four tape drives (with associated controllers).
- IBM PC/AT and AT compatibles (Zenith Z-248), including the following components:
  - A single 80286 microprocessor (with optional Intel287 NPX).
  - Gemini System Controller for PC/AT Compatibles (GSC-AT).
  - A single hard disk drive (with associated controller).
  - Up to two floppy disk drives (with associated controller).

[1] Personal Computer/AT (PC/AT) platforms are modified through the addition of a Gemini System Controller board and replacement of the standard boot Programmable Read-Only Memory (PROM) with a Gemini-developed boot PROM. The Gemini boot PROM runs the necessary portions of the kernel and completely replaces the Basic Input/Output System (BIOS).

- Elephant-12 or Intel Above Board RAM expansions (up to 8 Mbytes).
- Up to two serial I/O ports and one parallel I/O port (provided by Intel Above Board products and/or an IBM serial/parallel adapter. One serial and one parallel port is standard on all Zenith Z-248 systems.).
- An Enhanced Graphics Adapter (EGA) (with associated monitor).
- An 84- or 101-key keyboard.

Unique to both platforms is the Gemini System Controller. This is a board that contains devices that provide special support for the GTNP. This includes the Data Ciphering Processor (DCP); read-only memory for the storage of encryption/decryption keys and other system configuration information; and an internal bus used for transmitting these keys to the DCP without exposing them on the system bus. Additional information on the Gemini System Controller may be found in section 3.3.6, page 36 and section 3.4.4, page 44.

The following sections describe the protection mechanisms provided by the 80286, i386, and i486 microprocessors. This is followed by discussions of the Gemini Trusted Base, and the IBM PC/AT/Zenith Z-248 version of the GTNP. The phrases "Trusted Base," "IBM PC/AT," and "Zenith Z-248" are used when the reference applies to a specific version of the GTNP. The more general phrase "PC-AT platform" is used to refer to all PC/AT compatible systems collectively, and "GTNP" is used in the most generic sense.

3.2 Hardware Protection Mechanisms

The underlying hardware protection mechanisms of GTNP play a vital role in supporting the enforcement of the GTNP security policy. Principally, these protection mechanisms are provided by the GTNP microprocessors, and include the following: the protected mode operating environment, segment and page protection, address translation, task management, and I/O protection. These mechanisms are fundamental to the support of domain separation, process isolation and I/O protection discussed throughout the remainder of this report. This section summarizes the various processor protection mechanisms, and explains how they are used by the GTNP. For additional information on these and other mechanisms of the Intel 80286, i386, and i486 microprocessors, refer to appendix D, page D-1 [80286], appendix E, page E-1 [i386], and appendix F, page F-1 [i486].

The Intel 80*x*86 processor architecture has evolved with the addition of mechanisms and features, while remaining backward compatible. Not all of these mechanisms and features need be used by a security kernel, including protection mechanisms (e.g., the i386 I/O protection bit map). Additionally, some mechanisms and features may be configured differently within various security kernel implementations. Table 3.2 provides a brief summary of various features and mechanisms implemented in the 80*x*86 architecture, and identifies those employed by GTNP, and how they are configured.

3.2.1 Privilege Levels

A hardware *privilege level* (PL), not to be confused with a software *ring* abstraction (see section 4.1.2, page 47), is a mechanism provided by the 80286, i386, or i486 hardware base that implements multiple protection domains. Four hardware PLs are provided, numbered zero to three (PL0-PL3). Privilege levels are organized such that the set of segments accessible for reading and writing in higher numbered PLs are a subset of the

Feature/ Mechanism	**Applicable Processors**	**Used**	**Description of Use in GTNP**
Privilege Level	80286–i486	Yes	Kernel uses PL0; other processes may range in privilege from PL1-PL3.
Segmentation	80286–i486	Yes	One GDT that maps an LDT array, TSS Table, Core Manager data and kernel data area. All descriptors in GDT have a DPL = 0. One LDT per non-kernel process; non-kernel processes do not share LDTs.
Segments > 64 Kbytes	i386–i486	No	Not used by kernel, not virtualized for non-kernel processes.
Multiple segments sharing same physical memory (segment aliasing)	80286–i486	No	Not used by kernel, not virtualized for non-kernel processes.
Direction of segment expansion	80286–i486	No	Not used by kernel, not virtualized for non-kernel processes. The size of segments is fixed at segment creation.
Sharing of LDTs	80286–i486	No	Each LDT is process-specific; multiple processes do not share a common LDT.
Paging	i386–i486	Yes	A static page directory map and page table are setup at system initialization. Processes share the PDT (same **CR3** for all processes) and page tables. Pages are 4 Kbytes in length.
Conforming Segments	80286–i486	Yes	Creation and use available through the GTNP kernel interface.
Task Switching	80286–i486	Yes	Intel task switching mechanism employed by GTNP for all context switches.
Cache	i386–i486	Yes	Caching is enabled and used.
IOPL	80286–i486	Yes	IOPL = 0 for all processes.
IOP bit map	i386–i486	No	Not supported.
Test Registers	i386–i486	No	Not used by kernel, not virtualized for non-kernel processes.
Debug Registers	i386–i486	No	Not used by kernel, not virtualized for non-kernel processes.
V86 Mode Support	80286–i486	No	Not supported.
Floating Point Emulation	80286–i486	No	If coprocessor not-available, FPU instructions raise exception.

Table 3.1. 80x86 Features and Mechanisms Used By GTNP

set of segments accessible for reading and writing in lower numbered PLs. Hence, a task[2] has the greatest access privilege in PL0, and the least in PL3.

Typically, a task can only execute code at its own PL. Gates may be used to transition to more privileged PLs. Conforming segments are an exception to this; see section 3.2.1.2, page 18 and section 4.1.2.1.4, page 50 for more information on this.

In the GTNP, only the three least privileged levels are available outside the kernel (PL0 is used exclusively by the kernel). A task may execute in any one of the three available PLs. Whenever a task introduces a segment into its address space, it must specify a PL for that segment. Once a segment is introduced at a given PL, the hardware allows a task to access that segment only when the task is executing in a PL of equal or greater privilege.[3]

3.2.1.1 Traps and Gates

The hardware of the GTNP supports two methods for transferring control from less-privileged subjects to more-privileged subjects: traps and gates. *Traps* are synchronous interrupts whose handlers are more-privileged subjects. A *gate* is a well-defined interface to a more-privileged subject that is invoked in a similar manner as a subroutine call. See section 3.2.3.2.1, page 21 for more detail.

3.2.1.2 Conforming Segments

When a task introduces a segment into its address space, it must specify a PL for that segment. If the segment was introduced for execution, this specified PL restricts the PL at which the segment may be executed; normal segments may be executed only at the PL at which they were introduced. However, the hardware supports a special class of segments, called *conforming segments*, that may be executed from any PL between the specified PL to the least-privileged PL (PL3). These segments "conform" in the sense that they always execute at the PL of the caller, even if that is different than the PL of the segment. Conforming segments cannot be invoked through gates.

Conforming segments are typically used for code shared between PLs in the same process, such as library functions. The use of this type of segment from software is described in section 4.1.2.1.4, page 50.

Any code segment can potentially be conforming. When a task makes a segment known as a conforming PL segment, the process must indicate: a PL less-privileged or equal to the executable segment; *execute-conf* or *read-execute-conf* access mode; and no gate, since conforming segments cannot be called through gates.

3.2.2 Privileged Instructions

The 80286, i386, and i486 microprocessors support a number of instructions—each of which is either an application or system instruction. All application instructions can be executed by any task, regardless of PL. System instructions can be further broken down into those that are unprivileged, and privileged; instructions

[2] Intel uses the term *task* to refer to a single thread of execution on the 80286, i386, or i486 microprocessors.

[3] This is only the hardware view of what happens, the Trusted Computing Base (TCB) view will be described later in this report. Later, it will be apparent that processes specify ring numbers, and the kernel assigns PLs based on the ring to the segment.

may be more restrictive with regard to the privilege level from which they may execute. Additionally, there exist I/O-sensitive instructions as discussed in section 3.2.4, page 23.

Application instructions are the general set of instructions that can be used to write application software (without privilege). They include data movement, stack manipulation, type conversion, arithmetic, logical, conditional, test, flow control, interrupt, string and character manipulation, flag control, floating point coprocessor interface, and a few miscellaneous instructions.

Unprivileged system instructions are available to all tasks, but are normally not used by application software. The primary examples of this class of instructions are the pointer-parameter validation instructions.

Privileged system instructions are executable only by tasks running in PL0. The instructions in this set are all of the instructions that can affect hardware data structures, and hence control the system. Specific examples are the instructions to load the IDT, LDT, and GDT registers or the instructions to halt the processor.

3.2.3 Segmentation

The 80286, i386, and i486 microprocessors view physical memory as a collection of *segments*, each segment being from 1 byte[4] to 64 Kbytes in size and residing anywhere in the processor's address space. Although the i386 and i486 segment limit can exceed the 64 Kbyte segment limit of the 80286, the GTNP does not use this feature. Segments can be defined to either occupy unique memory or to overlap one another (i.e., multiple segments can share the same physical memory concurrently).[5]

3.2.3.1 Paging

On the i386 and i486, the GTNP operates with the paging mechanism enabled. During kernel initialization, a single page directory map and all of the page tables needed to map linear to physical addresses, including the Multibus address space, are created. The mapping is one-to-one: linear addresses and physical addresses are the same. All tasks share the single page directory map and page tables; the map and tables remain static once initialized. Although the paging mechanism is enabled, it is not utilized to support page swapping.[6]

Descriptors, in the i386 and i486, contain within them a linear address consisting of an index into the page directory map, an index into a page table, and an offset into a physical page frame. **CR3** points to the base of the page directory map, and each entry in the page directory map points to the base of a page table. The page table index portion of the linear address is used to select a page table entry. The page table entry points a physical page frame, which is then combined with the offset of the linear address to form a physical address.

[4] The GTNP does not support 1 byte segments, but rather requires that they be a minimum of 2 bytes in size.
[5] The GTNP does not use the latter capability.
[6] The Kernel was designed to utilize the paging mechanism in order to support use of Virtual 8086 mode. Although Virtual 8086 mode is no longer supported in the current version of the Kernel, the use of paging is retained to ease future enhancements to support virtual memory.

1 bit	2 bits	5 bits	24 bits	16 bits
P	DPL	Type/Access	Base Address	Limit

Figure 3.1. Segment Descriptor

3.2.3.2 Descriptors

A *descriptor* (see figure 3.1, page 20) is a data structure that defines each segment known to the processor. Each segment can only be accessed through its descriptor, which has the following attributes:

P	The present bit.
DPL	The descriptor privilege levels (zero through three). Given the one-to-one correspondence of descriptors to segments, this is also the associated segment's PL.
Type/Access	This field contains the following information: • The segment type (data segment, code segment, special system segment (e.g., descriptor table), or gate). • The type of access allowed: data segments can be read only or read/write; code segments can be execute only or read/execute, and can be conforming or non-conforming; special system segments and gates have special access controls defined below. • The direction the segment will expand (this feature is not used in the GTNP.)
Base Address and Limit	The base address determines where the segment is in memory, and the limit is an offset from the base defining the extent of the segment. (The GTNP uses a 64 Kbyte segment limit on all processors).

All descriptors are physically located in memory-resident tables; each such table is actually a segment itself. The 80286, i386, and i486 microprocessors recognize three types of descriptor tables: the Local Descriptor Table (LDT), the Global Descriptor Table (GDT), and the Interrupt Descriptor Table (IDT).

LDT	LDTs are used for defining process-local references to segments. There can be any number of these in memory, but only one for a given process. The hardware allows multiple processes to share a common LDT in order to share a semi-private, as opposed to completely global, set of segments, however, the GTNP does not use this capability.
GDT	The GDT is used to define system-wide, global segments. Although the GDT is accessible from outside the kernel, all segments described by the GDT in the GTNP are restricted to kernel access only (DPL = 0).
IDT	The IDT is used to store interrupt vectors in the form of interrupt and trap gates. Up to 256 such interrupt and trap gates may be defined in the single IDT.

The GDT, IDT, and LDTs (as well as the Task State Segment (TSS))[7] are not directly accessible by tasks, but rather are accessed indirectly in making reference to segments.

3.2.3.2.1 Gates

Gates are special descriptors used for transferring control indirectly. There are four types of gates: call gates, trap gates, interrupt gates, and task gates. Only call gates are available for user operations.

Call Gate	A *call gate* contains the PL of the gate, base address, and index (defining the entry point) for a code segment.
Trap Gate	A *trap gate* is similar to a call gate, except that it is used in the IDT to specify a trap service routine.
Interrupt Gate	An *interrupt gate* is similar to a call gate, except that, when it is invoked, interrupts are automatically disabled.
Task Gate	A *task gate* contains a descriptor referencing a TSS. Hence, when such a gate is used, a complete context switch (i.e., to the new task definition) occurs.

A call or task gate can be invoked by tasks via hardware instructions (e.g., CALL). However, the gate must be accessible (i.e., at the same or less privileged level) and the code must also be accessible (i.e., at the same or a more privileged level). See figure 3.2, page 22 for examples of valid and invalid inter-privilege level calls. If the code segment is at a more privileged level, then control is transferred to the new level during the execution of such code. In order to transition back out to a less privileged level, the task must return (e.g., RET instruction) from the call.

Interrupt and trap gates must pass a similar protection check as call and task gates when invoked, but are used exclusively in the IDT to handle interrupts (internal or external, maskable or non-maskable) and traps (e.g., a protection violation prevents an instruction from completing normally). Task gates can also be used in the IDT for interrupts that need to be serviced in a different context. Call gates are defined exclusively in LDTs and GDTs to provide a well-defined interface to more privileged functions.

3.2.3.2.2 Aliasing

Recall that the hardware allows multiple descriptors to refer to the same memory segment. An example might be to create a descriptor making a read/writable segment at the same location in memory as the GDT; this would allow a trusted task to alter the contents of the GDT (e.g., adding new segments to it or changing access attributes). A more common example occurs when multiple tasks share a segment. In this case, each task has a descriptor for the shared segment in their LDT. When multiple descriptors refer to the same segment, but with different attributes (other than base or limit), it is referred to as *aliasing*.[8] The GTNP does not utilize nor construct multiple descriptors referring to the same segment that have different bases and/or limits.

[7] A TSS is the system special segment that defines a task.

[8] The term aliasing used here should be differentiated from the term alias used in reference to the GTNP segment naming.

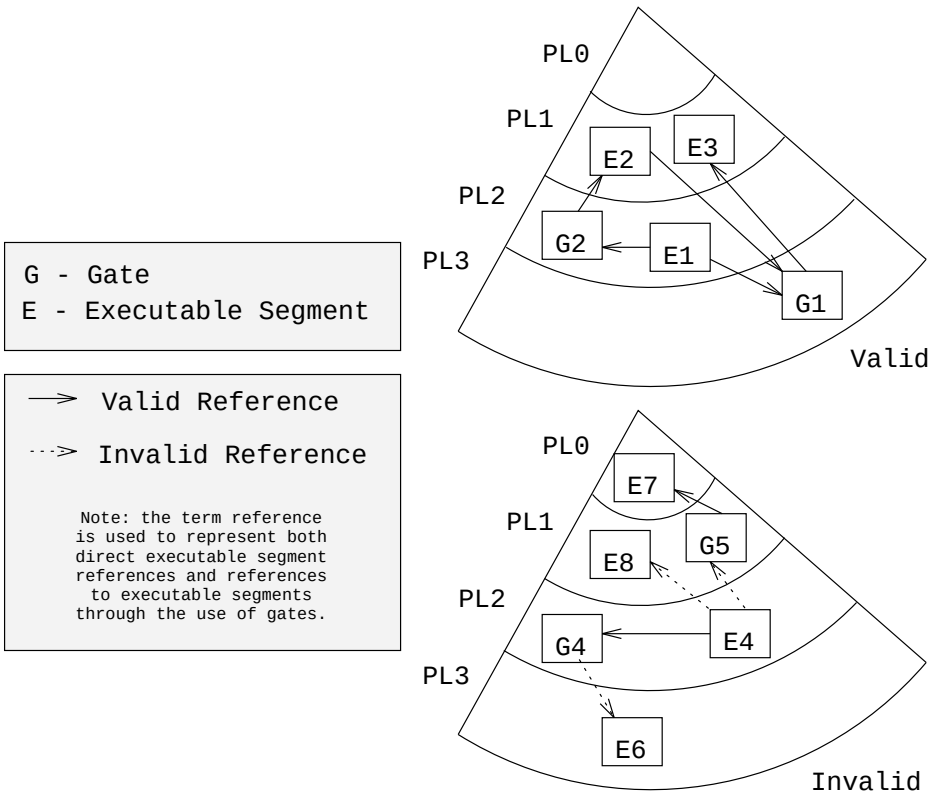


Figure 3.2. Example Valid and Invalid Call Paths

3.2.3.3 Selectors

On the 80286, segment *selectors* are used by tasks to select the descriptor to use in order to reference a segment (see figure 3.3, page 23). (Section 3.2.3.1 discusses address translation on the i386 and i486.) The selectors are made up of an index, a descriptor table selector bit, and a Requested Privilege Level (RPL). The descriptor table selector bit specifies whether to use the GDT or the LDT. The index specifies the descriptor to use within that table. The RPL indicates the privilege level of the access request.[9]

An executing task has access to the following four segment selector registers: Code Segment (**CS**), Data Segment (**DS**), Stack Segment (**SS**), and Extra Segment (**ES**). There are two additional extra segments on the i386 and i486—**FS** and **GS**. Before a segment can be accessed, the selector must be loaded into the corresponding selector register. Hence, this set of segments defines the subset of the task's instantaneous address space. However, the set of segments defined (indirectly through descriptors) in the GDT and LDT define the task's address space (inasmuch as these are the sources of new descriptors that can be selected).

When a selector is loaded into one of the selector registers, it must be well formed and must pass a series of tests:

[9] The GTNP kernel uses RPL only to ensure that the caller is not trying to access a PL0 segment.

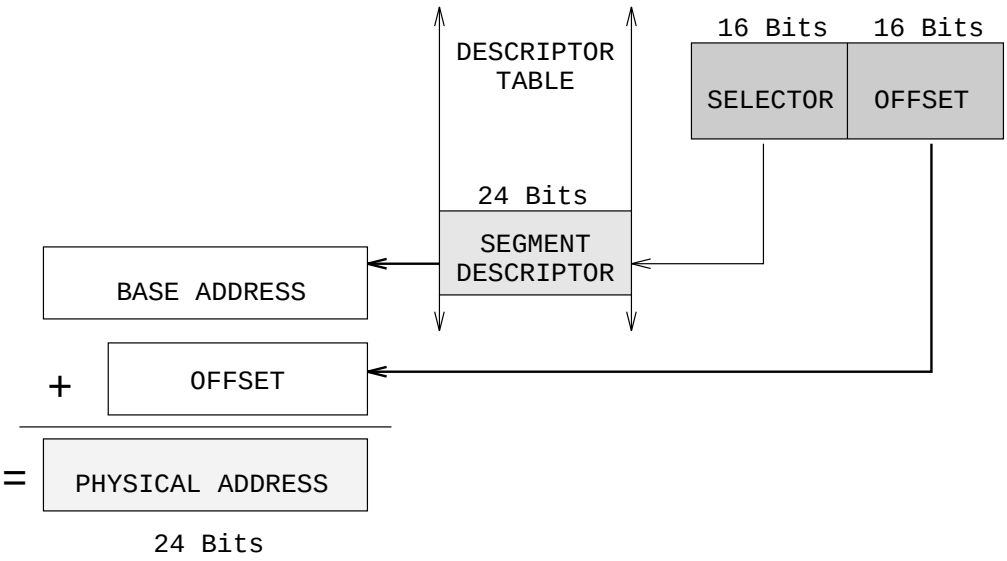


Figure 3.3. Address Translation

- Instructions that load selectors into the **DS**, **ES**, **FS**, and **GS** must refer to data segment descriptors or readable code segment descriptors.
- Instructions that load selectors into the **SS** must refer to writable data segment descriptors.
- Instructions that load selectors into the **CS** (i.e., transfer control) must refer to code segment descriptors.

In all cases, privilege requirements must be met (i.e., read or write access is allowed to the same or less privileged level only; execution is allowed only to the same privilege level). Any violations in the privilege check will result in a general-protection exception and the appropriate trap handler will be invoked.

Also, subsequent accesses through a loaded selector are compared against the base and limit of the segment, and no accesses outside the range of the defined segment are allowed. Again, a general-protection exception will occur.

3.2.4 I/O Protection

The 80286, i386, and i486 microprocessors also provide the ability to restrict use of I/O instructions, forcing tasks to use the kernel interface to perform I/O. Similar to how a task's privilege level determines whether a *privileged* instruction executes successfully, the IOPL flag, located in the **FLAGS** register, determines whether an I/O instruction can be executed by the current task. Such instructions are referred to as *IOPL-sensitive*. Tasks wishing to perform I/O instructions must be operating at a privilege level at least as privileged as that specified within the IOPL flag. The following instructions are IOPL-sensitive: IN, INS, OUT, OUTS, CLI, and STI.

In the GTNP, the IOPL flag is set to PL0, requiring tasks that perform I/O to be operating at PL0. Non-PL0 tasks are thus restricted to using the GTNP kernel gates to perform I/O. Additionally, the i386 and i486 introduce an I/O permission bit map, increasing the ability to restrict I/O to the granularity of an individual I/O port. This feature is not used by the GTNP.

3.3 Gemini Trusted Base Platform

Each Trusted Base uses an IEEE 796 Multibus-I backplane (with either 12 or 26 slots) as a base to support a maximum of three or eight microcomputers, depending on model and configuration. Each processor board may have a maximum of 8 or 16 Mbytes of local RAM and 1 or 2 two serial I/O (SIO) ports per processor (depending on processor type). There may also be up to 8 Mbytes (depending on processor type) of shared (global) memory. Additionally, slots on the Multibus may be used to support a maximum of 64 additional (shared) SIO ports, and up to 2 each of any of the following: hard disk, floppy disk, 1/4″ cartridge (QIC), and 1/2″ tape drives. A variety of manufacturer's options are available, such as an HDLC or Ethernet controller.

The following hardware features are offered on most Trusted Bases:

- One or more SBCs with 80286, i386, or i486 microprocessors. Gemini uses the designation *Standard* for the iSBC 286/10 and 286/10A; HP for iSBC 286/12, 286/14, and 286/16 processors; SP for the i386-based processors; and UP for the i486-based processors.
- A minimum of 1 Mbyte of local memory, per SBC.
- A minimum of 1/2 Mbyte of global memory. Standard and HP systems support a maximum of 7 3/4 Mbytes of global memory, with 1/4 Mbyte reserved for CPU PROMs. The SP and UP systems support a maximum of 8 Mbytes of global memory; on these platforms, the PROMs reside at the top of the fourth Gbyte page (i.e., $\mathrm{FFF80000}_{16}$).
- A math coprocessor.[10]
- Multiple RS-232 compatible SIO ports.
- One Gemini System Controller that provides a real-time clock, non-volatile storage of security parameters (such as the security labels of the devices on the system, stored in the System PROM), a Data Encryption Device, a programmable timer, and the bus control for up to 16 bus masters.
- One Direct-Current (DC) power supply sufficient to support a fully-loaded card cage and four disk drives.

In addition to the Multibus, the 80286-based CPU boards support the Intel Local Bus eXtension (iLBX). This bus is a high-speed bus that can be used for off-board local memory expansion on 80286-based CPUs. The number of iLBX boards allowed per 80286 CPU varies based on the model of the chassis. The models 6, 9, 12 and 15 allow one board per CPU; the model 26 allows one or two boards per CPU (depending on

[10] There are two series of Trusted Bases, original and "A" series, in the evaluated configuration. For Standard and HP processors in the original series, the math coprocessor is optional. The original series also uses the Seagate Technology 506 (ST506/ST412) disk interface, whereas the "A" series uses the Enhanced Small Device Interface (ESDI) interface. The original series is no longer available from Gemini, but is still in use in fielded systems.

board location). Each board provides 1/2 to 2 Mbytes of local memory. Thus, a HP CPU with 4 Mbytes of on-board memory could be expanded to a maximum of 8 Mbytes in a model 26, and 6 Mbytes in all other models. A Standard CPU can have a maximum of 4 Mbytes in a model 26, and 2 Mbytes in all other models.

The following sections provide more detail regarding the boards that can be connected either directly or indirectly to the Multibus.

3.3.1 Standard and HP (80286) Processors

The Standard and HP processors supported in the GTNP can be viewed simply as extensions to the basic 80286 architecture (see appendix D, page D-1). The Standard processor requires one slot; HP processors require two slots on the Multibus, and potentially another slot for optional expansion memory.

The Standard processors are the iSBC 286/10 and 286/10A. The 286/10 operates at a speed of 6 Megahertz (MHz); the 286/10A at 8 MHz. Both standard processors have no on-board memory, and require the use of the iLBX memory expansion (which takes an additional chassis slot). Additionally, the 286/10 lacks the Intel Bus Arbiter, and uses a piggy-back bus arbiter that takes an additional chassis slot instead.

The HP processors are shipped in the following configurations. Other than memory capacity, there are no functional differences between the models.

- iSBC 286/12: 1 Mbyte local RAM.
- iSBC 286/14: 2 Mbytes local RAM.
- iSBC 286/16: 4 Mbytes local RAM.

The iSBC 286 SBC [106] is a single circuit board consisting of an 80286 microprocessor and supporting circuitry. The board provides standard IEEE 796 Multibus connectors, with an iLBX bus interface, and two RS-232 SIO connectors. Each connector will be described in more detail later.

Briefly, the features available with the board, in addition to the set of 80286 features (see appendix D, page D-1), are as follows:

- A clock speed of 8 MHz (except for the 286/10, which runs at 6 MHz).
- An Intel287 NPX (optional on the original series Standard and HP boards).
- Two RS-232 programmable asynchronous serial I/O (ASIO) interfaces.
- For the HP processor, a "piggy-back" erasable programmable read-only memory (EPROM) Expansion Module with four sites (1/4 Mbytes each). These serve as boot PROMs.[11] The Standard processor has four on-board PROM sites.
- For the HP processor, byte-parity generation and checking Dynamic RAM (DRAM).
- An iLBX interface for local memory expansion.
- A Multibus interface using 7 3/4 Mbytes of global memory and a full 64-Kbyte I/O address space.

[11] Gemini uses their own boot PROM.

- 15 levels of vectored interrupt control.
- An 8254 Programmable Interrupt Timer (PIT).

3.3.1.1 Serial I/O (SIO) Interface

The SIO consists of an 8274 Multiple Protocol Serial Controller (MPSC) device controlling two 26-pin serial connectors. Each connector is connected to one of the two channels (A and B respectively) provided by the MPSC; because the two channels are independent they can operate concurrently.

Channel A is configured as an RS-232 data circuit-terminating equipment (DCE) interface by default. It may also be configured (via its programmable controller chip) as an RS-232 data terminal equipment (DTE). Channel B is configured as an RS-232 DCE interface and cannot be altered.

The MPSC may be programmed to operate in either an interrupt-driven mode or a polled-driven mode. By default, the MPSC operates in interrupt-driven mode; an interrupt request signal is activated when either channel needs service.

3.3.1.2 On-Board Memory

The on-board memory consists of EPROM (i.e., the Boot PROM) and DRAM[12] that is accessible only by the on-board processor. The EPROM devices contain the bootstrap routine for the board and must always be located at the highest addresses in the CPU memory address space. One, two, or four Mbyte of on-board zero-wait-state "local" memory is supported by the HP CPUs. The on-board memory is implemented on a piggy-back RAM module that extends into an adjacent Multibus slot.

The iLBX bus interface is used to allow the processor to access a full 8 Mbytes of local memory space. The memory is provided by the iSBC 0*xx*CX series RAM boards, described in section 3.3.4.1, page 31. As the HP CPUs have on-board zero-wait-state memory, the iLBX memory expansion is optional for these processor boards. Due to the lack of on-board RAM, the standard processors require memory expansion using this interface; hence two Multibus slots are used.

3.3.1.3 Multibus Interface

The iSBC 286 uses its Multibus interface to provide and/or access various Multibus control signals during bus arbitration, data transfer, and interrupt servicing. For Multibus compatibility, the board uses the 82289 bus arbiter and 82288 bus controller. The following list includes the Multibus signals used in GTNP systems:

Bus arbitration signals	Bus clock, bus priority in (BPRN), bus busy, bus request (BREQ), and common bus request.
Bus control signals	Memory read command, memory write command, I/O read command, I/O write command, transfer acknowledge, lock, reset (called initialize), and bus clock.

[12]HP processors only.

Address and data signals	Address lines 0 through 23, byte high enable, and data lines 0 through 15.
Interrupt signals	Interrupt 0 through 7 and interrupt acknowledge.

The processor boards are capable of 16-bit (word) and 8-bit (byte) data exchanges.

3.3.1.4 Controller Subsystem

The Controller Subsystem controls on-board and Multibus memory and I/O cycles for the local CPU. It also performs on-board tasks, such as refreshing memory and generating the READY signal.

Part of the controller subsystem arbitrates for the local memory between the on-board CPU and the refresh logic. The refresh logic always gets priority over all other requests. During a transfer cycle, the requesting agent controls memory; that control cannot change during the cycle (i.e., the transfer must complete).

3.3.1.5 I/O and Interrupt Control

A PIT provides three 16-bit timers:

- One connects to the master interrupt controller; it is used to generate real-time interrupts.
- One connects to the slave Programmable Interrupt Controller (PIC); it is used to generate real-time interrupts or to time operations.
- One connects to the transmit and receive clocks on the serial controller (see below); it is used to generate various baud rates.

Two PICs are arranged as a master and slave, processing up to 15 interrupt levels (see figure 3.4, page 28). Three of the interrupts have dedicated purposes, leaving 13 interrupts that can be configured (not counting the Non-Maskable Interrupt (NMI)) on the CPU. Only the PICs operate in direct-vectored mode.

The PICs monitor all interrupt signals from the on-board devices and the Multibus interface except Multibus interrupt two (INT2). INT2 is mapped directly to NMI. In addition, interrupt level 7 (IR7) on the master PIC is connected to the slave PIC (an on-board direct-vectored interrupt device). A direct-vectored interrupt device is one that can generate an interrupt signal to the master PIC and pass an interrupt vector to the processor.

3.3.2 SP (i386) Processors

The SP processors supported in the GTNP can be viewed simply as extensions to the basic i386 architecture (see appendix E, page E-1). Each requires two slots on the Multibus. The i386 SBCs are shipped in the configuration supporting 2, 4, or 8 Mbytes of local memory, with clock speeds of either 16 or 20 MHz. No iLBX expansion memory is available with these processors.

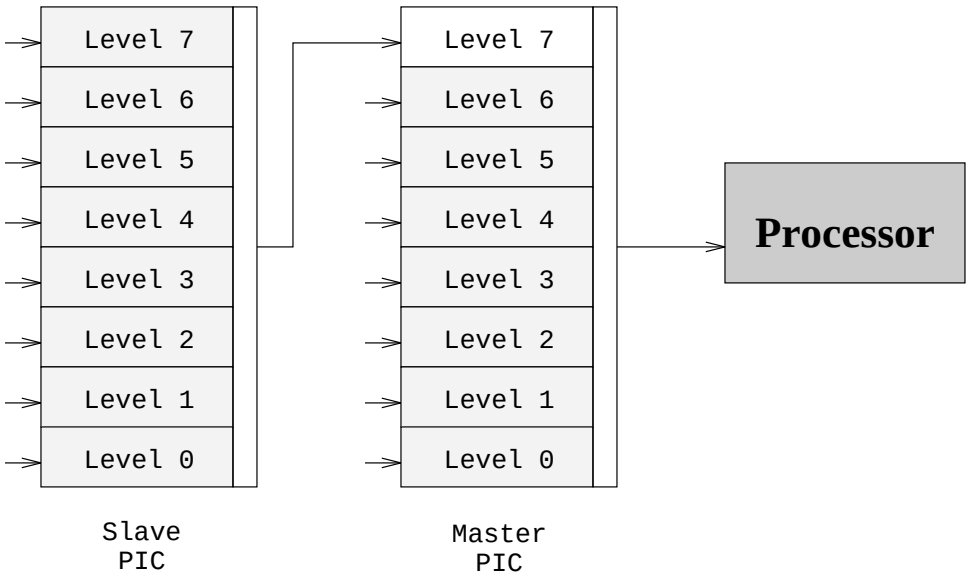


Figure 3.4. Cascaded PIC Devices

The iSBC 386 SBC [108] is a single circuit board with a piggy-back memory module consisting of an i386DX microprocessor, an Intel387 math coprocessor, and supporting circuitry. The board provides standard IEEE 796 Multibus connectors and one asynchronous RS-232 SIO connector. Each connector is described in more detail later. The features available with the board, in addition to the set of i386 features, are as follows:

- A clock speed of 16 or 20 MHz.
- Two EPROM sites (up to 512 Kbytes on-board).
- 64 Kbytes of Static RAM (SRAM) cache memory.
- Byte-parity generation and checking of Dynamic RAM (DRAM).
- Multibus interface using the full 8-Mbyte global memory space and full 64-Kbyte I/O address space.
- One RS-232 SIO asynchronous interface.
- Two programmable timers for system timer interrupts.
- An 8254 programmable interval timer.
- Numerical coprocessing with the Intel387.
- On-board control registers for signaling errors, configuring the board, and reporting board status.
- Two PICs providing a total of 15 interrupts, 13 of which are jumper-configurable.

3.3.2.1 Serial I/O (SIO) Interface

The serial I/O interface for the i386 boards provides a single asynchronous RS-232 SIO port.

3.3.2.2 On-Board Memory

The on-board memory includes EPROM, DRAM, and SRAM cache memory. The EPROM contains bootstrap code. DRAM is volatile memory that is configured as local memory. Only the i386 can use local DRAM. SRAM cache memory automatically stores some of the same information that is in the on-board DRAM (i.e., it caches local DRAM for performance reasons and does not cache global memory).

Anywhere from 8K×8 to 256K×8 EPROMs are supported. In real mode, during initialization, the EPROM occupies the highest portion of the processor's 1-Mbyte memory space. In PVAM, the EPROM occupies the highest portion of memory in the processor's 4-Gbyte memory space.

Support is present for 2, 4, or 8 Mbytes of local DRAM.[13] Any i386 accesses to memory addresses above 16 Mbytes will map to the first 16 Mbytes (with the exception of EPROM accesses). For example, an access to the $17th$ Mbyte or $33rd$ Mbyte will go to the $1st$ Mbyte. In the GTNP, all on-board DRAM is local memory. The GTNP does not configure any DRAM as dual-port DRAM (which would allow access from the Multibus). Only the CPU can transfer 32-bit data streams; data transfers across the Multibus are 8 or 16 bits wide.

The SRAM is a 64-Kbyte write-through cache (between the DRAM and CPU) supporting 16,384 32-bit entries, each of which is associated with an 8-bit tag field. The tag is in a smaller bank of SRAM that holds addresses,[14] but no data. Address bits 2 through 15 are the index to these fields; hence all DRAM locations that have those address bits in common will map to the same data field (but each byte of DRAM maps to only a single data field in the cache).

3.3.2.3 Multibus Interface

The Multibus Interface is the same as that supported on the iSBC 286. See section 3.3.1.3, page 26 for more information.

3.3.2.4 Controller Subsystem

The Controller Subsystem controls on-board and Multibus memory and I/O cycles for the local CPU. It also performs on-board tasks, such as refreshing memory and generating the READY signal.

Part of the controller subsystem arbitrates for the local memory between the on-board CPU and the refresh logic. The refresh logic always gets priority over all other requests. During a transfer cycle, the requesting agent controls memory; that control cannot change during the cycle (i.e., the transfer must complete).

[13] The local memory of this board is implemented on a piggy-back module, hence two Multibus slots are required for this board.

[14] Actually, it holds the upper eight bits of the address, so that a 24-bit address space is supported.

3.3.2.5 I/O and Interrupt Control

A PIT provides three 16-bit timers:

- One connects to the master interrupt controller; it is used to generate real-time interrupts.
- One connects to the slave PIC; it is used to generate real-time interrupts or to time operations.
- One connects to the transmit and receive clocks on the serial controller (see below); it is used to generate various baud rates.

Two PICs are arranged as a master and slave, processing up to 15 interrupt levels (see figure 3.4, page 28). Three of the interrupts have dedicated purposes, leaving 13 interrupts that can be configured (not counting the NMI) on the CPU. Only the PICs operate in direct-vectored mode.

The iSBC 386 board has six control registers to control the status of the system. These registers allow control of functions such as indicating whether the CPU is operating in PVAM, resetting Multibus interrupts, and turning the on-board memory cache on and off.

3.3.3 UP (i486) Processors

The i486 processor supported in the GTNP is an extension of the basic i486 architecture (see appendix F, page F-1), and is compatible with the 80286 and i386 processors. The GTNP uses the iSBC 486/12 SBC computer to provide the i486 processor. This board contains the following:

- Intel486 DX2 processor (66 MHz internal, 33 MHz external).
- Minimum local memory: 8 Mbytes; Maximum local memory: 16 Mbytes.
- Two EPROM sites (up to 1/2 Mbytes of EPROM).
- An 8 Kbyte internal cache.
- Byte-parity generation and checking DRAM.
- A Multibus interface using the full 8-Mbyte global memory space and the full 64-Kbyte I/O address space.
- Two RS-232 asynchronous SIO interfaces.
- Two programmable timers for system timer interrupts.
- One 8254 PIT.
- On-chip Intel487 numerical coprocessor.
- Two PICs providing 15 interrupts, 13 of which are jumper-configurable.

From the board perspective, the i486 is a performance-enhanced version of the 80286. The primary difference is the increased clock speed of 33/66 MHz. The i486 also has an 8-Kbyte instruction and data cache on the processor chip. In addition, the floating point coprocessor has been moved onto the processor chip.

3.3.3.1 Serial I/O (SIO) Interface

The SIO interface on the i486 processor board is the same as that used on the 80286 processor boards. This interface is described in section 3.3.1.1, page 26.

3.3.3.2 On-Board Memory

The On-Board Memory available on the i486 processor board is the same as available on the i386 processor boards (described in section 3.3.2.2, page 29), with the following exceptions:

- The i486 supports 8 or 16 Mbytes of RAM.
- The i486 has 8 Kbytes of cache (on the i486 chip).

3.3.3.3 Multibus Interface

The Multibus Interface is the same as that supported on the iSBC 286 (described in section 3.3.1.3, page 26), except that the i486 board uses two Programmable Array Logic (PAL) chips, instead of the 82288 and 82289 chips.

3.3.3.4 Controller Subsystem

The Controller Subsystem on the i486 processor board is the same as is used on the 80286 processor boards. This subsystem is described in section 3.3.1.4, page 27.

3.3.3.5 I/O and Interrupt Control

The I/O and Interrupt Control mechanisms on the i486 processor board is the same as is used on the 80286 processor boards. These mechanisms are described in section 3.3.1.5, page 27.

3.3.4 Memory Boards

Memory expansion boards available for Gemini Trusted Bases are categorized as either volatile or non-volatile. Volatile RAM is used for either local or global memory, while non-volatile RAM is used exclusively for global memory.

3.3.4.1 Volatile RAM

Both processor local and global memory can be provided by iSBC CX-Series RAM Boards [105]. Each board provides a dynamic memory storage capacity of 512 Kbytes, 1024 Kbytes, or 2048 Kbytes and can support 8- or 16-bit Multibus or iLBX bus accesses. DRAM chips are used, and error checking and correcting (ECC) circuitry is provided on-board. These boards are both Multibus and iLBX bus compatible.

ECC is accomplished with the Intel 8206 error detection and correction unit along with other supporting circuitry. The ECC allows for the detection of single-bit errors, double-bit errors, and most multiple-bit errors. On all of these boards, the ECC is programmed to correct single-bit errors (double-bit and multiple-bit errors are not correctable). Also, the ECC is programmed by the GTNP to interrupt the processor on non-correctable errors. Communication between the ECC and processor is done through a single byte I/O port. This port is used for control status and error status. It is mapped to one of eight memory locations, selectable by on-board jumpers.

The maximum system (Multibus I) memory size supported by this series of boards is 16 Mbytes (8 Mbytes local and 8 Mbytes global).

The 8-Mbyte Multibus global address space is divided up into two 4-Mbyte pages. On-board jumpers are used to specify which of the two 4-Mbyte pages into which the memory is to be placed. Further, each page is partitioned into 256 16-Kbyte blocks, with on-board jumpers selecting a base address, on a 16-Kbyte boundary, within the 4-Mbyte address space. Hence, the board will provide a contiguous series of 16-Kbyte RAM blocks beginning at the specified base address within the specified page. The RAM provided by a board cannot extend beyond a page boundary.

The iLBX address space is divided up into 128 64-Kbyte partitions from 0_{16} to $7E0000_{16}$. On-board jumpers are used to select a base address, on a 64-Kbyte boundary, within the address space.

In the Trusted Base, the Multibus is used for global memory (8 Mbytes) and on-board memory or the iLBX bus is used for local memory (8 Mbytes). Due to the number of iLBX slots bussed, a maximum of 4 Mbytes is accessible in the 8-Mbyte local memory range. The CX-series memory boards do not provide the capability to completely disable the Multibus addressing. Therefore, in order to provide isolated local processor memory, the Multibus base address is set to an inaccessible Multibus address (000000_{16}); this disallows access to local memory via the Multibus (which is significant for isolation in a multiprocessor system).

3.3.4.2 Non-volatile RAM

The Central Data CD21/2032 High Density EPROM/RAM Board (CD21) provides additional global memory. The board can accept a variety of different EPROMs or Complementary Metal-Oxide Semiconductor (CMOS) SRAMs. The CD21 has the following functionality:

- Thirty-two sockets capable of accepting EPROM or SRAM.
- Battery Backup (SRAM only).
- 24-Bit Multibus addressing.
- Joint Electronic Device Engineering Committee (JEDEC) pin standards.

The CD21 can operate in either 8- or 16-bit data mode depending on the Multibus mode. It allows a system-wide Multibus address space of 16 Mbytes of which only the upper 8 Mbytes is accessible in Gemini's configuration.

3.3.5 I/O Boards

The Gemini Trusted Bases support I/O boards that provide SIO ports and Ethernet and RS-422 HDLC connections. Each board is described below.

3.3.5.1 iSBC 188/56 Extended Serial I/O Board

The iSBC 188/56 Extended Serial I/O Board is a communications board consisting of the iAPX 80188 processor and supporting circuitry. The following are the features that are used by Gemini:

- Memory-mapped I/O controller.
- Support for RS-232 interfaces on eight channels.
- Four 82530 Serial Communication Controllers (SCC)s with two channels each supporting a total of eight channels.
- Nine on-board Direct Memory Access (DMA) channels.
- iAPX 80188 processor, with a clock speed of 8 MHz.
- RAM: 256 Kbytes, of which 48 Kbytes are dual-ported.

The iSBC 188/56 can be conceptually divided into the following three major subsystems: processor, memory, and I/O support, which are described below.

3.3.5.1.1 Processor Subsystem

The iSBC 188/56 uses an iAPX 80188 operating at 8 MHz as its CPU. The iAPX 80188 has an 8-bit data bus interface with a 16-bit internal architecture that enables the processor to provide the following functionality: clock generator, two DMA channels, programmable interrupt controller, and 1 Mbyte direct addressing.

3.3.5.1.2 On-Board Memory

The iSBC 188/56 contains 1 Mbyte of address space that is accessible by the iAPX 80188. This space is divided into four 256-Kbyte blocks: local memory (0_{16} to $3FFFF_{16}$), unused (40000_{16} to $7FFFF_{16}$), Multibus window (80000_{16} to $BFFFF_{16}$, unused by the GTNP) and universal site memory (EPROM $C0000_{16}$ to $FFFFF_{16}$), of which only the top 64 Kbytes are used by the GTNP as shown in figure 3.5, page 34. In Gemini's configuration, the 256 Kbytes of Multibus access memory is disabled, as the board is not configured as a bus master. Universal site memory consists of two 28-pin sockets, one of which is used for the 188/56's boot PROM.

The iSBC 188/56 has DMA controllers that support nine DMA channels. Only seven of the DMA channels can be used with the SIO chips. DMA operations are limited to SCC-to-memory transfers.

3.3.5.1.3 I/O Subsystem

The I/O Subsystem of the iSBC 188/56 can be broken down into the following three major areas: serial I/O, local parallel I/O, and DMA functionality.

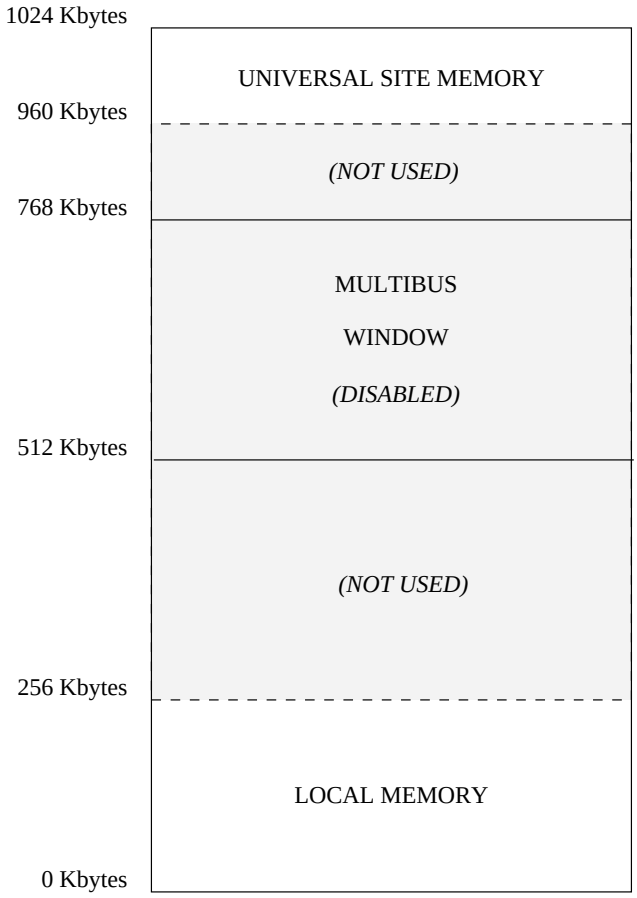


Figure 3.5. iSBC 188/56 Local Memory Space

3.3.5.1.3.1 Serial I/O

The iSBC 188/56 has four on-board SCCs that provide eight channels of half/full duplex serial I/O. These channels are configured to RS-232.

3.3.5.1.3.2 Local Parallel I/O

The iSBC 188/56 has an on-board 8-bit parallel output port, and an 8-bit input port for on-board functions. A programmable latch is used to input various DMA, interrupt, and controller control signals through this port.

3.3.5.1.3.3 Direct Memory Access (DMA)

The iSBC 188/56 provides DMA functionality by making use of two DMA controllers that reside on the board and two of the iAPX 80188's controllers. Each of the on-board controllers has four independent channels; however, the iSBC 188/56 uses a master-slave DMA controller implementation that limits the number of DMA channels to seven. The on-board DMA controllers are used to service SCC components only; while the iAPX 80188 internal DMA controllers are used for memory-to-memory transfers. Hence, only seven of the eight RS-232 channels have half-duplex DMA (read). The remaining channel has no DMA.

3.3.5.2 Ethernet/HDLC Network Controller Base

The Ethernet/HDLC Network Controller Base (NCB) is a commercial-off-the-shelf (COTS) Intel 186/51 Ethernet Communications Computer. It contains a number of Intel-authorized modifications specifically for Gemini. A maximum of 16 NCBs are supported (providing 16 Expandable Ethernet (EETH) devices and/or 16 Expandable HDLC (EHDLC) devices).

The iSBC 186/51 IEEE 802.3/Ethernet Communication Computer [44, 118] (hereafter referred to as the iSBC 186/51) is designed around an Intel 16-bit 80186 microprocessor and the Intel 82586 LAN coprocessor. The following are the features that are used by Gemini:[15]

- Memory-mapped I/O controller.
- 80186 processor, with a clock speed of 8 MHz.
- One Ethernet interface, utilizing an Intel 82586 LAN coprocessor (for Ethernet).
- Local Memory: 128 Kbytes, of which 64 Kbytes are dual-ported to the Multibus.
- One RS-422/449 HDLC channel.
- Additional Board-local Memory: 128 Kbytes.

A flag byte is available to the Multibus; this allows signalling between the NCB and the general purpose processors.

3.3.5.2.1 Processor

The iSBC 186/51 uses an 80186 operating as its CPU. The 80186 is a 16-bit microprocessor that, with supporting circuitry, provides the following functionality: clock generator, two DMA channels, three timers, and 1 Mbyte direct addressing.

3.3.5.2.2 Memory

The iSBC 186/51 has 128 Kbytes of local RAM that is configured by Gemini with 64-Kbyte dual-ported to the Multibus. Further, up to 192 Kbytes of RAM, Read-Only Memory (ROM), or EPROM can be installed using the six JEDEC sockets provided. Gemini uses four of the available sockets. All of this memory is available to both the 80186 and the 82586 LAN coprocessor, with the 80186 performing the necessary arbitration.

[15] The iSBC 186/51 also provides two serial I/O ports that are not used in the GTNP.

3.3.5.2.3 Ethernet Interface

The Ethernet interface consists of an 82586 LAN coprocessor and an Ethernet interface.

The 82586 LAN coprocessor is an intelligent, high-performance local communications controller. It is designed to relieve the 80186 microprocessor of many of the tasks associated with controlling a local network.

Upon command from the 80186, the 82586 LAN coprocessor moves data from memory, gains access to the Ethernet serial interface, formats the data into packets, and then initiates data transmission. In a receive operation, upon command from the 80186, the LAN coprocessor detects the beginning of a packet, performs address checking, decomposes the packet, then moves data to memory. The 80186 has no further involvement beyond the initial command.

3.3.5.2.4 HDLC Interface

The HDLC interface consists of an 8274 Multi-Protocol Serial Controller (MPSC) and the RS-422 differential receivers and drivers. The 8274 MPSC is a high-speed serial communications controller with multiple serial protocols. The GTNP initializes this controller to provide an HDLC Bit Synchronous protocol. Data is moved to and from the MPSC by the two DMA channels located on the 80186.

The HDLC data packets are formed by the general purpose secure processors (80286, i386, or i486) and written to the NCB through circular buffers in the NCB's dual-ported memory.

The HDLC ports are physically accessed via DC37 connectors that conform to the RS-449 specification.

3.3.6 Gemini System Controller

The Gemini System Controller for the Trusted Base contains devices that support the GTNP by performing data encryption and bus arbitration operations. The devices that make up the GSC include: the Data Ciphering Processor (DCP); an EPROM containing non-changeable encryption/decryption keys and other static system information; a CMOS RAM with battery back-up containing changeable encryption/decryption keys and other dynamic information; a real-time clock (RTC) used for periodic interrupts and time stamps; and a PIT.

3.3.6.1 On-Board Memory

The GSC has support for three types of on-board memory devices:

System ID PROM This is a read-only memory that can't be modified by system generation software. It contains keys for the DCP and permanent system data. It is implemented using the EPROM on the GSC board.

System PROM This is non-volatile (read/write) memory that contains passwords for users of system generation software (offline administrators and operators), access labels, RAM Disk definitions, disk volume definitions, interrupt condition definitions and shared memory locations. When configured in accordance with the Trusted Facility Manual, System PROM is accessible only by the operating system and

system generation tools provided by Gemini. It is implemented using the battery-backed SRAM, and is addressed through the I/O space as it has a static address.

Auxiliary PROM This is a socket reserved for future expansion.

3.3.6.2 Bus Interfaces

The GSC has two Multibus connections:

- One connection is used for Multibus I/O-mapped communications to and from the GSC. Other Multibus devices can communicate with the GSC by issuing writes and reads to the specified I/O locations.
- One connection is a customized interface that is connected to Gemini's Priority Controller. The priority controller is used to determine the bus priority resolution for all bus masters.

3.3.6.3 Data Ciphering Processor

The DCP, which is implemented via a dedicated Z8068 [172] , is responsible for encrypting/decrypting data (domestic systems only) and producing cryptographic seals (both domestic and export systems).

3.3.7 Peripheral Device Controllers

There are essentially two types of peripheral devices in the Trusted Base: disks and tapes. Consequently, there are two types of device controllers. The following sections describe the set of disk controllers and the set of tape controllers that are available for the Trusted Base, as well as the devices that can be connected to those controllers.

3.3.7.1 Disk Controllers

The Storager controllers are the only disk controllers in the Trusted Base. There are four models of Storager controllers in the evaluated configuration (Storager I, Storager II, Storager II-2, and Storager III). All of these controllers are essentially the same with regards to functionality, but the Storager III has differences in memory availability and can interface to a more limited set of peripheral devices. Subsequently in this section, the term "Storager" will imply Storager I, Storager II, or Storager II-2, and "Storager III" will refer to the remaining controller. The Storager controller is used in the original series of the Trusted Bases; the Storager III is used in the "A" series of the Trusted Base.

3.3.7.1.1 Storager and Storager III

All Storager controllers [125] are intelligent multifunction Winchester disk/tape controllers. They are based on the 68000 microprocessor and are designed for use with the Multibus. They can support up to two ST506 Winchester hard disk drives, one QIC tape drive, and any mix of up to two 5 1/4″ floppy disk drives operating in double-, or high-density modes (360-Kbyte or 1.2-Mbyte).

The 68000-based intelligence of Storager provides a multitasking, virtual-buffer architecture environment. Both disk and tape operations function concurrently and independently. Also, disk and tape operations can proceed independent of Multibus activity.

Two bipolar state machines are used to manage high-speed data streams. They operate independently to allow for concurrent data movement between the Storager, bus, and any peripherals. Although the controller also supports a queue mode of operation (where up to fourteen operations may be pending), the GTNP does not use this mode.

The following list describes the major functionality of the Storager controllers:

Virtual Buffer Architecture:	Each Storager has 16 Kbytes of on-board memory. At any given instance, individual buffers may be allocated to either a disk, tape, or the Multibus. The 68000 microprocessor dynamically allocates and deallocates buffers as they are requested or released by any of the devices listed above.[16]
Zero Latency:	Conventional controllers, when a multisector request is made, will wait until the first requested sector is encountered before beginning to read and transfer data. A Storager, in contrast, begins reading data as soon as the heads are floating over the track and begins transferring data as soon as it encounters a sector of interest.
Intelligent Caching:	When a Storager has completed a read operation and transferred the requested data, it will continue to read sectors until all available buffers are full or another request is received from the host. Thus, if subsequent requests are for logically contiguous sectors, the requests can be fulfilled immediately without necessitating further media reference.

Communication between requesting devices and a Storager is handled via two interfaces: the Input/Output Parameter Block (IOPB) and a set of registers. IOPBs are stored in global memory (i.e., memory that is common to the requesting device (trusted processor) and a Storager) and are used to list the parameters that define the exact transfer function to be performed. A Storager provides two sets of four 8-bit registers (**R0**-**R3** and **R0′**-**R3′**). These registers are I/O mapped into the Multibus I/O address space as specified by on-board jumpers.

The registers allow commands and status to be communicated between a requesting device and the Storager, and also allow the requesting device to indicate where the IOPB is located. The second set of registers (**R0′**-**R3′**) is provided to enable concurrent tape and disk operations. The second set of registers can be disabled if concurrent operations are not necessary. In this case, all operations may still be performed, but not simultaneously.

After a request, the Storager operates in DMA fashion. When the Storager completes a transfer, the status and error registers are updated and an interrupt is sent. The status register indicates if the transfer was successful. If the transfer was not successful, the error register indicates the reason with an appropriate error code. If the transfer was successful, the error register is updated in order to provide information that might be helpful in predicting future problems or in indicating device degradation. The lowest four error

[16] It should be noted that no buffer will be written to a device unless it has been written by the requester, nor will any buffer be given to (read by) a requester unless it has been read from a device (i.e., there is a write-before-read type of information flow scheme; hence object reuse is not a problem).

register bits indicate the number of retries required to complete the transfer. Bit 7 indicates whether error correction was applied. Bit 6 indicates whether a restore and reseek sequence was required. Bit 4 indicates whether bad data (uncorrectable) has been moved into system memory. The Storager performs automatic error correction via a 32-bit ECC associated with each sector of information.

3.3.7.1.2 Storager III

The Storager III [126] supports up to two enhanced small device (ESDI) Winchester hard disk drives, two QIC tape drives, and any mix of up to two 5 1/4″ floppy disk drives operating in double or high density modes (360-Kbyte or 1.2-Mbyte). The ESDI interface is described in the ESDI Specification [154]. The only other significant difference between this and earlier versions of Storager is in the memory available for the virtual buffer architecture. Storager III provides 12 Kbytes of on-board memory for tape devices and 48 Kbytes for disk, whereas earlier versions simply provided 16 Kbytes for use with all devices. This larger disk buffer permits full track caching from the larger capacity (380- and 760-Mbyte) ESDI drives.

3.3.7.2 Tape Controllers

The Storager/Storager III controllers and Tapemaster 1000 are the only tape controllers in the Trusted Base.

3.3.7.2.1 Storager and Storager III

The Storager and Storager III controllers can also serve as a QIC streaming[17] tape controller, as described in section 3.3.7.1.1, page 37.

3.3.7.2.2 Tapemaster 1000

The Tapemaster 1000 [5] is an intelligent high-performance 1/2″ magnetic tape drive controller that is designed to be compatible with the Multibus. It controls one to eight *start/stop* or *streaming tape* drives conforming to the industry standard interface specification. The Tapemaster 1000 accommodates either an 8-bit or 16-bit data bus, and 24-bit Multibus memory addressing capability.

The Tapemaster 1000 controller is controlled through sixteen 1-byte I/O-mapped read/write registers (at switch selectable addresses).[18] Tape commands are executed by means of parameter blocks resident in system memory (at programmable addresses). All data transfers to the Trusted Base are performed via DMA. Also, once an operation is requested, the device marks itself *busy* (with a bit in one of the registers) and will not accept further requests until that one has completed.

3.3.7.3 Storage Devices

The following sections describe the hard disk, floppy disk, and tape drives that can be used in the Trusted Base.

[17] A streaming tape drive is designed to run the whole length of the tape, normally without any interruption.

[18] As with all other memory mapped device controllers, the memory used for this mapping is global and is protected by the GTNP kernel.

3.3.7.3.1 Disk Drives

Each of the hard disks in the Trusted Base is connected via the Storager controller using industry standard interfaces; Storager (I, II, and II-2) supports ST506 disk drives, while Storager III supports ESDI disk drives. Each of the floppy disks in the Trusted Base is also connected via any of the Storager controllers. Table 3.2, page 40 includes the set of disk drives supported by the Trusted Base.

Product Name	Interface	Platter Surfaces	Size of Platter	Approximate Capacity
——— Hard Disk Drives ———				
Maxtor XT-1085	ST506	8	5 1/4″	85 Mbytes
Maxtor XT-1140	ST506	15	5 1/4″	140 Mbytes
Sequel (Maxtor) XT4170E	ESDI	5	5 1/4″	106 Mbytes
Seagate WREN III (94216-106)	ESDI	5	5 1/4″	106 Mbytes
Seagate WREN VI (94246-383)	ESDI	7	5 1/4″	383 Mbytes
Seagate WREN VI (94196-766)	ESDI	15	5 1/4″	766 Mbytes
Newbury NDR-1140	ST506	15	5 1/4″	140 Mbytes
——— Floppy Disk Drives ———				
Epson SD-580L	—	2	5 1/4″	1.2 Mbytes
Epson SD-621L	—	2	5 1/4″	360 Kbytes
Epson SD-680L	—	2	5 1/4″	1.2 Mbytes

Table 3.2. Trusted Base Disk Drives

3.3.7.3.2 Tape Drives

Each tape drive in the Trusted Base is connected via either the Storager controller (see section 3.3.7.2.1, page 39), in the case of QIC drives, or the Tapemaster 1000 (see section 3.3.7.2.2, page 39), in the case of 1/2″ drives; both tape drives use industry standard interfaces. Table 3.3, page 40 includes the media dimensions and approximate capacity for each tape drive supported by the Trusted Base.

Product Name	Tape Size	Tracks	Tape Length	Capacity
Tandberg Data TDC 3630 QIC	1/4″	15	600′	125 Mbytes
Tandberg Data TDC 3650 QIC	1/4″	15	600′	125 Mbytes
Kennedy Model 9600A	1/2″	9	200 – 3600′	1.9 – 138 Mbytes

Table 3.3. Streaming Reel and Cartridge Drive Capacities

3.4 PC-AT Platform

This section describes the architecture of the GTNP PC-AT platform. There are two machines (models) included in the PC-AT platform (IBM PC/AT [100, 101, 102, 103], and Zenith Z-248 [171]) that are essentially the same (e.g., they use largely similar processors and make use of the same peripheral controllers and devices). First a number of peripheral controllers and devices that are usable with each model of the PC-AT

Name	Base Memory	Piggy-back Memory	Total Memory	Parallel Ports	Serial Ports
Above Board PS/AT	1.5 Mbytes +	2 Mbytes	= 3.5 Mbytes	1	1
Above Board 286	2 Mbytes +	2 Mbytes	= 4 Mbytes	0	0
Above Board PS/286	2 Mbytes +	2 Mbytes	= 4 Mbytes	1	1
Above Board Plus	2 Mbytes +	6 Mbytes	= 8 Mbytes	0	0
Above Board Plus I/O	2 Mbytes +	6 Mbytes	= 8 Mbytes	1	1
Above Board Plus 8	8 Mbytes +	6 Mbytes	= 14 Mbytes	0	0
Above Board Plus 8 I/O	8 Mbytes +	6 Mbytes	= 14 Mbytes	1	1

Table 3.4. Above Board Products Summary

platform will be described, and then each model will be described separately. The specific models included in the evaluation are listed in appendix A.2.4, page A-7.

3.4.1 Memory and I/O Boards

There are a number of boards available that can be installed to extend the memory and I/O capabilities of the IBM PC/AT.

3.4.1.1 AMI Elephant-12 (16-bit slot)

The AMI Elephant-12 [1] supports 0 to 12 Mbytes of DRAM (in 2-Mbyte increments) on a single card. The board utilizes a set of jumpers and a programmable array logic chip to establish the address space that the memory will occupy and decode addresses to it. For each byte of data on the board, the Elephant-12 also provides one bit for parity checking.

The memory on this board is used as extended memory. This means that the memory on this board must be configured to occupy addresses above 1 Mbyte and below 14 Mbytes (100000_{16} to $CFFFFF_{16}$).

3.4.1.2 Intel Above Board Products (16-bit slot)

Each of the Intel Above Board products (i.e., Above Board PS/AT, Above Board 286, Above Board PS/286, Above Board Plus, Above Board Plus I/O, Above Board Plus 8, and Above Board Plus 8 I/O) is a single card that can be used to extend the memory, and possibly I/O capabilities, of the IBM PC/AT.

Each board supports up to some maximum amount of memory, located on the board itself or an optional piggy-back module, (see table 3.4, page 41) that must be configured to occupy addresses above 1 Mbyte and below 14 Mbytes (100000_{16} to $CFFFFF_{16}$). This board is used as extended memory.

In addition, some of the boards provide both a serial and a parallel I/O port (see table 3.4, page 41). Each of these ports can be defined to be associated with the various I/O instructions supported by the 80286, or can be disabled. Further, the serial port is fully programmable; allowing software to control the protocol (e.g., parity and baud rate) that will be used by the port.

Multiple Above Board products can be used in conjunction to further increase the amount of memory and number of I/O ports available. The upper bound is based on the address space reserved for the GSC-AT ($D00000_{16}$ to $F7FFFF_{16}$) and the 80286 boot PROMs ($F80000_{16}$ to $FFFFFF_{16}$ when operating in PVAM).

3.4.1.3 Zenith Memory Board (16-bit slot)

This board is usable with Zenith Z-248 models only, and allows up to an additional 1 1/8 Mbytes. The 1/8 Mbyte portion of the memory is used to expand the processor memory from 512 Kbytes to 640 Kbytes. The 1 Mbyte portion is used as expanded (100000_{16} to $CFFFFF_{16}$) memory.

3.4.1.4 IBM PC/AT Memory Expansion Options (16-bit slot)

The IBM PC/AT 128 Kbyte Memory Expansion Option [103] can be used to provide memory in the address space between 512 Kbytes and 640 Kbytes. The board operates only in this address range, and will also generate an NMI if a parity error occurs when memory on the board is referenced.

3.4.1.5 IBM PC/AT Serial/Parallel Adapter (8-bit slot)

The IBM PC/AT Serial Parallel Adapter [103] provides both a parallel port and a serial port, and connects to the system via a system-board expansion slot.

The serial portion of the adapter is fully programmable and supports asynchronous communications. It provides its own baud rate generator, and allows for a number of baud rate, character size, and stop bit combinations. All of the control and data registers are I/O-mapped on the bus (as defined by jumpers), and therefore map to 80286 I/O instructions.

The parallel portion of the adapter makes it possible to attach to devices that accept eight bits of parallel data. Its control and data registers are also I/O-mapped (as defined by jumpers).[19]

3.4.2 Disk Controllers and Drives

The following disk controllers and disk drives are supported by the PC-AT platform.

3.4.2.1 IBM PC/AT Fixed Disk and Diskette Drive Adapter

The IBM PC/AT Fixed Disk and Diskette Drive Adapter [102] connects to the system board using one of the system expansion slots. The adapter controls 5 1/4″ floppy drives and hard disk drives. Connectors on the adapter supply all the signals necessary to operate two hard disk drives and two floppy drives. The adapter allows concurrent data operations on one floppy and one hard disk drive.

[19] This port allows parallel (Centronics) printers to be used by the GTNP. Note that there are no printers in the evaluated configuration.

The hard disk function features 512-byte sectors; high-speed, programmed I/O data transfers; ECC correction of up to five bits on data fields; multiple sector operations across track and cylinder boundaries; and on-board diagnostic tests. The adapter will support 2 hard disks with up to 16 read/write heads and 1024 tracks.

Further, a number of registers are used to control and allow data transfers to and from the adapter (and therefore the disk drive). These registers are I/O-mapped on the bus and therefore map to I/O instructions of the 80286. Both the primary and secondary hard disk drives are controlled through primary and secondary sets of these registers (i.e., there are two independent sets).

The 5 1/4″ floppy drive function is an integral part of this adapter. One or two floppy drives can be attached to the adapter through an internal daisy-chained cable. The adapter will support 320/360-Kbyte and 1.2-Mbyte floppy drives.

There are two registers for both the primary and secondary floppy drives. One is read-only and is used to indicate controller status, and the other is a read/write data register used to move data to and from the floppy. As with the hard disk registers, these registers are also mapped to I/O instructions of the 80286 (selected by jumpers).

Both functions of this adapter allow either programmed I/O[20] or DMA.[21]

3.4.2.2 PC-AT Platform Disk Drives

Each disk drive for the PC-AT platform, whether hard or floppy, is a direct-access device; the interfaces allow control of the drive motor and read/write head, as well as transfer of data. Further, each disk drive is compatible with the IBM PC/AT Fixed Disk and Diskette Drive Adapter (see section 3.4.2.1, page 42).

Table 3.5, page 43 includes the media dimensions and approximate data capacity for each disk drive.

Product Name	Platter Surfaces	Size of Platter	Approximate Capacity
——— Hard Disk Drives ———			
IBM PC/AT 20MB	4	5 1/4″	20 Mbytes
Seagate ST4026	4	5 1/4″	20 Mbytes
——— Floppy Disk Drives ———			
IBM PC/AT Double-Sided	2	5 1/4″	320 or 360 Kbytes
IBM PC/AT High-Density	2	5 1/4″	1.2 Mbytes

Table 3.5. PC-AT Platform Disk Drives

3.4.2.3 Other Disk Drives

The following hard and floppy disk drives are also supported in the PC-AT platforms: Maxtor XT-1085, Maxtor XT-1140, Epson SD-680L, Epson SD-580L, and Epson SD-621L (see section 3.3.7.3.1, page 40 for descriptions of these disk drives).

All of these disk drives can also be used with the IBM PC/AT Fixed Disk and Diskette Drive Adapter.

[20] Programmed I/O transfers to the hard disk are 16-bit; all others are 8-bit.
[21] The GTNP uses only DMA.

3.4.3 Video Controllers

3.4.3.1 IBM Enhanced Graphics Adapter (EGA)

The IBM EGA is a display controller that supports color and monochrome displays. The EGA contains 64 Kbytes of storage that are used as a display buffer. This memory is protected by segmented memory and other hardware protection mechanisms.

3.4.4 Gemini System Controller for PC Based Systems

The Gemini System Controller for the IBM PC/AT (GSC-AT) is a board that has the same general functionality and features as the GSC board (see section 3.3.6, page 36), but is adapted to the IBM PC/AT. The differences in these boards are the I/O addresses that access the board; the physical interrupt mapping; some minor interface differences for mutually supported devices; and the addition of a parallel port latch.

3.4.5 Specific PC-AT Platform Models

The following sections briefly describe each of the PC-AT platform models. For both PC-AT platform models, Gemini replaces the boot PROM with a Gemini-developed boot PROM.

3.4.5.1 IBM PC/AT

The IBM PC/AT is the system for which the PC-AT platform was defined. An IBM PC/AT is essentially a box consisting of an 80286 microprocessor, and its supporting logic (i.e., the motherboard), and a bus. The bus allows 16-bit and 8-bit peripheral devices (e.g., memory and I/O boards) and device controllers (e.g., disk controllers) to be made available, as resources, to the central processor.

3.4.5.2 Zenith Z-248

The Zenith Z-248 PC Computer is an IBM PC/AT compatible computer. The GSC-AT can run on this system with new boot PROMs that are installed in the processor board.

Section 4

Software Architecture

The GTNP software consists of five logical pieces. Many of these pieces share design and code with the GEMSOS product under development. These five pieces are as follows:

1. *GEMSOS Security Kernel.* The kernel is the heart of the TCB and executes at Privilege Level 0 (PL0) and provides the Mandatory Access Control (MAC) policy enforcement for the GTNP.

2. *GEMSOS Security Kernel Initialization (GKI).* GKI executes just prior to the operational kernel, and is responsible for determining the operational system configuration and bringing the kernel to a secure initial state. It also plays a role in trusted distribution. As it executes before the GTNP enters an operational state, it is considered to be system generation software.

3. *GEMSOS Kernel Gate Library (KGL).* KGL is a collection of routines that serve as an interface between the kernel gates (invoked by traps) and high-level programming languages such as Pascal or C.

4. The *GEMSOS Network Processor System Parent (NPSP).* The NPSP executes in Ring 1 (R1) at PL1, and is the TCB component responsible for starting up the initial untrusted application processes that will be running on the GTNP. The KGL is linked into the NPSP to provide an interface to the kernel gates, and thus is also part of the TCB. The NPSP also includes the run-time intersegment linkage mechanisms provided by the GEMSOS Multilevel Subjects Intersegment Linkage Tool (MIT).

   Note that the evaluated version of the NPSP prevents application processes from creating R1 subjects; this is because a R1 subject could modify the NPSP data (and hence violate the requirement for TCB isolation).

5. *System Generation Utilities.* The system generation utilities do not execute as part of the TCB, but are built upon the same kernel base and execute on the same hardware platforms. These utilities must be started ("booted") from a logical volume distinct from that of the operational system. These utilities provide a means to define the system configuration, ensure the integrity of the system configuration, generate the system from distribution media, and aid in recovery from system failure.

Figure 2.2, page 11, illustrates the GTNP TCB. As shown in this figure, the TCB of the system includes the kernel, KGL, and the NPSP (including portions of the MIT). All remaining code in the operational system is application code and is not part of the TCB. With the evaluated product is also delivered a minimal amount of application code; this code includes the GEMSOS Applications Generator Utility (APGEN),[1] Model Application Environment (MAE), and the GTNP Acceptance Tests.

The picture does not show the portions of the GTNP (GKI and the system generation utilities) used to support achievement of a secure initial state. These portions execute either during system generation, or

[1] APGEN is an applications generator, similiar to GEMSOS System Generation Utility, SYSGEN.

just before the TCB.[2] Both of these portions are considered to be system generation utilities, and not a part of the TCB. They are both maintained under strict configuration management; and are subject to a subset of the system architecture requirements.

The following sections describe the software architecture of the GTNP. The discussion begins with a presentation of some background concepts that need to be understood in order to comprehend the software architecture. This is followed by discussions of the kernel (including GKI), the KGL, and the NPSP architectures. Following that is a discussion of the system generation utilities.

4.1 Background Concepts

The basic goal of the design of the GTNP was to implement a high-assurance reference monitor on top of a hardware base consisting of multiple tightly-coupled microprocessors that provide support for multiple privilege levels. This goal led to a number of concepts that appear throughout the architecture of the GTNP: the notion of access labels; execution domains, rings, and privilege levels; virtual processors; segment aliasing; volumes; synchronization and signals; object label tranquility, devices, and the use of encryption and cryptosealing. These concepts are discussed below.

4.1.1 Labels

The GTNP is designed to enforce mandatory access control restrictions between the subjects and objects under its control. To do this, labels are associated with all subjects and objects; these labels are used to make the mandatory control decisions. In the GTNP, this label is called an *access label*; with the specific portions of the access label that specify the lattice-based access policy being called an *access class*.[3]

An access label is more than just the conventional sensitivity levels and categories. In the GTNP, an access label contains the following information (the first two forming the access class):

- Two hierarchical levels (16 levels each).
- Two sets of non-hierarchical categories (providing a total of 96 potential categories divided between secrecy and integrity).
- Three ring bracket fields.
- A gate field that specifies the gate jump type.[4]

The algorithm used to compare access labels is discussed in more detail in section 4.2.1.6, page 70. Information on labels and their use may be found in section 6.1, page 109.

[2] GKI is the only software portion that executes just before the TCB. In practice, GKI is not made unavailable since it is required to reinitialize the system after a successful shutdown. However, the kernel does not invoke GKI during online operation (except during the reinitialization after a successful shutdown); nor can it be accessed by untrusted subjects because of ring bracket restrictions.

[3] That is, the formalisms for the MAC policy enforcement are based on the access class.

[4] For more information on this, see section 4.2.2.1.4, page 86.

4.1.2 Execution Domains

In the GTNP, a *process* is a software abstraction, constructed from hardware tasks, that is characterized by a single execution point and an address space defined by the process' Local Descriptor Table (LDT) and Global Descriptor Table (GDT).[5] A process is made up of multiple *subjects*, who share the process' LDT but have different access abilities to the segments therein. Each subject thus executes with a different set of accessible objects that defines that subject's *domain*. The set of potentially accessible objects is determined by the *ring* of the subject. The set of potentially accessible rings of subjects within a process are nested, with the set of potentially accessible rings of more-privileged subjects containing the set of potentially accessible rings of the less-privileged subjects.

The GTNP uses a system of eight rings numbered from zero to seven. Each subject in a process is associated with a single ring. The set of objects accessible to subjects associated with ring j (Rj) are also accessible to subjects associated with ring i, where $j > i$. This access can be further constrained by other security controls (e.g., MAC). As a result of this, a process may be executing in one of eight potential domains; the particular domain being defined by the ring associated with the currently active Privilege Level (PL) of the process. In the GTNP, subjects are defined as process/domain pairs; thus, for any process, there are up to eight potential subjects (each one corresponding to a ring). A process can be thought of as a single thread of execution, with the process changing rings to have access to different domains, or as a collection of subjects, with the subjects communicating among themselves.

A subject whose domain is bounded by ring i is said to execute in ring i or is called a *ring i subject*. A subject executing in a more privileged (i.e., lower numbered) ring has greater access privilege than a subject executing in a less privileged ring. An R0 subject, therefore, has the most access privilege within a process; conversely, a R7 subject has the least access privilege.

Note that the scope of access of a domain may include multiple rings; since a subject in a given ring has access to entities in less-privileged rings. As a result of this, subjects in more-privileged rings are considered to be *trusted with respect to a ring integrity policy* across a range of protection rings extending from their ring of execution to the least privileged ring in the system.

The GTNP uses three of the four hardware privilege levels (PL1, PL2, and PL3) provided by the 80286, i386, or i486 processors (see section 3.2.1, page 16) to implement and enforce the eight-ring hierarchy (see figure 4.1, page 48).[6] This means that only three of the eight possible subjects in a process can be active (i.e., mapped to a privilege level) at any one time. The initial set of active subjects is specified when a process is created. This specification identifies the subject (by ring number) that will occupy each privilege level. Once a process is created, the initial set of active subjects may be changed only if the subject in PL1 takes explicit action to do so by invoking the `k_activate_subject` gate (see section 4.2.2.3.4, page 90). The `k_activate_subject` gate only allows the active subjects that occupy PL2 and PL3 to be changed.

One restriction that is always enforced is that when a ring is mapped to a privilege level, that ring must be more privileged than all of the rings that are currently mapped to less privileged levels in that process. In other words, the ring-privilege mappings must be properly ordered so that the hardware privilege level mechanism will enforce the ring hierarchy.

[5] All processes share the same GDT, which references only kernel segments.

[6] PL0 is reserved for the GTNP kernel. The kernel executes in PL0 and provides the ring abstraction, thus it does not map to any ring.

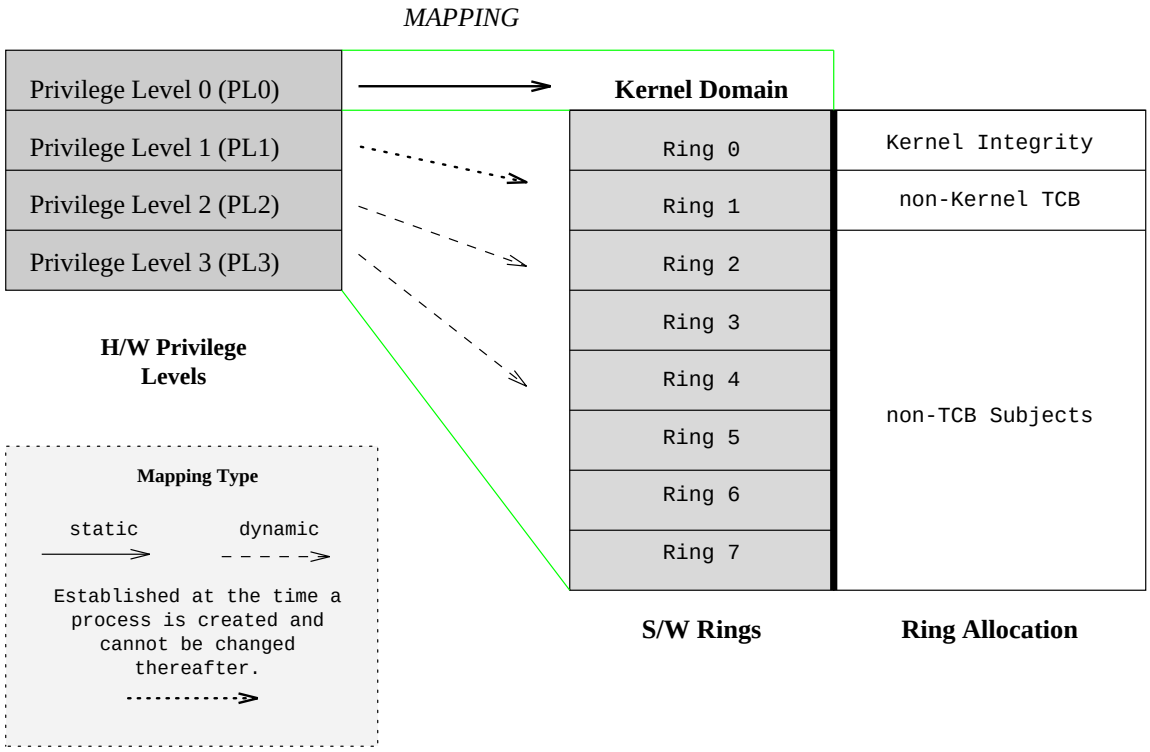


Figure 4.1. Privilege Level to Ring Mapping

When a process requests that a segment be brought into its address space (see section 4.1.4.2, page 56 for a description of this process), it may specify the ring number of any active subject (thus indicating the least privileged subject that may access the segment). The kernel selects a PL for the segment.

4.1.2.1 The Ring Mechanism

The following paragraphs address some specific details related to the Gemini ring mechanism as implemented in the GTNP.

4.1.2.1.1 Ring Usage

The GTNP is designed as an M-Component. Rings are used in the GTNP as follows:

R0 No online processes contain active R0 subjects. This is because R0 is used to protect the integrity of the kernel code and data. The kernel is not considered to be a subject operating in R0, since it is the kernel that provides the ring abstraction. Rather, the kernel is distributed in the address space of every process, where it runs in PL0 and has access to all rings. Calls

to the kernel are viewed as invocations of the reference monitor rather than communication between subjects.

Active R0 subjects exist in some of the system generation utilities that are run when the system is non-operational. These utilities must also have a legitimate need to directly access the segments that make up the kernel; an example is the GTNP SYSGEN utility used to generate bootable GTNP configurations. These tools are part of the mechanism that sets up a secure initial state for the GTNP. The R0 subject in such tools always runs in hardware PL1.

R1 In the operational GTNP, R1 is reserved for the NPSP. Subjects in other processes cannot be assigned to R1 (for this undermines the integrity of the NPSP data structures labeled so as to restrict access to R1). This restriction is enforced by the evaluated version of NPSP, which does not allow redefinition of the R1 subject.

Active R1 subjects exist in some of the system generation utilities that are run when the system is non-operational. These utilities have a legitimate need to directly access NPSP data structures and System Programmable Read-Only Memory (PROM) I/O devices that are not accessible by subjects executing outside R1. An example of this usage are the GTNP Configuration Tools that define the application processes to be started by the NPSP. The R1 subject in such tools always runs in PL1.

R2-R4 These rings are not specifically used in the GTNP.

R5-R7 These rings are allocated to untrusted subjects in the GTNP.

4.1.2.1.2 Ring Brackets

Ring brackets on segments (analogous to those found in Multics) further restrict modes of access rather than assigning specific domains. A segment's ring brackets define the ranges of rings in which subjects have the ability to introduce a segment with a specific access mode. Every segment is given a set of ring brackets when it is created. These ring brackets are part of the information used by the security kernel to validate requested accesses as part of the process of introducing segments into a process' address space. The ring brackets are assigned to each segment at the time of creation and are fixed thereafter.

A segment's ring brackets are encoded in a set of three ring numbers (RB1, RB2, and RB3, where RB1 $\leq$ RB2 and RB1 $\leq$ RB3). For most segments, RB1 $\leq$ RB2 $\leq$ RB3. Figure 4.2, page 50 shows the meanings of the possible patterns of ring brackets. Gemini has additional encodings for the case RB3 $<$ RB2.

A segment may be introduced by a process for write access in the most privileged ring (R0) out to the ring whose number is denoted by RB1. This range (0–RB1) is known as the *write bracket*. There is a corresponding *ring bracket* that defines the allowed ring numbers into which a segment may be introduced for read access. The ring bracket's lower end is R0; the upper end varies depending on the relationship between RB1, RB2, and RB3 as shown in figure 4.2, page 50.

The range RB1 to RB2 defines the *execute bracket* for a segment. Since RB1 is both the maximum ring number allowed write access and the minimum ring number allowed execute access, there is an implicit mandatory program integrity mechanism (i.e., executable segments that contain executable code cannot be maliciously modified by subjects in lower privileged rings).

The range RB2+1 to RB3 is the *gate extension*. Gates are not a form of access; they are a mechanism to allow processes to transfer execution from lesser privileged rings to more privileged rings in a manner

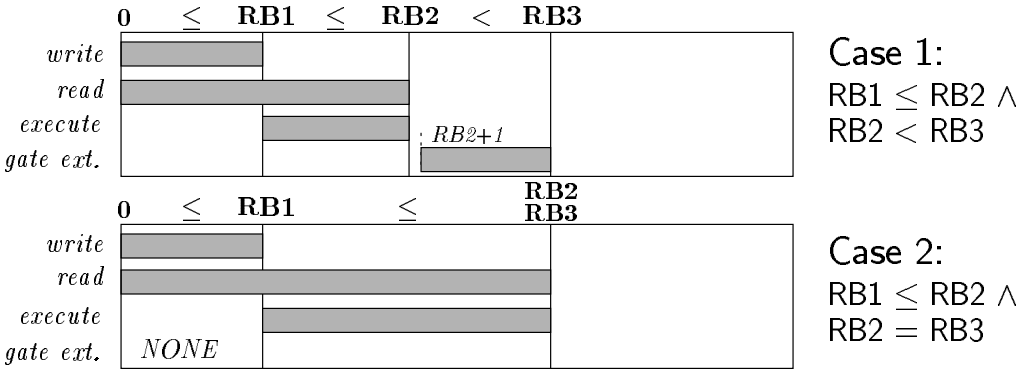


Figure 4.2. Ring Brackets

controlled by the more privileged ring. Gates are described more fully in section 3.2.3.2.1, page 21; briefly, a subject with execute access to a segment can set up a *gate* accessible by less-privileged subjects running in rings delimited by the gate extension. These less-privileged subjects cannot execute the segment directly, but can use the gate to invoke the more-privileged subject. The process will change rings, from that of the less-privileged to that of the more-privileged, and begin execution at a controlled location in its code segment.

Because a segment can be directly executed by a subject from RB1 up through RB2, only subjects in rings greater than RB2 need to use a gate to execute routines in the segment. The GTNP provides a limit (RB3) that specifies the least privileged ring that may use the gate. Subjects in rings less privileged (numerically greater than) RB3 may never directly access the segment, nor directly call through a gate associated with the segment. No segment with RB2 $\geq$ RB3 may have a gate, because the gate extension bracket is obviously empty (RB2+1 to RB3).

4.1.2.1.3 Using Ring Brackets

When a segment is introduced, the GTNP checks the segment's ring brackets to determine if the segment may be accessed with the specified access mode in the specified ring. If the kernel determines that a violation has not occurred, it makes the segment available by establishing the actual limits to be enforced (e.g., PL) by the hardware.

It is possible for a segment's ring brackets to permit the segment to be introduced simultaneously by two processes in two different rings with two different access modes. For example, a segment with the correct ring brackets could be introduced in R1 with read-write access by one process, and in R2 with read-only access by another. Note that each of these access modes must be allowed based on the MAC and ring policy relationships between the subject and the segment.

4.1.2.1.4 Conforming Segments

A segment introduced with read-execute or execute-only access may be executed only by the process when it executes in the ring to which the segment is assigned. A special class of executable segments are defined in

the processor hardware architecture. Such segments, called *conforming segments* (see section 3.2.1.2, page 18), may be introduced so that they can be executed from a given ring out through the ring of the least privileged subject defined for a process. The most privileged ring that may execute the segment is specified when the process introduces the segment into its address space.

In order for a segment to be used as a conforming segment, its RB2 must equal the ring of the least privileged subject that is allowed to execute the segment. No conforming segment, therefore, may ever be called as a gate in that process, since the gate extension starts at RB2+1. When a process introduces a segment as a conforming segment, the process must specify a ring number greater than or equal to the segment's RB1, and an access mode of execute-conforming or read-execute-conforming.

Conforming segments are useful to save memory. For example, simple library functions could be shared across a broad range of subjects in a process without needing individual copies. This also reduces unnecessary ring crossing.

4.1.2.2 Subjects in Rings

A process that moves from a less-privileged ring to a more privileged ring gains the additional access capabilities of the more-privileged ring. The GTNP provides two hardware-supported methods for processes to move from less-privileged to more privileged rings: *traps* and *gates* (see section 3.2.3.2.1, page 21).

4.1.2.2.1 Traps

All traps result in execution of code at PL0. The kernel is designed to handle two types of traps. Most traps (the first type) are reflected out to the subject executing at PL1 (there is always a PL1 subject). There are two traps (invoked via the INT instruction) that are the second kind of traps. If (and only if) the trap occured at PL1, these are specifically interpreted by the kernel's trap handler as requests for kernel services.

4.1.2.2.2 Gates

A gate is a well-defined interface that allows a less-privileged subject to invoke a more-privileged subject. Gates are more efficient than traps for changing rings, since no "side-trip" through the security kernel is necessary to invoke a gate. Gates also allow hardware-assisted parameter passing from less-privileged to more-privileged rings.

Each gate is set up by a more-privileged subject to control invocation by subjects executing in less-privileged rings. A less-privileged subject calls the gate by pushing parameters onto its stack, and then issuing a gate call. The gate call copies a fixed number of words of parameter data from the less-privileged stack onto the stack of the more-privileged subject (the number of words, the starting location in the code segment, and the ring number of the more-privileged subject are set when the gate is created). Additional information can be passed in registers. The process then changes rings to the more-privileged subject, and execution starts at the specified location. When the called subject is finished, the RET or IRET hardware instruction is used to clean up the stack and return execution to the less-privileged subject.

In general, the only mechanism for moving from a ring with higher privilege (lower numerical value) to a ring with lesser privilege (higher numerical value) is by using one of the hardware return instructions. When this is done, to ensure continued execution, the subject in the higher-privileged ring must ensure that the

Central Processing Unit (CPU) registers contain only information specifically intended to be visible to the subject in the lower privileged ring.[7]

The return mechanism cannot be used to move to a higher-privileged ring. When the process executes the return instruction, the hardware checks the privilege level of the segment in the return address popped off the stack (the privilege level is listed in the segment descriptor table). Only if the privilege level of the return address segment is greater than or equal to the process' current privilege level will the hardware automatically assign to the process the access privileges associated with the privilege level of the return address. Otherwise, the hardware will trap any attempt by a process to return to a more-privileged ring.

For example, assume a process has a R5 subject in PL2 and a R7 subject in PL3. If the process executing in R5 executes a RET instruction, the hardware checks the return address. If the return address is in a PL3 segment, the hardware recognizes this and changes the access privileges to conform with the PL3 subject. This way, the process moves from R5 to R7 without the intervention of the security kernel.

4.1.3 Virtual Processors

In the GTNP, physical processors[8] are not directly visible to processes. Rather, the GTNP kernel virtualizes the physical processors into a number of Virtual Processors (VPs), each of which supports a number of processes. This is illustrated in figure 4.3, page 53. This architecture allows for the design of a loop-free scheduler, where the blocking of virtual processors waiting for resources can be hidden from the portion of the kernel that schedules processes onto virtual processors.

4.1.3.1 Virtual Processor Multiplexing

VPs are scheduled much like processes in a more conventional system. They can be in one of three states (see the right side of figure 4.4, page 53): READY, RUNNING, or BLOCKED. Each VP also has a priority. When a VP is RUNNING, it executes until preempted by a higher priority VP (going to READY) or until it needs to wait on an event (going to BLOCKED). When BLOCKED, it waits until the pending event happens and then moves to the READY state. Lastly, when READY, it moves to the RUNNING state whenever no higher priority VP is RUNNING.

4.1.3.2 Process Multiplexing

The kernel is also responsible for the multiplexing processes on the available VPs. The processes are scheduled in much the same way as VPs, except that they can be created and destroyed during the lifetime of the system. Furthermore, while a VP is permanently bound to a single physical processor, processes are not bound to a single VP.

Processes can be in one of four states (see the left side of figure 4.4, page 53): READY, RUNNING, BLOCKED, or PENDING DELETION. When a process is RUNNING, it executes until preempted by a higher priority VP (going to READY), until it needs to wait on an event (going to BLOCKED), or until it deletes itself (going to PENDING DELETION). When BLOCKED, it waits until the pending event happens, and then moves to the

[7]Note that this is not a MAC policy concern, as the ring integrity policy is separate from the MAC policy.
[8]Note that the terms CPU and physical processor are used interchangeably in this report.

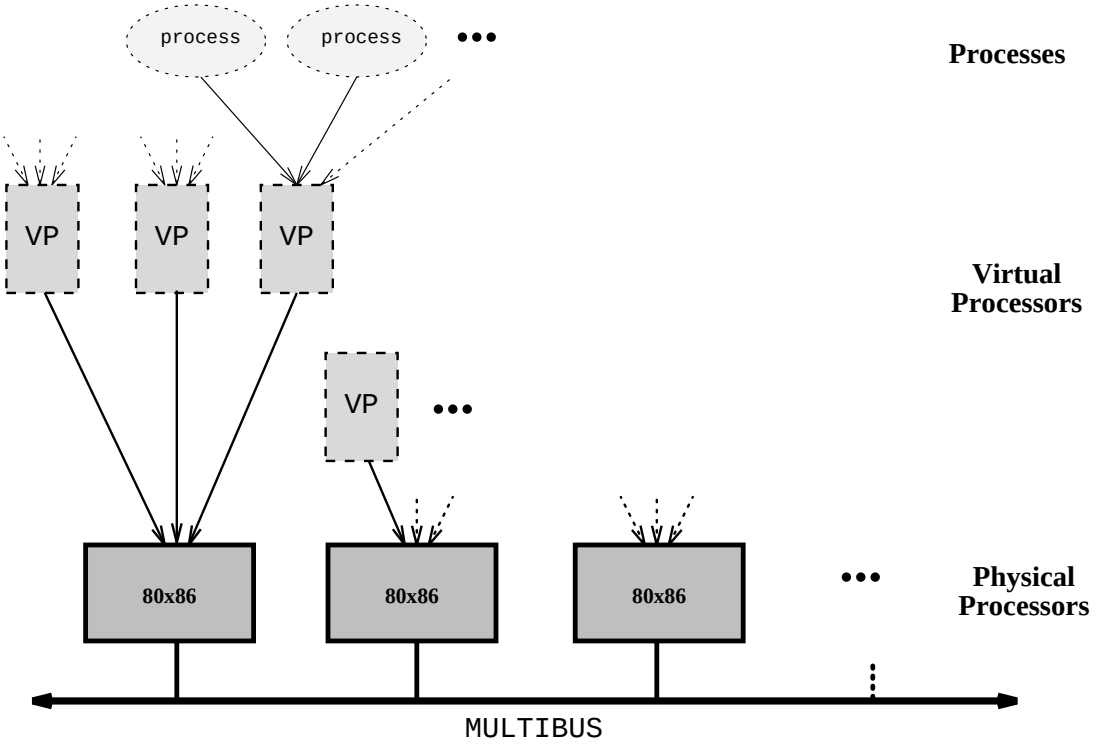


Figure 4.3. Virtual Processor (VP) Architecture

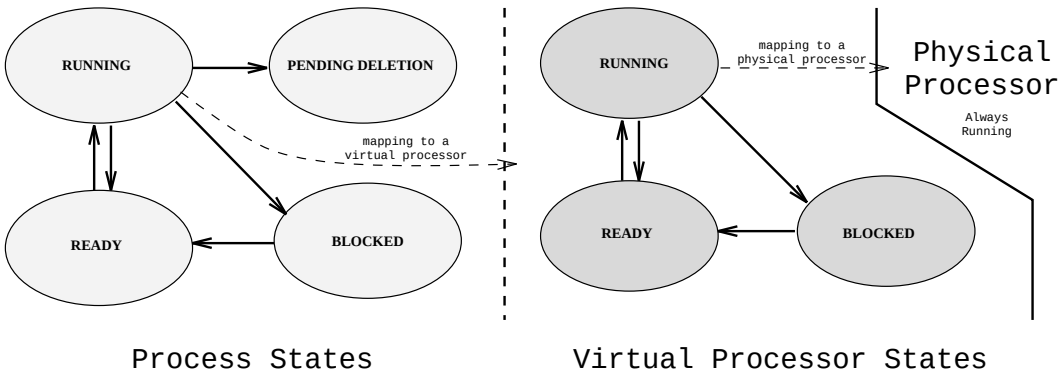


Figure 4.4. Process and Processor States

READY state. When READY, it moves to the RUNNING state whenever no higher VP is RUNNING. Finally, when a process is PENDING DELETION, it waits until its parent process issues the appropriate instruction to delete it and then it ceases to exist.

4.1.3.3 Support for Multiprogramming

This two-level scheduling structure allows the GTNP kernel to provide support for multiprogramming. Just as processes may block and be rescheduled waiting for a resource, so can the portion of the kernel that provides the virtual process abstraction. When this happens, the virtual processor providing the processor abstraction to that kernel portion is blocked and rescheduled (freeing up the physical processor). An available virtual processor can then continue to operate while the first virtual processor waits for its resource.

4.1.4 Segments

A segment in the GTNP is a logical unit of storage (either in memory or on disk) that is either 0 bytes[9] or has a size ranging from 2 bytes[10] to 64 Kbytes. As described in section 3.2.3.2, page 20, these segments are referenced through the LDTs.

4.1.4.1 Segment Naming

Internally, every segment on the system has a unique identifier that is different for every object created in every GTNP.[11] This unique identifier is not visible to the applications. Instead, a process-local aliasing scheme is used outside the kernel to refer to segments; the unique name of a segment is visible only within selected layers of the kernel. As a result of this aliasing scheme, the only way that applications executing on the GTNP can reference segments is in a process-local fashion using a selector (convertible to a pointer in a programming language) constructed from a Process Local Segment Number (PLSN). There is one PLSN for each segment that is known to a process; each PLSN corresponds to an entry in the process' LDT. The PLSN is also used to reference the synchronization objects that are associated with a segment; for more information on synchronization objects, see section 4.1.6, page 58.

When a child process is created, the parent process passes the child process an initial set of segments that are to be introduced into the child's address space at initiation. To access more than these initial segments, the application process must name other segments in the system and introduce these segments into its address space. Once introduced, a segment can then be accessed by constructing a selector from the PLSN. The segments that are introduced in this fashion are named using the GTNP aliasing mechanism.

An alias in the GTNP consists of two parts: a process-local component and a system-wide component. These are paired to form the alias that is used as the handle to name segments. The process-local component is the PLSN of a segment that is already known to the process. For naming purposes, this known segment serves as a parent (the Gemini term for this role is *mentor*) to a fixed number of other segments. The disk addresses for the other segments are stored in an *alias table* that is associated with (not stored with) the mentor; the

[9] No secondary storage (as opposed to control information) is allocated for storing data for logical segments of zero size. The LDT entries for zero-size segments have zeroed (and thus, invalid) descriptors.
[10] The GTNP software interface does not allow segments to have one and only 1 byte.
[11] The unique name combines a volume-local identifier with a unique volume ID.

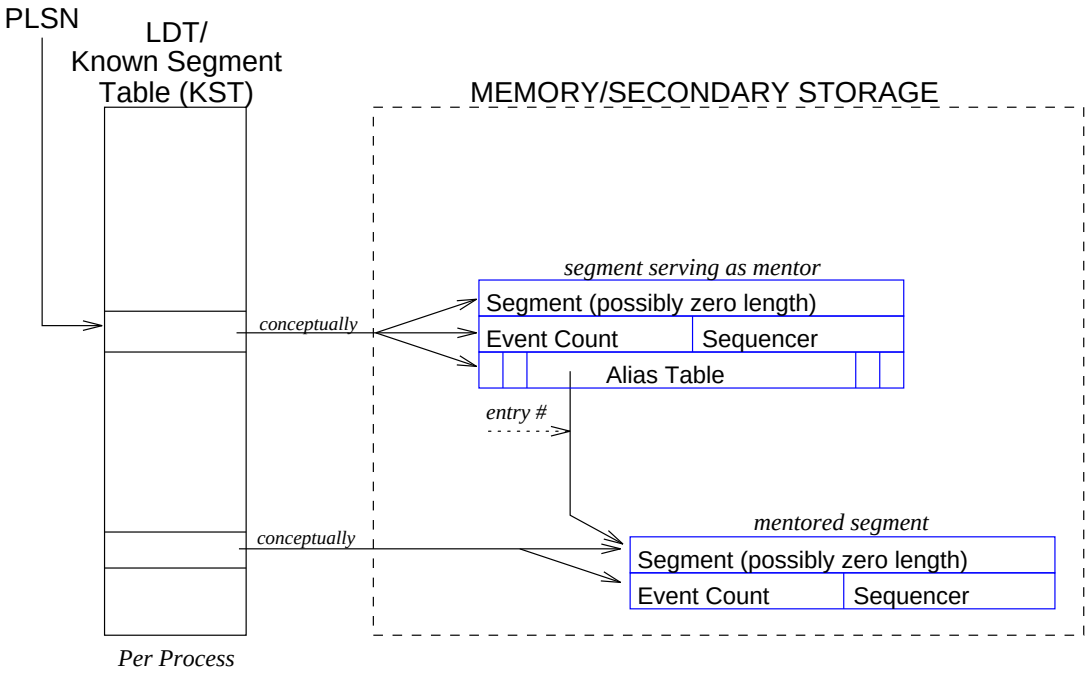


Figure 4.5. Conceptual View of Segments from the Application Process Point of View

application selects which of these other segments is to be obtained by providing an index into this alias table called an *entry number*. Through the combination of the mentor's PLSN with the entry number, the GTNP TCB is able to obtain the disk address of the segment (which the application subject never sees).

Figure 4.5, page 55 presents a *conceptual* view of the PLSN serving as a process-local name for these three different aspects of a segment: the alias table (when the segment is serving as a mentor), the segment contents, and the synchronization objects. This is the view seen by application processes outside the kernel.[12] Every segment that exists on the system has synchronization objects; if the segment has a length greater than zero, it also has a hardware segment for its data. Additionally, if the segment serves as a mentor to other segments, it has an alias table for naming purposes.

Segments introduced in this fashion can, in turn, serve as mentors to other segments. The GTNP restricts segments to being referenced by only one alias table entry; this results in the presentation to the user of a tree-like naming hierarchy of segments not unlike names in a UNIX directory structure (assuming links are not present). The system enforces certain policies relative to this hierarchical naming structure; for example, a segment that serves as a mentor cannot be deleted until all of the segments it mentors are deleted. The GTNP also enforces a compatibility property on the assignment of access labels to segments, which requires that the access label of the mentor be dominated by the access label of each segment it mentors.[13]

[12] Note that this is not the internal representation. Note also that the alias table is not directly visible through the interface; its existence is implied in the ⟨mentor,entry⟩ notation.

[13] See section 10.4, page 185, for a comment regarding this property.

4.1.4.2 How Segments are Used

Aliases are used to name a segment only when the segment is not already known to the application process; in other words, when the segment is being created, deleted, or introduced. All other references to the segment use the PLSN.

During process creation, a parent process specifies the segments to be passed to its child process.

In order to access segments other than those passed during creation, new PLSNs must be defined. This is done through the "makeknown" operation. When an application process "makes known" a segment (requesting that the segment be introduced into its address space), it provides not only the alias of the segment to be introduced, but the PLSN to be assigned to the new descriptor if the introduction is successful. This operation is analogous to the shared memory attach call (*shmat*) in systems supporting UNIX System V Interprocess Communication (IPC) objects; the application process also provides the requested access modes for the segment and for the synchronization objects (these are drawn from different sets of possible permissions)[14] and the destination ring. As in the shared memory attach call, it is at this point that access to the segment is validated, based on the access labels of the segment and the subject, the requested access modes, the ring bracket of the segment, and the ring number into which the segments are being introduced.[15] If all the access checks pass, the LDT entry corresponding to the PLSN specified by the application process is defined to reference the indicated segment.

Once made known, the PLSN can be used as a mentor in other segment creation, deletion, and makeknown operations, and in accessing the synchronization objects associated with the segment. However, the data in the segment is not yet available.[16] To load the contents of the segment into memory, an operation called "swapin" is used. This operation may be used on the PLSN of any accessible (e.g., a segment that was not made known in "no-access" mode) non-zero length segment not currently swapped in by the application process. Once a segment has been swapped in, the application process can construct a selector (pointer) to access the segment, and manipulate the data in the segment. A protection violation trap is generated for any attempted type of access that is not allowed by the access mode specified when the segment was made known.

All manipulations done on a swapped-in segment affect only the copy of the segment in memory. To ensure that the version of the segment on secondary storage is consistent with the version of the segment in memory, the application process must "flush" the segment, which writes the contents back to secondary storage. This action does not prevent other processes from accessing the segment being flushed during the flush operation.

When use of the segment is completed, the process removes the segment from its memory space with a "swapout" operation. When this happens, the descriptor for the segment (referenced by the PLSN) is marked to indicate the segment is "not present," the segment is written back to secondary storage and the memory is released for reuse.

Swapping out a segment does not remove the segment from the address space of the application process. In order to release the descriptor table entry (and allow the PLSN to be reused), an application process must "terminate" the segment from its address space. Termination of a segment removes the specified segment

[14] Segments can be made known in the following modes: read-only, read-write, execute-only, read-execute, no-access, or as conforming read-execute or execute-only. Synchronization objects can be read-only, write-only, read-write, or no-access.

[15] The specific access checks performed during a makeknown operation are described in more detail on page 86.

[16] Specifically, the entry for the segment in the LDT has its limit and access rights initialized based on the makeknown parameters. The physical address is set to zero and the segment is marked as not present.

from the address space of the application process, releases the PLSN, and writes the synchronization objects back to secondary storage.

It is important to note the differences between a segment storage system and the more conventional "filesystem" concept. In a filesystem, each user has a copy of the file information in a local buffer; updates to this buffer are typically not seen until the buffer is written to disk and reread. In contrast, a segment storage system (as in the GTNP) has a closer analogue in virtual memory. Segments made known by multiple processes share the same portion of memory; updates to the segment in memory are immediate. Any process may swapout or flush the segment; write permission controls the ability to change the contents of the segment, not the ability to update the image of the segment on secondary storage.

4.1.5 Volumes

Segments on secondary storage are organized into distinct logical groups called volumes. The volume on which a particular segment resides must be mounted to make known that segment. In general, volumes are used only for organizing and storing segments. Bootable volumes may boot the operational system, or they may boot one of the system generation utilities that execute on top of the kernel. Volumes are identified using a volume number and a volume unique ID used to identify the volume.

From the point of view of a GTNP application, the GTNP presents a single segment naming hierarchy for all segments in the system. In this single naming hierarchy, segments are aliased using a ⟨mentor,entry⟩ path that begins from a common fixed point—the System Mentor of the boot volume. During GTNP initialization, the kernel provides the NPSP with a PLSN for the boot volume's System Mentor. The NPSP restricts the application process to some subtree of segments within the volume. This restriction is implicit and is determined by the set of segments that are initially available in the address space of the application process.

For those subjects to whom the boot volume's System Mentor is available, any segment accessible under the security policy can be reached by iteratively making known a series of segments along a path. For example, consider the path: "SM-5-6-9-10" (this is an arbitrary notation). This segment could be reached by making known entry 5 off the System Mentor. Using the PLSN obtained from that makeknown as a mentor, entry 6 is then introduced. Using that new PLSN, entry 9 is introduced. Finally, using the PLSN just obtained, entry 10 is introduced. Suppose a parent application (or NPSP) wanted to restrict a child process to the subtree mentored by "SM-5-6." To do this, it would provide the child with the PLSN for "SM-5-6" during process creation (assume that it was placed into LDT slot 15). The child process would have no knowledge of this path above the subtree, and would only see the segment as "PLSN15-9-10."

All kernel code and data segments are R0 segments; hence, they are accessible only by the kernel and R0 subjects, such as the SYSGEN utility. Kernel and initial process parameters may be accessible from outside the TCB; however, in the GTNP evaluated configuration, they are restricted to TCB access.

Through the eyes of the aliasing mechanism, the GTNP provides a single segment naming hierarchy that may consume an infinite amount of storage; however, in the real world, there are physical media boundaries and limited sizes. As noted above, the GTNP supports multiple uniquely identified logical volumes, identified by volume numbers.[17] Support is provided in the GTNP to connect these logical volumes to form the volume space seen by applications. In all cases, a given segment exists on only one physical volume.

[17] Several volume numbers may refer to the same physical device (such is the case for partitions of a disk).

A volume is connected to the segment naming hierarchy through the use of the "mount" operation that associates the naming hierarchy on the new volume with a specific mentor segment (called a *mount segment*) on the existing volume. After the operation, the mount segment serves as a mentor to the collection of segments occupying the mounted volume. There are two types of volumes: *recoverable* and *non-recoverable*.

For *recoverable* volumes, the initial mount of the volume creates a permanent association between the volume and the segment upon which the volume was mounted. All subsequent mounts of that volume are restricted to that mount point. Furthermore, it restricts subsequent mounts to a specific system.

This fixed association is bidirectional; i.e., the mount segment is associated with the mounted volume. This limits the mount capability of the mount segment to a single volume. A system generation utility is provided to restore the information from a volume containing TCB segments if the mount-segment is deleted; user volumes can be recovered through the TCB interface.

Non-recoverable volumes may be mounted on any mount segment; this restricts subsequent mounts on that mount segment to that particular non-recoverable volume.

The ability to transfer volumes between systems is controlled by information maintained in the System Programmable Read-Only Memory (PROM). The PROM contains a table that defines, for each volume, the Media Resource and Internal I/O read and write classes (see section 4.1.8.2, page 60), along with the attributes of that volume device: if volumes mounted must be sealed and/or encrypted, and if non-local volumes are allowed. In the GTNP evaluated configuration, all volume devices accessible to untrusted subjects are restricted to local volumes only (this is enforced by checks on the particular keys used to format the volumes).

The evaluated configuration also requires removable volumes (other than those mounted during TCB initialization) to possess a degenerate range (read label equal to write label) in order for them to be mountable by untrusted subjects (which also have a degenerate range). As segments that have labels that are outside of the range of a volume cannot be stored on the volume, this results in all media being not only single-level (as are all devices in the GTNP, since trusted labels for export purposes are not provided), but single-label.

Physically, volumes can represent any variety of storage. In the GTNP system, there are three types of media that can be formatted as volumes: logical partitions of a hard disk; floppy disks; and RAM disks.[18]

Regardless of the support provided by the underlying secondary storage hardware, any type of volume can be made a *read-only* volume. This restriction, enforced by the kernel, is done through an indication in a special part of the volume called the Volume Table of Contents (VTOC). In addition to the read-only indication, the VTOC contains the back-link and a format type indication for the volume. Since the VTOC must be written when a volume is mounted the first time, the ability to mount physically write-protected volumes is limited to either non-recoverable volumes, or volumes that have been previously mounted.

4.1.6 Synchronization and Signals

Process synchronization operations are used to synchronize and communicate information between processes in the system. Every segment has one eventcount and one sequencer[19] associated with it. Each is initialized

[18] This is a special type of memory that is treated as a disk. RAM disks include volatile RAM as well as non-volatile RAM, PROM, or ROM. Other than the physical storage media (and the impermanence of volatile RAM), they behave as any other volume media.

[19] Eventcounts and sequencers were introduced in a paper by Reed and Kanodia [147].

to zero when the segment is created, and has the access label of the segment associated with them. Intuitively, an eventcount is a monotonically increasing integer variable used to count the occurrences of some event of interest to more than one process. The occurrence of an event is reported by a process by using the "advance" operation that increments the eventcount by one.

Sequencers are used to efficiently serialize actions in the application domain (e.g., entry into a critical section). Eventcounts alone are insufficient for this task, as many processes are allowed to wait for any event and there is no way to tell if processes are already waiting. A "ticket" operation returns the current value of the sequencer and increments it by one. The sequencer is usually used in conjunction with an eventcount: each process, after receiving its ticket, waits on a unique value (obtained from the sequencer) of the eventcount so that only one process can be unblocked at a time. The "await" operation is used to wait for a particular event (represented by a value of an eventcount). The "read eventcount" operation is used to determine if events have occurred.

Processes have three signal channels associated with them. Signals support asynchronous awaits for events using eventcounts. The "await signal" operation associates an eventcount with a signal channel. When the eventcount reaches the value specified in the "await signal" operation, an interrupt is generated. The signal channel's interrupts can be masked by using the "signal mask" operation.

4.1.7 Object Label Tranquility

A key underlying concept in the GTNP is object label tranquility. The GTNP provides no online mechanisms that allow the access labels assigned to existing subjects or objects to be changed during their lifetime (thus, no mechanism to handle revocation is required). Segments, volumes, and the objects named by the segment alias obtain their label at the time of creation; subjects obtain their label at the time of process creation. These labels cannot be changed once established.[20] Devices obtain their label from a configuration table in the System PROM; this table can only be changed using system generation utilities.

4.1.8 Devices

The GTNP supports a wide-variety of I/O devices. These devices include the mountable media as well as Serial I/O devices, High-level Data Link Control (HDLC), and Ethernet—to name a few. Device Management in the GTNP is provided by the kernel, which treats devices as TCB *subjects*.[21] Communication between application subjects and device subjects is done through the use of shared segments; coordination of requests is done through the use of eventcounts.

4.1.8.1 How Devices Are Used

In order to communicate with a device, an application subject first "attaches" the device. This operation provides the application subject with a process-local identifier used to access the device. A synchronization object may also be specified to allow coordination between the programming subject and the device subject.

[20] Note that if a multilevel subject were to be defined (although doing so invalidates the rating), the multilevel subject may change the access label range of outer ring subjects via the "activate subject" operation.

[21] The treatment of devices as subjects is not that common, although it is permitted by the TNI. The rationale behind this treatment is discussed in section 5.1.2, page 106.

Depending on the nature of the device, the number of processes that may attach a device at the same time may be limited to some finite number. However, certain devices can be attached at the same time by all processes in the system. Devices are also distinguished by the intended mode of access (i.e., sequential or random).

Once a device is attached, a process uses I/O calls to communicate with the device. These calls conceptually introduce segments into the address space of the device subject. I/O operations on devices can either be sequential or random-access, depending on the nature of the device.

Devices are not attached with a particular access mode (as is typical in other operating systems); instead, the access mode is determined based on the nature of the particular device attached. For each physical device that supports reading, there is a "read" device. For each physical device that supports writing, there is a "write" device. The system also supports bidirectional devices. Typically, there are also "control" devices that are used to send control information to, or read status information from, the physical device. These control devices can be either unidirectional (i.e., "read control" or "write control") or bidirectional.

Information on the devices supported and their capabilities can be found in section 4.2.1.8.2, page 75. All devices supported in the evaluated configuration are single-level devices. Although the architecture of the kernel conceptually allows multilevel devices, no multilevel device drivers are provided in the evaluated configuration. See section 10.3, page 184 for an evaluators comment regarding this.

When a subject completes the use of the device, it "detaches" from the device. This call, similar to the terminate call for segments, invalidates the process-local identifier used to name the device.

4.1.8.2 Physical vs. Logical Devices

Each physical device in the GTNP is partitioned into some number of logical hardware units (LHUs). Most devices have a single LHU; hard-drives may have many. Each LHU on a device is configured to support some number of I/O devices; these I/O devices are used by application processes to access the media via the device interface at the kernel. Furthermore, the LHU may be configured to support a logical "Volume Drive" that is used by the kernel to manage the segment and volume abstractions. This results in three possible states:

1. LHUs that provide device I/O only. For disks, this implies the availability of only the "disk device" accessible through the device interface.

2. LHUs that provide volume access only. This implies the presence of a volume device (available through the device interface) and a logical volume drive (indirectly available through the segment and volume management calls).

3. LHUs that provide both disk and volume access, as well as a logical volume drive.

To eliminate potential conflicts between devices, the kernel performs some special checks. On attach, the kernel verifies that the volume drive is not busy (i.e., mounted) and that the other I/O device (if the LHU has more than one I/O device) is not already attached. On a mount, the kernel verifies that the associated I/O devices are not attached.

4.1.8.3 Mountable Media

Special consideration is given to the mountable media devices (volumes), which do not always go though the "attach" and "detach" operations. As described in section 4.1.5, page 57, mounted volumes contain segments; each segment has a label associated with it. These segments are accessed by first "mounting" a volume. From that point on, the accessing of segments on the volume is considered by the GTNP to be *Internal I/O*. Volume drives used by the kernel for Internal I/O have an Internal I/O Media access label range (defined in the System PROM) that reflects the physical environment in which the physical media exist. This range is in addition to the device access label range. More information on the use of this range may be found in section 6.3, page 114, and in the description of the "mount" operation in section 4.2.2.2, page 88.

4.1.9 Use of Encryption/Cryptosealing

As described in section 3.3.6, page 36 and section 3.4.4, page 44, the GTNP system controller provides mechanisms that allow the encryption of data and the generation of cyptographic checksums using the DES. The GTNP depends on the ability of the DES to generate cryptographic authenticators (using the cipher block chain encryption algorithm to generate 64-bit checksums) in the following four areas:

- *Label Integrity*. Cryptographic checksums are depended upon to associate labels with information on media used to implement the volume abstraction, where that media is either removable or accessible via device interfaces.
- *Trusted Distribution*. Cryptographic checksums are used as part of the software and hardware distribution process, as described in section 7.8, page 142.
- *Trusted Recovery*. Cryptographic checksums are depended upon to insure the integrity of kernel code and data stored in PROMs and on volume media following a system failure or other discontinuity.
- *Generation of Unique IDs for objects*. The DES cipher block chain encryption algorithm is depended upon to generate unique IDs for each created segment.

4.2 GTNP Kernel

The first portion of the GTNP to be discussed is the GEMSOS kernel. The discussion begins with a description of the architecture of the kernel. This is followed by a description of the kernel interface. The discussion concludes with a description of kernel initialization.

4.2.1 Kernel Architecture

The GEMSOS kernel is the heart of the system, and provides the M-component's reference monitor. The kernel executes at PL0, which isolates it from the rest of the operational software on the system (which executes at privilege levels above PL0) and gives it the ability to manipulate global kernel data structures available to all processes on a processor and all processors in a multi-processor system.

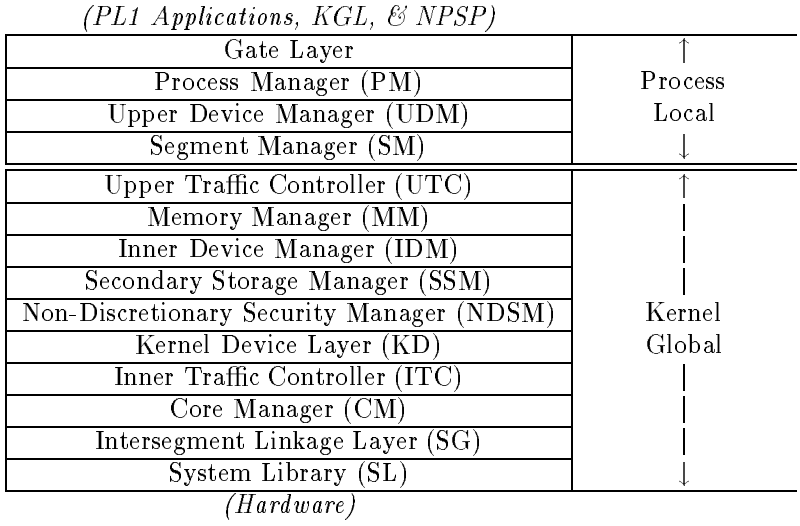


Figure 4.6. GEMSOS Kernel Layers

The GTNP kernel is made up of 14 layers; it exports 29 gates to PL1. The gates are described in more detail in section 4.2.2, page 85. The layers themselves are organized in a strict hierarchical fashion with no loops. There is extensive use of information hiding; the kernel does not have any "global" (in the sense of being *directly* accessible in multiple kernel layers) data structures. The structure of the design restricts access to data structures to single kernel modules. On disk, all kernel data structures are accessible only by the kernel (executing at PL0) and any active R0 subjects.[22] The layers of the GEMSOS kernel are shown in figure 4.6, page 62.

The top four layers of the kernel are process-local. Information contained in these layers is specific to a given process; the information is stored in the process' address space. These layers are the following:

- **Gate Layer**. This layer is a trap handler that manages the interface to the kernel.
- **Process Manager**. This layer provides the management of the application process structures.
- **Upper Device Manager**. This layer provides the interface to I/O devices for non-kernel subjects. It is responsible for the checking of the mandatory security attributes of the devices.
- **Segment Manager**. This layer manages all non-kernel access to segments. It is responsible for making the mandatory security checks on segment access. This layer provides a per-process Known Segment Table (KST) and maps process-local segment names into unique, system-wide names.

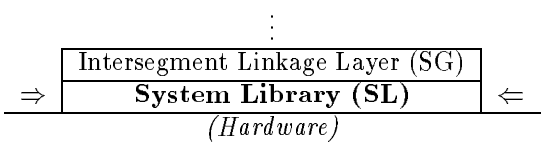
[22] Data maintained on disk does not have a privilege level associated with it. Static kernel data structures on disk are stored with ring brackets restricting them to R0. When the kernel is initialized, these structures are brought into the kernel address space at PL0. Thus, a R0 subject could manipulate only the static structures. However, this is precluded in the online system, since there is no way for active R0 subjects to be created. NPSP, the TCB subject responsible for starting all non-TCB application processes, executes in R1. The kernel restricts any process to creation of child processes in the parent process' ring and above. Thus, there can be no non-TCB applications with an active R0 subject. Use of R0 is reserved for the SYSGEN utility.

The remaining ten layers are kernel-global; their databases contain information concerning either all processes on a given physical processor, or all processes in the system as a whole. These tables are maintained in either local memory (for CPU-specific information) or global memory (for system-wide information). These layers are the following:

- **Upper Traffic Controller**. This layer schedules non-kernel processes onto VPs (an abstraction provided by the Inner Traffic Controller, below). It is also responsible for the process scheduling that results from interprocess synchronization with eventcounts and sequencers.
- **Memory Manager**. This layer manages the multiplexing of the physical memory into the memory available to the processes. It manages the segment descriptor tables, as well as the mapping of segments on processor local and system global memory.
- **Inner Device Manager**. This layer manages the configuration of devices available in the system, including the control of the access labels assigned to devices. This layer also provides the device drivers for all devices accessible outside the kernel.
- **Secondary Storage Manager**. This layer provides the interface to (and manages the use of) internal I/O to the disk devices used for permanent storage of segments. It is also responsible for making the mandatory security checks for volumes.
- **Non-Discretionary Security Manager**. This layer is responsible for the interpretation of access labels in terms of the specific security policy implemented by the GTNP.
- **Kernel Device Layer**. This layer provides the hardware-specific device drivers for all devices directly used by both the kernel TCB and the non-kernel TCB.
- **Inner Traffic Controller**. This layer provides the abstraction of VPs to higher layers. Each physical processor has a fixed number of these VPs that are multiplexed onto it by this layer.
- **Core Manager**. This layer provides the interface to the processor hardware, and "hides" the information about the unique characteristics of a given processor. It also provides a controller interface to the interrupt processing mechanism.
- **Intersegment Linkage Layer**. This layer provides explicit support to create and maintain the linkage across code segments within the kernel.
- **System Library**. This layer provides functions for the manipulation of data passed as parameters. It maintains no databases and performs no global state changes.

The following subsections discuss each of these layers in more detail, presenting the functions of the layer and the abstractions the layer provides to layers above it in the kernel. The discussion begins with the two utility layers—the System Library and Intersegment Linkage Layer. This is followed by the first non-utility layer closest to the hardware, the Core Manager, and continuing up the layers until the Gate Layer is reached.

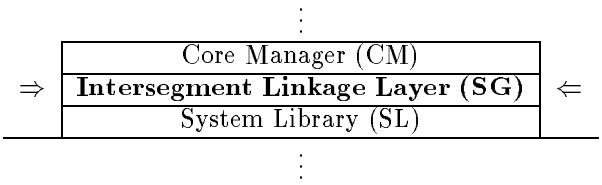
4.2.1.1 System Library Layer (SL)



The lowest layer in the kernel layer hierarchy is the System Library layer. This layer provides library functions to the other layers of the kernel, with all data manipulated being passed as parameters (e.g., there is no global data manipulated). The basic functions provided by the routines in the System Library include the following:

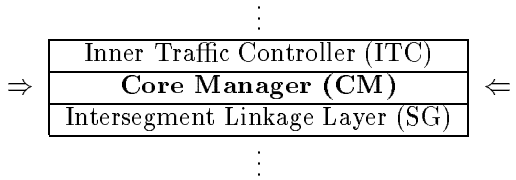
- Conversions of values between different size data types.
- Arithmetic on Gemini-defined types (such as 3-byte addresses).
- Functions to construct pointers and selectors.
- Access to hardware functions for system reset unavailable through Pascal (PC-AT platform only).
- Access to hardware functions for input and output from the I/O Ports on the hardware unavailable through Pascal.
- Access to hardware functions for basic hardware control and status functions unavailable through Pascal.

4.2.1.2 Intersegment Linkage Layer (SG)



The Intersegment Linkage (SG) layer provides a mechanism to create and maintain the linkage across code segments within a module and between modules (a facility not supported by the implementation programming environment). Each code segment is executed in its own environment. When a transition between code segments must be accomplished, this layer saves the processor state of the current code segment and transfers control to the new code segment.

4.2.1.3 Core Manager Layer (CM)



The Core Manager (CM) layer is responsible for the management of the hardware and physical resources that are used to build segments and processes; and executes appropriate code depending on the hardware base (this is also true of the KD and IDM layers).[23] It consists of the following modules:

[23] For the Core Manager, there are multiple configurations in the evaluated product: one for each Trusted Base type and one for the PC-AT platforms.

- The *Core Manager Module*.
- The *Interrupt Manager Module*.
- The *Metering Support Module*.
- The *System ID PROM Module*.
- The *Encryption Support Library Module* (ESLM).
- The *Console and Error Handling I/O Module*.

The CM layer hides the differences in the underlying processors from the layers above it. For configuration management, there is a single Core Manager; the system loader (part of kernel initialization) selects the appropriate modules for the underlying system at boot time.

4.2.1.3.1 Core Manager Module

The Core Manager Module (CMM) is responsible for scheduling physical processor tasks[24] onto physical processors under the direction of the ITC layer (see section 4.2.1.4, page 68), and performing the necessary data structure management associated with this function.

The functions of the Core Manager Module fall into the following areas:

- Support for the allocation of processor tasks, including management of the LDTs and TSSs for the tasks.
- Management of the Global Descriptor Table (GDT).
- Management of the page tables used for memory references. Note that page tables are available only on the SP and UP processors, and that they are strictly static and set up during initialization. The paging is enabled for the purpose of physical address calculation, but there is no dynamic management of page tables (see section 3.2.3.1, page 19).
- Creation and initialization of the kernel stack frame.
- Provision of information about the current task and physical processor.
- Completion of system shutdown.
- Trap support with respect to the PL1 trap frame and the double fault interrupt.
- Assignment of specific kernel tasks and processes to specific physical processors.

[24] Gemini makes a distinction between *processes* and *tasks*. A task is the basic, schedulable, active entity within the kernel; it is scheduled onto a physical processor. A process is a task that is visible outside of the kernel and is scheduled on a virtual processor.

4.2.1.3.2 Interrupt Manager Module

A second area of functionality provided by the CM layer is that of interrupt management. This functional area is the responsibility of the Interrupt Manager Module (IMM). This module provides support in four principal areas:

1. Interrupt Handling
2. Non-interruptible movement of data
3. Numeric Coprocessor Extension (NPX) and Signal Support
4. PC-AT Binary Clock (ABC) Simulation (PC-AT platform only)

4.2.1.3.2.1 Interrupt Handling

Interrupts are received by the Interrupt Manager from both hardware and software. Hardware interrupts can result from either Multibus or Personal Computer/AT (PC/AT)-compatible I/O Channel interrupts, or from devices connected directly to the physical processor. All maskable interrupts arrive via the Programmable Interrupt Controllers (PIC).[25] Software interrupts (commonly referred to as *traps*) arise from either physical processor-detected interrupts or programmed exceptions.

When an interrupt occurs, the physical processor vectors to an address defined for the given interrupt in the Interrupt Descriptor Table (IDT). The IDT entry that is chosen is determined from the interrupt number. Interrupts 0-31 are software interrupts generated by the physical processor (although applications do have the ability to directly generate breakpoints and overflows). Interrupts 32-47 are hardware interrupts generated through the PIC. Interrupts 48-49 are software traps initiated for applications by higher kernel layers.[26] Each IDT entry contains, in addition to the address, information used to restrict the ability to trigger the interrupt by privilege level, and information about the context in which the interrupt is handled. Some of the more significant interrupts are: "Breakpoint," "Overflow," "Math Coprocessor Not Available," "Segment Not Available," "Stack Exception," "General Protection Violation," "Signal Mask Interrupt," and "Gate Interrupt."

4.2.1.3.2.2 Safe Moves

The IMM provides other kernel layers with two entry points that allow for movement of data between kernel and non-kernel segments (two calls), and one entry point that allows for movement of data between non-kernel segments. This latter call is used to support the `k_replace` gate described in section 4.2.2.4, page 91. For these calls, rescheduling cannot occur during the data transfer.

4.2.1.3.2.3 Numeric Processor Extension and Signal Support

The IMM is responsible for the coordination of use of the Numeric Processor Extension (NPX) among tasks, as well as the signalling of tasks when synchronization conditions are met.[27]

[25] Multibus interrupts are delivered to all physical processors. Note that PC-AT platforms do not have a Multibus and receive only local bus interrupts.

[26] If somehow the system vectors to an IDT entry above 49, the system will halt. This occurs because all IDT entries above 49 are initialized with a vector that vectors to the kernel's illegal interrupt handler.

[27] Signals have already been described in section 4.1.6, page 58.

4.2.1.3.2.4 PC/AT Binary Clock Simulation

The IMM also provides support for the implementation of the PC/AT binary clock device. Entry points are provided to higher kernel layers to support initialization of the counter, setting the alarm, and reading the current counter value.

4.2.1.3.3 Kernel Metering Support Module

The Metering Support Module in the CM layer provides the implementation of the kernel meters that support the Kernel Metering Support device in the KD and IDM layers.

4.2.1.3.4 System ID PROM Module

The System ID PROM Module in the CM layer provides the equivalent of a low-level device driver for accessing the System ID PROM.

4.2.1.3.5 Encryption Support Library Module

As noted earlier, the GTNP is delivered with built-in support for encryption using the DES algorithm. The Encryption Support Library Module provides library functions for performing the following key-related operations:

- Data encryption with encryption key
- Data decryption with decryption key
- Key encryption
- Cryptosealing
- Loading of keys into a keys database

In addition to supporting application processes, encryption is used in the GTNP as an integral component of the Trusted Recovery and Distribution schemes (see section 7.4, page 125 for a discussion on Trusted Recovery; and section 7.8, page 142 for a discussion on Trusted Distribution), and of label integrity.

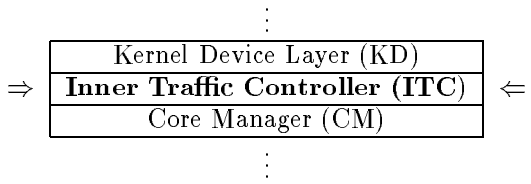
Keys are composed of eight bytes, with the low-order bit of each byte being the parity bit. The hardware encryption unit checks this parity bit; an exception results if it is not correct (odd parity is used).

4.2.1.3.6 Console and Error Handling Module

This module provides a standard interface for doing console[28] and error-handling I/O. It is present in the CM layer so that various messages from higher kernel layers can be output to the console.

[28] Every GTNP has one serial I/O line that can be used as a console for any messages that must be issued during the system initialization and boot process. This console is always a fixed local serial I/O port on a physical processor, defined in the System PROM.

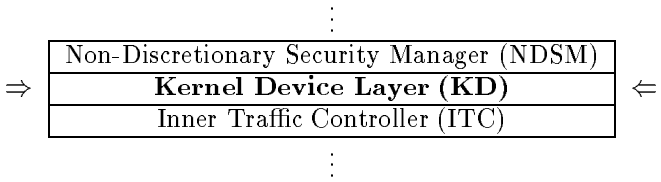
4.2.1.4 Inner Traffic Controller (ITC) Layer



Processor multiplexing has two layers. The bottom layer, the ITC Layer, provides what is sometimes referred to as a *separation kernel*[26, 148]. Each physical processor has a fixed number of VPs, that are multiplexed onto it by the ITC. VPs are available (i.e., abstracted up) to the upper layer, the Upper Traffic Controller (UTC) Layer (described in section 4.2.1.10, page 81). The ITC also provides primitives for synchronization between VPs. This means that the ITC supports multiprogramming by scheduling VPs to run on the physical processor with which they are (permanently) associated.[29]

The synchronization primitives provided by the ITC are for the higher layers of the kernel. These primitives include event counts and sequencers and locks. ITC event counts and sequencers are used by the kernel only. Non-kernel subjects use eventcounts[30] and sequencers managed by the UTC and MM.

4.2.1.5 Kernel Device (KD) Layer



Directly above the ITC layer is the KD layer. The KD layer provides higher kernel layers (e.g., the IDM and SSM layers) with access to the internal devices used by the kernel to implement visible abstractions (such as the segment system or selected attachable devices). These devices fall into four major groupings: encryption device support, System PROM support, Random Value Generator support, and disk (and tape) support.

4.2.1.5.1 Encryption Support

Support for encryption is provided by the Encryption Driver module within the KD layer. This module manages I/O operations to the encryption device (no direct I/O services are provided). It is the responsibility of this module to ensure exclusive access to the device.

[29] Note that this structure implies that the security kernel can be (and is) interruptible. In addition, this layer provides all the multiprocessing interactions between individual physical processors, using interrupts. An additional benefit of the strict layering in the design is that the existence of multiple processors is visible only at this lowest level. Thus, the multiprocessing adds no difficulty to the design of the rest of the kernel.

[30] Note the distinction between the use of the terms "event count" (with a space) and "eventcount" (without a space). "Event count" is used to refer to the static set of synchronization abstractions managed by the ITC layer and used by other kernel layers; "eventcount" is used to refer to the UTC synchronization abstraction that is available above the UTC layer and outside the kernel.

4.2.1.5.2 System PROM Support

Support for the System PROM is provided by the System PROM Driver module within the KD layer. This module is very similar to the Encryption Driver. It is responsible for ensuring exclusive access to the System PROM and performs no direct I/O.

4.2.1.5.3 Random Value Generator (RVG) Support

Support for the RVG is provided by the Random Value Generator Device Driver module within the KD layer. This driver provides functions for generating 64-bit random values and 64-bit random keys. Random keys differ from random values in that each byte is set to odd-parity so that it may be used as a DES key.

4.2.1.5.4 Disk and Tape Device Support

The disk device support available through the KD layer is provided in two forms: volume-oriented and unit-oriented. Volume-oriented access is done through the Disk Volume Manager (DVM) module; whereas "raw" or unit-oriented access is done through the Disk Unit Manager (DUM) module. Both of these managers utilize the Disk Unit Driver (DUD) module to actually perform the I/O operations; the DUD module, in turn, calls the specific device drivers. Since the Storager device also controls the tape device, the KD layer also handles tape interactions.

4.2.1.5.4.1 Volume-oriented Interface

The DVM module provides a volume-based interface to all physical devices handled by the DUD. This interface is used by higher layers such as SSM to perform internal I/O to volumes. Each physical device is identified by higher kernel layers through the use of a logical volume number (LVN). This LVN serves as an index within a DVM-local Volume Table to determine the actual unit number. LVNs may identify hard disk partitions, floppy disks (or partitions thereof), RAM disks, or streaming tapes (Trusted Base only).

The DVM provides higher kernel layers with the ability to ensure exclusive access to a volume through the use of attach and detach calls.

In order to process I/O requests, the DVM maps from the volume abstraction (volume number, volume-relative physical sector number) into the unit abstraction (unit number, physical sector number) used by the DUM and DUD. It then interacts with the DUM and the DUD to perform the I/O.

4.2.1.5.4.2 Unit-oriented Interface

The DUM module provides unit-level access to disk hardware.[31] It is either invoked directly by higher kernel layers, or indirectly as a result of calls to the DVM as described above. As a result of this, the DUM provides an attachment mechanism similar to that of the DVM to ensure exclusive access to a unit.

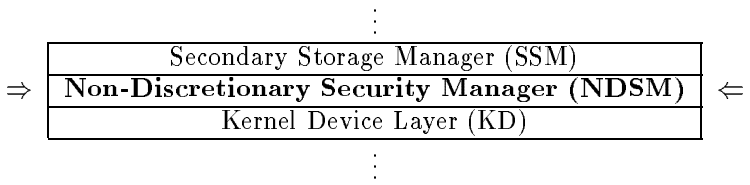
[31] Note that the DUM also provides unit-level access to streaming tape devices in the Trusted Bases. The ability to perform I/O and position the tape is provided. All tape I/O support through the DUM is provided by the Storager device driver.

4.2.1.5.4.3 Kernel Device Layer Device Drivers

The DUD module in the KD layer performs its actions through the use of entry points provided by modules specific to a given disk driver. These modules provide basic device driver services and are not discussed further. There are three such modules:

- The *Storager Driver*, which controls the disk device and the streaming tape device in the Gemini Trusted Bases.
- The *Western Digital Driver*, which controls the fixed and floppy disk devices in the PC-AT platforms.
- The *RAM Disk Driver*, which controls the RAM disk.

4.2.1.6 Non-Discretionary Security Manager (NDSM) Layer



The next kernel layer is the Non-Discretionary Security Manager (NDSM) layer.[32] This layer is responsible for all interpretation of the contents of an access label. This allows an implementation with a different policy to replace the module in this layer with one that supports the new policy (of course, this would require re-examination through the ratings maintenance process).

All access labels contain an access class that specifies the mandatory access policy information, as well as other information used to provide a ring integrity policy. The set of access labels forms a partial ordering with respect to the "dominates" relation. The partial ordering has the property that Least Upper Bounds (LUBs) and Greatest Lower Bounds (GLBs) are defined for any pair of labels. The ordering is a lattice since it also has the property that there is a Universal Upper Bound (System High) and a Universal Lower Bound (System Low).

In addition to the System High and System Low values, the NDSM maintains an *Effective System High* value. This is the read label of the access label range of the initial process on the system. Since process creation enforces the restriction that a upper-end of a child process' access label range is dominated by the upper-end of the parent's access label range, this effectively bounds all processes on the system. The effective system high value is the only data maintained in this layer.

Access labels managed by the NDSM have the structure shown in figure 4.7, page 71. This figure shows the following information:

RB*n*: *Ring Brackets*. Note that the RB1 value of an access label is defined to be a part of the non-mandatory-policy comparisons in the dominance relationship. The RB1 value of the subject's write label identifies the ring number of the subject.

[32] The term "Non-Discretionary" is used because the policy enforced by the NDSM layer includes both the "Mandatory" policy as well as the Ring Integrity policy.

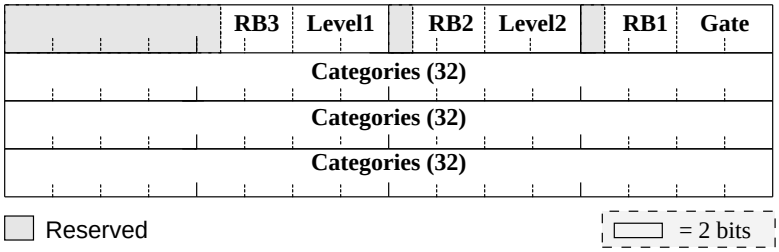


Figure 4.7. Structure of an Access Label

Level$n$: *Hierarchical Levels (16 levels).* Note that the GTNP supports *two* level fields. These two fields are used to provide secrecy and integrity policies. The latter is implemented by inverting the sense of the integrity label field and the integrity compartment bits. This allows the same comparison test to be done for both secrecy and integrity. See section 10.5, page 185 for an evaluator's comment regarding this.

Categories: *Non-hierarchical Categories (96 bits)*

Gate: *Gate Parameters.* This defines the allowed gate entry points for the segment. For more information, see section 4.2.2.1.4, page 86.

Given two access labels $\mathcal{A}$ and $\mathcal{B}$, the NDSM defines the dominance relation as follows:

$$\begin{aligned} \mathcal{A}\ \textbf{dominates}\ \mathcal{B} \quad \Leftrightarrow \quad & (\text{Level1}_{\mathcal{A}} \geq \text{Level1}_{\mathcal{B}}) \quad \wedge \\ & (\text{Level2}_{\mathcal{A}} \geq \text{Level2}_{\mathcal{B}}) \quad \wedge \\ & (\text{Categories}_{\mathcal{A}} \supseteq \text{Categories}_{\mathcal{B}}) \end{aligned}$$

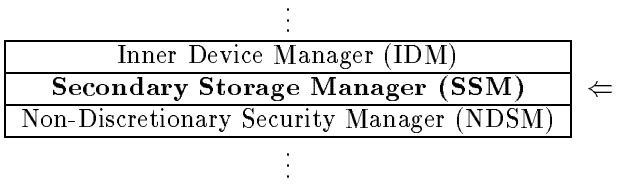
The NDSM dominance relation also enforces the non-mandatory policy based requirement that ($\text{RB1}_{\mathcal{A}} \geq \text{RB1}_{\mathcal{B}}$).

Entry points are provided in the NDSM to do the following:

- Test a dominance relation.
- Set and obtain the effective system high.
- Set and obtain the ring brackets in an access label.
- Obtain the gate parameter information from an access label.
- Obtain the "system high" and "system low" access labels.

The position of this layer in the kernel layering hierarchy is due to the fact that no layers beneath this layer need to make use of security policy enforcement. Layers above NDSM call NDSM entry points to perform security tests between labels.

4.2.1.7 Secondary Storage Manager (SSM) Layer



The layer above the NDSM is the Secondary Storage Manager (SSM). The SSM is responsible for defining and implementing the GEMSOS disk format on secondary storage and RAM disk devices. It provides segment I/O services to the Memory Manager layer, and uses the volume- and unit-oriented disk I/O services provided by the KD layer.

The SSM layer is responsible for providing the abstractions of segments and the segment naming hierarchy (described in section 4.1.4, page 54), and the abstraction of volumes (described in section 4.1.5, page 57). SSM provides higher kernel layers with the ability to open, close, create, delete, read and write segments to/from secondary storage; and to mount and dismount volumes. The SSM also provides higher kernel layers with the ability to format and manipulate volumes provided through the volume devices.

4.2.1.7.1 Segment Abstraction Implementation

In addition to the data segments on secondary storage or the hardware segments in memory, the implementation of the segment abstraction provides a segment naming mapping (used when the segment serves as a mentor) and synchronization objects; both of which are referred to using the segment name.

On secondary storage, each segment[33] is described in an *alias table entry* (which is the implementation of the segment naming mapping) associated with the segment's mentor. The alias table entry is part of a larger structure called an alias table. Every segment may have associated with it an alias table; if one is present, the segment serves as a mentor to a set of segments beneath it in the segment naming hierarchy. The alias table header contains a pointer to the segment's data segment, and a cryptographic checksum for the structure. The latter is part of the mechanism that provides label integrity.

Alias tables are not associated only with segments; every volume has a special structure called the *Volume Alias Table* (VAT). When the volume is mounted, the segment with which that volume's VAT is associated serves as the naming hierarchy base for all segments on that volume.[34]

4.2.1.7.2 Volume Abstraction Implementation

The other principal abstraction provided by the SSM Layer is the volume. Volumes represent collections of segments stored on secondary storage. The naming hierarchies are referenced through the use of the volume alias tables described above.

In order to connect the VAT of a volume to a known segment naming hierarchy, the volume must be mounted onto a segment in that naming hierarchy, which is then called the *mount segment* of that volume.

[33] This term will be used henceforth to refer to the abstraction; the terms *data segment* or *hardware segment* will be used to refer to the resource that contains arbitrary data.

[34] The VAT of the boot volume serves as the *System Mentor*.

The most critical structure on a volume is the VTOC, which contains header information about the volume and pointers to the other critical volume data structures. The VTOC is created when the volume is formatted through the volume device. It is in the VTOC that the volume format and volume ID are stored. There is a back link that specifies the absolute segment ID (Volume ID + Volume-relative Segment ID) of the mount segment. There are also four access labels maintained in the VTOC:

- A *write* access label, specifying the minimum (in terms of the dominance relation) access label of information that can be stored on the volume. This label is specified at the time the volume is formatted.
- A *read* access label, specifying the maximum (in terms of the dominance relation) access label of information that can be stored on the volume. This label is specified at the time the volume is formatted.
- A *VTOC* access label, specifying the access label required to mount the volume. This label is specified at the time the volume is formatted.
- A *Volume Attributes* access label, specifying the access label of information stored in the VTOC. This label is derived from the write label of the volume device that was used to format the volume.

The VTOC access label must be dominated by the volume's write access label, which must, in turn, be dominated by the volume's read access label: $label_{read}$ **dominates** $label_{write}$ **dominates** $label_{VTOC}$.

The information in the VTOC, along with the characteristics of the volume device upon which the volume is mounted, determine the attributes of the volume. These attributes include:

Sealed/Unsealed	If a volume is *sealed*, then all data segments on the volume are cryptographically sealed (i.e., have cryptographic checksums). Note that in all volume formats, the kernel data structures (VTOC, Alias Tables, etc.) on the volume are cryptographically sealed.
Encrypted/Non-encrypted	If a volume is *encrypted*, then all data segments on the volume are encrypted.
Removable/Non-removable	If a volume may be physically removed from the system, it is a *removable* volume. This is determined based on the device through which the volume is accessed. In the evaluated configuration, if a volume is removable, it must be sealed.
Bootable/Non-bootable	A volume is considered *bootable* based on an indication in the VTOC.
Read-only/Writable	The ability to write to a volume is determined at the time the volume is mounted based on the following information: the read-only indication in the VTOC, the mode specified on the mount, and the capability of the hardware to write the volume.
Recoverable/Non-recoverable	A volume is considered *recoverable* if there is a fixed association with a mount segment.[35] The restrictions imposed on recoverable volumes

[35]This recovery is limited to the ability to determine where in a naming hierarchy a volume had been mounted, if the information in the mentor is lost due to some form of corruption.

	are lifted for *non-recoverable* volumes; non-recoverable volumes may be mounted on any mount segment.
Local/Non-Local	A volume is *local* if its internal volume data structures are sealed using one of the local volume keys. *Non-local* volumes are those that were formatted with a non-local key. The type of key is specified at the time the volume is formatted.

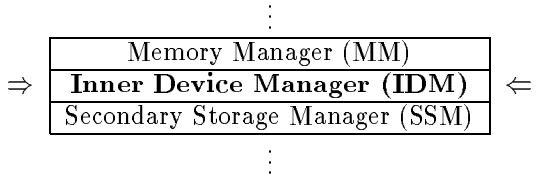
The use of any volume by the SSM layer requires that the SSM have access to the DES encryption/decryption facilities provided by the KD layer. This is because all volumes contain sealed TCB data structures, and volumes may have sealed data (if sealed) or encrypted data (if encrypted). When reading sealed structures from disk, the seal must be regenerated and checked; if the structure is modified, it must be recalculated before returning the object to disk.

4.2.1.7.3 Security Enforcement

Within the SSM, calls are made to the NDSM layer to check access labels and enforce the mandatory policy. These calls are made at the following times:

- *During Segment Creation:* A check is made to ensure that the access label of the segment to be created falls within the bounds of the volume's access label range.
- *During Mount Operations:* A check is made to ensure that the access label of the mount segment and the VTOC access label of the volume being mounted dominate each other. Checks are also made against the Internal I/O range of the volume: the access label range of the volume being mounted must be within the Internal I/O range of the volume device. Lastly, unless the volume is read-only or has a read access label equal to its write access label, the range defined by the volume's write and read access labels must be enclosed by the media resource exhaustion limit access label range defined for the volume device.
- *During Write Sequential Operations to Volume Devices* (i.e., volume formatting operations): A write sequential operation to a volume device ensures that, for a newly created volume, the read access label of the volume dominates the write access label of the volume, and that the write access label for a volume dominates the VTOC access label for the volume.

4.2.1.8 Inner Device Manager (IDM) Layer



The next layer in the kernel is the Inner Device Manager (IDM). This layer provides higher kernel layers with all interfaces to device drivers. For the devices used by the kernel (i.e., disk devices, selected tape devices,

PROMs, and encryption devices), it makes appropriate calls to the Kernel Device layer; for non-kernel devices, it calls the device drivers in this layer.

Entry points are provided in the IDM to attach and detach devices, to read and write (both sequentially and random), to obtain information on the next device event to be handled, and to obtain information on specific devices.

The IDM layer consists of multiple modules: an *IDM Common* module that provides the layer interfaces and forwards requests to the appropriate drivers, and the modules for each *device driver* that actually interact with their corresponding devices or the KD layer.

4.2.1.8.1 IDM Common

IDM Common accepts incoming device-related calls and forwards them to the appropriate drivers. It enforces access control to devices, and propagates I/O completion signals to higher kernel layers.

IDM Common defines a generic device driver interface; all actual device drivers conform to the requirements of this interface. IDM Common has no knowledge of the actual devices associated with the drivers; it knows only the interface. This allows for easy replacement and addition of device drivers.[36]

Higher kernel layers issue requests to the IDM to perform various operations with devices. Each of these requests refers to the device using a major/minor number pair. The major number[37] identifies a specific device category (e.g., Extended Serial I/O Device or Streaming Tape Device); the minor number identifies the particular instance of a device.

4.2.1.8.2 Specific Device Drivers

The other major conceptual portions of the IDM layer are the individual drivers for each device (one driver per device category). Each driver provides an interface to the device through a collection of logical devices, each device providing a number of functions. The driver generally translates these functions into controller requests that are then passed on to the device controller; kernel devices (disks, tapes through the Storager controller, System PROM, kernel metering, and devices utilizing encryption facilities) are an exception to this—they interact with the drivers in the KD layer.

Devices are categorized as either *finitely-attachable* or *infinitely-attachable*.[38] Finitely-attachable devices may only be attached by a finite number of subjects (typically one) at any given time. This could result in a covert channel through the resulting "busy" indication. To close the channel, if a device is finitely-attachable, the GTNP does not allow a subject to attach to the device unless the subject has observe and modify access at the device's write label.

Infinitely-attachable devices are not subject to this restriction; the mode required to attach is determined by the following checks:

- Unidirectional *Read* devices require observe access.

[36] Note that the introduction of drivers not covered by this evaluation invalidates the rating of the system.

[37] Major numbers are statically assigned by Gemini and never reused in the evaluated configuration of the kernel. Although there will be specific major numbers allocated for Gemini customer-linked device drivers, Gemini customers in general have the option to replace any driver in the system. In this sense, any major number may be reused by a Gemini customer, though such use will result in an unevaluated configuration of the system.

[38] "Infinitely-attachable" is Gemini's terminology.

- Unidirectional *Write* devices require modify access.
- Bidirectional devices require both observe and modify access.[39]

The following sections discuss the potential devices available in a GTNP system. For the purposes of discussion, these devices are grouped based on general characteristics. Not all devices are typically present in a GTNP system; the particular device drivers included in a configuration of a GTNP kernel is determined by the hardware platform and any optional devices ordered. If a device is not available, the appropriate device driver entry points are replaced with stub entry points that return a not-available indication to the IDM common.

4.2.1.8.2.1 Serial I/O Support Devices

The IDM layer provides four different types of serial I/O device drivers to support asynchronous RS-232 communications:

1. *Local Serial I/O (LSIO) Driver.* This driver interacts with the two serial I/O ports present on an 80286 or i486 board.
2. *Extended Serial I/O (ESIO) Driver.* This driver interacts with any of the eight serial I/O ports on an extended serial I/O board. The board is controlled via the iAPX 80188 processor in a manner similar to the Expandable Ethernet and HDLC devices.
3. *PC-AT Serial I/O (ASIO) Driver.* This driver interacts with either of the two potential serial I/O ports available on the PC-AT platform.
4. *i386 Local Serial I/O (SSIO) Driver.* This driver interacts with the single serial I/O port on an i386 board.

All of the serial I/O device drivers provide some mechanism that allows transmission of a non-spoofable indication that can serve as a secure attention key (e.g., <BREAK>). This is significant for the case where a virtual machine supporting users is composed on top of the GTNP. See section 10.6, page 186, for a discussion regarding this.

4.2.1.8.2.2 Network Devices

The GTNP provides the following two types of network device drivers:

1. *Ethernet Drivers.* These drivers provides a means to transmit and receive frames using both the standard Ethernet Data Link layer or IEEE 802.3 Media Access Control format.
2. *HDLC Driver.* This driver provides a means for transmitting and receiving HDLC-format serial frames, that can serve as a basis for constructing an X.25 protocol link.

[39] Two Bidirectional devices only require observe access. They are both clock devices (RTC and ABC). Writing to these devices only sets a process-local alarm clock; when the alarm expires, an eventcount (specified at the time the alarm was set) is advanced.

Both of these devices are built around the 186/51 processor board, and consist of two sets of components: one set that executes on the main GTNP processors, and one set that executes on the 80186. An Ethernet driver and an HDLC driver on a given 186/51 board share the same CPU. A given GTNP system may support up to 16 186/51 boards; each board supporting a single-level HDLC and/or a single-level Ethernet device.

4.2.1.8.2.3 Disk-related Devices

The GTNP provides the following three device drivers that interact with the various forms of disk media supported by the GTNP:

1. *Disk Device Driver*. This driver provides raw sector I/O and control functions for disk partitions.

2. *Volume Device Driver*. This driver provides logical formatting functions for creating and modifying GEMSOS volumes on disk partitions.

3. *Disk Unit Device Driver*. This driver provides installation and maintenance operations for all disk units in the system. A disk unit device corresponds directly to a physical device driver; its primary function is to provide bad track mapping and format/verify interfaces.

For the volume device driver, the control devices select the next operation to be done through the data device. The data devices are then used to format volumes and create and manipulate the VTOC.

All disk-related device drivers in the IDM do not interact directly with the device controller; rather, they interact with the lower-level device drivers in the KD layer.

4.2.1.8.2.4 Tape-related devices

The GTNP provides two device drivers that interact with the tape devices supported on the system:

1. *Streaming Tape Device Driver*. This driver is used to provide I/O and control functions to a Quarter-Inch Cartridge (QIC) streaming tape drive. Since the Storager controller serves as the device controller for the QIC streaming tape drive as well as the disk, this driver interacts with the Storager driver in the KD layer as opposed to directly to the controller.

2. *Half-Inch Tape Device Driver*. This driver provides access to the 1/2″ tape controller hardware, supporting multiple individual tape drives, each of which can be individually configured.

4.2.1.8.2.5 System PROM Device

The GTNP provides one device driver to interact with the System PROM. This device driver does not directly interact with the System PROM; instead, it uses the System PROM module in the KD layer. The System PROM is a battery-backed Complementary Metal-Oxide Semiconductor (CMOS) RAM that provides non-volatile storage of system configuration information. There is one System PROM per system.

4.2.1.8.2.6 Data Ciphering Processor-related Devices

The GTNP provides five device drivers that interact with the Data Ciphering Processor (DCP) on the Gemini System Controller (GSC) board. The DCP provides support for use of the DES algorithm. These drivers also interact (to some extent) with the System and System ID PROMs (also on the GSC board), for it is in these PROMs that the keys are stored. The DCP-related device drivers are:

1. *Key Management (KM) Driver.* This driver is used to construct composite keys from, and load keys extracted from composite keys into, the key tables maintained in the System PROM and System ID PROM.[40] This driver also allows Original Equipment Manufacturers (OEMs) to load end-user site keys for generating their own distribution volumes for end-user sites, with support from Gemini.
2. *Data Encryption Device (DED) Driver.* This driver is used to interact with the DCP for the purposes of data encryption.
3. *Data Sealing Device (DSD) Driver.* This driver is used to interact with the DCP for the purposes of generating cryptographic seals.
4. *Hardware Distribution Support (HDS) Driver.* This driver provides functions to support trusted distribution of hardware components. The basis of the distribution mechanism is the generation of an authenticator for a hardware component. The authenticator includes unique identification of the component, source of the distribution and destination of the distribution. The functions provided include obtaining authenticators for hardware components in a system and generation of authenticators for components being introduced into a system. The HDS device is intended to be used only by GTNP tools that support trusted distribution.
5. *Random Value Generator (RVG) Driver.* This driver is used to generate random 64-bit values (one at a time) using the Cipher Block Chaining (CBC) mode of the DES algorithm. These values are generated by using DES to encrypt a data stream using a hidden, random key stored in the System PROM.

Both the DED and DSD drivers virtualize the DCP and provide a configurable number of DED devices, each with its own data buffer and device security attributes.

None of the DCP-related device drivers interact directly with the DCP. Instead, they interact with the appropriate modules in the KD layer, which in turn interact with the CM layer.

4.2.1.8.2.7 PC-AT Human Interface Devices

The GTNP supports the following PC-AT human interface devices or device ports:

1. *PC-AT Serial I/O Device Driver.* This is the ASIO device discussed in section 4.2.1.8.2.1, page 76.
2. *PC-AT Keyboard Device Driver.* This driver provides the capability to receive hexadecimal codes transmitted from the PC-AT keyboard when keys are pressed. It also supports detection of a user-definable secure-attention key, which is reported as an exception on a data read operation.

[40] These key tables are not readable–either from the kernel or from outside the kernel; the System PROM (unlike the ID PROM) may have keys written into it; once written, these keys cannot be read by software.

3. *PC-AT Enhanced Graphics Adaptor (EGA) Device Driver.* This device driver provides an interface to the PC-AT EGA graphics controller. This controller primarily supports alphanumeric display capabilities. Although limited bit-mapped graphics are also supported, a light-pen is not.
4. *PC-AT Parallel Printer Port Device Driver.* This driver provides the capability to send data in parallel to a Centronics-compatible printer port.

4.2.1.8.2.8 Clock Devices

The GTNP supports three different clock devices—one for the Trusted Base, and two for the PC-AT platforms:

1. *Real-time Clock Device Driver.* This driver supports interaction with the battery-backed calendar clock on the Gemini Multibus platforms. This clock has a resolution of 10 milliseconds and counts up to 99 years, and provides support for alarms.
2. *PC-AT Binary Clock Device Driver.* This driver supports interaction with an awaitable 48-bit binary clock (with a resolution of 7.8125 milliseconds). This driver also provides support for alarms.
3. *PC-AT Calendar Clock Device Driver.* This driver is used to access the PC-AT Calendar Clock, which is a battery-backed device supporting a 100-year calendar and a time-of-day clock.

4.2.1.8.2.9 Kernel Metering Support Device

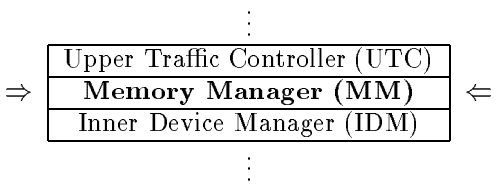
Kernel meters are a software abstraction that count the occurrence of specific events internal to the kernel. The specifics of these events are not detailed in the documentation with one exception; one kernel meter counts the number of times the ITC idle process has been scheduled.

These meters are used primarily as part of the kernel testing; thus, the kernel tested is the kernel fielded. The Trusted Facility Manual (TFM) directs that the system be configured with these metering support devices labeled as System High.

4.2.1.8.3 Security Enforcement

There are no specific IDM-common entry points that invoke the NDSM layer. However, individual device drivers may invoke the NDSM to perform security checks; these device drivers having been invoked by IDM entry points. In particular, multilevel device drivers (which are not part of the evaluated product) would be expected to invoke the NDSM to ensure that data passing through the device is within the device range defined by the write and read access labels of the device.

4.2.1.9 Memory Manager (MM) Layer



Above the IDM is the Memory Manager (MM). The MM is responsible for managing the memory allocated for storage of segments in both local and global memory. In addition to performing this function for segments visible outside the kernel, the MM layer provides memory management services for kernel segments. In both cases, it is the MM's function to place segments in the appropriate sections of memory (local or global) based on usage, and to handle allocation and deallocation of memory space for segment storage.

Segments that are visible outside the kernel are always stored on disk (although the disk may be a RAM disk), and swapped into and out of memory. To do this, the MM interacts with the SSM.

As described in section 4.1.4, page 54, application subjects obtain access to segments through the "make-known" operation, which introduces the segment into the process' LDT. Within the MM, this descriptor manipulation is done through interaction with the CM layer (note that the MM manipulates process-local descriptors only—no GDT manipulation is done). The manipulation associated with the various basic operations is as follows:

Creation The state of the process address space is not affected by segment creation.

Makeknown When a segment is made known, a specific LDT slot (specified by the application process) is initialized with the specified access and descriptor privilege level. The descriptor is marked "not present" and the base address set to zero.

Swapin When a segment is swapped-in, memory is allocated for the segment and the base address set. The descriptor is marked as "present."

Swapout When a segment is swapped-out, the base address is set to zero and the segment marked as "not present." The segment is removed from memory only when all processes using the segment have swapped it out.

Terminate When a segment is terminated, the LDT slot is invalidated.

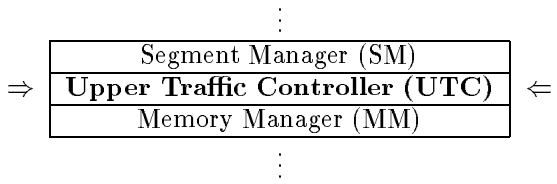
Deletion Deletion of a segment does not affect the calling process. However, if the segment is made known at the time of its deletion by another subject, the descriptors that contain the deleted segment are invalidated before the deletion operation is complete.

The MM layer manages the use of local memory (specific to a given processor) and global memory (available to all processors). In order to optimize performance and reduce bus contention, memory is allocated locally if at all possible. A segment is placed in global memory only if it is writable by one process and another process on a different CPU also has access to the segment. If multiple processes on different CPUs all have read-only access to a given segment, the segment is duplicated in the local memory of each CPU. If another processor then introduces the segment in write mode, the MM transparently moves the segment to global memory and updates the descriptors.

Security Enforcement

There is no security enforcement provided explicitly by the MM layer; enforcement is provided by the lower kernel layers.

4.2.1.10 Upper Traffic Controller (UTC) Layer



The Upper Traffic Controller (UTC) layer manages processes on virtual processors for given physical processors. The UTC maintains a table of all processes in the system and performs scheduling. Scheduling is done by strict process priority (a kernel gate is provided to allow a process to adjust its priority), and is preemptive. There is no time slicing; when scheduled, a process runs until preempted.

Processes are scheduled on *virtual processors* managed by the ITC. The UTC has several VPs available for each physical processor. Each process is scheduled on the physical processor whose local memory contains its code segment at the time of the process scheduling decision. Processes remain bound to a physical processor, but at a given time may be scheduled on any of the VPs associated with the physical processor. Scheduling uses an algorithm that uses priority, with pseudo-round-robin rescheduling for equal-priority processes.

The UTC maintains an active process table (APT) that contains state information about each process in the system.

4.2.1.10.1 Synchronization

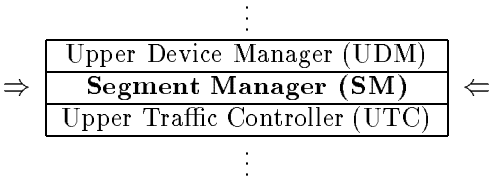
The UTC layer provides an eventcount and sequencer mechanism. Interprocess synchronization is accomplished through the use of eventcounts and sequencers, managed by MM, that allow asynchronous control of cooperating processes. Communication between processes is provided by the use of shared segments.

The *awaits* and *signals* supported by the UTC are supported through the use of *signal channels*. Each process has four signal channels. Signal channel 0 is used for supporting synchronous awaits. Synchronous awaits are provided for internal kernel use. The other channels are used to support signal channels available through the TCB interface; signal channel 1 is mapped to await channel 1, and so on. The await channels keep track of the eventcount and the eventcount value being awaited.

4.2.1.10.2 Security Enforcement

Any relevant security enforcement is provided by other layers.

4.2.1.11 Segment Manager (SM) Layer



Type of Operation	Entry Points	
Synchronization Operations	advance await signal read eventcount	await eventcount ticket
Segment Operations	create segment makeknown segment	delete segment
Process-Related Segment Operations	activate subject	
Volume-Related Operations	mount	dismount
Device-Related Operations	attach	detach

Table 4.1. Security Enforcement Done By The Segment Manager

All kernel layers to this point have either been global across all processors or processor-local. The remaining four kernel layers, however, are only *process*-local; i.e., all data structures in these layers contain information specific to, and accessible only by, a given process.

The first process-local layer is the Segment Manager (SM). The SM layer is responsible for the management of the per-process segment information. This includes the following:

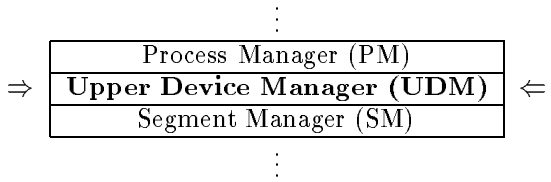
- Performing most segment-related security enforcement checks.
- Managing the association between subjects and access label ranges and privilege levels.

These functions are performed using information contained in the *Known Segment Table* (KST), which is managed by this layer and contains state information regarding segments made known by a process.

Security Enforcement

The SM layer is the principal locus of enforcement for access to segment-related information. Most of the SM entry points make calls to the NDSM layer to check access. These entry points are enumerated in table 4.1, page 82. In addition to checks made for segment-related and synchronization-related operations, the SM layer performs checks when creating processes and attaching devices. The process-related operations are checked because the SM is responsible for maintaining the mapping of subjects to privilege levels and access label ranges.[41] The device-related operations are checked because segments are used to synchronize with external devices.

4.2.1.12 Upper Device Manager (UDM) Layer



[41] This includes the checks that the range of a child process is enclosed by the range of the parent.

The second process-local kernel layer is the Upper Device Manager (UDM) layer. This layer manages the per-process device information, keeping track of the devices that have been attached by a process.

To the UDM layer, devices are complete abstractions. There is no knowledge of the physical characteristics of the device. All device-specific information is managed by the device drivers in the IDM layer. From the point-of-view of the UDM, the only significant characteristics of any device are the following:

- The type of I/O supported: *random access* or *sequential access*.
- The security characteristics of the device: *multilevel* or *single-level*.[42]
- The access mode required to attach the device: *observe* and/or *modify*.
- The "finite/infinite attach" attribute.

In order for an application process to use a device, it must first *attach* the device. It is at this time that security checks are performed, and the Process Local Device Number (PLDN) is assigned. Once attached, the PLDN is used to address the device via the *Known Device Table* (KDT) managed by the UDM. The KDT keeps track of all devices attached by the process.

Device operations can be asynchronous. If a synchronous operation is desired (and supported by the device), the application process may specify an eventcount to be used to signal completion of I/O.

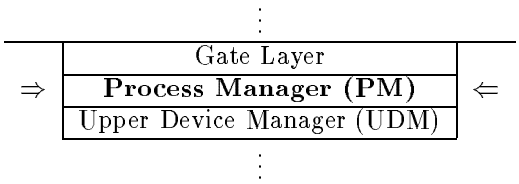
Once the device has been attached, I/O calls are issued to transfer information to and from the device. The type of call that is appropriate for the device (random or sequential) is part of the information obtained when the device is attached. When the process is finished using the device, it detaches it. The process cannot delete itself until it detaches all devices known to it.

Security Enforcement

In additional to the security enforcement performed during device attachment, the UDM layer also makes checks during subject activation. These latter checks verify that the new subject's ranges are consistent with the ranges of the currently attached devices. This validation of the subject ranges also occurs during process creation.

[42] It is important to note the distinction between the use of the terms *multilevel* and *single-level* as defined in the Trusted Computer Security Evaluation Criteria (TCSEC), and the actual access label range of a device. TCSEC multilevel devices must know how to label data as it is imported and exported, regardless of the specifics of the access label range. TCSEC single-level devices do not label data; the top and bottom of the access label range are typically the same (this is called a *degenerate range*). Note that it is possible for a multilevel device to have a degenerate range; this occurs when the top and bottom label of the multilevel device's range are the same. On the other hand, a single-level device could have a non-degenerate range. In the GTNP, this would require the range of the device to be enclosed by the range of any subject attaching the device. The subject would also be responsible for properly establishing the label of any data passing through the device. Since, in the evaluated configuration of the GTNP, all subjects have degenerate ranges, no devices with non-degenerate ranges can be attached.

4.2.1.13 Process Manager (PM) Layer



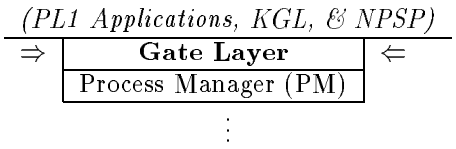
The third process-local layer, the PM Layer, provides the interface from outside the kernel for the management of the application process structure. The PM includes functions to create and delete processes.

Processes follow a tree structure. Every process (other than the initial process on each processor) has a parent that created it. Only a parent process can delete a child process, and this can be done only after the child process has marked itself for deletion and does not itself have children remaining.

Security Enforcement

All relevant security enforcement is performed in other kernel layers.

4.2.1.14 Gate Layer



The top-level of the kernel layer hierarchy is the Gate Layer. This layer receives all calls from the PL1 subject and directs them to the appropriate kernel routines. It also performs functions related to trap and signal handling for subjects outside of the kernel.

Subjects in PL1 enter the kernel by generating a specific kernel trap. After control is passed to the Gate layer, the Gate layer calls the appropriate kernel entry point. In these calls, the Gate layer utilizes parameters placed on the PL1 stack by the calling subject.

In addition to the use of hardware traps by PL1 to enter the kernel, the hardware trap mechanism is used to respond to exception conditions and to implement signal channels. If the trap was an exception trap or a signal trap, the Gate layer places an interrupt frame on the PL1 stack prior to returning control to PL1.

The sequence of events that occurs when PL1 issues a kernel call is as follows:

1. The PL1 subject loads the parameters onto the PL1 stack as appropriate for the call.
2. The PL1 subject then loads the kernel call number into the **AL** register and generates a Gate Interrupt.
3. The parameters are then copied into the kernel space and assembled.

4. The appropriate kernel call is then invoked.

5. After processing is complete, the return parameters are copied back into the PL1 space.

6. An interrupt return then occurs.

4.2.2 Kernel Interface

All of the kernel layers just discussed are unified into a single interface from the point of view of processes in PL1 (both untrusted application processes and the NPSP). This interface, which serves as the TCB interface, provides 29 gates,[43] as well as an interface to support PL1 trap handlers [48]. These gate calls, in conjunction with the non-privileged hardware instructions and PL1 trap handlers, provide a virtual machine monitor base for virtual machines implemented in PL1. Note that these gates are restricted such that they may only be called from PL1. In order to call them from PL2 and above, an appropriate PL1 layer must propagate the interface.

The gates provided by the GEMSOS Security Kernel fall into six groupings:

- Segment Management
- Process Management
- I/O Management
- Volume Management
- Process Synchronization
- Other Gates

The following sections discuss these gates. For each gate, a brief description of the gate is given, along with a synopsis of the security relevant checks made by the gate. After the discussion of the gates, a brief summary of the PL1 trap handling interface is given.

4.2.2.1 Segment Management Gates

The GTNP provides eight gates for the management of segments. These gates are described below.

4.2.2.1.1 `k_create_segment`

This gate is used to create a segment on secondary storage and introduce it into the segment naming hierarchy. The caller must supply the process-local alias ⟨mentor PLSN, entry number⟩ used to uniquely identify the new segment, the size of the segment (non-zero values specify the limit of the data segment), and the access label of the segment. A successful call results in the creation of the segment[44] and the synchronization objects. The data segment and synchronization objects are initialized to zero upon creation.

In order for the creation to be successful, the caller must have the mentor made known and have observe-modify access to it. Furthermore, the access label of the segment being created must dominate the access label of the mentor segment. The ring brackets of the new segment must also be in the correct order. Additionally, the volume upon which the segment is being created must not be read-only.

The creating process is guaranteed to have modify access to the new segment. There is no equivalent guarantee of observe access unless the segment crosses a volume boundary (i.e., it is a mount segment with data). Within a volume, a process may create a segment for which it does not have observe access.

[43] Twenty-nine, that is, from the point of view of the application. There is not actually a separate hardware gate for each gate call.

[44] No secondary storage is allocated for the data of the segment if the size is zero.

4.2.2.1.2 `k_delete_segment`

This gate is used to delete the specified segment from secondary storage and remove its name from the segment naming hierarchy. The calling process specifies the segment by its process-local alias (⟨mentor PLSN, entry number⟩). The segment may not be deleted if any of the following non-security related conditions are true:

- The segment is currently made known by the calling process.
- There is a volume currently mounted on the segment.
- The segment is a mentor to other segments.

With respect to the security policy, the calling process must have observe and modify access to the segment's mentor segment and observe access to the segment being deleted. Additionally, the volume containing the segment to be deleted must not be read-only.

4.2.2.1.3 `k_makeknown_segment`

This gate is used to introduce a segment currently existing in the segment naming hierarchy into the address space of the calling process. The caller specifies the process-local alias ⟨mentor PLSN, entry number⟩ of the segment to be introduced, the requested modes of access to the segment and synchronization objects, and the ring number for which the segment is being introduced. The caller also specifies the LDT location into which the descriptor for the segment is to be stored. In addition to status, the call returns the size and access label of the segment.

After the access checks described below are completed, the kernel selects a privilege level to be assigned to the segment. Multiple processes may have the same segment made known.

If the segment has a non-zero size and is being accessed in a mode other than "no-access," then a successful makeknown results in the initialization of an entry in the LDT of the calling process. The limit and access mode in the descriptor are initialized and the segment is marked "not present."

In order for the makeknown to be successful, the subject for which the segment is being introduced must be allowed the requested modes of access[45] to the segment and the synchronization objects. Furthermore, the subject in the ring in which the mentor is made known must have observe access to the mentor. Lastly, the specified ring number must be within the allowable range for the requested access mode. The caller is also subject to the resource constraints on available PLSNs and segments specified by the calling process' parent at the time of process creation.

4.2.2.1.4 `k_make_gate`

This gate is used to allow the calling process to create a call gate for subjects in different rings, and to put a reference to the gate in the caller's LDT.

When creating a gate, the caller specifies the PLSN of the code segment for which the gate is being made, the PLSN to assign to the gate, the ring number of the least privileged subject that can access the gate, the

[45] Modify access to the segment or synchronization objects is not allowed if the segment is on a read-only volume. Note also that there are separate sets of access modes for segments and synchronization objects.

number of parameter words to copy from the gate caller's stack to the stack in the more privileged ring, and the jump table index.

The gate field in the access label of the code segment and the jump table index specify an offset into the code segment.

In order for the gate creation to be successful, the ring number specified by the caller must be greater than or equal to the ring number of the most-privileged active subject, and less than or equal to the ring number of the least-privileged active subject. The access mode of the code segment must be either execute-only or execute-observe. Furthermore, the specified ring number value must be greater than the code segment's RB2 value and less than or equal to the RB3 value.

Note that if the subject in the ring specified by the caller is not active, then the privilege level selected for the gate must be greater than the privilege level of the code segment.

4.2.2.1.5 `k_terminate_segment`

This gate is used to remove the specified segment (or gate) from the address space of the calling process. The calling process provides the PLSN of the segment or gate.

If all conditions for termination are satisfied, the segment or gate is removed from the address space and the PLSN is made available. If the PLSN specified a segment and no other processes had the segment made known, the values of the synchronization objects are flushed to disk. Lastly, if the LDT slot contains a descriptor (i.e., the segment had non-zero length and an access mode other than "no-access"), the descriptor is marked "invalid."

There are no specific security related conditions that must be met in order for termination to be successful. However, the segment cannot, at the time of the termination request, be the target code segment of a call gate, swapped in, used for external device synchronization, or used for a signal channel.

4.2.2.1.6 `k_swapin_segment`

This gate is used to bring the specified segment into the memory space of the calling process. The calling process specifies the segment by PLSN. Once the segment has been swapped in, the descriptor is marked "present." The process can then construct a selector to access the segment by combining the privilege level of the segment with the PLSN of the descriptor of the segment.

There are no security relevant constraints on the swapin operation; however, the process cannot exceed the resource constraints on memory use specified by its parent. However, in order for the swapin to be successful, the segment cannot already be swapped in, have a zero length, or have been made known in "no-access" mode.

4.2.2.1.7 `k_swapout_segment`

This gate removes the segment specified (by the PLSN) from the memory space of the calling process. If no other processes have the segment swapped in, the segment is written back to secondary storage at this time. Additionally, the descriptor for the segment is marked "not present;" this invalidates any current selectors for the segment in the calling process.

The only restriction on this gate is that the segment must currently be swapped in.

4.2.2.1.8 `k_flush_segment`

This gate updates the image of the segment specified (by PLSN) on secondary storage. The segment must currently be swapped in.

This gate does not keep other processes from accessing the segment while it is being flushed. In order to guarantee that the secondary storage image of the segment is consistent with the memory image, there must separately be synchronization between the processes manipulating the segment.

4.2.2.2 Volume Management Gates

The GTNP provides two gates for the management of volumes, as described below.

4.2.2.2.1 `k_mount_volume`

This gate is used to attach a segment naming hierarchy associated with a volume on specific storage media to the current segment naming hierarchy. The volume number is an integer that reflects a partition on the storage media; volume numbers are assigned over storage media in the system that support volumes. If this is the first time the volume has been mounted (and it is recoverable), a permanent relationship is established between the volume and the mount segment.

A segment used as a mount segment must be made known by the calling process and must not currently be a mentor to other segments. Once the volume is mounted, that segment can only serve as a mentor to the segments on the mounted volume.

When a volume is mounted, the calling process specifies the PLSN of the segment upon which to mount the volume, the volume number of the volume to mount, and the requested mode of access to the volume. The calling process also specifies whether the mount is restricted to recoverable volumes only. The call returns the read, write, and VTOC access labels of the volume, and the format type of the volume. This information was specified when the volume was formatted through the I/O management interface.

In order for the mount operation to be successful, the following security constraints are enforced:

- The calling subject must have observe and modify access to the mount segment. Furthermore, the access label of the mount segment and the VTOC access label must dominate each other.
- The write access label of the volume must dominate the VTOC label.
- If observe-modify access was specified or this is the initial mount of a recoverable volume, the volume must not be a read-only volume.
- The calling subject must have observe access to the volume attributes.
- The access label range of the mounted volume must be enclosed by the Internal I/O access label range.
- If the volume is not read-only or single-label (i.e., read access label equal to write access label), then the volume's access label range must be enclosed by the media resource exhaustion label range defined for the volume in the System PROM.

- The subject must have observe and modify access at the associated disk device's write label and at the volume device's write label.
- The LHU for the specified volume must not currently be attached for raw disk I/O.
- If the volume device for this volume is configured to restrict mounts to local volumes only, then the volume must have been formatted using a local volume key.

There are a number of other non-security related restrictions enforced. These include checks that the volume is not already mounted, that the mentor segment does not have segments created off it, that the mentor segment did not have a different volume previously mounted on it, and that the limit for the total number of mounted volumes is not exceeded. The kernel also verifies that the volume to be mounted is appropriate for the device; that is, that the attributes for the volume device (sealed and/or encrypted) agree with those of the volume being mounted.

4.2.2.2.2 `k_dismount_volume`

This gate is used to make a segment naming hierarchy mounted on a known segment inaccessible. A dismounted volume may normally only be remounted onto the same mount segment. A volume may be dismounted only if no segments on the volume are currently made known by any process and there are no volumes currently mounted on the volume being dismounted. As a result of this, actions by subjects up to the System High access label can prevent a volume from being dismounted. As a result, the following security constraint is enforced:

- The subject requesting the dismount must have observe access at System High in order to dismount a given volume.

In order to dismount the volume, the subject for which the mount segment is made known must have observe and modify access to the segment, and must have a read label that dominates the effective system high label. Additionally, the mount segment must exist and have a volume mounted on it.

4.2.2.3 Process Management Gates

The GTNP provides five gates for the management of processes, as described below.

4.2.2.3.1 `k_process_create`

This gate is used to create a new child process; the calling process is the parent of the process. The parent process must specify the maximum ring range of subjects that the child process is allowed to have; a process may have up to eight subjects numbered 0 to 7. The parent process must also specify certain resources to allocate to the child (e.g., maximum number of processes the child can create). These resources are then subtracted from the parent's current allocation of resources.

The parent process also specifies a priority to be assigned to the child process. This will subsequently affect when the child process can be scheduled to run.

To terminate the child process and recover its resources, the child process must terminate voluntarily with a `k_self_delete` operation. The parent process can detect that the self-delete has occurred either by (a) attempting to delete the child and failing, or (b) through the use of an eventcount specified by the childprocess. After the child has self-deleted, the parent may perform a `k_child_delete` operation to remove the child process from the system and recover its resources.

In order for a process to be created successfully, the following conditions must be satisfied:

- The resources to be allocated to the child process must be available to the parent process (i.e., within its current resource allocation).
- Each subject specified for the process must contain valid access class ranges that are within the ranges of the parent.
- All segments passed to the child must be in within the access class range of the child.
- For each segment passed to the child, the child must have observe access to that segment's mentor (i.e., the child's read class must dominate the class of the segment's mentor).

Additionally, the (process local) child number assigned to the new process must not already exist, and the priority of the child process must be a valid priority value.

4.2.2.3.2 `k_self_delete`

This gate is used to place the calling process into a suspended state, waiting to be deleted by the parent process. If no exception occurs, an eventcount is advanced to notify the parent process of the child process' self deletion, and this call never returns.

The process may have segments made known at the time of self deletion. Those segments that are both made known and swapped in by the process are terminated and swapped out when the process is deleted by the parent process. Further, the process must have no attached devices.

The process will no longer be scheduled to run after a successful self deletion.

In order for a process to delete itself successfully, the PLSN of the segment whose eventcount will be advanced must be provided, and the process must have no child processes.

4.2.2.3.3 `k_child_delete`

This gate is used to delete a specified (by process local child number) child process and recover the resources allocated to the child during process creation. Note: a parent may delete only its *own* child processes.

In order for a child to be successfully deleted, it must be a valid process and that child must have previously performed a successful self-delete operation.

4.2.2.3.4 `k_activate_subject`

This gate is used to activate one or two subjects for a process. Subjects in PL2, PL3, or both privilege levels may be activated in a single call.

This gate also allows new subjects to be defined. A new subject is defined by a change in access label range or the addition of a subject in a ring for which there was not previously a subject defined. A process may have up to eight subjects defined.

In order for subjects to be successfully activated, the access modes for all objects that are currently made known in the process' address space must be allowed by the access label range(s) of the newly activated subject(s). Also, all devices that are currently attached by the calling process must still be accessible to the new subjects in the rings for which the devices are attached.

4.2.2.3.5 `k_set_priority`

This gate is used to change the priority of the calling process, and returns the original priority.

If the new priority is NULL (i.e., $FFFF_{16}$), the current priority of the calling process remains unchanged.

If the new priority is greater than or equal to the current priority at which the process is running, the process continues to run, and no rescheduling is performed. If the new priority is lower than the current priority, the process may be rescheduled.

In order for the priority to be changed successfully, the new priority must be in the valid range (0 to 32000).

4.2.2.4 Process Synchronization Gates

The GTNP provides seven gates for the synchronization of processes, as described below.

4.2.2.4.1 `k_advance`

This gate is used to increment the eventcount associated with the specified segment. The segment is specified by PLSN. This gate has two effects. First, it increments the value of the eventcount associated with the specified segment. Second, any processes that are waiting for the new value of the eventcount are unblocked and rescheduling takes place. The subject must have modify access to the synchronization object.

4.2.2.4.2 `k_await_evc`

This gate is used to wait upon the eventcount associated with the specified segment. If the current value of the eventcount associated with the specified segment has not reached the specified value, the process will be blocked. The process must have observe access to the synchronization object.

4.2.2.4.3 `k_read_evc`

This gate is used to read the eventcount associated with the specified segment. The process must have observe access to the synchronization object.

4.2.2.4.4 `k_ticket`

This gate is used to read and increment the sequencer associated with the specified segment. Typically, this gate is used in conjunction with the await gate to perform mutual exclusion. The process must have both observe and modify access to the segment.

4.2.2.4.5 `k_await_signal`

This gate is used to set up a signal await channel. A signal await channel provides the capability for a process to await eventcounts asynchronously. Processes have three signal channels available to them. The process supplies the signal channel number, the segment number associated with the eventcount, and value to await. For the gate to be successful, the subject in the ring in which the specified segment is made known must have permission to observe the synchronization object. When the value of the eventcount reaches the value specified in the await gate, an interrupt is generated. The **`k_signal_mask`** gate can be used to control the signal await channel mask.

4.2.2.4.6 `k_signal_mask`

This gate is used to change the current mask of the signal await channels for a process. The process provides the new value for the signal mask, and is returned the old mask and a success indication.

4.2.2.4.7 `k_replace`

This gate provides support for implementing mutually exclusive, non-interruptible updates of arbitrary data regions. It can be used when multiple processes need to read and write a shared data region in a consistent manner. The process must have observe access to the source segment and modify access to the destination segment. The process cannot be rescheduled during the call.

4.2.2.5 I/O Management Gates

The GTNP provides the six gates for device I/O, as described below.

4.2.2.5.1 `k_device_attach`

This gate is used to attach a device for use by the calling process. A device cannot be accessed by a process until it has been attached. The calling process specifies the major and minor numbers of the device, and the PLDN to be used as a handle for the device after attachment. The calling process also specifies the ring number of the subject for which the device is attached, the PLSN of a segment to be used for synchronization of I/O (NULL if there is to be no synchronization), and a pointer to any device-specific initialization information.

After a device has been attached, data transfer may proceed using the appropriate form of read and write calls. This I/O is asynchronous; however, the process may synchronize with the device subject through the use of the eventcount associated with the supplied PLSN. A segment whose eventcount is being used for this form of synchronization may not be terminated by a process until all devices using the segment for synchronization in that process have been detached.

Every device has associated with it an access requirement, as described in section 4.2.1.8.2, page 75. This requirement is either observe, modify, or observe-modify. In order for the attachment to be successful, the following conditions must hold:

- If *observe* access is needed, the subject's read label must dominate the device's read label, and the subject's ring must be in the read bracket of the device read label.

- If *modify* access is needed, the device's write label must dominate the subject's write label.
- If *observe-modify* access is needed, both conditions must be satisfied.

Additionally, if a segment is specified for external synchronization, the subject in the ring in which the segment is made known must have modify access to the synchronization object.

4.2.2.5.2 `k_detach_device`

This gate is used to detach the device specified (by PLDN) from the calling process. The PLDN can be reused after a successful detach.

The only significant restriction on detachment is that the device is currently attached.

4.2.2.5.3 `k_read_random`

This gate is used to read data from a currently attached random-access device. The calling process specifies, in addition to the device's PLDN, the address from which to begin reading data and the number of blocks to be read. The calling process also specifies the address of a buffer into which to place the data read. For single-level devices, the call returns the read class of the PL1 subject.[46]

Security enforcement is primarily done at the time of attachment. Additional enforcement may be done based on the security nature of the device. For example, when a request is made of a volume device to read volume attributes, the volume device enforces the constraint that the volume device read class dominates the volume attributes class. Constraints such as this are specific to the volume device.

In general, in order to read successfully from a random-access device, the device must be attached and be a random-access device.

4.2.2.5.4 `k_write_random`

This gate is used to write data to a currently attached random-access device. The calling process specifies, in addition to the device's PLDN, the address of a buffer from which the data is to be written, and the device address into which the data is to be written. The calling process also specifies the number of blocks to be transferred and the access label of the data. The access label is ignored if the device is single-level.

The only security enforcement provided directly by this gate is for multilevel devices; the call ensures that the access label of the data written is within the device's range. All other enforcement is done at the time of attachment. In order to write successfully, the device must be attached and be a random-access device.

4.2.2.5.5 `k_read_sequential`

This gate is used to read data from a currently attached sequential-access device. The calling process specifies, in addition to the device's PLDN, a pointer to a buffer to hold the data read. The number of bytes of data that will be input on each sequential read as a single data unit is specified during device configuration. For some devices, this number is a maximum; other devices always read the same number of bytes. The call

[46] The call interface is designed to support multilevel devices, even though they are not in the evaluated configuration. Multilevel devices would return the access label of the data read. This behavior was not evaluated.

returns the number of data units input. In addition, the call interface is designed for the eventual support of multilevel devices; for single-level devices (i.e., in the evaluated configuration), the call returns the read class of the PL1 subject.[47]

All security enforcement is done at the time of attachment. In order to read successfully, the device must be attached and be a sequential-access device.

4.2.2.5.6 `k_write_sequential`

This gate is used to write data to a currently attached sequential-access device. The calling process specifies, in addition to the device's PLDN, a pointer to a buffer containing the data that is to be written, a count of the number of data units in that buffer, and the access label of that data (this label is ignored if the device is single-level).

The only security enforcement provided directly by this gate is for multilevel devices; the call ensures that the access label of the data written is within the device's range. All other enforcement is done at the time of attachment. In order to write successfully, the device must be attached and be a sequential-access device.

4.2.2.6 Other Gates

There is one additional gate that does not fall into any of the previous categories. It is described below.

4.2.2.6.1 `k_shutdown`

This gate is used to stop the kernel and (optionally) reboot the system. The calling process supplies the mode of shutdown, as well as a message to be displayed on the console.

There are three classes of shutdown modes:

- The kernel reenters the kernel Boot PROM and prompts for a new boot volume.
- The kernel reenters the kernel Boot PROM and reboots the previously booted volume; no boot volume prompt is given.
- The kernel is halted and only manual reinitialization is allowed.

There are no restrictions on the use of this gate other than the fact that it may only be called from PL1. This allows applications to implement arbitrary restrictions on kernel call invocations in outer rings, as would typically be the case in an operating system or virtual machine executing in an outer ring. However, the Kernel Parameter Table may by configured to restrict the ability to use selected shutdown modes.

4.2.2.7 PL1 Trap Handling Interface

In addition to the gate calls exported, application processes may interface with the kernel through the use of traps. This is possible because the kernel provides a facility that allows PL1 to service traps that occur in PL1 or above by invoking a trap handler specified by the process.

[47] The design calls for multilevel devices to return the access class of the data read; this behavior is not evaluated.

4.2.3 Kernel Initialization

GEMSOS Security Kernel Initialization [60] provides the function of initializing the kernel and the associated hardware base. This includes the establishment of the initial hardware synchronization and environment, the bootstrapping from this environment to a fully initialized kernel one layer at a time, and the creation of the initial process on each CPU. Though GKI is not considered to be in the TCB, it is clearly crucial to achieving a secure initial state. It runs in PL0.

Interfacing with GKI is done through input and output databases. Many of these databases have stock configurations defined as part of the Kernel Initialization Tables CSCI [63, 69, 70, 55, 56], although use of these configurations is not mandatory (they can be entered through the system generation utilities). The input databases (either stored on the boot volume or in PROM) include the following:

Initial Process Parameter Table (IPPT)
: This table contains the process creation parameters of the initial processes to be created by GKI (i.e., the NPSP processes).

Kernel Parameter Table (KPT)
: This table is used to define internal kernel and certain system-wide limits.

Kernel Loader Table (KLT)
: This table contains the locations and unique IDs of the set of kernel code segments that must be loaded from the boot volume.

Boot PROM
: There is a Boot PROM for each processor. The Boot PROM contains the code that is first executed by the processor upon reset.

System ID PROM
: This defines selected encryption keys, model and serial numbers for the hardware base, the bus type, and the maximum number of CPUs for the system.

System PROM
: This PROM contains a number of databases that define the physical configuration of the system. Significant databases contained in the System PROM are the following:

  - A database that defines the volume partitions of the known volumes in the system.
  - A database that defines the access labels for known volumes and devices in the system.
  - A database that defines other characteristics of the known volumes in the system.
  - A database that defines the known devices in the system.
  - A database that defines the behavior of the system after system discontinuity (e.g. auto reboot).
  - Various tables that define information about I/O devices in the system, such as the Serial I/O devices.
  - Key Management tables.

The output databases include the Initial Parameter Record (IPR) provided on the stack of the initial process on each CPU (in the evaluated configuration, NPSP), the initial register contents, the kernel databases for each layer, and the Loader Environment database.

4.3 Kernel Gate Library (KGL)

The Kernel Gate Library [15, 18] presents a high-level language interface into the kernel. It may be used by PL1 subjects to invoke the kernel functions. It allows better access to the kernel (avoiding the inconvenience of using trap gate calls and packing data from a higher-level language) and provides 29 gates corresponding to the kernel interface in section 4.2.2, page 85. When the KGL is linked with the NPSP to provide a high level language interface to the kernel, KGL is considered part of the TCB. Copies of KGL code may also be linked within application processes, however these copies are not considered to be part of the TCB when executed within a non-TCB subject.

The KGL is divided into two sub-layers: the high-level language interface layer and the kernel interface layer. The high-level language layer presents the high-level language interface to the kernel functions. Due to the limited number of parameters that can be passed by a trap gate call, this layer organizes or "packs" simple data types into kernel-specific data structures. The kernel-specific data structures are then passed to the kernel interface layer. Upon return, the high-level language layer "unpacks" the data back into the high-level language structures.

The kernel interface layer contains a single module that provides the trap interface to the kernel. Calls that do not require any parameter packing go directly to this module. This module implements the kernel interface defined in the GTNP Security Features Users Guide (SFUG) Volume 2 [48]. This module is written in assembly language to use the hardware interrupt instructions necessary to invoke the kernel.

4.4 Network Processor System Parent (NPSP) Architecture

The NPSP [25, 16] provides for initiating the execution of customer-defined application processes on the GTNP. The NPSP is allocated to R1, which in the GTNP executes in PL1. The NPSP's function is provided only once per CPU per system boot. After the kernel has completed its initialization, execution on each CPU is transferred to the NPSP, a TCB multilevel subject. The result of the NPSP's processing is either the creation of a set of customer-defined application processes, or system shutdown (if an exception condition has occurred).

4.4.1 NPSP Database-Defined Interfaces

The NPSP has no human interfaces or programming interfaces to less-privileged rings. Information is passed only through input and output databases. The input databases include the following:

Offset Table	This table contains the starting segment offset values of the Process Definition, Segment Creation, Volume, and Volume Format Tables, along with indicators that reflect whether space has been allocated for these tables.
Physical CPU Table	This table provides a mapping between the physical CPU ID and CPU Indices (used within the NPSP to name CPUs).
Process Definition Table	This table specifies parameters for the creation of the customer-defined application processes.

Segment Creation Table	This table specifies the segments that are to be created by the NPSP prior to the mounting of volumes, as well as the segments that are to be created subsequent to the mounting of volumes.
Volume Format Table	This table specifies the format parameters for the volumes to be formatted.
Volume Table	This table specifies the parameters for the volumes to be mounted.
System Maximums Table	This table defines the size of the following tables: Physical CPU; Process Definition; Segment Creation; Volume Format; and Volume.
Initial Parameter Record	This record specifies the resources given to the initial process by the kernel.
Initial Process Register Contents	This record specifies the initial register contents for the initial process.
Image Segment	This is constructed by the Multilevel Subjects Intersegment Linkage Tool (MIT) and is used by the NPSP to build the Intersegment Linkage Output Database, a database needed by the run-time portion of the MIT.[48]

The output databases include:

Environment Definition Record	This information is provided to each process created by the NPSP. It is functionally equivalent to the Initial Parameter Record provided by the kernel to the initial process.
Child Process Initial Register Content	This database provides the initial contents of the registers of the processes created by the NPSP.
Mounted Volumes Table	This database is stored in a segment in the Non-TCB domain and contains information about the volumes that have been mounted by the kernel and NPSP.
ISL Output Database	This database is used by the MIT to provide run-time intersegment linkage (ISL).

4.4.2 NPSP Layers

Figure 4.8, page 98, depicts the layers in order from top to bottom, where the top layer in the figure is also the top layer (i.e., that layer upon which no other layer depends) in the NPSP CSCI. Note that the Self Initialization Layer and the Child Initialization Layer are both depended upon by other modules—even though they are not depicted at the bottom of the diagram.

[48] The Multilevel Subjects Intersegment Linkage Tool (MIT) provides intersegment linkage functions for the NPSP, similar in functionality to the services provided by the Intersegment Linkage Layer in the kernel. The MIT includes run-time intersegment linkage mechanisms linked into the NPSP code segments. These parts of the MIT are thus part of the GTNP TCB. The MIT also is responsible for generating databases used by the NPSP to construct the databases used by the run-time linkage mechanism of the MIT. The functions that generate these databases are not part of the GTNP TCB, but are part of system generation.

Self Initialization
System Startup
Child-/Re-initialization
System Configuration
External Storage
System Maximum
Process Mode
Miscellaneous
External Database Path
Non-discretionary Security
PLSN
Type Operator
Shutdown

Figure 4.8. NPSP CSCI Layers

Self Initialization Layer. This layer performs NPSP self-initialization. It includes the initial code that executes upon transfer of execution from the kernel to the initial processes. First the set of segments used by the NPSP are made known, and the ISL output database is initialized. Once the NPSP environment is set up, the NPSP modules are initialized. The self-initialization of each module includes allocation of external databases managed by the module, snapping the data links to a process-local database and/or the external databases managed by the module, and initialization of the module. Self-initialization is performed from the bottom up (i.e., modules in the lowest layer are initialized first, followed by the modules in the layer above it, and so on). Since different types of processes have different self-initialization requirements, there is a separate entry point for each type of process. Note, however, that in the GTNP there is only one type of process.

System Startup Layer. This layer starts up the GTNP system. It sets up the environment for user-defined processes, then creates the application processes, then goes to sleep without any provision for waking up.

Child-/Re-initialization Layer. This layer performs re-allocation and re-linkage for the NPSP configuration.

System Configuration Layer. This layer manages the Non-TCB Pre-mount Segment Definition database, the Non-TCB Post-mount Segment Definition database, the Non-TCB Volumes Definition database, the CPU Identifier database, the Customer-defined Trusted Subject Process Definition database, and the Volumes Format Definition database.

Higher layers depend on this layer to create the non-TCB segments prior and subsequent to the mounting of the non-TCB volumes, to manage the non-TCB volumes specified in the Non-TCB Volumes Definition database, including which access classes exist on which volumes, common access classes within each volume, and the algorithms that walk the path to the volume mount segment and mount the non-TCB access class leaf volume, to create and initialize the Mounted Volumes Table, to obtain the physical CPU identifiers, to obtain definitions of user-defined processes, and to format non-TCB volumes.

External Storage Layer. This layer manages the segments used to store various indirect external databases used in the NPSP. It also verifies that application ring numbers are outside of the NPSP's domain.

Higher layers depend on this layer to verify that the ring numbers of the application processes are valid, to verify that the ring (RB1) of a Non-TCB segment is valid, and to allocate the storage space for and return the location of the Non-TCB Volumes Definition database, the Customer-defined Trusted Subject Process Definition database, and the Volumes Format Definition database.

System Maximum Layer. This layer manages the database that reflects the system maximums (e.g., the maximum number of TCB volumes, non-TCB volumes, pre-mount segments to create, volumes to be formatted, and the maximum number of processes within each process family) defined by the security administrator. It also reflects the system's state transitions from the build state to the system configuration state.

Higher layers depend on this layer to obtain the system maximums, including the maximum number of definition entries allowed in each of the System Configuration Layer databases, to verify that the system is in the system configuration state when any of the System Configuration Layer databases is being accessed, and to return the current state of the system.

Process Mode Layer. This layer reflects whether or not the system has been started and whether or not the process is in the address space initialization mode.

Higher layers depend on this layer to verify that the system is still being initialized.

Miscellaneous Layer. This layer manages per-process device attachments and PLDNs. During process initialization, ranges of PLDNs are allocated for the NPSP. This layer manages the PLDNs within these ranges. This layer allows for the attachment and detachment of devices while enforcing the use of PLDNs that are allocated to the associated range.

Higher layers depend of this layer to allocate and manage the PLDN ranges, and to attach and detach devices.

External Database Path Layer. This layer manages the location and creation of segments that contain external static databases used in the NPSP.

Higher layers depend of this layer to find, create, make known, and swap in external storage space segments.

Non-discretionary Security Layer. This layers stores the access class range of the outer ring subject and stores the system access class range. It manages the interpretation of access classes and provides a dominance function.

Higher layers depend on this layer to provide access class manipulation services, to obtain the system-low access class, and to verify that the access class of the outer ring subject is process-low.

PLSN Layer. This layer manages PLSNs allocated to a process by dividing those PLSNs into four non-overlapping ranges. For the PLSNs within the NPSP PLSN range, this layer keeps track of the PLSNs that are in use and knows specifically the PLSNs of the NPSP stack segment, the synchronization segment, and the system mentor segment. This layer ensures that only the NPSP is allowed to use these PLSNs.

This layer also stores and protects the ring number of the NPSP, stores the hardware privilege level of the NPSP, stores the constant value 'zero-address,' stores and returns the ring number of the active PL2 subject, stores the information about logical volumes mounted by the kernel, keeps track of the amount of the memory consumed by the process, determines the size of the child processes' NPSP stack

segments, builds data links to indirect internal static databases, and makes known and terminates the PL2 subject's stack segment.

Higher layers depend on this layer to make known, terminate, swap in, and swap out, and return the PLSN of the synchronization segment, the NPSP stack segment, the system mentor segment, the Common Mentor segment, and the segment upon which non-TCB segments are mounted. In addition, higher layers depend on this layer to obtain the ring number and hardware privilege level of the NPSP, to get the PLSN of the stack segment for the NPSP within a child process, to allocate PLSN ranges, and to obtain global values.

Type Operator Layer. This layer interprets and manipulates specific data types. It performs arithmetic operations on unsigned 32-bit numbers. It also returns the NPSP CSCI version numbers and dates, and the version of the NPSP external databases.

Higher layers depend on this layer to manipulate pathnames, to return the NPSP CSCI version and version date, to obtain the version of the NPSP external databases, and to perform arithmetic functions on unsigned 32-bit numbers.

Shutdown Layer. This layer provides the capability for higher-layered modules to perform system and process shutdown.

Higher layers depend on this layer to shutdown the system when a fatal error is detected.

4.4.3 NPSP Initialization

When the kernel initially boots, it may mount a number of volumes along the path to the NPSP stack segments. The volume structure existing after system boot is determined by the Initial Process Parameter Table (IPPT). After the kernel has completed booting, distinct volumes may already be mounted on the Common Mentor, the Common Stage, the TCB Root, or the Acute AC Stage segments. These are a set of segments that SYSGEN builds so that the remainder of the GTNP can be booted.

As the first step in the initialization of the NPSP CSCI, the NPSP walks the path from the System Mentor to the Offset, Process Definition, Volume, Segment Creation, Volume Format, Physical CPU, and System Maximums Tables. The NPSP then references the Offset Table to obtain the starting segment offset values of the Segment Creation, Volume, Volume Format, and Process Definition Tables. The System Maximums Table defines the number of entries in each of these tables, except the Offset Table. The NPSP then references the Segment Creation Table and creates the pre-mount segments. It then references the Volume Format Table and formats the specified volumes. The NPSP then references the Volume Table to retrieve a list of volumes to be mounted, along with the paths to each of their mount segments. It walks the path to each mount segment and mounts each volume in the table. The NPSP also copies the Volume Table into the Mounted Volumes Table in the non-TCB domain.

The Segment Creation Table is referenced again and the NPSP creates the specified post-mount segments. The NPSP references the Initial Parameter Record, provided on the Initial Process stack, to retrieve the Physical CPU ID corresponding to the CPU on which that Initial Process is running. The NPSP uses this value to acquire the corresponding CPU Index from the Physical CPU Table.

The CPU index is used by the NPSP to determine the processes specified in the Process Definition Table that are to be created by the Initial Process running on a particular CPU. The NPSP creates each of those processes, subsequent to writing the Environment Definition Record database on the processes' stacks using information from the Initial Parameter Record, the initial processes' initial register content, and the Process

Definition Table. After the NPSP has created each of the processes that correspond to the CPU on which this instantiation of the NPSP is running, it goes to sleep without the provision for waking up.

4.5 System Generation Utility Software

System generation and system configuration functions are provided by utilities (collectively called the "system generation utilities," even though they also provide maintenance and configuration capabilities) that run on the same hardware base as the GTNP. These tools serve to generate and configure the GTNP by defining the attributes of the initial environment and application processes which are to be created. These tools are viewed as "offline" in the sense that they execute before the establishment of the secure initial state and separately from the runtime GTNP which they generate and configure, and are not supported by or available through any runtime GTNP interface. In addition, the only human interfaces provided by the GTNP are through the system generation and configuration tools described below. Note that some of the tools below are used in trusted recovery (see section 7.4, page 125).

The functions provided by the GTNP's system generation utilities can be divided into the following categories:

- Administrator.
- Operator.
- System Maintenance.
- System Generation.

Many of these functions must be restricted to specifically authorized personnel. Such restrictions are enforced by combinations of physical protection and user authentication.[49] However, the authentication mechanism does not distinguish any specific user role other than that of Mandatory Security Administrator (MSA). Of those functions protected by user authentication, certain ones are available only to authorized MSAs; the remainder are available to any user defined by the MSA, regardless of actual role. It is the administrator's responsibility to restrict personnel access to the utilities that provide these functions, when appropriate, via physical controls. The following sections summarize the functions provided for each category organized by the system generation utilities used to perform them.

4.5.1 Administrator Functions

The GTNP Configuration Tools (NPCT) are used to define authorized users of the system generation utilities, password lifetime and expiration time for users of system generation utilities, the access class names used by system generation utilities, access class ranges and access requirements for devices used both by GTNP systems and system generation utilities, the fundamental GTNP system maximums, the attributes of initial application processes' environment that is to be established during GTNP system start-up, and attributes of initial application processes to be created during system start-up. It is also used to backup and recover System PROM files. The Original User Installation (OUI) program is used to recover original preinstalled

[49] Note that this user authentication is used only during system generation. The user is authenticated by providing a password. Passwords are generated by the system and stored in the System PROM and can not be modified by untrusted users.

user definitions. The Hardware Distribution Utility (HDU) is used to verify the trusted distribution of hardware components, authenticate currently installed hardware components, and modify the Hardware Distribution Table (HDT) to reflect site-local hardware component changes.

Functions identified here may be performed only by an authorized MSA, as enforced by user authentication. The MSA position must be requested during login to obtain access to such information. The only exception to this requirement is in exercising the system recovery functions, which cannot perform user authentication due to the lack of reliable user information in the System PROM. Original user recovery may be performed by anyone who has physical access to OUI. The TFM warns that this utility should be carefully protected, since it is capable of rendering the system inoperable if misused.

4.5.2 Operator Functions

The Hard Disk Conditioning Utility (referred to as DSKGEN) is used to physically condition hard disks. The NPCT is used to set the system clock, compose and/or load crytographic keys, copy, backup, and recover system volumes, define autoboot paramaters, and define the operator console. The System PROM Bootstrap Utility (SPBU) is used to load the random value generation key, recover trusted distribution keys, and reset System PROM files to default values.

These functions may be performed by any authorized user, as enforced by user authentication, who has physical access to DSKGEN, NPCT, and SPBU. An exception to this is in exercising the system recovery functions, where login cannot be required due to the lack of reliable user information in the System PROM. Operator functions that require login are not available in the MSA position.

4.5.3 System Maintenance

The System Maintenance Utility (SMU) is used to define physical disk partitions, define RAM disks, allocate logical volumes to disk partitions, and modify physical device parameters as necessitated by routine maintenance.

Certain hardware maintenance and configuration functions clearly affect system security, yet are not functions typically performed in the role of a security administrator. These are functions that would be performed only by the manufacturer if there were no need to configure or upgrade the system on-site for site-specific requirements. The system maintenance role identified here is executed by one who effectively completes the on-site building of the system by knowledgeably performing such maintenance and configuration functions. These functions may be performed by any authorized user, as enforced by user authentication, who has physical access to SMU. These functions are not available if the MSA position is requested during login.

4.5.4 System Generation

The System Generation Utility (SYSGEN) is used to format volumes, set the volumes' read-only attribute, establish links between volumes and their mount segments, manipulate segments, define system resource limits used by the kernel, and define the configuration and initial environment of initial GTNP TCB processes (the NPSP).

The installation and correct configuration of the GTNP TCB's software is also security-relevant, but is not a function typically performed in the role of a security administrator. The system generation role identified is executed by one who knowledgeably performs these functions. These functions may be performed by any authorized user, as enforced by user authentication, who has physical access to SYSGEN. These functions are also not available in the MSA position.

This page intentionally left blank

Section 5

TCB Protected Resources

As the GTNP is a network M-component, its subjects and objects are not the same as might appear in a more conventional operating system. Subjects, for example, are not associated with a single, identified user; this must be done by other network components, as outlined in the Network Security Architecture and Design [97]. Similarly, objects have no "owners;" the only concern is the mandatory label enforcement.

Analysis by the vendor and the team determined that the active entities in the system are device subjects and programming subjects. The objects are segments and various structures associated with segments (synchronization objects, volume attributes, and mentor information). Each of these is explained in further detail below.

5.1 Subjects

The types of subjects in the evaluated configuration of the GTNP are as follows:

- *Programming subjects*, which are processes executing in domains outside the kernel boundary. With the exception of the Network Processor System Parent (NPSP), all programming subjects in the GTNP are untrusted.
- *Device subjects*, which are abstract trusted subjects that execute within the kernel, but separate from the reference monitor.

Each of these types of subjects are discussed in more detail below.

5.1.1 Programming Subjects

Programming subjects are processes executing in non-kernel domains. A process is a hardware task defined by an address space and a single point of execution; the "domain" of a subject is the ring at which the process is executing. The GTNP provides an eight-ring hierarchy (numbered zero (R0) through seven (R7)) to processes outside of the kernel. Thus, each process may be simultaneously associated with up to eight subjects; each subject being uniquely characterized by the process and the ring.

Only three of the eight possible subjects in a process may be active at any one time. This is because the eight potential subjects are mapped to the three available non-kernel hardware privilege levels. Since a process is characterized as having a single point of execution, at any point in time the process is executing within one of the three currently active subjects. Thus, process/domain pairs are subjects because they identify active entities (defined by the process) that possess a particular set of privileges and access rights (identified by the domain).

Each subject has a *read* (maximum) and a *write* (minimum) access label, referred to as an access range. As one moves from more-privileged rings to less-privileged rings (i.e., numerically increasing ring values), the access range of the subject in the less privileged ring must be enclosed by the access range of the subject in the more privileged ring. This is not significant for the GTNP evaluated configuration, which requires that the read label and the write labels be equal; however, other configurations of the GTNP support subjects (called *multilevel subjects*) with true ranges (read label strictly dominating the write label). The Formal Security Policy Model and Formal Top-Level Specification make no distinction between the single-level and multilevel subjects.

A programming subject in a given privilege level has the ability to redefine the ranges of the subjects associated with numerically higher privilege levels. In the evaluated configuration, this ability is meaningless, as the single-label nature of untrusted programming subjects prevents any label changes.

While in some systems subjects also qualify as objects, this is not the case in the GTNP. There is no way for one subject to communicate with another subject other than through objects. Since programming subjects do not share or store information, they do not qualify as objects.

5.1.2 Device Subjects

Devices also are considered to be subjects in the GTNP [132]. Device subjects are a conceptual entity created each time a programming subject attaches a device. When a programming subject invokes the kernel to attach a device (initiating a device driver), it includes as parameters a synchronization object to be used for signalling. A device subject is created by the kernel, having the same access label range as the caller of the attachment. The device subject brings the synch object into its address space. Thereafter, data transfer takes place in a straightforward manner: the programming subjects makes a gate call specifying an Input/Output (I/O) buffer; the device subject introduces the I/O buffer into its address pace, transfers the information to or from the device, and terminates the buffer from its address space and increments the eventcount when the transfer is completed. When the programming subject detaches the device, the device subject removes the eventcount from its address space and becomes inactive.

The effects of a device read or write (data transfer and signalling using the synch object), therefore, are performed by the device subject. These activities occur asynchronously with respect to actions of the programming subject; in particular, these actions may be simultaneous with actions of the programming subject. The device subject therefore takes an active role in data transfer, uncontrolled by the programming subject that interacts with it.

Device subjects are within the kernel and have a domain of PL0. They are considered to be trusted subjects. While device subjects could be implemented in a separate domain between that of the reference monitor and the programming subjects, the vendor has chosen to restrict them to the reference monitor domain. Note that this differs from their treatment in some other systems.

5.2 Objects

Analysis by the vendor and the team identified four resources that qualified as objects meriting Mandatory Access Control. The resources are as follows:

Segments	Segments are units of main memory of variable size (see section 4.2.1.7, page 72) that may be mirrored on permanent storage. They are accessible (as a class) to trusted and untrusted subjects, have a distinct interface, a distinct name space, and access to them is mediated by the mandatory access control policy.
Eventcounts and Sequencers	These are integer values associated with segments, used for process synchronization (see section 4.1.6, page 58). As outlined above, subjects have shared read and write access to them. The name space and the access label are derived from the segments with which they are associated. They are protected by the same mandatory policy that protects the segment.
Mentor Information	The mentor information object is a data structure containing selected attribute information (such as alias table entry usage) about a segment. The name space and the access label of a mentor information object is derived from the segment with which it is associated. It is protected by the same mandatory policy that protects the segment.
Volume Attributes	Each volume (see section 4.2.1.7, page 72) has a collection of attributes that is considered to be an object; this information includes the format type of the volume, the volume's access label range, and the volume's key type. These attributes are stored in the Volume's Volume Table of Contents (VTOC), classified with the Volume Attributes label.

5.3 Devices

As described above, the GTNP system models devices as a special class of subjects. However, the Trusted Computer System Evaluation Criteria (TCSEC) describes characteristics that are incumbent upon devices irrespective of their treatment as subjects or objects. This section describes these characteristics of GTNP devices.

The set of devices available in a given GTNP system is defined using the system generation utilities and stored in a table in the System Programmable Read-Only Memory (PROM). This information can only be changed via the system generation tools; there are no facilities to update this information through the TCB interface. For each device, this table defines the write and read labels of the device. Additionally, for volume drives, the table defines the device busy exemption. There is no explicit single-level/multilevel indicator maintained in the table; the single-level/multilevel nature of the device is implicit from the particular device driver used.[1] All devices in the evaluated configuration have degenerate (read label equal to write label) access label ranges.

In order to attach a device for read/write access, the range of the device must be enclosed by the range of the attaching subject. Read-only devices require that the subject's read label dominate the device's read label. Write-only devices require that the device's write label dominate the subject's write label.

[1] For the evaluated configuration, Gemini provides only single-level device drivers.

This page intentionally left blank

Section 6

TCB Protection Mechanisms

The GTNP provides a number of protection mechanisms for its resources: as per the requirements for an M-component, all TCB resources (see section 5, page 105) have labels (*access labels*) permanently attached to them. Labels are a multi-field data structure containing security information about the resource; they are used by the Non-Discretionary Security Manager layers to provide enforcement of the Mandatory Access Control (MAC) policy, outlined below, between GTNP subjects and objects. Other network components (e.g., a D-component or I-component running on top of the GTNP) can also use the MAC facilities to perform access control on the objects that they directly control.

The following sections describe the protection policies provided by the GTNP. The discussion begins with a description of the label format and the policies enforced. Next, the specific access checks performed when accessing objects are described. This is followed by a discussion of how these protection mechanisms are applied to devices. Lastly, the application of object reuse to storage objects is discussed.

6.1 Protection Policies

The GTNP protects objects through both a *mandatory access control* policy and a *ring integrity* policy. These policies are defined by the Philosophy of Protection. The mandatory access control policy is formally described in the formal model; the ring integrity policy is described in the descriptive top-level specification. Both policies are implemented by the Non-Discretionary Security Manager (NDSM) layer of the kernel (and of GEMSOS Network Processor System Parent (NPSP), which uses the same code).

Access modes are defined as follows:

- In order for a subject to read or derive information from an object, that subject must have *observe* access to the object.
- In order for a subject to alter the contents of an object, the subject must have *modify* access to that object.

The set of allowed access modes for any subject and object pair is determined based on the access labels of the objects. A GTNP access label is a 128-bit data structure as shown in figure 4.7, page 71. The first 32 bits contain a reserved byte, two 4-bit hierarchical security levels, three 3-bit ring brackets, a 4-bit gate-parameters field, and three non-adjacent reserved bits. The remaining 96 bits are used for non-hierarchical categories. Each bit represents one category, and the bit value indicates the presence or absence of that category. Each object has an access label associated with it. Each subject has two labels associated with it; one is a maximum or *read* label and the other is a minimum or *write* label.

As detailed in section 5, page 105, those objects protected by mandatory access control are segments, eventcounts/sequencers, volume attributes, and mentor information. The GTNP associates a label with

each of these objects whenever they are created. Once the objects are created, the labels cannot be changed. When subjects make kernel calls to access these objects, the NDSM layer is called to determine if the proper dominance relations exist. The results are used to determine if access is allowed.

6.1.1 Mandatory Access Control Policy

When a subject attempts to access a protected object, the NDSM is called by a higher kernel layer to establish the presence or absence of a dominance relation between the two labels. The dominance relation includes the following MAC checks as described in section 4.2.1.6, page 70: given two labels $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A}$ is said to dominate level $\mathcal{B}$ if both hierarchical levels of $\mathcal{A}$ are greater than or equal to the hierarchical levels of $\mathcal{B}$, the categories of $\mathcal{A}$ contain all the categories of label $\mathcal{B}$.

The GTNP uses the two hierarchical labels and the categories to support the notion of secrecy (*à la* Bell and LaPadula [2]) and integrity (*à la* Biba [3]). For the secrecy component of the label, the lowest hierarchical secrecy level has the lowest numerical value, and category bits are set to one to indicate the presence of a secrecy category. For the integrity portion of the label, the lowest hierarchical integrity level has the highest numerical value, and category bits are set to zero to indicate the presence of an integrity category. Thus, the lowest access class in the system is all zeros (lowest secrecy level, no secrecy categories, highest integrity level, all integrity categories) and the highest access class in the system is all ones (highest secrecy level, all secrecy categories, lowest integrity level, no integrity categories).

6.1.2 Ring Integrity Policy

The ring integrity policy supplements the MAC policy of the GTNP and allows the creation of privilege domains. Part of this is implemented by additions to the dominance relation; the remainder is implemented through ring bracket checking and other specific ring-based checks. These checks are summarized in table 6.1, page 111.

As described above, all subjects and objects have ring brackets stored as part of their access labels. For segments, these three values (RB1, RB2, and RB3) define the ring brackets used to further restrict accessibility, as described in section 4.1.2.1.2, page 49. Furthermore, the RB1 of an segment not only defines the highest privileged ring that can write the object, but effectively defines the ring in which the segment is said to reside. Thus, kernel segments, which have an RB1 value of zero (limiting the ability to write these segments to the kernel and the GEMSOS System Generation Utility (SYSGEN)), can be called R0 segments.

For subjects, the interpretation of the ring brackets in the access label range is as follows:

- The RB1 in the *write label* of the subject defines the ring of the subject.
- The RB1 in the *read label* of the subject defines the least privileged ring in which the subject may activate another subject.

For programming subjects, the GTNP does not use the RB2 and RB3 values in the read and write labels; they are set to the RB1 values. The RB2 value in the read label of the device does have special meaning; this is described in section 6.3, page 114.

Requested Access Mode	Relationship Between Ring Brackets of Obj.	Condition Required to Allow Req. Access Mode
Read Only	All cases	$0 \leq \mathcal{R} \leq$ RB2
Write Only	All cases	$0 \leq \mathcal{R} \leq$ RB1
Read Write	All cases	$0 \leq \mathcal{R} \leq$ RB1
Execute Only	All cases	RB1 $\leq \mathcal{R} \leq$ RB2
Read Execute	All cases	RB1 $\leq \mathcal{R} \leq$ RB2
Execute Only Conforming	All cases	((RB1 $\leq \mathcal{R} \leq$ RB2) $\wedge$ (RB2 = *ring of least priv'd def'd subject*))
Read Execute Conforming	All cases	((RB2 = *ring of least priv'd def'd subject*) $\wedge$ RB1 $\leq \mathcal{R} \leq$ RB2
Gate Access	RB2 < RB3	(RB2 < *gate ring number* $\leq$ RB3)
	RB2 =RB3	*No Gate Access Possible*
$\mathcal{R}$ is the requested ring for the object. RBn is a ring bracket of the object.		

Table 6.1. Ring Bracket Checks

As described in section 4.2.1.6, page 70, the dominance relation for access labels also includes a check that is part of the ring integrity policy. Given two labels $\mathcal{A}$ and $\mathcal{B}$, this additional check requires that not only must the MAC conditions be met for $\mathcal{A}$ to dominate $\mathcal{B}$, but the RB1 value of $\mathcal{A}$ must be numerically greater than or equal to the RB1 value of $\mathcal{B}$.

Note that the inclusion of RB1 in the dominance test has the effect that, within a process, all subjects of a lower privilege have read and write labels that are enclosed—both in terms of MAC labels and RB1 values—by the read and write labels of higher privilege subjects (since the access label range of a subject in a ring must be enclosed by the access label ranges of all higher-privileged rings).

6.2 Object Protection

The following sections describe the types of access required for each class of object controlled by the GTNP. In general, the following restrictions apply:

- For *observe* access, the subject's read access label must dominate the object's access label. This implies the following:
  - The subject's maximum label dominates the label of the object.
  - The object must reside in a ring of greater or equal privilege than the least privileged ring in which the subject can activate another subject (since the read label of the subject must dominate the label of the object).
- For *modify* access, the object's access label must dominate the subject's write access label. This implies the following:

- The subject's minimum label is dominated by the label of the object.
- The object must reside in a ring of lesser or equal privilege to that of the subject.

6.2.1 Segments

Segments are a software abstraction, as explained in section 4.1.4, page 54. Segments have an access label permanently associated with them, stored in the mentor's alias table entry (see section 4.1.4, page 54) for that segment. This access label cannot be changed after segment creation. Protection policy checks are required during creation, deletion, or makeknown operations on segments. All of these operations are initiated at the Kernel Gate layer, which calls the Segment Manager (SM) layer to carry them out. In particular, the SM layer calls the NDSM to return whether or not the proper relationships exist. When this decision is made, the higher layers enforce the policy decision.

In order to create a segment with a given access label, a subject makes the kernel gate call `k_create_segment` (see section 4.2.2.1, page 85). The caller must have observe and modify access to the segment's mentor. Furthermore, the mentor access label must be dominated by that of the segment to be created; this results in the naming hierarchy characteristic that access labels increase (i.e., move towards the system high access label and the least privileged ring) as the depth of the naming tree grows. Additionally, the RB1 value in the segment label's ring bracket must be less than or equal to the RB2 and RB3 value. Segments cannot be created on a read-only volume.

In order to delete a segment, the calling subject must have observe and modify access to the mentor, and observe access to the segment.[1] The deleting subject must not have the segment currently in its address space (i.e., the segment must be terminated), and the volume must not be read-only.

In order to make a segment known to a subject, the subject must satisfy security constraints for the requested type of access (observe, modify, or observe-modify), based on the access mode requested. In addition, the requested ring number must satisfy the ring bracket test for the requested mode of access.

Other operations on segments perform no significant access checking.

6.2.2 Eventcounts and Sequencers

Every segment in the GTNP system, including zero-length segments, has associated with it a pair of synchronization objects: an eventcount and a sequencer. These synchronization objects are stored along with the segment attributes in the segment's mentor's alias table. The security label associated with the synchronization objects is the same as the label of the segment.

There are six kernel gates through which subjects may access eventcounts and sequencers. The SM invokes the NDSM at the time synchronization objects are made known and computes the allowed access. When the `k_advance`, `k_await_evc`, and `k_await_signal` calls are invoked, the kernel gate layer invokes the SM layer. The SM layer then invokes the Upper Traffic Controller (UTC) layer, which calls the Memory Manager (MM) layer to either advance or read the eventcount value. The UTC then performs scheduling activity based on the eventcount value. When the `k_ticket` and `k_read_evc` calls are invoked, the kernel gate layer

[1] This has the implication that single-level subjects cannot delete upgraded segments. This concern is specifically discussed in the Security Features Users Guide (SFUG) and noted in section 10.4, page 185.

invokes the SM layer. The SM layer then invokes the MM layer to either ticket the sequencer or read the eventcount value.

The synchronization object is placed into the subject's address space along with the associated segment by the `k_makeknown_segment` call; hence, all checks for makeknown are performed. The synchronization object and the segment each have a distinct access mode specified when the segment is made known; in order for the makeknown to be successful, the checks must pass for both the segment and synchronization object access modes. Since the sets are different (e.g., the synchronization object set contains a write-only mode), it is permissible to request no access for either the segment or synchronization object.

The following is the mapping between the type of access checks performed and the particular kernel gates:

- Observe: `k_ticket`, `k_await_evc`, `k_read_evc`
- Modify: `k_ticket`, `k_advance`

6.2.3 Mentor Information

As part of certain kernel calls, subjects may observe and modify the mentor-info object (which corresponds to the visible non-synchronization fields in the alias table of the mentor). The mentor-info object contains attributes of its associated segments (i.e., the subordinate segments' size and access label). The `k_create_segment` and `k_delete_segment` functions observe and modify the segment mentor's mentor info object. Since the mentor-info object has the access label of the segment's mentor, the Secondary Storage Manager (SSM) calls the NDSM layer to verify that the subject has observe and modify access to the mentor. The `k_makeknown_segment` function observes the mentor info object; the SSM calls the NDSM layer to verify that the subject has observe access to the mentor.

Furthermore, the SSM calls the NDSM layer to verify that for `k_create_segment` and `k_delete_segment`, the subject has observe access to the mentor. SSM also calls the NDSM layer to verify that for `k_mount_volume` the calling subject has observe and modify access to the mount segment.

6.2.4 Volume Attributes

Every volume has a set of volume attributes that are considered storage objects. These attributes include the format type, access label range and key type. Volume attributes are specified at the time a volume is created using the volume devices and thus are assigned a security label defined by the write label of the volume device. This security label is recorded in the Volume Table of Contents (VTOC) at the time the volume is created as the Volume Attributes label.

On a `k_mount_volume` call, the calling subject must have observe access to the volume attributes of the volume being mounted. Similarly, when a subject uses the volume devices to manipulate an existing volume, a check is made to ensure the calling subject has observe access to the volume attributes.

There are other access checks performed on volume attributes. When mounting a volume, the calling subject must have observe and modify access at the volume device's write label and the associated disk device's write label. In addition, the access label range of the mounted volume must be contained within the Internal I/O access label range. Also, if the volume is not read-only or single-label, then the volume's access label

range must be enclosed by the media resource exhaustion range associated with the Logical Hardware Unit (LHU). In order to dismount a volume, a subject must have a read label that dominates the system high access label.

6.3 Device Protection Mechanisms

Every device in the GTNP has an access label range defined for it in the System PROM. These labels can only be changed through the use of the system generation utilities. The range provides both a maximum (read) access label and a minimum (write) access label.

In the evaluated configuration, all GTNP devices are single-level. Although the system conceptually supports the notion of multilevel devices,[2] there are no multilevel device drivers supplied by Gemini in the evaluated configuration.

The GTNP provides six kernel calls for Input/Output (I/O) management, only one of which is used to control access. This call, `k_device_attach`, is used to attach a device for a subject in the calling process. In order to attach a device, the subject access label range must properly compare with the device's range, as follows: For a subject to attach a write-only device, the device's write label must dominate the subject's write label. For a subject to attach a read-only device, the subject's read label must dominate the device's read label. If the device is bidirectional, both sets of constraints must be met.

The ring integrity policy imposes the following additional constraints during device attachment (some of which are implied by the dominance tests):

- In order to attach a device for observe access, the following two tests must be successful:
  - The RB1 of the device's read label must not be greater than the ring of the least privileged subject that can be created by this subject (i.e., the RB1 in this subject's read label). In other words, the device's read label's RB1 specifies the least privileged subject that can be activated by $\mathcal{A}$ and still have observe access to the device.
  - The RB2 value of the device's read label must be numerically greater than or equal to the ring number of the subject (the RB1 in the subject's write class). Thus, the administrator may configure devices which may not be attached above a specified ring.
- In order to attach a device for modify access, the ring of the subject (RB1 of its write class) must not be greater than the ring of the device (RB1 of the device's write class).

No access check is made to detach a device from a subject. Mandatory Access Control for accessing devices is done at the time of attaching the device.

[2] That is, devices that provide trusted access labels on export and interpret labels on import.

6.4 Object Reuse Mechanisms

6.4.1 Segments

Segments exist primarily on secondary storage. Whenever a segment is created, the storage is overwritten with zeroes. The size of the segment is set at creation time and never varies. In this way, the GTNP ensures that no leftover data is ever available when a new segment is defined.

This memory overwrite is completed before access to the segment is granted to a subject.

Whenever a segment is swapped into main memory, the memory area is overwritten with the contents of the segment. The amount of storage available to the subject extends only to the size of the segment. The GTNP thus ensures that no data residue is available to a process.

6.4.2 Eventcounts and Sequencers

Eventcounts and sequencers are created as a result of a create segment call and are initialized to zero. During the life of the segment, the associated eventcount may be incremented from 0 to 65,535, and then back to 0. Subjects are not able to store values in eventcounts and sequencer objects.

6.4.3 Mentor Info

Mentor Info is created as a result of a create segment call, and is initialized at that time. Mentor Info is deleted (zeroed) as a result of a delete segment call.

6.4.4 Volume Attributes

Volume Attributes are created when a volume is formatted. Subjects are not able to directly store values in volume attributes.

6.4.5 Other Storage Variables

Other storage areas include hardware registers, device I/O buffers, and the Numeric Processor eXtension (NPX). Hardware registers are cleared between every context switch. Device I/O buffers are zeroed between every device attachment. In the case of read-only and write-only devices, the kernel ensures through the mandatory access control rules (see section 6.1, page 109) that residual data cannot flow to a subject with a read (or write) label that is dominated by the read or write label of the device. The kernel also ensures that the device state is cleared. I/O buffers are a fixed size determined by the device block size. Since there is always a write before read condition, there is never space at the end of a buffer open for scavenging. Furthermore, the virtual buffer is managed as a set of sector buffers and data movement to/from the virtual buffer must occur in units of a sector buffer.

The Numeric Processor eXtension also has a number of registers, used for storing operands and results, that tasks may read and write. The NPX state is treated similarly to the general purpose processor register state. The kernel maintains the complete register state of each task. The complete register state includes both the general purpose processor register state and the NPX state; this information, when saved, is stored in the task's TSS. The NPX state for each task is initialized to a well-defined state. Each time a context switch occurs, the kernel insures that the general purpose processor registers and the NPX contain the state of the new task. Loading the complete register state of a new task during a context switch effectively erases the values put there by the previous task. Thus, no data is shared between tasks via the NPX.

Section 7

Assurance

One of the key requirements of a Class A1 system is the assurance that the system properly enforces its policy (and accountability requirements, if applicable). This assurance provides the basis for trusting that the protection mechanisms of the system work as advertised. In the GTNP M-component, this assurance is provided by eight factors: system architecture, integrity testing, covert channel analysis, trusted recovery, security testing, formal design specification and verification, configuration management, and trusted distribution. The following sections explore each of these factors in more detail.

7.1 System Architecture

The kernel portion of the GTNP executes at Privilege Level 0 (PL0) of the 80286, i386, and i486. No other software is permitted to execute at PL0. Also, in-memory copies of the kernel's data structures are associated with PL0, and therefore cannot be accessed or tampered with by subjects executing at other privilege levels. The Network Processor System Parent (NPSP) executes at Privilege Level 1 (PL1), and is associated with Ring 1 (R1). Applications are not allowed to have R1 subjects, thus assuring the NPSP its own domain of execution.

The 80286, i386, and i486 segmentation and privilege level features are also used to accomplish process and subject separation. A process executing on the GTNP can have up to eight subjects associated with it. The separation of these eight subjects is accomplished by associating each subject with a different ring. Three of these subjects can be active at a given time, and execute at PL1, PL2, and PL3. Data structures associated with process management are kept in the kernel, and so are protected from corruption. The process management data structures keep track of segments (code, data, stack) that are associated with the process; the hardware privilege level mechanism works with the hardware segmentation mechanism to ensure process and segment isolation.

The 80286, i386, and i486 hardware segmentation feature has been used internal to the Trusted Computing Base (TCB) to support logically distinct storage objects with separate attributes. Kernel data has been separated by mode of access through the use of hardware segmentation. Data that does not have to be accessed for write is kept in read-only segments. Also, kernel modules have access only to the data that is needed to accomplish their function. This restriction is ensured by strict adherence to Gemini development conventions.

The use of the 80286, i386, and i486 privilege level and segmentation features has also allowed Gemini to create a conceptually simple protection mechanism with well-defined semantics. Much of the protection is accomplished in hardware rather than software, where performing the same actions would be much more complex and cumbersome.

In addition to making effective use of the 80286, i386, and i486 protection features, Gemini has also used good software engineering practices to satisfy the Trusted Network Interpretation (TNI) system architecture

requirements. The GTNP has been structured into well-defined largely independent modules. Use of global variables has been restricted by not allowing sharing of databases across modules. This restriction is ensured by strict adherence to Gemini development conventions. Modules are further broken down into one or more components. Each component is a collection of subroutines, and generally contains no more than 100 source statements. The GTNP has been implemented in Pascal and assembly language only. All assembly language that interfaces with Pascal is specifically designed to conform to the MetaWare Professional Pascal calling conventions.

Another use of software engineering in the GTNP design is the implementation of layering. The kernel is composed of 14 layers; the NPSP is composed of 13 layers; and the GEMSOS Kernel Gate Library (KGL) has 2 layers. Each layer provides well-defined services to upper layers and relies upon only the services of lower layers. This relationship guarantees that there will not be any circular dependencies between layers. An important characteristic of the layered design of the GTNP is that a two-level scheduler is implemented in multiple layers that virtualizes the multiple processor architecture and allows for correct management of concurrency. For a complete description of the layers of the kernel, see section 4.2.1, page 61. The layers of the NPSP are described in section 4.4, page 96.

A number of software engineering techniques have been directed towards the requirement for least privilege in the GTNP. The standards for applying these design techniques are detailed in the Gemini Software Development Standards [89]. These include modularity, layering, data hiding, and minimization of global variable usage. Also, the use of segmentation for separation of storage objects with separate attributes contributes to least privilege.

Gemini has devoted system engineering to minimization of the TCB. This minimization is displayed in the way that the design is laid out, and the assignment of functions so as to eliminate duplication of effort. Gemini includes a review for minimization as part of their design process and reviews, as well as in the overall definition of their products, services, and interfaces. This can be seen by the limited functionality provided by the kernel interface, the absence of services such as a filesystem built on top of the segments, and the movement of many restrictions on the abilities to use various calls outside the Mandatory Network TCB (M-NTCB) partition.

The last aspect of the GTNP system architecture evidence is documentation. The Descriptive Top Level Specification (DTLS) is included in the Software Development Specification for the GEMSOS Security Kernel [61] and the Software Development Specification for GEMSOS Network Processor System Parent (NPSP) [25]. It defines the program interface to the GTNP, and it completely and accurately describes the TCB in terms of exceptions, error messages, and effects. The elements of the TCB are completely described in the GTNP functional [61, 25, 15, 14] and product [65, 16, 18, 17] specifications.

7.1.1 Reference Validation Mechanism

The GTNP kernel software implements the reference monitor concept as described in the Trusted Computer System Evaluation Criteria (TCSEC) [136]. The reference validation mechanism mediates all accesses between subjects and objects in the GTNP. The requirements for the reference validation mechanism are that the mechanism be tamperproof, must always be invoked, and must be small enough to be analyzed and tested.

As already discussed, the 80286, i386, and i486 provide a privilege level mechanism so that tasks can execute in separate domains (i.e., PL0 through PL3) and provide a virtual memory mechanism that can associate

specific segments with those domains to allow tasks to protect their data. Given these basic mechanisms, satisfaction of the requirement that the mechanism be tamperproof is achieved in four ways. First, only kernel tasks are able to execute at PL0. This is ensured by passing control only to well-defined locations in kernel code when all interrupts and traps occur. Note also that PL0 is the most privileged level and non-PL0 tasks cannot access PL0 data or code structures. In the evaluated configuration, the NPSP executes at PL1 in R1. Kernel code and data segments are protected with ring brackets restricting access to R0; thus, they cannot be manipulated in the evaluated configuration by any non-PL0 subject (as they all must have rings greater than zero). Second, only PL0 tasks are allowed to access hardware Input/Output (I/O) instructions. This ensures that devices (specifically those containing critical TCB data structures and code) cannot be accessed by non-PL0 entities. Third, all kernel segments are non-conforming and are associated with PL0. Finally, the kernel checks all pointers passed to it to ensure that kernel (PL0) segments are not referenced. This check occurs after the pointer is controlled inside the kernel.

The only way to invoke code that will run at PL0 is by using the 80286, i386, and i486 "gate" mechanism; the "gate" mechanism allows a predefined set of kernel entry points to be set up and only these entry points can be used to invoke the kernel. Furthermore, the objects in the GTNP can be accessed only through the use of data structures that are associated with PL0. Since these data structures can be accessed only by code running at PL0, and the kernel is the only code that executes at PL0, the reference validation mechanism is always invoked when subjects access objects. In other words, subjects must always invoke the kernel in order to gain access to objects, and the kernel has been designed to mediate any such accesses to ensure that the security policy is enforced.

System engineering effort went into design and implementation of a minimal reference validation mechanism. The minimization is exemplified by the relatively primitive set of functions, all being security relevant, provided at the TCB interface and by the absence of extraneous code and functions inside the TCB. The reference validation mechanism is shown to be correctly implemented by providing a correspondence between the mechanism's code and the Formal Top-Level Specification (FTLS) and formal model of the system.

7.1.2 Architectural Decomposition

As shown in figure 7.1, page 120, the GTNP TCB can be decomposed into many logical levels of abstraction.

System: The GTNP, of course, is the highest level of abstraction. There are four CSCIs in the GTNP TCB.

CSCI: The four CSCIs are the kernel, the KGL, the run-time component of the Multilevel Subjects Intersegment Linkage Tool (MIT), and the NPSP. Each CSCI consists of a number of layers.

Layer: The layers are hierarchically organized with no upward dependencies. Each layer consists of one or more modules.

Module: Each module is essentially an object manager and is largely independent from all other modules. Modules are made up of compilation units.

Compilation Unit: Compilation units are the smallest unit that the language (Pascal) will produce as an executable output. Compilation units are made up of CSCs.

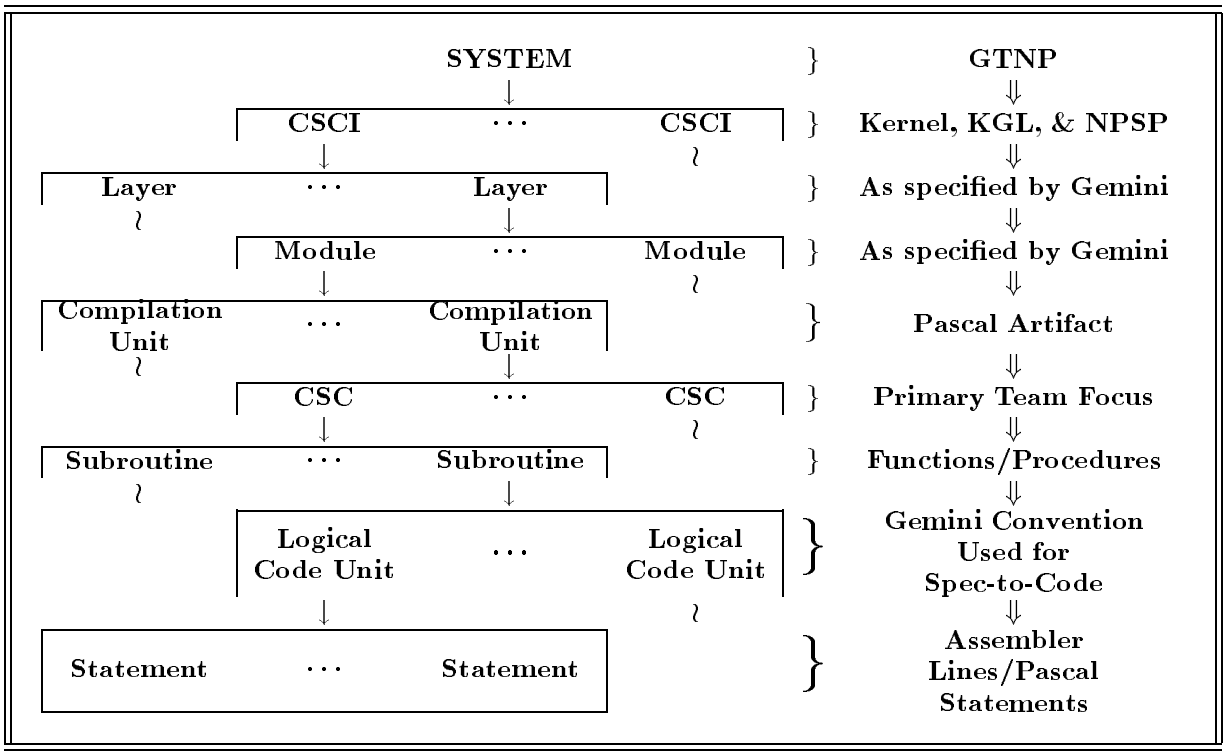


Figure 7.1. Architectural Decomposition

CSC: CSC are typically a Pascal source code file and a number of include files, though they can include more than one source code file. Each CSC provides some related set of functions, and is made up of one or more subroutines.

Subroutine: Subroutines are simply Pascal procedures and functions. Each such subroutine is further broken down into logical code units.

Logical Code Unit: Logical code units are small groups of code statements that perform some relatively simple action. They are identified in the code by comments, and every code statement falls within a logical code unit. These units are used for specification to code mapping purposes.

Statement: Statements are simply single lines of executable code.

The GTNP documents are divided among the CSCIs. Inside the documents, layers, modules, and CSCs are identified and described. Logical Code Units are also identified in documentation, but not for architectural purposes. All other abstractions can be identified in the code and code file structure itself.

7.2 System Integrity

Diagnostics are executed during system boot time [60] to verify correct operation of the hardware and firmware. The tests verify the minimal set of functions necessary for the correct operation of the secure environment. This set pertains mainly to the 80286, i386, and i486 hardware—specifically: privilege levels and segmentation. Additional diagnostic tests are run on each processor within a low priority process that is executed whenever no higher priority processes are available to be run on the processor (either due to blocking or termination).

Gemini has divided its hardware diagnostics into two general classifications:

1. Processor board (Trusted Base) or System board (PC-AT platform) hardware diagnostics
2. System level hardware diagnostics

Hardware diagnostics are performed as feasible and reasonable. The components are split into four categories for testing purposes:

Untestable — Untestable components are those that are not normally used during a system boot. Therefore, they can only be tested while the system is offline. The Non-Maskable Interrupt (NMI) and reset circuitry of the Gemini System Controller (GSC) is an example.

Tested by use — Components that are used during a normal system boot, but are difficult to test without external test equipment fall into this category. Since diagnostics cannot be practically performed, verification of these component's integrity is essentially performed through the system's ability to use the component to boot successfully.

Operator observation — Components that are tested through operator observation are similar to those tested by use. These components can be exercised by a diagnostic and an operator may observe the results for verification. These tests can be performed during a normal boot sequence. Currently, the Light Emitting Diodes (LEDs) of the processor board, the GSC, the Storager, the serial I/O boards, and the PC-AT keyboard have this property.

Automatically tested — Automatically tested components are those components that can be independently tested and reported on during a normal boot sequence. Before the operator console is available, the processor LEDs are used to report processor board component failure, and the GSC LEDs are used to report system level component failure. After the console is available, the diagnostic errors are reported on the operator console.

7.2.1 Processor and System Board Hardware Diagnostics

In Trusted Bases, processor board hardware diagnostics consist primarily of exercising the processor and on-board memory. The LEDs, Programmable Interrupt Timer (PIT), local timeout circuitry, and on-board serial I/O interface are also used. The diagnostic focus for PC-AT platforms is the same as that for processor board hardware. See table 7.1, page 122 for a summary of components that are tested.

	Processor Board	System Board
Automatic Test	• Processor • On-board Memory	• Processor • On-board Memory
Manual Observation	• LEDs	• Keyboard LEDs
Tested By Use	• PIT • On-board serial I/O Interface • Local Time-out Circuitry	• Programmable Interval Timer (PIT) • Keyboard Controller
Untestable	• Reset Circuitry • Central Processing Unit (CPU) and I/O Support Circuitry	• Reset Circuitry • I/O Channel Check and Parity Check Circuitry • CPU and I/O Support Circuitry

Table 7.1. Summary of Diagnostic Tests

7.2.2 System Level Hardware Diagnostics

Any components that are not on a processor board are considered system level components. The following components are included in this category:

- Shared Memory
- Devices
- Gemini System Controller Board for the Trusted Base (GSC)
- Gemini System Controller Board for PC-AT platforms (GSC-AT)

Shared memory is tested after processor board level testing and shared memory mapping functions have been completed. In Trusted Bases, Error Checking and Correcting (ECC) Circuitry diagnostics are performed on the CX memory boards.

Devices are tested during device initialization. Tests are implemented as is feasible and reasonable.

Circuits on the GSC and GSC-AT are tested at various steps during the initialization sequence (see below).

7.2.3 Boot-time Hardware Diagnostics

The following hardware diagnostics are performed during boot sequences:

◇ Test Bootstrap Memory
◇ Test Processor (level 1 tests)
◇ Test GSC Global Timeout Circuits[1]
◇ Test Global Memory
◇ Test Processor (level 2 tests)
◇ Test GSC/GSC-AT Device Acknowledge Circuit
◇ Test Processor LEDs[1]
◇ Enter Protected Mode
◇ Test System PROM
◇ Test CX Board ECC Circuitry
◇ Test Local Memory

The processor diagnostics performed during the boot process include the following:

- Verification of memory accesses for conformance to the access rights and privilege attributes specified in the segment descriptors.
- Verification of control transfers among different tasks and among executable segments within a task for conformance to protection parameters specified in the Task State Segment (TSS) descriptor and gate descriptors.
- Verification of the isolation of each task's address space.
- Verification of correct operation of other protection-relevant registers (e.g., Global Descriptor Table Register (**GDTR**), Interrupt Descriptor Table Register (**IDTR**), Local Descriptor Table Register (**LDTR**), Machine Status Word (**MSW**))

Both valid and invalid operations are tested, and tests are executed in all privilege levels.

7.2.4 DCP Diagnostics

The following Kernel Initialization functions are used to test the DCP device and the GSC key loading functions:

- Boot PROM Recovery Seal Verification
- System ID PROM Recovery Seal verification
- Boot Volume (i.e., Volume table of contents (VTOC)) Recovery Seal Verification.

The complete testing of the DCP device and related hardware is performed by the GTNP offline diagnostics.

7.3 Covert Channel Analysis

A covert channel analysis [66, 67] in the GTNP was performed using a combination of formal and informal techniques. The analysis found 12 covert storage channels; all of which are closable by appropriate system configuration as described in the Trusted Facility Manual. The analysis identified a number of different types of covert timing channels, none of which are auditable. Channel bandwidth estimates were made for all channels detected by the analysis. These estimates were based on the fastest available processor using

[1] These items are Multibus specific.

optimum exploitation assumptions. For storage channels, the estimates also take into account differential times for signalling "1"s and "0"s, using the Shannon [161] approach as described by Millen [135].

Gemini used the Formal Development Methodology (FDM) Shared Resource Matrix (SRM) [6] tool set to satisfy the requirement that formal techniques be used in the covert storage analysis. The methodology developed and used by Gemini [130], was actually a synthesis of SRM [128] and an information flow analysis. Using this method, each transform in the FTLS was used by the SRM tool to generate a list of referenced and modified variables. Gemini then defined security labels for all variables within the specification, and used this information to produce flow theorems using the Interactive Theorem Prover (ITP). Once identified, each theoretical flow violation was examined manually to determine exploitability.

The covert timing channel analysis was conducted informally, and began with an enumeration of all timing-sensitive resources. A timing-sensitive resource refers to a resource that a TCB interface may be required to wait a varying amount of time to access due to an access or state change caused by another process. Both hardware and software resources were enumerated. For each resource identified, Gemini either provided a rationale for why that resources could not be exploited to support covert communication or developed a scenario to illustrate an optimal use of the channel.

The timing channel analysis was divided into three parts. The first focused on those channels that could be exercised solely through available 80286, i386, or i486 processor instructions. This analysis identified two channels, both of which provided an extremely high signalling capacity. The vendor introduced configurable countermeasures that will allow an administrator to reduce or close these channels (at the expense of performance). A third processor timing channel was also identified as a result of penetration testing.

The second part of the timing channel analysis examined those timing sensitive resources whose response times are modifiable and detectable through the kernel gate interface (such as device operations). For every kernel gate call that supported the reading or writing of a timing sensitive resource, Gemini calculated the optimum theoretic (N-character) bandwidth. Gemini used these calculations to provided an overall maximal bandwidth for channels driven through a specific kernel gate call; all channels using a given gate call would be limited (in terms of potential maximum bandwidth) to the fastest speed allowed through that call.[2] As with other timing channels, the channels drivable through the gate interface had the potential for moderate bandwidth. For installations that have a concern about timing channel bandwidths, the GTNP product incorporates a mechanism that allows the execution speed of the kernel gate calls to be reduced, thereby reducing the speed of their associated timing channels.

The third part of the timing channel analysis considered network channels, i.e., those associated with communication between components. In the exploitation scenario for channels in this class, the reader process runs on one processor, a clock on another, and the writer process is on some external network component sending data to the reader's component. The clock process was employed to provide the reader and writer processes a finer granularity clock than the fastest clock provided through GTNP.

In the general network timing channel scenario, the writer utilizes interrupt activity due to I/O to influence the reader's execution time. Such channels using serial I/O ports could support a high signalling capacity, and would be moderated or closed by constraints on the external component supporting the writer. If the writer's component is a GTNP, the bandwidth is limited to the intrinsic rate associated with the write sequential call.

[2] As a result, for example, all channels that took advantage of timing characteristics of random access devices would be limited by the fastest channel possible through the `k_read_random` gate.

7.4 Trusted Recovery

Trusted recovery is concerned with assuring that when the system detects a failure or is forced into resetting, it will not resume operation without ensuring that it proceeds from a secure state. The GTNP takes a simple, uniform approach to recovery.

System failures may be detected during kernel initialization (section 4.2.3, page 95) as a result of hardware integrity testing (see section 7.2, page 121). The system can also be halted with the hardware boot switch. Error handling software always calls the kernel shutdown function, which provides an opportunity to reboot the system.

Corruption of writable portions of the System PROM, containing various keys and user information, may be detected by verifying its cryptoseal. It is possible to restore the correct contents of the System PROM using system generation software (see section 4.5, page 101). The System PROM Bootstrap Utility (SPBU) recovers preinstalled keys, and the Original User Installation (OUI) tool recovers the original offline operator ID and password.

Each of the utilities resides on a bootable floppy disk and is not intended to be installed on the system's fixed disk. They are not intended for use during the normal course of facility operation, but are only required should the contents of the System PROM become corrupted or mismanaged so as to prevent the other system generation programs from executing. The SPBU and OUI utilities recover only enough of the system keys and PROM files to allow the system generation software to be reinstalled, if necessary, and for the GTNP Configuration Tools to execute. Using the latter, any other required keys may be loaded and a previously saved set of correct PROM files may be recovered.

Rebooting is performed by the same functions used during kernel initialization (see section 4.2.3, page 95). As part of the boot sequence for bringing up the TCB, the same diagnostics are performed whether rebooting was due to a successful system shutdown or due to a system discontinuity.

At any time a segment is read in from disk, its label is checked by verifying the cryptoseal. An error would invalidate the segment, and it would have to be restored from backup. Segments can be backed up individually simply by storing copies, but system generation software is also available to restore entire volume images from backup media such as floppy disk or streaming tape.

7.5 GTNP Security Testing

Functional security tests of the GTNP TCB (including the NPSP) are composed of interface and special engineering tests that demonstrate the adherence to the specification of the kernel interface. In general, the tests include the following:

- Interface Tests

  These tests verify the functionality of the interface, and include exception tests, no-exception tests, and device tests.

- Special Engineering Tests

  These tests demonstrate proper performance of key features, functions or subsystems that are not directly mappable to the interface, or that demonstrate other specific scenarios of interest.

Tests are automated, to the extent possible, with test scripts that can be run singly or in groups, depending on the purpose of the test session. Automatic and manual tests are grouped to be run separately, as are those with unique hardware requirements. Test scripts are provided to run groups of the tests on the various hardware and software configurations selected for test.

The protection domain-based structure and the strict adherence to loop-free layering in the GTNP TCB form the basis for confidence in the TCB functional security tests. The kernel is tested and verified, and then the TCB interface, which depends on the correct behavior of the kernel, is tested and verified. Similarly the layers within each component (e.g., kernel) are tested from the bottom up, with dependencies between the layers bound to those layers already tested.

7.5.1 Interface Tests

Exception tests exercise each possible exception of the TCB program interface, and verify: that all exceptions are returned; proper ordering of exceptions that are security relevant; and correct behavior of the kernel at the interface after exceptions. Exceptions that are generated by common internal capabilities of the kernel are tested during the explicit tests of the capability itself. These exception tests are thorough in that the various combinations of conditions that may produce the exception are tested.

No-exception tests exercise the interface with valid parameters. They verify correct return of parameters from successful interface calls and correct behavior of the kernel at the interface after the successful call. The no-exception tests are normally incorporated as setup for the general exception tests and the special engineering tests. Those interface functions that are not tested in this way are identified for specific testing.

In addition to the exception and no-exception tests, the team verified, during the code study and the specification to code mapping, that, while it is possible for a process to specify an arbitrary kernel call number, the gate layer will generate an exception if that call number is for anything other than the 29 kernel gates.

Device tests demonstrate the correct interface to the devices supported by the kernel, including the proper return of device-specific exceptions. For the purposes of the functional security tests, the security checks performed upon device attachment are demonstrated. The interface to the device, once attached, is demonstrated with sample read and write operations. For each I/O device that utilizes buffer areas, or indirect objects, for temporary data storage, a test will be performed to confirm that no residual data remains in the buffer when the device is attached for use by another process. These tests also demonstrate that the kernel interface supports detection of the break key to provide access to a trusted subject (utilities) or support for a secure attention key.

7.5.2 Special Engineering Tests

Special engineering tests can be divided into several groups. This section discusses the various types of special engineering tests.

Common Internal Capability Tests. The common internal capability tests exercise certain common internal capabilities of the kernel to verify their functionality. These include the following: tests of the validity checks performed on memory address pointers that are provided as input at the kernel interface; tests of the checks performed on several aspects of a segment number used as a parameter to the call;

observe/modify access tests that check the subject for appropriate observe or observe/modify access to the object (including synchronization objects), as required by the requested access mode, and ring bracket tests that compare the input ring value with the ring brackets of the segment and determine if the requested mode of access is allowed; tests that verify the dominance relationships between the access class of two subjects by invoking the `k_activate_subject` function a number of times while strategically manipulating components of the access class; and, tests of the checks for several exception conditions of a memory move and the state change of the segment.

Memory Management Tests. The memory management tests demonstrate the proper results of memory compaction performed on local and global memory, the proper results of Local Descriptor Table (LDT) compaction, and the proper relocation of memory segments between global and local processor memory based on process usage.

Covert Channel Testing. The covert channel tests demonstrate the effectiveness of covert channel reduction techniques. Tests that exploit the covert channels are executed both with and without the techniques applied. The tests show that the covert channels are reduced. Mechanisms are provided to close or reduce the timing channels and to close all storage channels.

Concurrency Tests. Concurrency tests demonstrate the multiprogramming, multiprocessing capability and characteristics of the security kernel and the accompanying isolation of processes.

Hardware Protection Mechanism Tests. These tests verify the proper operation of the following features of the x86 hardware to provide TCB domain protection, process address space isolation, and control of point of execution throughout all aspects of processing: privilege levels, segment descriptors, registers, exceptions and traps. More specifically, the following are tested:

- Trap and exception vectoring. It is verified that traps and hardware-defined exceptions are vectored into the GTNP code.
- Privilege level when accessing data. It is verified that when referencing data, the code that is currently executing is not less privileged than the data segment being referenced. It is also verified that the privilege level of the stack is equal to the privilege of the code.
- Access to code and data segments. For code segments, various combinations of the readable, conforming privilege level, and present bits are tried. For data and stack segments, various combinations of the writable, privilege level, and the present bits are tried.
- I/O memory protection. It is verified that a general protection fault occurs if a segment is deleted during a protected I/O operation.
- Access of memory locations. It is verified that the various instructions that access memory locations encounter the appropriate traps if an invalid segment selector or offset is used.
- Selection of current privilege level. It is verified that when a process loads a selector, the new current privilege level is set to the greater of the old current privilege level and the privilege associated with the selector.
- Execution of co-processor instructions.
- Call accessibility. It is verified that the privilege level of the target code segment is less than or equal to the greater of the old current privilege level and the privilege associated with the selector, which in turn must be less than or equal to the privilege of the gate.

Hardware Instruction Set Tests. The tests of the instruction set are incorporated into the tests of the segment descriptors, segment registers, and the privilege levels. The instruction set tests covered the following four goals:

- Verify that privileged instructions cannot be executed by programs executing outside of PL0. Also, verify that IOPL-sensitive instructions executable only by those processes whose privilege levels are at least as privileged as the level specified in the IOPL field of the **(E)FLAGS** register (GTNP does not employ the IOPL permission bit map feature of the i486.)
- Verify that the invalid opcodes (as defined by Intel in each Processor's Programmer's Reference Manual) produce the invalid instruction exception (interrupt 6) when invoked on the processor. NPX invalid opcodes are included in this test set. (These tests were developed by the team during penetration testing, but the tests were later incorporated into the GTNP processor test suite. The team and vendor also took into consideration instructions such as LOADALL, XBTS, IBTS, and CMPXCHG, that appeared in certain steps of the processors)
- Verify that the reserved opcodes (as defined by Intel in each Processor's Programmer's Reference Manual) do not compromise the security state of the CPU. Unlike the invalid opcodes of the processor, which Intel identifies as producing a specific exception, the state change produce by a reserved opcode is explicitly "undefined." During penetration testing, the team ran tests on all reserved opcodes to determine whether these opcodes provided obvious ways for a non-PL0 process to violate the security features of the hardware (e.g., privilege level, memory management, and task-isolation mechanisms). However, comprehensive testing of these opcodes with operand variations and system state variations was not performed.
- Verify that traps and hardware-defined exceptions occur as expected. Tests included the confirmation of general protection faults for those instructions that raise this fault.

7.5.3 NPSP Security Testing

The NPSP Functional Security Tests (FST) test all of the NPSP code that can be executed through the use of the NPSP interface. All code not tested by the NPSP FST are tested by module testing.

The only interface presented by the NPSP, and consequently the only interface tested, is the database interface (see section 4.4.1, page 96). The only input databases of interest are the Physical CPU Table, the Process Definition Table, the Segment Creation Table, the Volume Format Table, and the Volume Table. These databases define the functions the NPSP is to perform to create child processes and establish the initial environment for the child processes. The remaining input databases do not define functions for the NPSP to perform nor do they affect the NPSP's ability to perform its functions. The FST uses the input databases constructed by the GTNP Configuration Tools (see section 4.5.1, page 101). Both valid and invalid classes of data sets for the input databases are used in the FST. The valid data sets are used to test the correctness of the following: the mounting of volumes; the creation of child processes; the creation of segments; and the formatting of volumes. The invalid data sets cause the system to shut down. They are used to test the correct handling of the following: an invalid request to mount a volume; an invalid request to create a child process; an invalid request to create a segment; and an invalid request to format a volume.

7.5.4 System Generation Software Testing

The security testing of the system generation software is performed via a combination of informal testing performed by the developers and formal testing performed by the test organization. The formal testing of the system generation utilities is combined with the formal TCB and kernel initialization testing, as the security mechanisms of the generation utilities are used as part of the testing setup; if they fail to setup

the system correctly, the TCB and kernel initialization tests will fail (as the security mechanisms of the system generation utilities are focused on the generation of the executable version of the TCB on disk and the initialization of TCB data structures).

7.5.5 Test Documentation

The GEMSOS TCB Test Plan [27] (including the test scripts themselves) includes the following for each test: the conditions being tested for; the environment in which the test must run; and the expected response if the test is successfully passed.

The following list comprises the set of test documentation used in the testing effort.

- Chapter 4 of Software Development Specification for GEMSOS Security Kernel (CSCI KER00) [61]
- Chapter 4 of Software Product Specification for GEMSOS Security Kernel (CSCI KER00) [65]

  Presents the test descriptions for the tests described in System Specification for Gemini Trusted Network Processor [97] and Software Development Specification for GEMSOS Security Kernel (CSCI KER00) [61].
- Chapter 4 of Software Product Specification for GEMSOS Multilevel Subjects (CSCI MLS00) [16]

  Presents the test plan for conducting the functional security test and informal internal test of the NPSP CSCI.
- Volume Monitor User Manual [29]
- Standard User Interface Package User Reference Manual [74]
- GT Segment Monitor User Reference Manual [23]
- NPSP Functional Security Test Description [38]
- Test Plan to Test Mapping Document [76]
- Gemini Test Controller User Manual [47]
- GEMSOS Test Application Environment User Reference Manual [51]
- GT Memory Monitor User Reference Manual [22]
- Kernel Call Test Monitor User Manual [24]
- GT Acceptance Test Documentation: Test Description Reference Manual [52]
- Device Test Monitor User Manuals [31]
- GEMSOS Application Generator User Manual [41].
- GEMSOS Model Application Environment [34].

The entire test suite is under configuration management and is repeatable.

7.5.6 Test Environment

GEMSOS Test Application Environment. The GEMSOS Test Application Environment [51] is the test environment used for testing the kernel and TCB program interface. It provides a basic test operating environment, a simple interface for running tests manually or automatically, and a simple UNIX-like file management.

Volume Monitor. The volume monitor [29] is used for manual examination of the disk structures on a volume.

Kernel Table Monitor. The kernel table monitor is used for examination of internal kernel tables. It provides a formatted dump of tables and the ability to write tables to a segment. A special test fixture is required.

Segment Monitor The segment monitor [23] provides a convenient interface to the segment and volume manipulation support provided by the kernel.

Device Driver Monitors. The device driver monitors [31] provide a convenient interface to the device drivers.

Kernel Call Monitor. The kernel call monitor [24] provides an interface to most kernel calls and allows the modification of parameters. It is used in areas of process management, segment management, I/O management, and process synchronization.

Specially Modified Tools. Tools are provided to examine internal kernel structures.

The GEMSOS Test Controller (GTC) Application provides an environment for scheduling and executing tests. The tests can be scheduled and executed in any arrangement across the CPUs in a system. In addition, the tests may be scheduled and executed interactively, by entering commands at the terminal, or in batch mode, by using previously written GTC scripts that automatically schedule and execute tests. The test application environment runs above the gate layer and provides a layer of support between the standard user interface package (UI) command interpreter and the user application. All the monitors also run as applications above the gate layer.

7.5.7 Test System Configuration

Testing was conducted on machines owned by Gemini. The team had access to the system source code that was used to build the system, the test suite code, and the test scripts. A system was generated by Gemini and was made available for use by the team. The system was installed by the team in accordance with product installation manuals. The GTAT was set up by the team in accordance with the test procedures.

The testing platforms consisted of an Multilevel Secure PC/AT Workstation Kit on an IBM 5170, a Model W26-1HP (with two 80286), and a Model W26A-8UP (with four 80286, one i386, and three i486). The system was not in the evaluated configuration for all of testing. Gemini's kernel tests exercise capabilities that are outside the evaluated configuration (e.g., creation of a multi-level subject), but that are useful to trusted application developers. There are no differences in the source code that is run on the IBM PC/AT and that run on the Zenith PC/AT. Although the team did not test the Zenith platform, the team felt that through the team's testing of the IBM PC/AT and the team's examination of the evidence from Gemini's complete testing, the PC/AT security features were sufficiently tested.

7.6 Design Specification and Verification

The GTNP model [40] and Formal Top-Level Specification (FTLS) [62] are written in Ina Jo, the specification language of FDM (The Unisys Formal Development Methodology) [149] version 12.5, an endorsed tool.[3] The Descriptive Top-Level Specification (DTLS) comprises the *Software Development Specification for the GEMSOS Security Kernel* [61] and the *Software Development Specification for GEMSOS Network Processor System Parent* [25]. A mapping of the FTLS to the source code is provided in the form of annotations to the FTLS referring to numbers in the TCB source code listing [73].

The specification and verification approach is essentially in accordance with the GEMSOS Verification Plan presented to the NCSC Verification Working Group (VWG) in 1987. There is no specific GTNP verification plan; Gemini is simply using those portions of the original plan applicable to the GTNP Mandatory Access Control (MAC) component. The vendor has provided a listing of the portions of the verification plan that are not relevant to the GTNP. Although the original plan was for a TCSEC evaluation, the actual verification approach is consistent with the TNI interpretation of the relevant requirements.

The portion of the Verification Plan about the FTLS-to-model correspondence describes the use of the Ina Jo transform mapping theorems, but does not mention the additional mapping evidence needed; this issue is explained in section 7.6.2, page 132. However, Gemini has supplied the needed information.

7.6.1 The Model

For a network component, there are potentially two levels of modeling that are applicable. An internal model characterizes the interface between untrusted processes within the component and the component Network Trusted Computing Base (NTCB) partition. This kind of model is a conventional access control model. An external model is sometimes used for communication-subnet components. It characterizes the interface between the component and other components, and typically employs protocol concepts and terminology such as connections, datagrams, etc. One purpose of an external model is to specify abstractly the working of non-kernel trusted software that implements protocols needed for correct labeling of imported and exported information. It is appropriate to have an external model when the network security policy is based on concepts that can be presented and understood better at a protocol-oriented level of abstraction.

The GTNP provides only single-level interfaces to other network components. For its correct operation, it does not support or require any protocols between itself and other components. It does not exhibit any non-trivial network behavior for which an external model would be useful. The only model provided for the GTNP is an internal model.

The model is the first level of the Ina Jo specification. It consists of ten transforms representing rules of operation, a criterion expressing a form of the $\star$-property, and some supporting material such as type declarations. Since the GTNP is an M-component, only a MAC policy is represented.

The declared types include subject, object, label, and access. There are four modes of access: read, execute, append, and write. As in the Bell-La Padula model [2], execute is treated abstractly as a kind of read access, append means write-only, and write combines read and modify access. Rings are not modeled.

[3] Verification systems on the National Computer Security Center (NCSC) Endorsed Tool List have been determined to be production-quality systems possessing the functionality to provide significant support for the use of formal methods in design specification and verification, and the integrity to provide confidence in the soundness of the verification evidence they produce.

A label is a structure with two fields: a secrecy part and an integrity part, whose further structure is not declared. Labels can be compared with a dominates function, assumed to be a partial ordering.

Each object has a single label, which is a constant. A subject has a `read_class` and a `write_class`. An invariant specifies that the `read_class` always dominates the `write_class`. These two labels will be equal for an untrusted subject. In the evaluated configuration of the GTNP, the Network Processor System Parent (NPSP) is the only multilevel subject. However, the model allows for other multilevel (trusted) subjects also.

The criterion states that if the current access set (a variable `cur_access`) indicates either form of read access, the `read_class` of the subject dominates the object label. If either form of modify access is indicated, the object label dominates the subject `write_class`.

The rules of operation are `activate_subject`, `child_delete`, `create_process`, `delete_segment`, `format_create`, `makeknown`, `modify_and_term`, `mount_return`, `observe_and_term`, `terminate_segment`, `copy`, `no_op`, `modify`, `va_makeknown_rd`, `volume_makeknown`, `volume_terminate`, `observe_and_modify`, `observe` and `replace`. `Copy`, `no_op`, `modify`, `observe_and_modify`, `observe` and `replace` are trivial transforms whose purpose is to serve as a mapping target for kernel calls and instructions that have no effect on the access state.

Consistency of the model with its axioms, in the sense of the TNI [137] is proved in the usual Ina Jo fashion by using the specification processor and Interactive Theorem Prover (ITP) to establish that the criterion, invariants, and constraints are preserved by every transform. The initial state is restricted only by the assumption that it satisfies the criterion and invariant.

7.6.2 The FTLS and Mapping

The kernel FTLS is the second level of the Ina Jo specification and has sufficient commentary and white space so that it is easy to read. Its level of detail is such that it can show effects concerning devices, volumes, the segment/mentor structure, eventcounts, tickets, and signals. Ring brackets are added to labels, and the secrecy and integrity components of labels are broken down into a level and category set. The transforms in the FTLS correspond closely to the kernel calls available at the kernel interface to untrusted processes.

The kernel FTLS listing includes Ina Jo mappings of types, constants, variables, and transforms. Since the model is the top "level" in the Ina Jo specification and the FTLS is the next level under the model, Ina Jo mappings are of the form:

$$\textit{model-entity} == \textit{FTLS-expression}.$$

In particular, each model transform is mapped to an expression calling an FTLS transform. Standard Ina Jo mapping theorems are proved using the specification processor and ITP.

The mapping of model to FTLS transforms in Ina Jo is not, by itself, sufficient to prove the necessary security correspondence between the two. The Ina Jo mapping theorem says that if the FTLS-expression is mapped to a model transform, the state change specified by the FTLS-expression is mapped to the state change specified by the model transform. However, the FTLS transform is called in the mapping expression with particular arguments derived from the model transform call. Thus, this mapping theorem shows only that the FTLS transform causes a secure transition (in the model sense) when it is called with those particular arguments. The problem is that, since the FTLS describes the TCB interface, an FTLS transform can be

called by untrusted software with any arguments at all, including, perhaps, arguments that are not mappable to arguments of any model transform call.

An adequate mapping between the model and FTLS is therefore incomplete without additional evidence that all possible arguments to an FTLS transform result in a state change that is mappable to a legal model state change. Gemini has chosen to provide this additional evidence informally, by showing that every set of arguments to an FTLS transform call is consistent with some set of model transform arguments via the mapping. This is possible because the mapping of arguments is not unique; given a set of model transform arguments, there are many possible sets of FTLS transform arguments that cause a corresponding state change. This approach has been checked, and it has been determined to be sound.

There is also an FTLS for the NPSP, the trusted subject that starts up the application processes in the GTNP (see section 4.4, page 96). The NPSP has no programming interface, so there are no exceptions or error messages to specify. However, the FTLS shows the effects of the NPSP on data structures that occur in the Kernel FTLS, and how they depend on the tables that drive the NPSP.

The NPSP FTLS has a single transform, `initialize_gtnp`. Input databases such as the segment creation table and volume table are represented by constant structures, and output databases such as the mounted volume table are represented as variables. Since the NPSP is not callable by untrusted processes, it need not be mapped to any part of an access control model. It does have a criterion stating that each active child process created by the NPSP is based on an entry in the process definition table. There is an initial condition stating that all child processes are initially inactive.

7.6.3 Code Correspondence

Gemini has provided a spec-to-code correspondence consisting of three parts: the annotated FTLS showing the corresponding code lines, an accounting of non-correlated code files, and an accounting of non-correlated code within correlated files.

Annotations in the FTLS refer to logical units of code, which are typically a few lines long, perform a conceptually unitary action, and bear a single comment as a group. Each logical unit is given a logical unit ID, which includes the file ID, subroutine ID, and a numerical string. These ID's appear in the code files. Also, in some cases, specific lines within logical units are referenced by line number. An FTLS annotation applies to the following one or more lines of specification. Transform effect statements are mapped to executable code, and constant and variable declarations in the front matter are mapped to data structure declarations. Some parts of the FTLS, such as no-change clauses in the effects, and invariants in the front matter, are not and need not be mapped.

Each file of code in the GTNP is listed in a C-spec appendix as correlated or not. Those that are not are marked with a categorical indication of why not. Uncorrelated logical units within correlated files are also listed. The categories of non-correlation are: language details, debugging aids, implementation details, initialization code, unused code, and device driver code.

7.7 Configuration Management and Ratings Maintenance

The process of *configuration management* (CM) involves identifying all hardware, documentation, software, fixtures, and tools used during the development and testing processes, and controlling and tracking all

changes to these items. Gemini has implemented a Configuration Management Plan (CM-Plan) [79] that establishes product baselines and tracks all configuration items.

The process of *ratings maintenance* (RM) extends the CM process to focus specific analysis on security-relevant changes to the configuration items, in such a way that the assurance in the trust of the evaluated product can be maintained. At the B2 rating and above, the RM process involves Vendor Security Analysts (VSAs) and, potentially, members of a Security Analysis Team (SA-Team) selected by the NSA. Gemini has implemented a Ratings Maintenance Plan (RM-Plan) [98] that establishes the processes and procedures that Gemini will follow to comply with the B2 and above Ratings Maintenance Program (RAMP) requirements [141], as well as defining the roles and responsibilities of individuals within Gemini.

The principal divisions of Gemini that participate in the CM and RM processes are Corporate Management, Product Development, Manufacturing, Product Quality Assurance, Customer Support, and Configuration Management.

7.7.1 Configuration Items

Gemini's process of CM/RM is centered on the notion of the *configuration item*. A configuration item (CI) is an entity (document, software, or hardware) that is managed by Gemini's CM process; it consists of a hardware or software component, and the documentation that is related to that component. The term CI is a generic designation; the following specific designations are also used:

- A configuration item that consists of software and related documentation is designated a *Computer Software Configuration Item* (CSCI). CSCIs are defined such that each has a meaningful functional interface specification; thus each CSCI is expected to have a set of well-defined properties and a well-defined interface. The software in a CSCI is, in turn, broken down into *Computer Software Components* (CSCs). This process was illustrated in figure 7.1, page 120.

- A configuration item that consists of hardware and related documentation is designated a *Hardware Configuration Item* (HWCI). A HWCI represents a distinct GTNP platform and constituent boards and sub-assemblies.

  HWCIs may contain other HWCIs, and are organized into a tree-like hierarchy. For example, the GTNP consists of two HWCIs: one for the Trusted Base models (GSB00) and one for the PC/AT Models (PCB00). The Trusted Base HWCI (GSB00) contains HWCIs for the two models in the evaluated configuration: the 12-slot system base (GSB01) and the 26-slot system base (GSB02). Each of these HWCIs contains HWCIs for the types of components that may be in that system; for example, the 12-slot system base contains HWCIs for the Central Processor Units (CPU00), Disk/Tape Controllers (DTC00), System Enclosures (ENC00), Floppy Diskette Drives (FDD00), Gemini System Controllers (GSC00), Hard Disk Drives (HDD00), Half-Inch Tapes (HIT00), I/O Control Devices (IOC00), Power Supplies (PWR00), Random Access Memory (RAM00), and Streaming Tape Drivers (STD00). Each of these component-type HWCIs contains HWCIs for each individual model or item. For example, the Gemini System Controller HWCI (GSC00) contains the HWCIs for the Multibus Gemini System Controller (GSC01) and the PC/AT Gemini System Controller (GSC0A). Similarly, the Central Processing Unit HWCI (CPU00) contains one HWCI for each class of CPU: Standard (CPU01), HP (CPU03), SP (CPU04), and UP (CPU05). A complete tree of these relationships is maintained by the Gemini Configuration Management office.

- A configuration item that consists solely of documentation is designated a *Computer Documentation Configuration Item* (CDCI). CDCIs tend to have a higher granularity, and are applicable to multiple CSCIs or HWCIs.

The chosen granularity for CIs ensures that all aspects of a particular component are examined when a CI is modified; for example, opening the kernel CSCI to make a software modification also opens the design documentation contained in that CI. Hardware and software CIs contain multiple smaller subunits that are configuration controlled as part of the control of the CI.

The following categories of material are considered to be CIs: TCB software and related design and test documentation; Evaluation documentation; User documentation (i.e., SFUG and TFM); software used in support of testing (e.g., test scaffolding) and supporting documentation; system generation software and supporting documentation; compilers and related tools; test scripts and documentation; and TCB hardware.

7.7.2 Baselines

Critical to the Gemini CM process is the notion of *baselines*. A baseline is a frozen version of a CI that has been past quality assurance and has been checked into the configuration management library. The process for establishing baselines (see figure 7.2, page 136) is the same for Gemini hardware and software and third party hardware. This process is used whenever a new product is developed.

A System Specification ("A-spec"), which is a system-level list of requirements for development work, is developed by an engineering team. Once completed, it is reviewed by the Configuration Control Board (CCB), whose members include: the Director of Quality Assurance (QA), Vice President of Engineering, lead engineers from each design group, and the configuration item managers in a requirements review. After the A-spec passes this review, it is given to the Vice President of Engineering and Director of Quality Assurance for final approval, and is entered into the library.

Next, a Hardware/Software Development Specification ("B-spec") is developed for each CI identified in the A-spec (it is possible for several CIs to be included in an A-spec). B-specs establish the requirements for the function, interface, performance, design, test and qualification of the configuration item, as well as providing the formal and descriptive top level specifications (as applicable). The B-specs are reviewed in a specification review by the CCB. During this review the CCB examines the design of the B-spec and compares it with the requirements of the A-spec. The CCB also conducts an audit of all CM documentation. If the CCB feels the CM policies are not being followed, a more comprehensive CM review is conducted by QA to determine all deficiencies. If deficiencies in CM procedures are discovered, the B-spec is not entered into the baseline and must be re-reviewed by the CCB. Upon successful review by the CCB and approval by management, the B-spec is placed into the allocated baseline.

Next, Hardware/Software Product Specifications ("C-specs") that include source code designs, and schematics are developed to meet the requirements of the B-spec and undergo a series of Preliminary Design Reviews (PDRs) and Critical Design Reviews (CDRs) by the CCB. After the design has been fully established, development work commences on the hardware or code CI. A Final Design Review (FDR) is conducted by the CCB after the development work is completed. The FDR includes verification and validation review, functional audit and physical audit. The new CI's product baseline is established from the previous product baseline plus changes. The CI product baseline is then tested and submitted to QA for verification and regeneration testing. Upon successful completion of this testing, QA enters the new code/hardware and documentation into the CI product baseline.

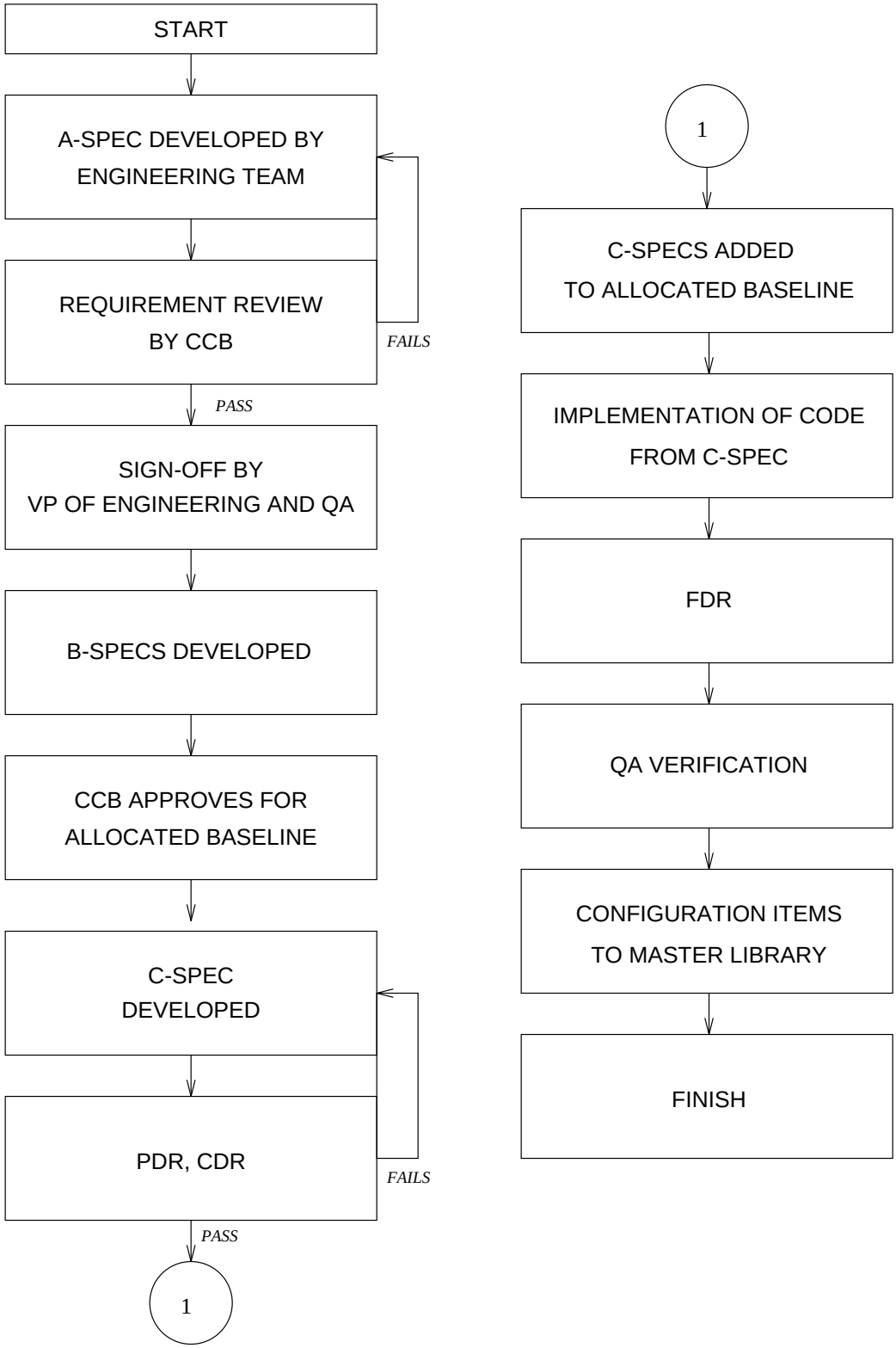


Figure 7.2. Configuration Management (CM) Baseline Process

7.7.3 Change Control

Once a CI has been initially baselined, it is subject to the CM control process described in the Gemini CM and RM plans. This process is initiated by Gemini employees through Engineering Change Proposals (ECPs), or by anyone (including customers) by sending Gemini a System Discrepancy Report (SDR) that may be developed into ECPs. There are actually two change control processes: one for ECPs, and one for SDRs that did not result in an ECP.

7.7.3.1 Engineering Change Proposals

Gemini uses Department of Defense Form 1692 as the basis for its Engineering Change Proposal. ECPs are used to request changes to an existing product, implement new features in software or hardware, remove hardware or software capabilities, and to undo past changes (as might be done if a change had a negative performance impact not discovered until the system was fielded). Once an ECP form is prepared, the configuration change control process begins (see figure 7.3, page 138). The ECP is submitted to the Director of Quality Assurance, who determines if the ECP affects a product in RAMP. If it does, the VSA is requested to prepare a preliminary security analysis.

Preliminary security analysis involves an examination of the ECP to determine the security relevance of the proposed change on the overall product. During this analysis, the proposed change is analyzed against the current state of the product, which is the most recent release as modified by any ECPs (both implemented and approved but not yet implemented). Once this analysis is completed, the VSA prepares a statement containing the analysis results. This statement includes the VSA's opinion on whether to accept the ECP, an indication of whether the changes affect the product's TCB (based on the configuration items affected), an indication of whether the change is security-relevant and the rationale for that determination, and the nature of the security relevance. During this analysis the VSA also notes any possible covert channel effects resulting from the change, as well as any penetration testing hypothesises that might be applicable.

Once the preliminary analysis is completed, a Configuration Control Board (CCB) is convened to evaluate the ECP. The CCB consists of the Director of QA, the Vice-President of Engineering, the Lead Engineers, and the configuration item managers. The VSA is an ex-officio member of the CCB. Other guests invited to a CCB meeting include engineers whose products may be affected by the change. The CCB meeting reviews the change, and identifies all configuration items and related documentation affected by the change. They also consider the technical feasibility of the change, the cost effectiveness, the scheduling effects, the contract effects, and the effects on the overall trust and assurance of the product. As a result of the CCB meeting, the CCB may either request that the ECP be modified to reflect information discussed in the CCB meeting, or present a recommendation to Gemini management.

Once the CCB has provided its recommendation to Gemini management, the decision is made whether or not to approve a proposed change. If the change is approved, resources are allocated and implementation begins. If an ECP that affects a product in RAMP is approved and the VSA is convinced that the ECP adversely affects the trust or assurance of the product, the VSA is authorized by the CM and RM Plans to appeal the approval to the RAMP Responsible Corporate Officer (RCO).

During the implementation process for an ECP, the VSA works with the engineers incorporating the change. They review the design of the proposed change and participate in all design reviews. They continually review the implementation with respect to system architecture considerations. They also verify that sufficient tests are developed, documented, and executed. This includes VSA participation in the Test Readiness Reviews

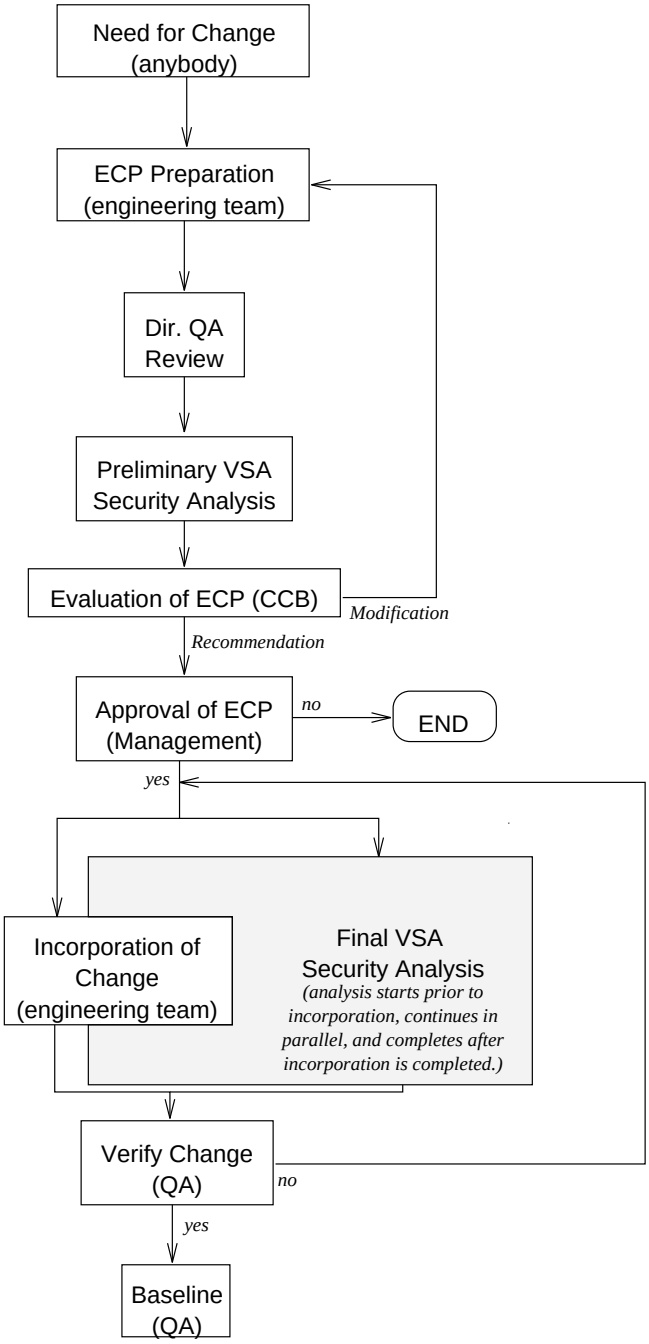


Figure 7.3. Change Control

and Functional Configuration Audits (integration). The VSA also verifies that all documentation needed to show compliance with the criteria, as well as user documentation, is updated properly as part of the implementation of the ECP. The RM plan directs the VSA to inform the RCO of any process failures.

Once implementation of an ECP is complete, the VSA completes a final security analysis of the implementation of the changes, the updates to the testing materials, and the updates to the affected documentation. This analysis, and all changed items or specifications (or related documentation) are submitted to QA (as the agent of the CCB) for verification and baselining. QA verifies that the change was properly implemented, and that only changes covered by the scope of the ECP were made. This analysis is done through the use of both manual and automated means. If a change is implemented incorrectly, it is returned to the engineering team for reimplementation. If the change was made correctly, it is submitted to the QA librarian for baselining.

The last step in the process is baselining. During this step, the QA librarian incorporates the changed items or specifications, assigns a new version or revision number, and archives them into the official baseline.

7.7.3.2 System Discrepancy Reports

The System Discrepancy Report is the means whereby Gemini customers and employees can identify problems and request fixes. If the SDR affects hardware, software, or a specification document, analysis and correction of the problem is handled through the ECP process. If the change affects a non-specification document (for example, the Configuration Management plan), a modified change control process is used. This process does not involve the configuration control board; rather, the change must be approved by the lead engineer for the document and the VSA. Furthermore, the change must be approved by the Gemini QA department before the document is incorporated back into the QA library.

7.7.3.3 Emergency Fixes

Gemini's CM procedures during emergency fixes are the same as the normal CM process, except that the CM procedures are greatly accelerated.

7.7.4 RAMP Cycles

Gemini's RM process defines both RAMP cycles, release cycles, and ECP cycles:

- An *ECP cycle* is the lowest granularity, and reflects the process of incorporating the change from a single ECP into a product.
- A *release cycle* reflects a product's release to customers, and incorporates the result of one or more ECP cycles.
- A *RAMP cycle* reflects the addition of a new version of an evaluated product to the Evaluated Products List (EPL). Each RAMP cycle begins with a Future Change Review Board (FCRB), and concludes with the publishing of the EPL entry for the RAMP-ed product.

7.7.4.1 Future Change Review Board (FCRB)

Before a RAMP cycle begins, Gemini schedules a meeting with the National Security Agency (NSA) FCRB to present the proposed changes to be incorporated in that RAMP cycle.

After a FCRB, if Gemini determines that a change not presented needs to be made, a new FCRB will be scheduled. However, the RM plan indicates the Gemini does not plan to schedule a new FCRB for minor deficiency corrections[4] determined to be necessary after the FCRB, but during a RAMP cycle. These unpresented corrections will be included in Ratings Maintenance Report and the presentation to the RAMP Technical Review Board (TRB).

7.7.4.2 RAMP Cycle VSA Security Analysis

Once per RAMP Cycle, the VSA, in conjunction with the Security Analysis Team (SA-Team) (as appropriate), performs a number of product-level analysis and evaluation activities. These activities include:

- Ensuring that the proper FTLS updates are made.
- Ensuring that the FTLS proofs are updated and reproved.
- Ensuring that the specification-to-code mapping is updated.
- Ensuring that the Covert Channel Analysis for the product remains complete, thorough, and accurate.
- Ensuring that proper testing of the product, both functional and penetration, is performed.
- Ensuring the model is updated and is consistent with its axioms.

7.7.4.3 RAMP Documentation

The VSA also collects during the RAMP cycle the RAMP evidence that is used to prepare the Ratings Maintenance Report (RMR). This evidence includes all ECP forms, the records of the development process for changes, the VSA analysis reports, and the ECP log from Quality Assurance.

This evidence is used by the VSA to prepare the RMR. The RMR includes a summary of the ECPs incorporated into the release (including identification of the security-relevant aspects and analysis thereof), a description of how these changes meet the TNI requirements, a summary of any process failures and how the process has been subsequently corrected, the results of the VSA-conducted RAMP audit, a summary of the covert channel report, a summary of the update to the architecture analysis report, a summary of the results of functional and penetration testing, and a summary of any specification changes and the specification-to-code mapping.

As part of the conclusion of the RAMP cycle, the VSA will also work with the NSA-assigned TPOCs to update the Final Evaluation Report for submission to the RAMP TRB.

[4] Whether a particular change falls into the category of a "minor deficiency correction" is determined jointly by the VSA and the Technical Point of Contact(s) (TPOC(s)).

7.7.4.4 Audits

The Gemini RM and CM process includes support for both RM and CM audits. The audits are conducted on a periodic basis by the VSA, with a minimum frequency of once per RAMP cycle. During an audit, the goal is to sample approximately 10% of the approved security-relevant ECPs. These ECPs may be in any state, and are selected to represent "typical" changes for the present RAMP cycle. The goal of the audit is to focus on whether the RM and CM procedures are being followed, and the results are included in the VSA's Quarterly Status Reports and in the RMR.

7.7.5 Third Party Hardware Acceptance Testing

Third-party hardware undergoes an acceptance procedure when received by Gemini that includes a physical inspection by an experienced hardware technician. The physical inspection is followed by an electrical inspection. The physical inspection and electrical inspection are component-specific; the steps for incoming inspection of each component are detailed in procedures maintained by the manufacturing department and include physically inspecting the incoming component for damage and missing parts, as well as functional testing, and burn-in. At the time these inspections are performed, a Component History Form is begun, which becomes part of the Configuration Management record for the component. The Component History Form records the "flow" of the component from the Incoming Physical Inspection through the Incoming Electrical Inspection, and through its Repair and Refurbishment History (including all Failure Reports for the component and their disposition).

When a system is assembled, prior to shipment, it is inspected by the QA Department to confirm that all the components that it contains are the ones that it should contain (i.e., correct part numbers, correct serial numbers), as specified on the System Configuration Form associated with the system being built. It is then confirmed that the system, as a whole, passes specific QA tests for systems of that configuration.

Interface design changes to third-party hardware are reviewed, especially for security relevance. Special engineering tests have been designed to exercise the hardware protection mechanisms to ensure correct functioning of hardware security mechanisms. These tests are part of the functional test suite.

7.7.6 Maintenance of CM/RM Process

Both the RM and CM plans are maintained under configuration management; any necessary changes must first go through the normal ECP/SDR cycles. Once all the changes are made to one of the plans, but before the plan is baselined, the updated document is submitted to the TPOC. If the TPOC approves the change, a cover letter is prepared and the updated RAMP process documents are submitted to the NSA for approval. If the change is not approved by either the TPOC or the NSA, Gemini will either modify the document to reflect comments made, or withdraw the change. Once approval has been received from the NSA, the document will be baselined.

If Gemini or the NSA identifies that the RM or CM process has, for some reason, failed during a cycle, a recovery process will be initiated. This process will identify the release(s) involved, determine where and how the process failed (i.e., identify the responsible person or procedures), and take appropriate actions to ensure that the type of failure will not occur again. Detailed security analysis of the affected modules and releases will be performed, and the missing or faulty evidence will be reestablished.

7.7.7 Configuration and Accounting Methodology and Tools

The Quality Assurance Department maintains a database that is used to determine the status of all configuration items. Official logs of SDR and ECPs containing information about status and related configuration item numbers are also maintained by Quality Assurance. Periodic reports of the status of all ECPs are prepared prior to CCB meetings. Audits of configuration management information are conducted by the CCB at every major review.

Gemini uses software tools on the UNIX operating systems to aid in its implementation of configuration management policies. Gemini also uses manual techniques to preserve software baselines.

A copy of each baselined configuration item and supporting documentation is kept in the CM library, and one copy is kept in a secure off-site location.

7.8 Trusted Distribution

Trusted distribution of the GTNP TCB is based on cryptographic seals and data encryption. Kernel software is distributed on bootable volumes. All segments on these volumes are encrypted and sealed. All formatting information on the volume, such as the Volume Table of Contents (VTOC) and alias tables, are sealed but not encrypted. With these cryptographic techniques, it becomes unnecessary to protect bootable volumes from alteration during shipping; only the keys used to verify the seals need be shipped securely.

Hardware distribution is implemented securely by creating an authenticator for each component. These hardware components include all board-level components in a system and all Gemini-produced firmware on those boards. An authenticator is a cryptographic seal. For a hardware component, the seal is based on a unique identifier such as the serial number; for a Programmable Read-Only Memory (PROM), the seal is computed on its contents.

Board-level protection by sealing of serial numbers is effective against shipping dock blunders and other relatively unsophisticated threats, but does not prevent knowledgeable malicious threats such as component-level substitution. Hence supplemental procedural measures must be agreed upon by the vendor and the receiving site in cases where those concerns exist. The cryptographic protection of GTNP firmware in PROMs is effective by itself when keys are sent securely, though it is redundant in situations where procedural hardware delivery assurances are employed. Partial assurance that hardware substitutions have not been made is also obtained when the system is booted due to the hardware diagnostics as described in section 7.2, page 121.

Encryption and the production of cryptographic seals are accomplished using a Data Ciphering Processor (DCP) on the Gemini System Controller (GSC) board, which implements the Data Encryption Standard (DES) [140]. A cryptographic seal is a checksum produced by enciphering data in the Cipher Block Chaining (CBC) mode, retaining only the last 64-bit block of ciphertext as the seal.

7.8.1 Kernel Boot Volume Authentication

A volume key is used to protect the information on a boot volume. The volume key is used to encrypt or seal data on the volume.

The volume key is sent to the user separately, in an encrypted form called a composite key. When the composite key is presented at the kernel interface for loading into the system, the kernel extracts the volume key from the composite key. The volume key is never exposed outside the kernel.

Decryption and cryptographic seal checking on the boot volume are performed during kernel loading. Kernel loading begins with a check on the VTOC seal by firmware in the boot PROM. The seals must be valid for all data read in, in order to boot the volume.

The cryptographic authentication procedures do not protect against the replacement of valid kernel segments with old or different kernel segments through use of the SYSGEN at a site. Gemini provides an additional mechanism to ensure that only valid combinations of kernel segments are able to be booted. Each code segment contains a unique ID. An R0 segment called the Kernel Loader Table (KLT) contains the list of valid kernel segments for a given distribution. During initialization, the ID of each code segment loaded is checked against the ID specified in the KLT.

7.8.2 Hardware Distribution Support

For the initial delivery of a system, all the authenticators for hardware components and firmware in a particular system are created with a single key, the Hardware Distribution Key (HDK). They are listed in a Hardware Distribution Table (HDT), which is shipped together with the HDK. The HDK and the HDT are shipped in a secure manner agreed upon by the vendor and the receiving site, separately from the hardware. Using a trusted utility called the Hardware Distribution Utility, the shipped authenticators are compared at the user site with authenticators produced on-site using the delivered components and the HDK.

If the HDT and HDK were shipped with the system, it would be possible for a malicious agent with access to the system during transit to substitute a different component, use the system (with its unique keys) to generate a correct authenticator for it, and place the authenticator in the HDT.

For a hardware update involving only components that are not repositorites for keys upon which the key management system depends, the process is slightly different. Authenticators for the components are generated using a key called the Hardware Update Key (HUK), which is sent with the authenticators and the component to the site. At the site, the Hardware Distribution Utility is used to check the authenticators. The assurance in this approach is based on the fact that a site has previously received the system containing the keys that are the basis for subsequent secure distribution of composite keys. Once the system is received and checked, all future composite key distributions may be made without concern about interception, because the already authenticated system is required to be present in order to extract the key from each composite key distribution.

This page intentionally left blank

Section 8

Evaluation as an A1 M-Component

This chapter presents an evaluation of the Gemini Trusted Network Processor against the Trusted Computer System Evaluation Criteria (TCSEC) [136], as interpreted by the Trusted Network Interpretation of the TCSEC (TNI) [137] for an M-Component. Each section begins with a statement of the TCSEC requirement, followed by the general TNI interpretation of that requirement. This is then followed, if necessary, by any additional interpretations or clarifications that are specific to an M-Component. Following this is the evaluation team's summarization of the applicable features, and the team's conclusions with respect to the interpreted requirement.

8.1 Object Reuse

Requirement

All authorizations to the information contained within a storage object shall be revoked prior to initial assignment, allocation or reallocation to a subject from the TCB's pool of unused storage objects. No information, including encrypted representations of information, produced by a prior subject's actions is to be available to any subject that obtains access to an object that has been released back to the system.

Interpretation

The NTCB shall ensure that any storage objects that it controls (e.g., message buffers under the control of a NTCB partition in a component) contain no information for which a subject in that component is not authorized before granting access. This requirement must be enforced by each of the NTCB partitions.

Applicable Features

The mechanisms for object reuse are explained in section 6.4, page 115. Storage media on disk is overwritten just before being allocated to a subject, while main memory is overwritten with the contents of a segment brought in from secondary storage. Hardware registers are cleared between every context switch. I/O buffers are treated the same as segments. The math coprocessor is cleansed when necessary.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Object Reuse requirement.

8.2 Labels

Requirement

Sensitivity labels associated with each ADP system resource (e.g., subject, storage object, ROM) that is directly or indirectly accessible by subjects external to the TCB shall be maintained by the TCB. These labels shall be used as the basis for mandatory access control decisions. In order to import non-labeled data, the TCB shall request and receive from an authorized user the security level of the data, and all such actions shall be auditable by the TCB.

Interpretation

Non-labeled data imported under the control of the NTCB partition will be assigned a label constrained by the device labels of the single-level device used to import it. Labels may include secrecy and integrity-components in accordance with the overall network security policy described by the network sponsor. Whenever the term "label" is used throughout this interpretation, it is understood to include both components as applicable. Similarly, the terms "single-level" and "multilevel" are understood to be based on both the secrecy and integrity components of the policy. The mandatory integrity policy will typically have requirements, such as the probability of undetected message stream modification, that will be reflected in the label for the data so protected. For example, when data is imported its integrity label may be assigned based on mechanisms, such as cryptography, used to provide the assurance required by the policy. The NTCB shall assure that such mechanism are protected from tampering and are always invoked when they are the basis for a label.

If the security policy includes an integrity policy, all activities that result in message-stream modification during transmission are regarded as unauthorized accesses in violation of the integrity policy. The NTCB shall have an automated capability for testing, detecting, and reporting those errors/corruptions that exceed specified network integrity policy requirements. Message-stream modification (MSM) countermeasures shall be identified. A technology of adequate strength shall be selected to resist MSM. If encryption methodologies are employed, they shall be approved by the National Security Agency.

All objects must be labeled within each component of the network that is trusted to maintain separation of multiple levels of information. The label associated with any objects associated with single-level components will be identical to the level of that component. Objects used to store network control information, and other network structures, such as routing tables, must be labeled to prevent unauthorized access and/or modification.

Applicable Features

All system resources in the GTNP have labels defined for them (i.e., programming subjects, segments, synchronization objects, mentor information, volume attributes, and devices). These labels are used as the basis for mandatory access control decisions whenever these decisions are made; for example:

- When segments and related structures (synchronization objects, mentor information) are made known, the label on the segment is checked.

- When volumes are mounted, the label on the volume information is checked.
- When import/export devices are attached, the label on the device is checked.

Import/export in the GTNP can only be done through the import/export devices available through the logical device interface provided by the kernel. These are all single-level logical devices in the evaluated configuration, with a degenerate range established by an authorized user via system generation utilities. These utilities run before the system is operational; hence, auditing is not required.

In the evaluated configuration, transfer of information from one instantiation of a Gemini system (GEMSOS or GTNP) to another is not possible through GTNP volumes, as the Trusted Facility Manual directs the administrator to restrict all volume devices accessible by untrusted application to local volumes only. Furthermore, procedural controls in the Trusted Facility Manual (TFM) (as well as cryptographic) prevent export to other systems, since volumes are protected with respect to disclosure to the same extent as the CPU.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Labels requirement.

8.3 Label Integrity

Requirement

Sensitivity labels shall accurately represent security levels of the specific subjects or objects with which they are associated. When exported by the TCB, sensitivity labels shall accurately and unambiguously represent the internal labels and shall be associated with the information being exported.

Interpretation

The phrase "exported by the TCB" is understood to include transmission of information from an object in one component to an object in another component. Information transferred between NTCB partitions is addressed in the System Integrity Section. The form of internal and external (exported) sensitivity labels may differ, but the meaning shall be the same. The NTCB shall, in addition, ensure that correct association of sensitivity labels with the information being transported across the network is preserved.

As mentioned in the Trusted Facility Manual Section, encryption transforms the representation of information so that it is unintelligible to unauthorized subjects. Reflecting this transformation, the sensitivity level of the ciphertext is generally lower than the cleartext. It follows that cleartext and ciphertext are contained in different objects, each possessing its own label. The label of the cleartext must be preserved and associated with the ciphertext so that it can be restored when the cleartext is subsequently obtained by decrypting the ciphertext. If the cleartext is associated with a single-level device, the label of that cleartext may be implicit. The label may also be implicit in the key.

When information is exported to an environment where it is subject to deliberate or accidental modification, the TCB shall support the means, such as cryptographic checksums, to assure the accuracy of the labels. When there is a mandatory integrity policy, the policy will define the meaning of integrity labels.

Applicable Features

In the GTNP, for objects created on media (e.g., disk) that are accessible through I/O devices, the accuracy of the label is assured through the use of a cryptographic checksum and is derived from the subject that created the object. Subject labels are actually assured by the kernel protecting its own internal data structures. The TCB, when creating an object, ensures that the label on the object fits within the (degenerate) range of the volume. This check is not done on access of an existing object; hence, the importance of the cryptographic checksum (as a volume could be accessed as a raw disk when not mounted).

The labels of subjects and objects are never exported, as all import/export devices in the GTNP are single-level. In particular, the resources used to create the volume abstraction (e.g., disk media) are never "exported by the TCB." Cryptographic seals are used to ensure that these resources remain as part of the TCB. Thus, although labels are written with segments to volume, these labels are in no way "exported."

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Label Integrity requirement.

8.4 Exportation of Labeled Information

Requirement

The TCB shall designate each communication channel and I/O device as either single-level or multilevel. Any change in this designation shall be done manually and shall be auditable by the TCB. The TCB shall maintain and be able to audit any change in the security level or levels associated with a communication channel or I/O device.

Interpretation

Each communication channel and network component shall be designated as either single-level or multilevel. Any change in this designation shall be done with the cognizance and approval of the administrator or security officer in charge of the affected components and the administrator or security officer in charge of the NTCB. This change shall be auditable by the network. The NTCB shall maintain and be able to audit any change in the device labels associated with a single-level communication channel or the range associated with a multilevel communication channel or component. The NTCB shall also be able to audit any change in the set of sensitivity levels associated with the information which can be transmitted over a multilevel communication channel or component.

Applicable Features

In the GTNP product, all communications channels and devices used for import/export are single-level. Each channel and device has an assigned range (minimum and maximum); however in the evaluated configuration these values are always equal (per the TFM). Changes in device characteristics can be done only when the system is not operational, as part of system generation, and hence are not subject to auditing.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Exportation of Labeled Information requirement.

8.5 Exportation to Multilevel Devices

Requirement

When the TCB exports an object to a multilevel I/O device, the sensitivity label associated with that object shall also be exported and shall reside on the same physical medium as the exported information and shall be in the same form (i.e., machine-readable or human-readable form). When the TCB exports or imports an object over a multilevel communication channel, the protocol used on that channel shall provide for the unambiguous pairing between the sensitivity labels and the associated information that is sent or received.

Interpretation

The components, including hosts, of a network shall be interconnected over "multilevel communication channels," multiple single-level communication channels, or both, whenever the information is to be protected at more than a single sensitivity level. The protocol for associating the sensitivity label and the exported information shall provide the only information needed to correctly associate a sensitivity level with the exported information transferred over the multilevel channel between the NTCB partitions in individual components. This protocol definition must specify the representation and semantics of the sensitivity labels (i.e., the machine-readable label must uniquely represent the sensitivity level).

The "unambiguous" association of the sensitivity level with the communicated information shall meet the same level of accuracy as that required for any other label within the NTCB, as specified in the criterion for Label Integrity. This may be provided by protected and highly reliable direct physical layer connections, or by traditional cryptographic link protection in which any errors during transmission can be readily detected, or by use of a separate channel. The range of information imported or exported must be constrained by the associated device labels.

Applicable Features

The GTNP does not support multilevel devices in its evaluated configuration. The GTNP does support a disk unit device, which allows access to all partitions on a disk. The TFM directs that this unit device must be defined to have a range that encloses the ranges of all the partitions on the disk unit. If that results in a non-degenerate range, the disk unit device cannot be accessed by non-TCB subjects in the evaluated configuration.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Exportation to Multilevel Devices requirement.

8.6 Exportation to Single-Level Devices

Requirement

Single-level I/O devices and single-level communication channels are not required to maintain the sensitivity labels of the information they process. However, the TCB shall include a mechanism by which the TCB and an authorized user can reliably communicate to designate the single security level of information imported or exported via single-level communication channels or I/O devices.

Interpretation

Whenever one or both of two directly connected components is not trusted to maintain the separation of information of different sensitivity levels, or whenever the two directly connected components have only a single sensitivity level in common, the two components of the network shall communicate over a single-level channel. Single-level components and single-level communication channels are not required to maintain the sensitivity labels of the information they process. However, the NTCB shall include a reliable communication mechanism by which the NTCB and an authorized user (via a trusted path) or a subject within an NTCB partition can designate the single sensitivity level of information imported or exported via single-level communication channels or network components. The level of information communicated must equal the device level.

Applicable Features

All devices in the evaluated configuration are single-level with a degenerate range (i.e., minimum equal to maximum). Hence, the sensitivity label of all data exported through a single-level device is implicitly that of the device. The sensitivity label of these devices can be changed only when the system is non-operational.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Exportation to Single-Level Devices requirement.

8.7 Labeling Human-Readable Output

Requirement

The ADP system administrator shall be able to specify the printable label names associated with exported sensitivity labels. The TCB shall mark the beginning and end of all human-readable, paged, hardcopy output (e.g., line printer output) with human-readable sensitivity labels that properly [1] represent the sensitivity of the output. The TCB shall, by default, mark the top and bottom of each page of human-readable, paged, hardcopy output (e.g., line printer output) with human-readable sensitivity labels that properly represent the overall sensitivity of the output or that properly represent the sensitivity of the information on the page. The TCB shall, by default and in an appropriate manner, mark other forms of human-readable output (e.g., maps, graphics) with human-readable sensitivity labels that properly represent the sensitivity of the output. Any override of these marking defaults shall be auditable by the TCB.

Interpretation

This criterion imposes no requirement to a component that produces no human-readable output. For those that do produce human-readable output, each sensitivity level that is defined to the network shall have a uniform meaning across all components. The network administrator, in conjunction with any affected component administrator, shall be able to specify the human-readable label that is associated with each defined sensitivity level.

Applicable Features

There are no human-readable output devices in the evaluated configuration.

Conclusion

The A1 Labeling Human-Readable Output requirement is not applicable to the Gemini Trusted Network Processor version 1.01.

[1] The hierarchical classification component in human-readable sensitivity labels shall be equal to the greatest hierarchical classification of any of the information in the output that the labels refer to; the non-hierarchical category component shall include all of the non-hierarchical categories of the information in the output the labels refer to, but no other non-hierarchical categories.

8.8 Subject Sensitivity Levels

Requirement

The TCB shall immediately notify a terminal user of each change in the security level associated with that user during an interactive session. A terminal user shall be able to query the TCB as desired for a display of the subject's complete sensitivity label.

Interpretation

An NTCB partition shall immediately notify a terminal user attached to its component of each change in the sensitivity level associated with that user.

Additional Network Component Interpretation

An M-Component need not support direct terminal input in which case this requirement is not applicable. Any M-Component which does support direct terminal input must meet the requirement as stated.

Applicable Features

The GTNP does not support interactive users when operational. Interactive users are supported only by the System Generation Utilities. See section 10.7, page 186 for an evaluator's comment with respect to supporting this requirement when the GTNP Mandatory Network Trusted Computing Base (M-NTCB) partition is composed with a partition supporting interactive users when operational.

Conclusion

The A1 Subject Sensitivity Levels requirement is not applicable to the Gemini Trusted Network Processor version 1.01.

8.9 Device Labels

Requirement

The TCB shall support the assignment of minimum and maximum security levels to all attached physical devices. These security levels shall be used by the TCB to enforce constraints imposed by the physical environments in which the devices are located.

Interpretation

This requirement applies as written to each NTCB partition that is trusted to separate information based on sensitivity level. Each I/O device in a component, used for communication with other network components, is assigned a device range, consisting of a set of labels with a maximum and minimum. (A device range usually contains, but does not necessarily contain, all possible labels "between" the maximum and minimum, in the sense of dominating the minimum and being dominated by the maximum.)

The NTCB always provides an accurate label for information exported through devices. Information exported or imported using a single-level device is labelled implicitly by the sensitivity level of the device. Information exported from one multilevel device and imported at another must be labelled through an agreed-upon protocol, unless it is labelled implicitly by using a communication link that always carries a single level.

Information exported at a given sensitivity level can be sent only to an importing device whose device range contains that level or a higher level. If the importing device range does not contain the given level, the information is relabelled upon reception at a higher level within the importing device range. Relabelling should not occur otherwise.

Applicable Features

All physical devices in the GTNP have minimum and maximum security labels. The explicit labels on all physical devices are used to constrain the levels at which logical devices can be attached.

In the case of volumes, each physical device used to mount a volume is assigned a minimum and maximum that reflect the physical environment of that physical device (per the TFM). When a volume is mounted, the TCB ensures that the volume limits are within the range defined by the administrator for the physical device.

For import/export devices, the labels on the physical device determine the allowable subject range that can attach the device. In order for an attach to be successful, the subject must, under the Mandatory Access Control (MAC) rules, be able to read or write the device (as necessary). This implies, for example, that in the case of read/write devices, the range of the subject must enclose the range of the device being attached.

For physical devices used to provide kernel resources, the device labels serve to determine the range of segments that can exist on the device. This is done by the TCB refusing to mount a volume when the range of the physical device does not enclose the range of the volume; the TCB subsequently does not allow segments to be created on the volume that are not within the volume range.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Device Labels requirement.

8.10 Mandatory Access Control

Requirement

The TCB shall enforce a mandatory access control policy over all resources (i.e., subjects, storage objects, and I/O devices) that are directly or indirectly accessible by subjects external to the TCB. These subjects and objects shall be assigned sensitivity labels that are a combination of hierarchical classification levels and non-hierarchical categories, and the labels shall be used as the basis for mandatory access control decisions. The TCB shall be able to support two or more such security levels. The following requirements shall hold for all accesses between all subjects external to the TCB and all objects directly or indirectly accessible by these subjects: A subject can read an object only if the hierarchical classification in the subject's security level is greater than or equal to the hierarchical classification in the object's security level and the non-hierarchical categories in the subject's security level include all the non-hierarchical categories in the object's security level. A subject can write an object only if the hierarchical classification in the subject's security level is less than or equal to the hierarchical classification in the object's security level and all the non-hierarchical categories in the subject's security level are included in the non-hierarchical categories in the object's security level. Identification and authentication data shall be used by the TCB to authenticate the user's identity and to ensure that the security level and authorization of subjects external to the TCB that may be created to act on behalf of the individual user are dominated by the clearance and authorization of that user.

Interpretation

Each partition of the NTCB exercises mandatory access control policy over all subjects and objects in its component. In a network, the responsibility of an NTCB partition encompasses all mandatory access control functions in its component that would be required of a TCB in a stand-alone system. In particular, subjects and objects used for communication with other components are under the control of the NTCB partition. Mandatory access control includes secrecy and integrity control to the extent that the network sponsor has described in the overall network security policy.

Conceptual entities associated with communication between two components, such as sessions, connections and virtual circuits, may be thought of as having two ends, one in each component, where each end is represented by a local object. Communication is viewed as an operation that copies information from an object at one end of a communication path to an object at the other end. Transient data-carrying entities, such as datagrams and packets, exist either as information within other objects, or as a pair of objects, one at each end of the communication path.

The requirement for "two or more" sensitivity levels can be met by either secrecy or integrity levels. When there is a mandatory integrity policy, the stated requirements for reading and writing are generalized to: A subject can read an object only if the subject's sensitivity level dominates the object's sensitivity level, and a subject can write an object only if the object's sensitivity level dominates the subject's sensitivity level. Based on the integrity policy, the network sponsor shall define the dominance relation for the total label, for example, by combining secrecy and integrity lattices.

Applicable Features

The GTNP enforces a mandatory access control policy between all subjects and objects (see section 5, page 105) as explained in section 6.1, page 109. The Non-Discretionary Security Manager (NDSM) layer of the kernel is always invoked whenever a subject obtains access to an object. The NDSM layer establishes the presence or absence of a dominance relationship between the subject's labels and the object's label. The label includes 2 hierarchical levels and up to 96 non-hierarchical categories. Information is also maintained to support a ring integrity policy. When labels are compared for a dominance relationship, both hierarchical levels and all non-hierarchical categories are used in the comparison.

The access labels provided by the GTNP can be used to implement various policies based on a lattice of labels. In the particular policy enforced by the GTNP, each label contains secrecy and integrity components represented by a hierarchical secrecy level, hierarchical integrity level, 64 non-hierarchical secrecy categories, and 32 non-hierarchical integrity categories.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Mandatory Access Control requirement.

8.11 Trusted Path

Requirement

The TCB shall support a trusted communication path between itself and users for use when a positive TCB-to-user connection is required (e.g., login, change subject security level). Communications via this trusted path shall be activated exclusively by a user or the TCB and shall be logically isolated and unmistakably distinguishable from other paths.

Interpretation

A trusted path is supported between a user (i.e., human) and the NTCB partition in the component to which the user is directly connected.

Additional Network Component Interpretation

An M-Component need not support direct user input (e.g., the M-Component may not be attached to any user I/O devices such as terminals) in which case this requirement is not applicable. Any M-Component which does support direct communication with users must meet the requirement as stated. In addition, an M-Component with directly connected users must provide mechanisms which establish the clearance of users and associate that clearance with the user's current session.

Applicable Features

The GTNP, when operating in the evaluated configuration, does not support direct user input. Thus, this requirement is not applicable for the M-component partition.

Section 10.6, page 186, provides an evaluator's comment regarding satisfaction of this requirement when the M-component is composed with other network components (implemented as virtual machines on top of the M-component) that do support users.

Conclusion

The A1 Trusted Path requirement is not applicable to the Gemini Trusted Network Processor version 1.01.

8.12 System Architecture

Requirement

The TCB shall maintain a domain for its own execution that protects it from external interference or tampering (e.g., by modification of its code or data structures). The TCB shall maintain process isolation through the provision of distinct address spaces under its control. The TCB shall be internally structured into well-defined largely independent modules. It shall make effective use of available hardware to separate those elements that are protection-critical from those that are not. The TCB modules shall be designed such that the principle of least privilege is enforced. Features in hardware, such as segmentation, shall be used to support logically distinct storage objects with separate attributes (namely: readable, writeable). The user interface to the TCB shall be completely defined and all elements of the TCB identified. The TCB shall be designed and structured to use a complete, conceptually simple protection mechanism with precisely defined semantics. This mechanism shall play a central role in enforcing the internal structuring of the TCB and the system. The TCB shall incorporate significant use of layering, abstraction and data hiding. Significant system engineering shall be directed toward minimizing the complexity of the TCB and excluding from the TCB modules that are not protection-critical.

Interpretation

The system architecture criterion must be met individually by all NTCB partitions. Implementation of the requirement that the NTCB maintain a domain for its own execution is achieved by having each NTCB partition maintain a domain for its own execution. Since each component is itself a distinct domain in the overall network system, this also satisfies the requirement for process isolation through distinct address spaces in the special case where a component has only a single subject.

The NTCB must be internally structured into well-defined largely independent modules and meet the hardware requirements. This is satisfied by having each NTCB partition so structured. The NTCB controls all network resources. These resources are the union of the sets of resources over which the NTCB partitions have control. Code and data structures belonging to the NTCB, transferred among NTCB subjects (i.e., subjects outside the reference monitor but inside the NTCB) belonging to different NTCB partitions, must

be protected against external interference or tampering. For example, a cryptographic checksum or physical means may be employed to protect user authentication data exchanged between NTCB partitions.

Each NTCB partition must enforce the principle of least privilege within its component. Additionally, the NTCB must be structured so that the principle of least privilege is enforced in the system as a whole.

The NTCB must be designed and structured according to the network security architecture to use a complete, conceptually simple protection mechanism. Furthermore, each NTCB partition must also be so designed and structured.

Significant system engineering should be directed toward minimizing the complexity of each NTCB partition, and of the NTCB. Care shall be taken to exclude modules (and components) that are not protection-critical from the NTCB.

It is recognized that some modules and/or components may need to be included in the NTCB and must meet the NTCB requirements even though they may not appear to be directly protection-critical. The correct operation of these modules/components is necessary for the correct operation of the protection-critical modules and components. However, the number and size of these modules/components should be kept to a minimum.

Each NTCB partition provides isolation of resources (within its component) in accord with the network system architecture and security policy so that "supporting elements" (e.g., DAC and user identification) for the security mechanisms of the network system are strengthened compared to C2, from an assurance point of view, through the provision of distinct address spaces under control of the NTCB.

As discussed in the Discretionary Access Control section, the DAC mechanism of a NTCB partition may be implemented at the interface of the reference monitor or may be distributed in subjects that are part of the NTCB in the same or different component. When distributed in NTCB subjects (i.e., when outside the reference monitor), the assurance requirements for the design and implementation of the DAC shall be those of class C2 for all networks of class C2 or above.

Additional Network Component Interpretation

An M-Component must meet the requirement as stated. In this interpretation the words "The user interface to the TCB shall be completely defined..." shall be interpreted to mean the interface between the reference monitor of the M-Component and the subjects external to the reference monitor shall be completely defined.

Applicable Features

Gemini has used a combination of hardware protection features, software engineering techniques, and design documentation to satisfy the system architecture requirement.

The GTNP maintains a domain for its own execution by executing at privilege levels and in rings that are unavailable to untrusted application processes:

- The kernel executes at PL0, and no untrusted applications may execute in PL0.

- Kernel data is stored on disk with ring brackets restricting it to Ring 0 (R0), which is available only to the System Generation Utilities. In-memory copies of the kernel data are only accessible at PL0.
- GEMSOS Network Processor System Parent (NPSP), the sole non-kernel trusted process, executes in R1 at PL1. NPSP data is stored on disk with ring brackets restricting it to R1. In the evaluated configuration, no other untrusted application processes may be executing in R1 (the minimum ring/PL for an untrusted process is R2 in PL1).

Process isolation is accomplished through the use of hardware task switching, hardware segmentation and the hardware privilege level mechanism. Software engineering techniques have been applied during the design and implementation of the system to produce a system composed of well-defined largely independent modules. Protection-critical elements of the TCB have been separated from non-protection-critical elements through the use of hardware segmentation and privilege level mechanisms. Least privilege has been enforced by allowing modules only the data needed to accomplish their tasks. The hardware segmentation feature has been used to separate TCB data by read and write modes of access. The TCB interface has been specified completely by the DTLS. The elements of the TCB have been identified by the GTNP system specification [97]. By implementing the reference monitor concept, the TCB has been designed and structured to use a complete, conceptually simple protection mechanism. This mechanism plays a central role in enforcement by being invoked on every subject to object access. Software engineering techniques have been applied to ensure a system that is layered and uses abstraction and information hiding. The GTNP was built from scratch by a small group of developers that used significant effort to ensure that only protection critical modules were included in the TCB. More information on the GTNP's satisfaction of the system architecture requirement can be found in section 7.1, page 117.

The evaluation team conducted an architecture study of the GTNP. This study included the code used in the boot Programmable Read Only Memory (PROM), which is also part of the GTNP kernel. In general, Gemini does not use commercial boot PROMs in their product as there is insufficient visibility of the design of the code in commercial PROMs; however, certain peripheral devices do use commercial boot PROMs.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 System Architecture requirement.

8.13 System Integrity

Requirement

Hardware and/or software features shall be provided that can be used to periodically validate the correct operation of the on-site hardware and firmware elements of the TCB.

Interpretation

Implementation of the requirement is partly achieved by having hardware and/or software features that can be used to periodically validate the correct operation of the hardware and firmware elements of each

component's NTCB partition. Features shall also be provided to validate the identity and correct operation of a component prior to its incorporation in the network system and throughout system operation. For example, a protocol could be designed that enables the components of the partitioned NTCB to exchange messages periodically and validate each other's correct response. The protocol shall be able to determine the remote entity's ability to respond. NTCB partitions shall provide the capability to report to network administrative personnel the failures detected in other NTCB partitions.

Intercomponent protocols implemented within a NTCB shall be designed in such a way as to provide correct operation in the case of failures of network communications or individual components. The allocation of mandatory and discretionary access control policy in a network may require communication between trusted subjects that are part of the NTCB partitions in different components. This communication is normally implemented with a protocol between the subjects as peer entities. Incorrect access within a component shall not result from failure of an NTCB partition to communicate with other components.

Applicable Features

During system boot time, hardware and software tests are executed to verify correct operation of the hardware and firmware (see section 7.2, page 121). The tests verify the minimal set of functions necessary for the correct operation of the secure environment. This set pertains mainly to the 80286, i386, and i486 hardware; specifically protection domains and segmentation.

Additional diagnostic tests are run on each processor within a low priority process. These tests are executed whenever no higher priority processes are available to be run on the processor (either due to blocking or termination).

With regards to validating the identity and correct operation of components prior to incorporation into a network and throughout network operation, the following features seem to apply:

- Correct operation is tested during GTNP boot time, hence correct operation can be assured prior to attaching it to a network.
- Since diagnostics are run during otherwise-unused CPU time, the GTNP will detect operational problems and shut itself down, hence other components of the network can detect this condition.
- Since the GTNP is intended to host network applications and other components themselves, error messages from the GTNP TCB will also allow correct operation of the GTNP TCB to be detected.
- Since the GTNP does not rely on other network components for its own correct operation, there is no need for it to be able to detect such a failure. However, if the other network component is executing as an application on the GTNP, its correct operation will be detectable through the same set of diagnostics that determine correct operation of the GTNP TCB.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 System Integrity requirement.

8.14 Covert Channel Analysis

Requirement

The system developer shall conduct a thorough search for covert channels and make a determination (either by actual measurement or by engineering estimation) of the maximum bandwidth of each identified channel. Formal methods shall be used in the analysis.

Interpretation

The requirement, including the TCSEC Covert Channel Guideline, applies as written. In a network, there are additional instances of covert channels associated with communication between components. The formal methods shall be used in the analysis of each individual component design and implementation.

Additional Network Component Interpretation

An M-Component must meet the requirement as stated. In addition, if the analysis indicates that channels exist that need to be audited (according to the Covert Channel Analysis Guideline), the M-Component shall contain a mechanism for making audit data (related to possible use of covert channels) available outside of the M-Component (e.g., by passing the data to an audit collection component).

Applicable Features

A covert channel analysis [66, 67] in the GTNP was performed using a combination of formal and informal techniques. Gemini provided two very complete covert channel analysis documents: one for storage channels [66], and one for timing channels [67]. These documents described the methodology the vendor used to find the channels, and presented a detailed description of the exploitation techniques and optimization methods. For each channel, the vendor provided a description of the channel and the optimal exploitation scenario. The covert storage channel analysis found 12 covert storage channels; all of which are closable by appropriate system configuration as described in the Trusted Facility Manual [87]. The analysis identified a number of different types of covert timing channels, none of which are auditable.

Formal techniques were used in the analysis of the covert storage channels. Gemini developed a methodology for identifying covert storage channels that combines the Shared Resource Matrix method with information flow analysis. For each storage channel found, Gemini employs Shannon's [161] method for bandwidth estimation as described by Millen [135]. Bandwidth calculations are based on the optimal execution speed of kernel calls when run on the fastest hardware platform. All storage channels are closable through administrative configuration; however, in the event an administrator chooses to operate the system with bandwidths that exceed the auditing threshold, information was provided on how to observe covert storage channel usage via an A-Component operating on top of the M-Component NTCB Partition.

The covert timing channel analysis was conducted informally, and began with an enumeration of all timing-sensitive resources. A timing-sensitive resource refers to a resource that a TCB interface may be required to wait a varying amount of time to access due to an access or state change caused by another process. Both

hardware and software resources were enumerated. For each resource identified, Gemini either provided a rationale for why that resource could not be exploited to support covert communication or developed a scenario to illustrate an optimal bandwidth supported by that channel.

The timing channel analysis [67] divided the potential channels into those that are exploited solely through 80286, i386, or i486 processor instructions, those that are exploited through the use of kernel gates, and those that are exploited through the network interface. The processor instruction analysis identified two high capacity timing channels, for which countermeasures have been implemented to remove one channel and reduce the speed of the other. A third processor instruction timing channel was latter identified as a result of penetration testing. Timing channels that were exercised through kernel gates were broken into equivalence classes based on the kernel gates utilized; the timing information for each gate provided a maximal bandwidth estimation for all channels that require the use of that gate for exploitation. The vendor identified all timing sensitive resources subject to timing attacks, and indicated the kernel calls that would be used to perform the attack. This, in combination with the kernel gate equivalence classes, provided maximal bandwidths for each kernel resource subject to a timing channel attack through kernel gates. A kernel call governor has been implemented to modify the individual execution speeds of the various kernel calls, thus providing a countermeasure to reduce the capacity of channels that use the kernel calls. Lastly, network timing channels were identified, in which one process may utilize I/O interrupt activity to affect the execution speed of a second process operating on the CPU from which the interrupt is handled. When both the writer and reader processes are connected to GTNP hosts, the bandwidth is limited to the intrinsic rate associated with the write sequential call (which is subject to control by the kernel call governor).

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Covert Channel Analysis requirement.

8.15 Trusted Facility Management

Requirement

The TCB shall support separate operator and administrator functions. The functions performed in the role of a security administrator shall be identified. The ADP system administrative personnel shall only be able to perform security administrator functions after taking a distinct auditable action to assume the security administrator role on the ADP system. Non-security functions that can be performed in the security administration role shall be limited strictly to those essential to performing the security role effectively.

Interpretation

This requirement applies as written to both the network as a whole and to individual components which support such personnel.

Additional Network Component Interpretation

An M-Component / D-Component / I-Component must meet the requirement as stated except for the words "The procedures for examining and maintaining the audit files as well as..." These words are interpreted to mean "the mechanisms and protocols associated with exporting of audit data must be defined." Also, the words "...to include changing the security characteristics of a user", shall not be applicable to an M-Component.

Applicable Features

The GTNP does not support interfaces for operator and administrator functions in the run-time system. There are operator and security administrator roles supported by the System Generation Utilities, but these tools are a part of the system configuration process that occurs before the secure initial state. Auditing is not required.

Conclusion

The A1 Trusted Facility Management requirement is not applicable to the Gemini Trusted Network Processor version 1.01.

8.16 Trusted Recovery

Requirement

Procedures and/or mechanisms shall be provided to assure that, after an ADP system failure or other discontinuity, recovery without a protection compromise is obtained.

Interpretation

The recovery process must be accomplished without a protection compromise after the failure or other discontinuity of any NTCB partition. It must also be accomplished after a failure of the entire NTCB.

Applicable Features

As summarized in section 7.4, page 125, a failure or discontinuity in the GTNP component caused by an error diagnostic, non-recoverable interrupt, or the boot switch is handled by shutting down the system. If the system is rebooted, software and procedures are used to restore a secure state as described in section 4.2.3, page 95.

Failures elsewhere in the network cannot affect the GTNP because all software external to the GTNP is not trusted by the GTNP to enforce or support its mandatory security policy. All initialization is performed

locally. Since all devices are single-level, the GTNP is not dependent on correct remote responses to protocols to label data from remote components. It is not dependent on other components for any other reason, since MAC is the only policy it provides.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Trusted Recovery requirement.

8.17 Security Testing

Requirement

The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. A team of individuals who thoroughly understand the specific implementation of the TCB shall subject its design documentation, source code, and object code to thorough analysis and testing. Their objectives shall be: to uncover all design and implementation flaws that would permit a subject external to the TCB to read, change, or delete data normally denied under the mandatory or discretionary security policy enforced by the TCB; as well as to assure that no subject (without authorization to do so) is able to cause the TCB to enter a state such that it is unable to respond to communications initiated by other users. The TCB shall be found resistant to penetration. All discovered flaws shall be corrected and the TCB retested to demonstrate that they have been eliminated and that new flaws have not been introduced. Testing shall demonstrate that the TCB implementation is consistent with the descriptive top-level specification. No design flaws and no more than a few correctable implementation flaws may be found during testing and there shall be reasonable confidence that few remain. Manual or other mapping of the FTLS to the source code may form a basis for penetration testing.

Interpretation

Testing of a component will require a testbed that exercises the interfaces and protocols of the component including tests under exceptional conditions. The testing of a security mechanism of the network system for meeting this criterion shall be an integrated testing procedure involving all components containing an NTCB partition that implement the given mechanism. This integrated testing is additional to any individual component tests involved in the evaluation of the network system. The sponsor should identify the allowable set of configurations including the sizes of the networks. Analysis or testing procedures and tools shall be available to test the limits of these configurations. A change in configuration within the allowable set of configurations does not require retesting.

The testing of each component will include the introduction of subjects external to the NTCB partition for the component that will attempt to read, change, or delete data normally denied. If the normal interface to the component does not provide a means to create the subjects needed to conduct such a test, then this portion of the testing shall use a special version of the untrusted software for the component that results in subjects that make such attempts. The results shall be saved for test analysis. Such special versions shall have an NTCB partition that is identical to that for the normal configuration of the component under evaluation.

The testing of the mandatory controls shall include tests to demonstrate that the labels for information imported and/or exported to/from the component accurately represent the labels maintained by the NTCB partition for the component for use as the basis for its mandatory access control decisions. The tests shall include each type of device, whether single-level or multilevel, supported by the component.

The NTCB must be found resistant to penetration. This applies to the NTCB as a whole, and to each NTCB partition in a component of this class.

Additional Network Component Interpretation

An M-Component must meet the requirement as stated except for the words "normally denied under the ... descretionary security policy," which are not applicable to an M-Component.

Applicable Features

Functional security tests of the GTNP TCB (including the NPSP) are composed of interface and special engineering tests that demonstrate the adherence to the specification of the kernel interface. In general, the interface tests verify the functionality of the interface, and include exception tests, no-exception tests, and device tests. The special engineering tests demonstrate proper performance of key features, functions or subsystems that are not directly mappable to the interface, or that demonstrate other specific scenarios of interest. A detailed description of these tests can be found in section 7.5, page 125.

The NPSP functional security tests (FST) test all of the NPSP code that can be executed through the use of the the NPSP interface. All code not tested by the NPSP FST are tested by module testing. More detail on these tests can be found in section 7.5.3, page 128.

The evaluation team conducted six weeks of functional and penetration testing on the GTNP. Not all of Gemini's functional tests were run on every test platform (See section 7.5.7, page 130 for the discussion of tested configurations). These tests could be categorized as follows: tests that were device specific (and the device was not configured on the platoform); tests that were platform specific; tests that were configuration specific (and that configuration was not available for testing); and, tests that were viewed as not security relevant.

The team generated and executed five functional tests, none of which identified any problems.

Using the System Developement Corporation (SDC) Flaw Hypothesis Methodology, the team formulated 47 penetration hypotheses, of which 46 were investigated. Of these, one identified a software problem and four identified documentation problems that have been corrected, and two identified covert channels that have been analyzed.

As of the conclusion of testing, no uncorrected problems identified by the test effort remained in the system.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Security Testing requirement.

8.18 Design Specification and Verification

Requirement

A formal model of the security policy supported by the TCB shall be maintained over the life cycle of the ADP system that is proven consistent with its axioms. A descriptive top-level specification (DTLS) of the TCB shall be maintained that completely and accurately describes the TCB in terms of exceptions, error messages, and effects. A formal top-level specification (FTLS) of the TCB shall be maintained that accurately describes the TCB in terms of exceptions, error messages, and effects. The DTLS and FTLS shall include those components of the TCB that are implemented as hardware and/or firmware if their properties are visible at the TCB interface. The FTLS shall be shown to be an accurate description of the TCB interface. A convincing argument shall be given that the DTLS is consistent with the model and a combination of formal and informal techniques shall be used to show that the FTLS is consistent with the model. This verification evidence shall be consistent with that provided within the state-of-the-art of the particular National Computer Security Center-endorsed formal specification and verification system used. Manual or other mapping of the FTLS to the TCB source code shall be performed to provide evidence of correct implementation.

Interpretation

The overall network security policy expressed in this model will provide the basis for the mandatory access control policy exercised by the NTCB over subjects and storage objects in the entire network. The policy will also be the basis for the discretionary access control policy exercised by the NTCB to control access of named users to named objects. Data integrity requirements addressing the effects of unauthorized MSM need not be included in this model. The overall network policy must be decomposed into policy elements that are allocated to appropriate components and used as the basis for the security policy model for those components.

The level of abstraction of the model, and the set of subjects and objects that are explicitly represented in the model, will be affected by the NTCB partitioning. Subjects and objects must be represented explicitly in the model for the partition if there is some network component whose NTCB partition exercises access control over them. The model shall be structured so that the axioms and entities applicable to individual network components are manifest. Global network policy elements that are allocated to components shall be represented by the model for that component.

An FTLS for a network consists of a component FTLS for each unique trusted network component, plus any global declarations and assertions that apply to more than one component. If the model for each component represents all the global mandatory policy elements allocated to that component, there may not be any global assertions needed, and in this case the collection of component FTLS, with any shared declarations, is the network FTLS. Each component FTLS shall describe the interface to the NTCB partition of its components. The requirements for a network DTLS are given in the Design Documentation section.

Additional Network Component Interpretation

Security Policy is interpreted to mean the MAC Policy supported by the component. Model is interpreted to be those portions of a reference monitor model that are relevant to the MAC Policy supported by the component (e.g., the representation of the current access set and the sensitivity labels of subjects and objects, and the Simple Security and Confinement Properties of the Bell and LaPadula Model)

Applicable Features

The model, FTLS, and verification evidence were described in section 7.6, page 131. The model is written in Ina Jo, a formal language supported by the NCSC-endorsed Formal Development Methodology (FDM) tool, Version 12.5. Consistency of the model with its axioms was shown by proving the correctness theorems generated by the Ina Jo processor, which verify the initial condition, criterion, and invariants. The model adequately expresses the mandatory policy required by the TCSEC as implemented in the GTNP.

The DTLS is in the form of software development specifications used by Gemini to implement the GTNP kernel and NPSP. It appears to be an accurate description of the TCB. Correspondence of the DTLS with the model is covered in section 2.4 of the *Philosophy and Design of Protection* document [26].

The FTLS has an adequate level of detail and its transforms can be matched directly with the kernel interface and the functions performed by the NPSP. Consistency of the FTLS with the model is shown by the following steps. First, an Ina Jo mapping between the FTLS and model is defined; the Ina Jo transform mapping theorem is proved formally using the Interactive Theorem Prover (ITP); then an informal argument is provided showing that the model-to-FTLS mapping is not restrictive, i.e., it does not exclude any possible FTLS transform call.

The mapping has been examined, and the informal argument for the mapping has been confirmed. The team has also conducted a review of the specifications of the GTNP, the GTNP kernel, and NPSP, identifying no significant problems. The team has also reviewed the specification to code mapping on the specification, and found it to be satisfactory. Finally, the team has examined the FTLS proof evidence from the use of FDM, and the proof was done properly.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Design Specification and Verification requirement.

8.19 Configuration Management

Requirement

During the entire life-cycle, i.e., during the design, development, and maintenance of the TCB, a configuration management system shall be in place for all security-relevant hardware, firmware, and software that maintains control of changes to the formal model, the descriptive and formal top-level specifications, other

design data, implementation documentation, source code, the running version of the object code, and test fixtures and documentation. The configuration management system shall assure a consistent mapping among all documentation and code associated with the current version of the TCB. Tools shall be provided for generation of a new version of the TCB from source code. Also available shall be tools, maintained under strict configuration control, for comparing a newly generated version with the previous TCB version in order to ascertain that only the intended changes have been made in the code that will actually be used as the new version of the TCB. A combination of technical, physical, and procedural safeguards shall be used to protect from unauthorized modification or destruction the master copy or copies of all material used to generate the TCB.

Interpretation

The requirement applies as written, with the following extensions:

1. A configuration management system must be in place for each NTCB partition.

2. A configuration management plan must exist for the entire system. If the configuration management system is made up of the conglomeration of the configuration management systems of the various NTCB partitions, then the configuration management plan must address the issue of how configuration control is applied to the system as a whole.

All material used in generating a new version of the NTCB and each NTCB partition must be protected, regardless of where it physically resides.

Applicable Features

Gemini has had some form of Configuration Management (CM) in place for the entire life-cycle of the configuration items that make up the GTNP. Since October of 1988, this configuration management has been via procedures spelled out in various versions of the *Gemini Configuration Management Plan* [79].[2] This plan calls for configuration control of all security relevant code, testing materials, verification materials, hardware, and documentation of the GTNP. Every Gemini division that is involved in the hardware and software development process participates in the CM plan; however, the Product Quality Assurance (QA) division and Configuration Control Board (CCB) have prime responsibility for enforcement of Gemini's CM procedures. The procedures and tools that Gemini uses to implement CM are specified in the list below:

- Establishment of Baselines: Gemini establishes baselines by entering configuration items, identified in Engineering Change Proposals (ECPs) and A-level (System Specifications), B-level (Software Development Specifications), or C-level (Software Product Specifications) specifications, into a QA library as they are completed by development groups, and approved by the CCB respectively.
- Development Procedures: Before a specification or ECP is entered into a baseline, the CCB conducts a review to determine if CM policies are being followed and if the design is complete, accomplishes its purpose, and matches the requirement. Also, during the Final Design Review QA conducts a CM

[2] Prior to October 1988, there was no formal CM system in place for control of changes. However, information such as product baselines, records of design decisions, and a list of known discrepancies were maintained. Further, code files contained maintenance records in their header comments, and each release identified known discrepancies that were corrected in the release.

verification of specifications or ECPs to determine if all related CM documentation of configuration items are complete and if CM policies are being followed. Finally the CCB may have QA conduct a CM verification at any of its specification or ECP reviews. If a specification or ECP does not pass a CM audit, it can't be entered into a baseline until the problems are corrected.

- Tools: Gemini makes use of a combination of computer databases and hand written logs to track the status of CM items and preserves the master copy of software baselines in a secure, off-site location.

The evaluation team conducted a CM audit of Gemini's CM procedures. The team found that information was maintained on all Configuration Item (CIs) in the GTNP for the lifetime of the product (and even before that, as some CIs in the GTNP, such as the GEMSOS kernel, predate the GTNP product). The team was able to identify the changes addressed in a given release of the product. The team also verified that the procedures enumerated in the Gemini CM Plan in effect at the time of a change were followed.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Configuration Management requirement.

8.20 Trusted Distribution

Requirement

A trusted ADP system control and distribution facility shall be provided for maintaining the integrity of the mapping between the master data describing the current version of the TCB and the on-site master copy of the code for the current version. Procedures (e.g., site security acceptance testing) shall exist for assuring that the TCB software, firmware, and hardware updates distributed to a customer are exactly as specified by the master copies.

Interpretation

This requirement applies as stated, with the additional requirement that, if down-line loading is used, there must be a trusted method of generating, sending, and loading any software involved.

Applicable Features

As described in section 7.8, page 142, TCB software and firmware are cryptographically sealed. Because the key management system employs system-specific keys burned into the System ID PROM (as well as other keys), it is not possible for a malicious agent to change the kernel software without also replacing the Gemini System Controller board. Because authenticators for the initial hardware delivery are shipped securely, the agent cannot substitute different circuit boards or firmware without detection.

To help counter the threat of substitution of components on boards, or whole boards with no firmware, Gemini's acceptance tests are made available to customers as a way of validating the hardware on delivery.

There is no provision for down-line loading of software.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Trusted Distribution requirement.

8.21 Security Features User's Guide

Requirement

A single summary, chapter, or manual in user documentation shall describe the protection mechanisms provided by the TCB, guidelines on their use, and how they interact with one another.

Interpretation

This user documentation describes user visible protection mechanisms at the global (network system) level and at the user interface of each component, and the interaction among these.

Applicable Features

Gemini has provided a Security Features User's Guide (SFUG) for the Trusted Network Processor. The users of this system would be software developers implementing untrusted applications. This document describes the protection mechanisms provided by the TCB, guidelines on their use, and descriptions on how they interact. The SFUG consists of the following four volumes:

- Volume 1 [49] provides an overview of the Trusted Network Processor security features.
- Volume 2 [48] provides a programmers reference guide. It is available in both Pascal and C forms.
- Volume 3 [33] is an appendix to volume 2 and describes the programming language interfaces for the supported devices. It is available in Pascal and C forms.
- Volume 4 [50] describes the language independent interface to the GEMSOS Security Kernel.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Security Features User's Guide requirement.

8.22 Trusted Facility Manual

Requirement

A manual addressed to the ADP system administrator shall present cautions about functions and privileges that should be controlled when running a secure facility. The procedures for examining and maintaining the audit files as well as the detailed audit record structure for each type of audit event shall be given. The manual shall describe the operator and administrator functions related to security, to include changing the security characteristics of a user. It shall provide guidelines on the consistent and effective use of the protection features of the system, how they interact, how to securely generate a new TCB, and facility procedures, warnings, and privileges that need to be controlled in order to operate the facility in a secure manner. The TCB modules that contain the reference validation mechanism shall be identified. The procedures for secure generation of a new TCB from source after modification of any modules in the TCB shall be described. It shall include the procedures to ensure that the system is initially started in a secure manner. Procedures shall also be included to resume secure system operation after any lapse in system operation.

Interpretation

This manual shall contain specifications and procedures to assist the system administrator(s) maintain cognizance of the network configuration. These specifications and procedures shall address the following:

1. The hardware configuration of the network itself;

2. The implications of attaching new components to the network;

3. The case where certain components may periodically leave the network (e.g., by crashing, or by being disconnected) and then rejoin;

4. Network configuration aspects that can impact the security of the network system; (For example, the manual should describe for the network system administrator the interconnections among components that are consistent with the overall network system architecture.)

5. Loading or modifying NTCB software or firmware (e.g., down-line loading).

6. Incremental updates; that is, it must explicitly indicate which components of the network may change without others also changing.

The physical and administrative environmental controls shall be specified. Any assumptions about security of a given network should be clearly stated (e.g., the fact that all communications links must be physically protected to a certain level).

The components of the network that form the NTCB must be identified. Furthermore, the modules within an NTCB partition that contain the reference validation mechanism (if any) within that partition must be identified.

The procedures for the secure generation of a new version (or copy) of each NTCB partition from source must be described. The procedures and requirements for the secure generation of the NTCB necessitated by changes in the network configuration shall be described.

Procedures for starting each NTCB partition in a secure state shall be specified. Procedures must also be included to resume secure operation of each NTCB partition and/or the NTCB after any lapse in system or subsystem operation.

Applicable Features

Gemini has provided a Trusted Facility Manual (TFM) for this evaluation. The TFM provides information on how to install and configure the system so as to provide the evaluated configuration. The TFM also provides information on how to validate trusted distribution, and to manage the system (from the point of view of the GTNP) once operational. The TFM is composed of the multiple documents; most of these documents are applicable only to the installation and configuration of the system.

A general overview of the system, as well as day to day operations of the system, are described in the following documents:

- *Guide to Trusted Facility Management for the Gemini Trusted Network Processor* [87], which provides an overview of the operation, installation, generation, and configuration procedures relevant to the Trusted Network Processor.
- *GEMSOS Mandatory Security Administrator Reference Manual* [81], which provides a reference for security functions to be performed in the role of GEMSOS Mandatory Security Administrator using the commands provided. A supplemental volume (*GTNP Mandatory Security Administrator Reference Manual* [85]) describing additional information specific to the GTNP is provided.
- *GEMSOS Security Features User's Guide: Volume 4: "User Guide to the GEMSOS Trusted Path"* [82], which describes how to use the trusted path facility provided by the System Generation Utilities.
- *GTNP Configuration Tools Operator Reference Manual* [84], which describes how an operator is to use the GTNP Configuration Tools.

The following documents are more focused on the installation processing, including verification of trusted distribution:

- *Product Installation Manual* [92], which provides the roadmap and instructions for product installation after delivery.
- *Product Creation Book for the Gemini Trusted Network Processor Product* [90], which provides information used to construct the GTNP.
- *Product Creation Book for the System Generation Tools* [91], which provides information used to construct the various system generation tools.
- *SYSGEN User Manual, System Generation Utility*[75], which provides detailed instructions for using the SYSGEN utility to generate a GTNP
- *Parameters for GEMSOS Security Kernel Generation* [57], a companion volume to the SYSGEN User Manual, which describes the significant kernel parameter tables in system generation.

- *DSKGEN User Manual, Hard Disk Conditioning Utility* [43], which describes how to condition the hard disk.
- *GEMSOS Hardware Distribution Reference Manual* [80], which provides information on how Gemini does hardware distributions, and how said distributions are verified.
- *Guide to Generating the Gemini Trusted Network Processor from Source Code* [86], which is provided by Gemini to source licensees and describes how to generate the GTNP given the supplied source code.
- *System Maintenance Utility User Manual* [95], which provides instructions for customizing the system configuration for site-specific requirements and for changes resulting from routine maintenance.
- *System PROM Bootstrap Utility User Manual* [96], which provides instructions for System PROM recovery.
- *Original User Installation Manual* [39], which describes how to install or recover the authentication database for the initial user of the system generation utilities.
- *Product Update Book for Gemini Trusted Network Processor Product* [93], which provides information used to update the GTNP from the previously released version.
- *Product Update Book for System Generation Tools Product* [94], which provides information used to update the system generation tools from the previously released version.

Also delivered with the product, and part of the TFM inasmuch as they support the installation and configuration process, are a series of Deficiency Summaries, and notes regarding volume allocation. [78, 42, 77, 36]

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Trusted Facility Manual requirement.

8.23 Test Documentation

Requirement

The system developer shall provide to the evaluators a document that describes the test plan, test procedures that show how the security mechanisms were tested, and results of the security mechanisms' functional testing. It shall include results of testing the effectiveness of the methods used to reduce covert channel bandwidths. The results of the mapping between the formal top-level specification and the TCB source code shall be given.

Interpretation

The "system developer" is interpreted as "the network system sponsor". The description of the test plan should establish the context in which the testing was or should be conducted. The description should identify

any additional test components that are not part of the system being evaluated. This includes a description of the test-relevant functions of such test components and a description of the interfacing of those test components to the system being evaluated. The description of the test plan should also demonstrate that the tests adequately cover the network security policy. The tests should include the features described in the System Architecture and the System Integrity sections. The tests should also include network configuration and sizing.

The mapping between the FTLS and the NTCB source code must be checked to ensure to the extent possible that the FTLS is a correct representation of the source code, and that the FTLS has been strictly adhered to during the design and development of the network system. This check must be done for each component of the network system for which an FTLS exists.

Applicable Features

The Kernel test plan is located in three separate documents. Section 4 and appendix III of the GTNP System Specification [97, 27] describe Gemini's grey-box test philosophy, and encompasses the derivation of the test requirements. Section 4 of the GEMSOS Security Kernel Software Development Specification [61] describes the requirements for grey-box testing at the interface-design level. Section 4 of GEMSOS Security Kernel Software Product Specification [65] describes the more detailed requirements for testing. The corresponding test procedures are found in the Gemini Test Acceptance Test (GTAT) Test Description Reference Manual [52], that describes the operating procedures for the tests, including a description of the condition being tested, step by step instructions for invoking each test, a list of resource requirements, any special setup requirements, and any notes regarding concurrency limitations, and in the test scripts themselves, that provide the actual content of the test.

The documentation describes tests that are divided into interface tests, that verify the functionality of the TCB interface, and special engineering tests, that demonstrate proper performance of key features, functions, or subsystems that are not directly mappable to the interface, or that demonstrate other specific scenarios of interest. The interface tests include the following: exception tests, exception ordering tests, no-exception tests, and device tests. The special engineering tests include the following: common internal capability tests, memory management tests, covert channel tests, concurrency tests, hardware protection mechanism tests, and hardware instruction set tests.

The NPSP test plan is located in Section 4 of the Software Product Specification for GEMSOS Multilevel Subjects [16], and provides the plan for conducting the functional security test of the NPSP Computer Software Configuration Item (CSCI). The corresponding test procedures are found in the NPSP Functional Security Test Description [38]. These procedures describe the operating procedures for the tests, including a description of the condition being tested and step by step instructions for performing each test.

Documentation is maintained on the tools used by the tests. These include the Application Generator (APGEN) [41]; the Device Test Monitor [31]; the GEMSOS Model Application Environment [34] the GEMSOS Test Application Environment [51]; the GT Memory Monitor [22]; the GT Segment Monitor [23]; the Gemini Test Controller [47]; the Kernel Call Test Monitor [24]; the Standard User Interface [74]; and the Volume Monitor [29]. All of these documents are maintained under configuration management.

The test sets and results are maintained under configuration management and are repeatable. For more detailed information refer to section 7.5, page 125.

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Test Documentation requirement.

8.24 Design Documentation

Requirement

Documentation shall be available that provides a description of the manufacturer's philosophy of protection and an explanation of how this philosophy is translated into the TCB. The interfaces between the TCB modules shall be described. A formal description of the security model enforced by the TCB shall be available and proven that it is sufficient to enforce the security policy. The specific TCB protection mechanisms shall be identified and an explanation given to show that they satisfy the model. The descriptive top-level specification (DTLS) shall be shown to be an accurate description of the TCB interface. Documentation shall describe how the TCB implements the reference monitor concept and give an explanation why it is tamper resistant, cannot be bypassed, and is correctly implemented. The TCB implementation (i.e., in hardware, firmware, and software) shall be informally shown to be consistent with the formal top-level specification (FTLS). The elements of the FTLS shall be shown, using informal techniques, to correspond to the elements of the TCB. Documentation shall describe how the TCB is structured to facilitate testing and to enforce least privilege. This documentation shall also present the results of the covert channels analysis and the tradeoffs involved in restricting the channels. All auditable events that may be used in the exploitation of known covert storage channels shall be identified. The bandwidths of known covert storage channels, the use of which is not detectable by the auditing mechanism, shall be provided. Hardware, firmware, and software mechanisms not dealt with in the FTLS but strictly internal to the TCB (e.g., mapping registers, direct memory access I/O) shall be clearly described.

Interpretation

Explanation of how the sponsor's philosophy of protection is translated into the NTCB shall include a description of how the NTCB is partitioned. The security policy also shall be stated. The description of the interfaces between the NTCB modules shall include the interface(s) between NTCB partitions and modules within the partitions if the modules exist. The sponsor shall describe the security architecture and design, including the allocation of security requirements among components.

The documentation includes both a system description and a set of component DTLS's. The system description addresses the network security architecture and design by specifying the types of components in the network, which ones are trusted, and in what way they must cooperate to support network security objectives. A component DTLS shall be provided for each trusted network component, i.e., each component containing an NTCB partition. Each component DTLS shall describe the interface to the NTCB partition of its component. Both the system description and each component DTLS shall be shown consistent with those assertions in the model that apply to it. Appendix A addresses component evaluation issues.

To show the correspondence between the FTLS and the NTCB implementation, it suffices to show correspondence between each component FTLS and the NTCB partition in that component.

As stated in the introduction to Division B, the sponsor must demonstrate that the NTCB employs the reference monitor concept. The security policy model must be a model for a reference monitor.

The security policy model for each partition implementing a reference monitor shall fully represent the access control policy supported by the partition, including the discretionary and mandatory security policy for secrecy and/or integrity. For the mandatory policy the single dominance relation for sensitivity labels, including secrecy and/or integrity components, shall be precisely defined.

Additional Network Component Interpretation

All components must meet the requirement as stated. In addition:

- The Design Documentation must include a description of the protocol used by the D-Component to communicate Subject permissions (i.e., user ids), where applicable, with other components. This protocol must be shown to be sufficient to support the DAC policy enforced by the D-Component.
- The Design Documentation must include a description of the protocol used by the I-Component to export Authenticated Subject Identifiers to other components.
- The Design Documentation must include a description of the protocol used by the A-Component to import Audit Data from other nodes.

Applicable Features

Gemini has provided a collection of design documentation for this evaluation. It includes the following documents:

- System Specification for Gemini Trusted Network Processor [97, 27, 26, 28, 40]: These documents provide the GTNP philosophy of protection, the Network Security Architecture and Design, the formal security policy model, the hardware base description, and the TCB protection mechanism description.
- Software Development [61, 15, 60, 25, 14, 30] and Product [65, 18, 68, 16, 17] Specifications: These documents provide the DTLS.
- Interface Requirements Specifications [54, 53]. These describe the interface requirements of the TCB.
- Software Development Standards [89]. This document describes the procedures followed by Gemini in developing software and documentation.
- GEMSOS Kernel FTLS [62] and the GEMSOS NPSP FTLS [25]: These documents provide the Formal Top-Level Specification.
- Code Correspondence Report for the GTNP [73]: This document demonstrates code correspondence between the code and the FTLS.
- GEMSOS Kernel Covert Storage Channel Report [66]: This report details the covert storage channels found in the GTNP.

- GEMSOS Kernel Covert Timing Channel Report [67]: This document details the covert timing channels found in the GTNP.
- Hardware Reference Manuals:
  - The general details of the Intel processors used in the product were described in publically-available Intel documents on the Intel network products [124, 121], Intel memory boards [112, 111, 110, 113, 105], 80186 [117, 118], 80188 [114], 80286 processor and processor board [106, 104, 107, 109], i386 processor and processor board [119, 116, 108, 115], and i486 processor and processor board [122, 123, 120].
  - The details of how the processor boards were integrated into the Gemini product were described in Gemini hardware reference manuals [44, 46, 45, 20, 19, 21].
  - The details of the Gemini System Controller board were described in Gemini documentation [88, 9].
  - The details of the hardware used in the PC-AT platform were described in manufacturer documentation [171, 101, 100, 102, 103].
  - The details of hardware from third-party vendors were described in vendor reference and installation manuals [1, 4, 125, 126, 129, 134, 142, 155, 156, 157, 154, 158, 159, 160, 166, 5, 172].

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the A1 Design Documentation requirement.

8.25 RAMP

Requirement

The vendor shall have in place procedures, mechanisms, tools, and personnel that comply with the Rating Maintenance Phase (RAMP) requirements for the Trusted Product Evaluation Program, as applicable to systems rated B2 and above as enumerated in `announce` transaction `[0268]`.

Applicable Features

Gemini has National Security Agency (NSA)-approved Ratings Maintenance (RM) and Configuration Management (CM) plans that detail the vendor's process and personnel used to meet the RAMP requirements specified in `announce` transaction `[0268]`. Based on the information in these plans and evaluation team-conducted audits of the procedures, the following summarizes how the RAMP requirements are met:

- The RM Plan considers as applicable all criteria and interpretations published by the NSA that (a) refer to a feature or assurance that is present in the product, and (b) was approved either prior to the most recent Evaluated Products List (EPL) date of the product or more than one calendar year prior to the submission of a Rating Maintenance Report (RMR) for the product version being considered.

The RM/CM plans contain details on the process used to comply with new interpretations as they arise.

- The RM and CM plans, in conjunction with the System Specification, detail the configuration items controlled by the RM/CM process. These items include the TCB, all system generation and testing tools, all user documentation, and the RM and CM plans. The granularity of these items is sufficient to support security analysis.
- The RM plan identifies all RAMP-responsible personnel, including the Vendor Security Analysts (VSAs) and the Responsible Corporate Officer. All Gemini VSAs have completed the NSA-required training; and the Responsible Corporate Officer (RCO) has sufficient authority to enforce decisions of the VSA.
- The RM plan identifies the original date of RM and CM plan approval, as well as providing space for identifying any approved changes to the RM and CM plans.
- The RM plan contains procedures for the VSA-performed RAMP audit, for RM-Plan maintenance, and for updates necessary in the case of RAMP process failures. The procedures for RM plan maintenance include approval of the RM plan by both the Technical Point of Contact and the NSA.
- The RM plan specifies the format of the RAMP evidience, as well as the information to be contained in the Ratings Maintenance Report prepared by the VSA. The RMR information detailed in the RM Plan includes all items called out in the B2 and above RAMP requirements.
- The RM plan identifies the security analysis to be performed on the product when changes occur. It includes both single-change analysis and cumulative-change (i.e., per RAMP cycle) analysis.
- The VSA has been submitting timely, quarterly, informal status reports to the vendor forum.
- The RM plan identifies the procedures that the VSA follows with respect to scheduling and presentations to an Future Change Review Board (FCRB).

Note that not all requirements detailed in transaction `[0268]` are addressed above. Omitted requirements fall into one of the following requirements:

- Requirements applicable to the SA-Team, not the vendor.
- Requirements that are unverifiable at the present time (for example, ensuring that something is done at a future date).
- Requirements that are not applicable because of the timing of the approval of this version of the RAMP requirements with respect to this evaluation. This evaluation entered formal prior to the approval of the B2 RAMP requirements; hence timing-related requirements with respect to the initiation of RAMP were delayed from the normal case.
- Requirements that are implicitly verified as satisfied by the approval of the RM/CM plans by NSA (for example, the signatures on the cover page).

Conclusion

Gemini Trusted Network Processor version 1.01 satisfies the RAMP requirement.

This page intentionally left blank

Section 9

Evaluation of Part II Requirements

This chapter discusses the GTNP from the point of view of the services discussed in part II of the *Trusted Network Interpretation (TNI) of the Trusted Computer System Evaluation Criteria* [137, Sections 5-9]. It begins with a brief overview of the purpose and scope of part II, and then addresses the functionality provided by the GTNP.

9.1 Background

Part II of the Trusted Network Interpretation addresses network security disjoint from Trusted Computer System Evaluation Criteria (TCSEC) interpretations discussed in section 8, pages 145-177. It addresses additional functionality that is specific to the network environment. This functionality is divided into three service groups, each containing three security services, as follows:

1. Communications Integrity.
   (a) *Authentication.* This service addresses the ability of the system to ensure that a data exchange is established with the addressed peer entity (and not with an entity attempting a masquerade or a replay of a previous establishment).
   (b) *Communications Field Integrity.* This service addresses the ability of the system to prevent the unauthorized modification of any of the fields (e.g., user-data or header fields) involved in communications. If this service is present, the network should ensure that information is accurately transmitted from source to destination.
   (c) *Non-Repudiation.* This service addresses the ability of the system to provide unforgeable proof that a message was both sent and received, thus preventing the receiver from falsely claiming that a message was never received when it actually was, in addition to preventing the sender from falsely claiming that a message was sent when it actually was not.
2. Denial Of Service
   (a) *Continuity of Operations.* This service addresses the ability of the system to provide a means to detect denial of service and notify the network manager.
   (b) *Protocol-Based Protection Mechanisms.* This service addresses the protocols provided by the system, and their robustness with respect to denial of service.
   (c) *Network Management.* This service addresses the ability of the system to provide network operational maintenance and monitoring of network health.
3. Compromise Protection
   (a) *Data Confidentiality.* This service addresses the ability of the system to protect data from unauthorized disclosure, including from passive wiretapping attacks.

(b) *Traffic Flow Confidentiality.* This service addresses the ability of the system to protect data from unauthorized disclosure through the analysis of message length, frequency, and protocol components (such as addresses).

(c) *Selective Routing.* This service addresses the ability of the system to apply rules during the process of routing so as to choose or avoid specific networks, links, or relays.

The purpose of the part II evaluation is to provide guidance to network managers and accreditors as to the reliance that can be placed in the system. This serves as input to the accreditor's decisions concerning the operational mode and range of sensitive information entrusted to the network.

9.2 Evaluation Of The GTNP Against The Part II Requirements

When delivered in the evaluated configuration, the GTNP TCB does not include any software that implements network protocols (other than hardware level protocols) or provides network services. It is intended to serve as a base upon which the desired services and network support can be built. As a result, in the evaluated configuration, the GTNP TCB does not provide any part II services. It does, however, provide underlying support for the implementation of these services.

The Trusted Computing Base (TCB) minimization requirements at Class A1 dictate that protocols and services that are not protection-critical be implemented outside of the TCB. If the additional services were included in the TCB (which is the only portion evaluated), they would either have to be protection critical (a condition not met by most part II services), or lead to a violation of the minimization requirement. The approach taken by Gemini, however, maintains the minimization while still being consistent with the introduction of part II of the TNI: "Part II services may be provided by mechanisms outside the Network Trusted Computing Base (NTCB)." Note that services provided outside the M-Component NTCB boundary are not covered by this evaluation.

The ability to provide support for these services is primarily due to three devices available on the Gemini System Controller boards: the Data Encryption Device, the Data Sealing Device, and the Key Management Device.[1] The services are also supported by the Mandatory Access Control (MAC) architecture provided by the NTCB MAC partition and the hierarchical domains provided by the ring mechanism. The relationship between this support and the services is as follows:

- The *Data Encryption Device* provides the ability to encrypt and decrypt data using any of the three Data Encryption Standard (DES) cipher modes defined by the National Institute of Standards and Technology: Electronic Code Book, Cipher Block Chaining, and Cipher Feedback. It conceivably could be used to support the following part II services: authentication, network management (distribution), data confidentiality, and traffic-flow confidentiality.

- The *Data Sealing Device* provides the ability to generate cryptographic checksums for data, again using the DES algorithm. It conceivably could be used to support the following part II services: authentication, communications field integrity, and network management.

[1] More information on these devices can be found in section 4.2.1.8.2, page 75 and in *Appendices to the Software Development Specification for GEMSOS Security Kernel* [30, Appendices A29 (Data Encryption), A35 (Data Sealing), and A28 (Key Management)].

- The *Key Management Device* provides the ability to generate and load keys intended for use by the data encryption and data sealing devices. The actual keys are maintained in the System Programmable Read Only Memory (PROM); users outside the TCB only manage encrypted representations of the key. This device could conceivably be used to support the following part II services: authentication, communications field integrity, network management (distribution), data confidentiality, and traffic flow confidentiality.
- The *Mandatory Access Control facilities* provide support for multiple hierarchical labels and compartments, allowing implementation of an integrity policy with a portion of the access label. This supports both separation of roles (via integrity levels) and isolation of domains (via integrity compartments). This could conceivably be used to protect any part II services implemented as virtual machines above the NTCB MAC partition.
- The *Hierarchical Domains* (ring) support provides support for hierarchical structuring of virtual machines implemented above the NTCB MAC partition. These virtual machines could conceivably be responsible for providing part II services. The ring mechanism would then serve to isolate the part II services from interference from higher level untrusted subjects.

Note that, to the extent that one believes "community keys" are not compromised, the GTNP could support non-repudiation within a community through the Data Sealing, Data Encryption, and Key Management devices. The key loading operation would include a "signature" of the source system transmitting the key. This would provide assurance that a particular composite key originated at a specific system. Thus, the origin of a composite key could be verified.

If a composite key is generated as a function of a message, then the origin of the message can be verified. This would be done by sealing the message using a public key and using this seal as the key ID when composing or loading the composite key.

Non-repudiation would then be implemented through a protocol that would prevent a sender from denying having sent a message; and would prevent a receiver from denying having received a message. The procedures on how this would be implemented are described in Gemini documentation [30, Appendix 28, Key Management Device].

This page intentionally left blank

Section 10

Evaluator's Comments

10.1 Cooperative and Knowledgeable Vendor

The benefits of working with a cooperative and knowledgeable vendor cannot be underestimated. The relationship between Gemini and the evaluation team has truly been that of a *team*, working together towards a common goal. The team commends the responsiveness of the vendor during entire GTNP evaluation. Although at times there were intense discussions over interpretations of requirements, the team feels these discussions led to an evaluation that stayed focused on the requirements without expanding their scope.

10.2 Multilevel Processes

Although the evaluation addressed only single-level application subjects, the GTNP does have the capability to define multilevel subjects. Defining multilevel subjects would be an alteration of the GTNP M-Component partition; as such, Section B.3 of the Trusted Database Interpretation [139] (TDI) should be consulted. However, the evaluation team feels that *in this specific case*, the GEMSOS Security Kernel would properly restrict such subjects to operating within their range (although no specific team tests were performed with respect to this, multilevel subjects are covered by the formal model and specifications, and are tested by the vendor test suite). Furthermore, the evaluation team feels that the existing M-Component Trusted Computing Base (TCB) would remain tamperproof and unbypassable, for the following reasons:

- The M-Component kernel executes at PL0. No multilevel process can execute at PL0. This prevents access to in-memory kernel data structures and requires all accesses to go through the kernel memory and device interfaces.

- All kernel static data structures (on secondary storage) have ring brackets restricting access to R0. GEMSOS Network Processor System Parent (NPSP) cannot create a R0 process; hence, no subject created by NPSP can access kernel data structures.

- NPSP executes at PL1/R1, and its data structures have ring brackets restricting access to R1. If the multilevel processes created by NPSP are in a ring greater than one (as is enforced by the evaluated configuration of NPSP), then said multilevel processes cannot access NPSP operational or static data structures.

  If NPSP is used in a non-evaluated configuration, or if the NPSP is bypassed, then there is no architectural protection for NPSP (although if the multilevel subject is in R1 or greater, the protection of the kernel remains). However, the NPSP only enforces policy with respect to the creation of initial application processes; the overall system security policy is enforced by the kernel.

Note that the term "single-level" is with respect to that portion of the Mandatory Access Control (MAC) policy allocated to the GTNP's Mandatory Network Trusted Computing Base (M-NTCB) partition. If, for example, the MAC policy allocated to the GTNP M-NTCB partition recognized secrecy levels but not secrecy categories, then a subject having a single secrecy level but a range of categories would be a single-level subject. This is significant in the determination of the extent of reevaluation required. In particular, defining subjects that are multilevel with respect to that portion of the network's MAC policy that is allocated to the GTNP M-NTCB partition would require more extensive analysis than the definition of subjects that are not multilevel with respect to the M-NTCB policy.

The extent of reevaluation of the GTNP M-Component would depend on the particular policy enforced by the multilevel subject. If the policy enforced by the multilevel subject is different from that of the M-Component, detailed examination of the M-Component should not be necessary. A new Formal Top-Level Specification (FTLS) would be required for the multilevel subject, along with a revised covert channel analysis. If the policy enforced by the multilevel subject is the same as the policy enforced by the M-Component partition, then the amount of work required is variable and, as the TDI notes, cannot be generalized.

It should be noted that the Gemini test suite for the GTNP, as part of the testing, verified the following:

1. The kernel properly enforces the ring integrity policy; thus, kernel data is protected in R0.
2. The kernel properly restricts multilevel subjects to operating within the defined range of the subject.
3. The NPSP properly creates multilevel subjects with their specified ranges.

In any case, the team feels that it is a very flexible mechanism that can be used to implement a wide variety of trusted applications.

10.3 Multilevel Devices

When configured in accordance with the GTNP Trusted Facility Manual (TFM), the evaluated configuration provides only single-level devices. However, the software does provide the capability for the addition of multilevel devices. This can be done in two ways:

1. A multilevel device could be added as a new device driver in the kernel. This would require the addition of a new device driver to the Inner Device Manager layer (and, potentially, changes to the Core Manager and Kernel Device layers to handle interrupts). This new device driver would have to have the capability to understand labels. Such an addition would also be an alteration of the GTNP M-Component Network Trusted Computing Base (NTCB) partition. This driver would need to be examined to verify its behavior. The evaluation team does feel that the remainder of the kernel would properly enforce information flows to the multilevel device; however, the analysis of such a driver by an appropriately qualified technical agency would be required.
2. A multilevel device could be added by building it as a virtual machine on top of the virtual machine monitor provided by the GTNP. This virtual machine would have to be multilevel; hence, this would require extension of the M-NTCB. By taking advantage of the fact that the kernel and NPSP provide assured integrity for data between a communication port and an application process, a multilevel application process could be written to interpret the contents of the data (for example, using the

Internet Protocol (IP) security option) and assign appropriate labels or start appropriate single-level untrusted protocol handlers to process a particular protocol stream. This multilevel application process would be subject to the analysis discussed in section 10.2, page 183. From the point of the GTNP, the device being used in this fashion would still be a single-level device, labeled with a range that specified the write and read labels of potential traffic. This device range would have to be encompassed by that of the multilevel process that is interpreting the labels.

Note that both approaches depend on the integrity of the data stream to the device. If an untrusted device, such as an unevaluated router, were to be placed between the network terminus and the GTNP, any label information in a message would be potentially subject to compromise or modification as it passed through the untrusted device. At that point, the multilevel device driver could not use information in that message to determine the ultimate label for that information.

10.4 Side-Effects of Compatibility Property

Due to the requirement that an application subject must have observe and modify access to the mentor in order to delete an observable segment, upgraded segments can be created but not deleted by untrusted application subjects in the evaluated system (which only supports single-level subjects). This has the potential for creating unremovable (except by regeneration or via system generation utilities) naming hierarchies. This is discussed in the Security Features Users Guide (SFUG).

10.5 Integrity Model Support

The GTNP documentation claims that the GTNP supports both a Bell-La Padula-style sensitivity model [2], and a Biba-style integrity model [3]. From one aspect, this is true; from another, it is not.

From the point of view of the kernel software, the model is one that supports two hierarchical levels and one set of category bits, all of which are compared bitwise. If an application supported a Biba-style model, it would have to invert the sense of the label for the hierarchical level (i.e., zero would be the highest integrity level), and invert the bit meanings for categories (i.e., zero indicates the presence of a category). On the other hand, a creative application could manipulate the label structure to support some other form of non-discretionary model. The team feels that flexibility is one of the GTNP's strengths.

However, from the point of view of the delivered system, a Biba-style integrity model is built-in. This is because the system generation utilities that define the labels automatically perform the necessary inversions. An application administrator could still support a non-Biba model with the extra labels and categories; however, they would have to know how to undo the inversions when configuring the system. This may be confusing, but the team has no idea how likely it is to be done in practice.

10.6 Support for Trusted Path in Virtual Machine Network Components

The GTNP, when operating in the evaluated configuration, does not support direct user input; hence, there is no need for trusted path. However, the GTNP does support terminal devices [Gemini local serial ports on the 80286, i386, and i486 hardware (LSIO, SSIO), Extended Serial I/O Boards (ESIO), the IBM/PC Keyboard device and serial I/O ports (ASIO)]; it also provides the ability to implement other network partitions as virtual machines in layers above the M-component partition. These other partitions may support direct user interaction; hence, the GTNP must provide mechanisms to allow these other components to provide trusted path, if necessary.

As described in section 4.2.1.8.2, page 75, all devices that could support keyboard input appear to provide mechanisms that can be used to support a form of a secure attention key:

- All serial I/O devices allow detection of the <BREAK> key; all but the SSIO device allow detection of loss of carrier.
- The IBM PC-AT Keyboard devices allows establishment of a user-definable key sequence for secure attention key.

Other component layers, when propagating the kernel interface, can force the appropriate modes to be set on these devices and can easily intercept the trusted path activation indications. Thus, the GTNP appears to provide the necessary mechanisms to allow virtual machines implemented on top of the M-component virtual machine monitor to satisfy the trusted path requirement.

10.7 Support for Subject Sensitivity Labels in Virtual Machine Network Components

The GTNP, when operating in the evaluated configuration, does not support interactive users; hence, there is no need for the GTNP M-NTCB to provide a mechanism whereby terminal users are able to query the TCB for a display of the subject's complete sensitivity label. However, the PL1 subject in a process is always aware of the sensitivity labels of all subjects in the process; an application could be developed to support reporting of that label to users. This reporting would be performed by a NTCB partition operating above the virtual machine interface provided by the M-NTCB partition.

10.8 Random Value Generator Risks

Gemini has brought to the team's attention a possible flaw concerning the use of the Random Value Generator device. Depending on how much one believes in Data Encryption Standard (DES) and the secrecy of the key (i.e., how much one believes the next random value is not predictable), the result of a call to this device may reflect the actions of another user. This "flaw"[1] is removable through a configuration option requiring

[1] Gemini places it in this category since they do not consider it a covert storage channel signalled through error codes.

that the caller pass a read-write security check on attaching the device (the subject range encloses the range of the device); alternatively the administrators may decide they believe in DES and the secrecy of the key, in which case they may choose to configure the device to only require a read type of security check (the subject read class dominates the read class of the device).

10.9 Effective Use of Hardware

The team would like to applaud the effectiveness with which Gemini has used their available hardware resources. Examples of effective use include: multiplexing three of the hardware privilege levels into eight rings, using the remaining privilege level to do so; and using many segments inside the kernel layers to separate code and data structures by modes of access.

10.10 Mapping FTLS to Model

The demonstration that the FTLS is consistent with the model was carried out with an unusually high degree of thoroughness, care, and automated support.

10.11 Test Suite and Scripting Language

Gemini has developed an extensive scripting language to support testing that provides an environment for test development that greatly reduces the need to use the Pascal compiler directly. This approach makes the test suite easier to understand, and simplifies the process of the development of tests.

10.12 Assignment of Processes to Processors

In the GTNP, the assignment of processes to physical processors is static. The relationship between a given process (and its descendants) and a given physical processor is established as part of system configuration, and cannot be changed once the system is booted. This is in contrast to other multiprocessor systems that perform dynamic assignment and load shifting.

The function of load shifting is not security-relevant; minimization of the security kernel directs that this functionality be implemented outside of the kernel, as it is not protection-critical. Rather, load balancing is an operating system function; in the GTNP, operating systems are implemented in non-TCB (with respect to the M-Component security policy) domains above the kernel.

The static assignment approach taken by Gemini has its strong points and its weak points. It allows a system architect to ensure an equitable loading of processes on a system, and to isolate processes so as to reduce global memory accesses. It allows localization of functions. It is, however, different from the approach taken in some other systems. The team felt that this difference was worth highlighting.

10.13 Usefulness of FTLS and Code Correspondence

The evaluation team would like to recognize an unanticipated benefit of the GTNP Formal Top-Level Specification (FTLS) and Code Correspondence. Gemini developed the FTLS and Code Correspondence with an obvious goal of having it be easily readable and understandable. As a result, the team found the documents invaluable during the Penetration Testing effort. In particular, the documents served as an easy cross-reference to locating security-relevant source files and lines of code. If a particular kernel call required analysis, the FTLS/Code Correspondence could be used to easily identify the source files used to implement that kernel call, the relevant data structures and where they were declared, and the significant lines of code in the source file.

Similarly, if the team needed to verify that a particular security-relevant test was implemented correctly, the team turned first to the FTLS/Code Correspondence. If it was implemented, it was visible in the specification (being security-relevant). The code correspondence identified the specific lines and source files involved in performing the test. The tester could then easily verify that the test was correct, and that the lines of code correctly implemented the test.

The usefulness of the FTLS/Code Correspondence as a super-cross-reference was invaluable to the team during testing and analysis.

10.14 Denial of Service Threats

During penetration testing, the evaluation team investigated a number of implementation flaws in the commercial Intel processors, based on information available in public literature [169, 165]. The team discovered that some specific steps of the i386 processor provide unprivileged subjects, through the use of non-privileged processor instructions, the unrestricted and unrestrictable ability to effectively halt the operation of GTNP. These flaws apply specifically to the i386 steps A0-B1. The team was unable to verify these claims, as the specific steps were not available for testing. Further, Gemini does not have the ability to identify if a customer has a specific step of the i386; nor is detection of the specific step easily performed, unless the enclosure is opened and the chip examined. As these are early production steps, the team does not feel the problem is extremely significant. However, if a GTNP i386 system obtained shortly after the i386 chip was released is to be used in an environment where denial of service is a concern, it is worth working with the vendor to determine if the steps in question are used in the particular system.

Additionally, the team confirmed that there are two undefined floating point opcodes that force the kernel into the unrecoverable timeout error. The team did not identify the i386 step from which this behavior was observed. Gemini is aware of the specific opcodes that trigger this behaviour, and the problem does not occur on other processors in the evaluated configuration.

Appendix A
Evaluated Hardware Configuration

The following sections summarize the hardware bases in the GTNP evaluated configuration. They are divided into Trusted Bases and PC-AT platforms.

A.1 Trusted Base

This section summarizes the Gemini Multibus system bases, including models 6, 9, 12, 15, and 26.

A.1.1 Base Configurations

Trusted Bases are available in (non-volatile) Random Access Memory (RAM)-based, floppy disk drive-based, and hard disk drive-based configurations. Each system contains at least one of the following common Multibus-compatible boards:

- Gemini System Controller (GSC) board (one only)
- Standard (80286-based), HP (80286-based), SP (i386-based), or UP (i486-based) processor
- Serial Input/Output (I/O) ports
  - Standard, HP, or UP processor: 2.
  - SP processor: 1.
- Local Memory
  - Standard or HP processor: 1 Mbyte.
  - HP processor: 2 Mbytes.
  - UP processor: 8 Mbytes.
- Global memory: 1/2 Mbyte.

The model number of a Trusted Base system identifies the type of secondary storage, the expansion capability, the model series, the number of processors, and the processor performance, as follows:

SnnA-mmXX

where:

S is the secondary storage identification:

- `R` *RAM-Based.* These require the purchage of non-volatile memory for the loading of GEMSOS and customer applications.
- `F` *Floppy Diskette Based.* These provide a single floppy diskette drive.
- `W` *Winchester (hard disk) Based.* These provide one floppy diskette drive and one 100 Mbyte hard disk.

nn is the expansion capability:

- `6` or `9` One processor.
- `12` Up to two processors.
- `15` Up to three processors.
- `26` Up to eight processors.

A is the series:

- ␣ *Original series Trusted Bases*, which are no longer available.[1] Original series models use the Storager disk interface (ST506).
- `A` *A series Trusted Bases.* This series uses the Storager III interface (ESDI).

mm The number of processors originally installed.

XX The designation for the fastest processor in the base:

- ␣ *Standard.* This uses the iSBC 286/10 and 286/10A Single Board Computers (SBCs). A math coprocessor is optional for original series standard processors.
- `HP` This uses the other iSBC 286 models in the evaluated configuration (see appendix A.1.3, page A-3). A math coprocessor is optional for the original series HP processors.
- `SP` This uses the iSBC 386 models in the evaluated configuration (see appendix A.1.3, page A-3).
- `UP` This uses the iSBC 486 models in the evaluated configuration (see appendix A.1.3, page A-3).

A.1.2 Gemini System Controllers for the Trusted Base

The following versions of the GSC are supported:

Model Number	Version Number	Revision ID
GSC-IV rev C	07 (domestic)	03
GSC-IV rev C	08 (export)	03
GSC-V rev A	10 (domestic)	01
GSC-V rev A	11 (export)	01

Note: If a GTNP system contains multiple processors, Gemini recommends the use of the GSC V board. The GSC IV board does not allow closing of one of the high-bandwidth covert timing channels.

[1] However, they are in use by Gemini customers.

A.1.3 Processor Types

The Trusted Base supports the following processors. Each processor contains its own boot PROM, providing code and data to boot the GTNP.[2]

1. Standard Processor
   - iSBC 286/10 Single Board Computer
     (80286/6 processor. Requires 1 Mbyte memory board.)
   - iSBC 286/10A Single Board Computer
     (80286/8 processor. Requires 1 Mbyte memory board.)
2. HP Processor
   - iSBC 286/12 Single Board Computer
     (80286/8 processor. On-board memory: 1 Mbyte.)
   - iSBC 286/14 Single Board Computer
     (80286/8 processor. On-board memory: 2 Mbytes.)
   - iSBC 286/16 Single Board Computer
     (80286/8 processor. On-board memory: 4 Mbytes.)
3. SP Processor
   - iSBC 386/22 Single Board Computer
     (i386DX/16 processor. On-board memory: 2 Mbytes.)
   - iSBC 386/24 Single Board Computer
     (i386DX/16 processor. On-board memory: 4 Mbytes.)
   - iSBC 386/28 Single Board Computer
     (i386DX/16 processor. On-board memory: 8 Mbytes.)
   - iSBC 386/32 Single Board Computer
     (i386DX/20 processor. On-board memory: 2 Mbytes.)
   - iSBC 386/34 Single Board Computer
     (i386DX/20 processor. On-board memory: 4 Mbytes.)
   - iSBC 386/38 Single Board Computer
     (i386DX/20 processor. On-board memory: 8 Mbytes.)
4. UP Processor
   - iSBC 486/12 Single Board Computer
     (i486DX2/66 processor. On-board memory: 8 Mbytes.)
5. Math Coprocessor: An Intel287 math coprocessor that provides an extension to the hardware instruction set. The Intel387 coprocessor is standard on the SP processors and all A-series models. It is optional on the original series standard and HP models. The i486DX2/66 processor used in the UP processors contains an integral Intel487 math coprocessor.

[2]Note: Selected steps of the i386 processor have known implementation errors that create the potential for denial of service attacks. Additional information regarding these steps may be found in section 10.14, page 188.

A.1.4 Optional Equipment

The following sections detail the optional Multibus boards that can be added to expand the capabilities of the base system.

A.1.4.1 Secondary Storage

The model 9, 12, and 15 Gemini Trusted Bases (excluding RAM-based) can support up to the following quantities of secondary storage drives in various combinations:

- Two half-height, 5 1/4″ removable media drives
  - Floppy disk drives (maximum of two)
  - Quarter-Inch Cartridge (QIC) streaming tape drive (maximum of one)
- Two half-height or full-height, 5 1/4″ hard disk drives
  - ST412/ST506 hard disk drives (with Storager I or II; maximum of two)
  - Enhanced Small Device Interface (ESDI) hard disk drives (with Storager III; maximum of two)

The model 26 Gemini Trusted Base (excluding RAM-based) can support up to the following quantities of secondary storage drives:

- Three half-height, 5 1/4″ removable media drives
  - Floppy disk drives (maximum of two)
  - Quarter-Inch Compatible streaming tape drives (maximum of two)
- Two half-height or full-height, 5 1/4″ hard disk drives
  - ST412/ST506 hard disk drives (with Storager I, II, or II-2; maximum of two)
  - ESDI hard disk drives (with Storager III; maximum of two)
- Two 1/2″ tape drives, 9-track

The floppy disk, hard disk, and QIC streaming tape drives are controlled by a single peripheral device controller – Storager I, II, II-2, or III. The 1/2″ tape drive requires a separate tape controller, Tapemaster 1000. See appendix A.3, page A-8 for a list of supported drives.

A.1.4.2 Memory Expansion

Memory boards can be added to the Gemini Trusted Base to provide additional local or global RAM in the increments described below.

1. Volatile RAM: 512 Kbytes (iSBC 012CX); 1 Mbyte (iSBC 010CX); or 2 Mbytes (iSBC 020CX).

2. Non-volatile RAM (Global memory only): 256 Kbytes; 512 Kbytes; or 1 Mbyte.

The following list describes the memory limits within the GTNP:

1. Local Memory
   - Standard Central Processing Unit (CPU): 2 Mbytes.
   - HP CPU: 6 Mbytes. Some Model 26 systems have a maximum of 8 Mbytes.
   - SP CPU: 8 Mbytes.
   - UP CPU: 16 Mbytes.
2. Global Memory
   - The global memory is expandable to 8 Mbytes with additional RAM boards. However, less than 8 Mbytes are available when 80286 processors are used. Note that memory-mapped devices are allocated part of the available 8-Mbyte address space.

Memory expansion may limit the number of processors and other hardware options available in particular configurations.

A.1.4.3 Communication Devices

The following types of communication options can be added to the Trusted Base, depending on bus slot availability:

1. Extended Serial I/O (ESIO) ports
   - The Trusted Bases can support up to 8 ESIO boards (iSBC 188/56), each of which provide 8 RS-232 ports, for a maximum of 64 ESIO ports.
2. Network Controller Base with Expandable Ethernet Interface
   - The Trusted Bases can support up to 16 Expandable Ethernet Interfaces. Each interface is implemented using a single Intel iSBC 186/51, which may also be used to simultaneously implement a single Expandable High-level Data Link Control interface. Each frame-level Ethernet interface (compliant with IEEE 802.3) provides a minimum of 1 megabit per second throughput.
3. Network Controller Base with Expandable High-level Data Link Control (HDLC) Interface
   - The Trusted Bases can support up to 16 Expandable High-level Data Link Control Interfaces. Each interface is implemented using a single Intel iSBC 186/51, which may also be used to simultaneously implement a single Expandable Ethernet interface. Each interface supports HDLC LAPB and reliably provides a minimum of 64 kilobits per second throughput.
4. Network Controller Base with Expandable HDLC and Expandable Ethernet Interfaces.
   - This combines the previous two options.

Selection of hardware options may limit the extent to which memory or number of processors can be expanded in particular configurations.

A.2 PC-AT Platform

This section summarizes the Gemini Multilevel Secure PC-AT Workstation Kit.

A.2.1 Standard PC-AT Compatible Equipment

The PC-AT platforms include the following common equipment:

- Selected Manufacturer's PC/AT compatible model with the following required equipment:
  - Enhanced Graphics Adapter (EGA) video controller with 64 Kbytes of memory.
  - Hard/floppy disk controller (ST412/ST506)
  - Standard memory: 512 Kbytes or 640 Kbytes
  - Memory expansion board with at least 1 Mbyte of extended memory
  - Keyboard: 84- or 101-key keyboard
  - Floppy disk drive
- Gemini System Controller for PC-AT (GSC-AT) board
- GTNP Kernel Boot PROMs
- Intel80287 math coprocessor (optional)

A.2.2 Gemini PC-AT Workstation Kit Model Numbers

There are a number of PC-AT workstation kit models, distinguished by the amount of user ROM available on the GSC for use as read-only volumes.

Model	User ROM
GSC-AT A0	no user ROM
GSC-AT A1	no user ROM
GSC-AT A2	256 Kbytes
GSC-AT A5	512 Kbytes
GSC-AT A10	1 Mbyte
GSC-AT A20	2 Mbytes

A.2.3 Gemini System Controllers for PC-AT Compatibles (GSC-AT)

The following versions of the GSC-AT are supported:

Model Number	Version Number	Revision ID
GSC-AT rev B	06 (domestic)	03
GSC-AT rev B	09 (export)	03

A.2.4 Manufacturer PC/AT Compatible Model Numbers

The following specific PC/AT and compatible models are supported:

1. IBM PC/AT Models
   (a) 5170, all other submodels (requires 32 Kbytes in Boot PROM) except 495, 496, 599, 739, 839, 899, 919, and 939.
   (b) 4459 TEMPEST PC (5170 TEMPEST equivalent) (requires 32 Kbytes in Boot PROM)
2. Zenith Z-248 Models
   (a) ZBF-2337-BK (requires 64 Kbytes in Boot PROM)
   (b) ZBF-2337-BK TEMPEST equivalent (requires 64 Kbytes in Boot PROM)

A.2.5 Optional Equipment

The following sections describe the various optional equipment that can be added to the PC-AT platforms.

A.2.5.1 Secondary Storage

The following secondary storage devices are supported:

- Up to two floppy disk drives
- Two hard disk drives

See appendix A.3, page A-8 for a list of supported drives.

A.2.5.2 Memory Expansion

Standard processor memory is composed of 512 or 640 Kbytes,[3] along with a combination of the following boards to provide up to an additional 8 Mbytes.

- Elephant-12 board: This board can support up to 8 Mbytes.
- Intel Above Board Family: This family of boards contain models that support a maximum of 3 1/2, 4, and 8 Mbytes per board depending on the model. The supported models are: PS/AT, 286, PS/286, Plus, Plus I/O, Plus 8, and Plus 8 I/O.
- Zenith Memory Board: This board is usable with Zenith Z-248 models only, and allows up to an additional 1 1/8 Mbytes. The 1/8 Mbyte portion of the memory is used to expand the processor memory from 512 Kbytes to 640 Kbytes. The 1 Mbyte portion is used as extended (100000_{16} to $CFFFFF_{16}$) memory.

[3] It should be noted that the IBM PC/AT 128 Kbyte Memory Expansion Option can be used to fill the difference between 512 and 640 Kbytes.

A.2.5.3 Communication Device Options

Additional I/O ports may be added to the standard equipment in two ways. A maximum of two serial I/O ports and one parallel I/O port is supported.

1. The addition of the Intel Above Board PS/AT, PS/286, Plus I/O, or Plus 8 I/O provides both a serial and parallel I/O port.
2. The addition of an IBM Serial/Parallel Adapter card, or compatible card, provides a parallel and serial I/O port.
3. Zenith systems come with a serial and a parallel port as standard equipment.

A.2.5.4 Math Coprocessor

An Intel287 math coprocessor is supported.

A.3 Secondary Storage Devices

The following lists identify all of the supported secondary storage devices. The floppy disk, hard disk, and streaming tape (not supported on the PC-AT platform) drives are controlled by a single peripheral device controller. The 1/2″ tape drive requires a Tapemaster 1000 tape controller. A Storager I, II, or II-2 (Multibus) or a hard/floppy disk controller (PC-AT) is required for ST412/ST506 hard disk drives. The ESDI hard disk drives require the Storager III disk controller. ESDI and ST412/ST506 hard disks cannot be combined in one configuration.

- Trusted Base
  - Maxtor XT-1085 85-Mbyte ST412/ST506 hard disk drive
  - Maxtor XT-1140 140-Mbyte ST412/ST506 hard disk drive
  - Newbury NDR-1140 140-Mbyte ST412/ST506 hard disk drive
  - Sequel (Maxtor) XT-4170E 106-Mbyte ESDI hard disk drive
  - Seagate Wren 3 100-Mbyte ESDI hard disk drive
  - Seagate Wren 6 380-Mbyte ESDI hard disk drive
  - Seagate Wren 6 760-Mbyte ESDI hard disk drive
  - Epson SD-680L 1.2-Mbyte floppy disk drive
  - Epson SD-580L 1.2-Mbyte floppy disk drive
  - Epson SD-621L 360-Kbyte floppy disk drive
  - Tandberg TDC 3630 120-Mbyte 1/4″ streaming tape drive
  - Tandberg TDC 3650 120-Mbyte 1/4″ streaming tape drive
  - Kennedy 9600 9-track 1/2″ tape drive

- PC-AT Platform
  - Seagate ST4026 20-Mbyte ST412/ST506 hard disk drive
  - Maxtor XT-1085 85-Mbyte ST412/ST506 hard disk drive
  - Maxtor XT-1140 140-Mbyte ST412/ST506 hard disk drive
  - Epson SD-680L 1.2-Mbyte floppy disk drive
  - Epson SD-580L 1.2-Mbyte floppy disk drive
  - Epson SD-621L 360-Kbyte floppy disk drive

This page intentionally left blank

Appendix B
Evaluated Software Configuration

Gemini markets the GTNP using a "system" approach; that is, customers order GTNP systems that include a specific set of hardware and software components bundled together. These orders do not specify a specific version number of a system; instead, they indicate whether or not they want the evaluated version.

The software that is delivered with an evaluated GTNP system is as follows [13, 32]:

GTNP Trusted Computing Base

Version number 1.01. The GTNP TCB includes the following:

Kernel *GEMSOS/GTNP Security Kernel*, Version number 2.01. The particular configuration of the kernel is based on the hardware to be supported in the system ordered. The following kernel configurations have been evaluated:

- 0007 Domestic PC/AT.
- 0029 Domestic Trusted Base (Standard, HP, SP, or UP) with Extended Serial I/O (ESIO), Expanded Ethernet/High-Level Data-Link Control (EETH/EHDLC), Streaming Tape, Half-Inch Tape (HIT).
- 0031 Domestic Trusted Base (Standard, HP, or UP) with EETH/EHDLC.
- 0032 Domestic Trusted Base (Standard, HP, or UP) with EETH/EHDLC, Streaming Tape.
- 0107 Export PC/AT.
- 0126 Export Trusted Base (Standard, HP, SP, or UP).
- 0127 Export Trusted Base (Standard, HP, SP, UP, or PC/AT) with ESIO, EETH/EHDLC, Streaming Tape, HIT.
- 0128 Export Trusted Base (SP) with ESIO, EETH/EHDLC, Streaming Tape.
- 0129 Export Trusted Base (Standard, HP, SP, or UP) with ESIO, EETH/EHDLC, Streaming Tape.

NPSP *GEMSOS Network Processor System Parent*, Version number 1.00e. This component, as delivered, also includes the following:

- *KGL* *GEMSOS Kernel Gate Library*, Version number 1.3.
- *MIT* *GEMSOS Multilevel Subject Intersegment Linkage Tool*, Version number 1.00b.

System Generation Software

The following software is also included as part of the evaluated configuration. It is not executed as part of the TCB; it is security relevant as it is responsible for bringing the GTNP to secure initial state.

GKI	*GEMSOS Security Kernel Initialization*, Version number 2.01. GKI is delivered as part of the kernel, although it is not accessible once the system completes initialization.
SYSGEN	*GEMSOS System Generation Utility*, Version number 2.03. Note that SYSGEN is not included with RAM-based or floppy disk-based systems. It may, however, be ordered separately as part of the optional Software Development Environment.
NPCT	*GTNP Configuration Tools*, version number 2.00a.
DSKGEN	*Hard Disk Conditioning Utility*, Version number 2.02b.
OUI	*Original User Installation*, Version number 1.00.
HDU	*GEMSOS Hardware Distribution Utility*, Version number 1.01b.
SPBU	*System PROM Bootstrap Utility*, Version number 1.01b.
SMU	*System Maintenance Utility*, Version number 1.04.

Unevaluated Non-TCB Software

The following unevaluated, non-TCB software is also shipped with an evaluated GTNP system. All of this software runs above the GTNP TCB:

APGEN *GEMSOS Application Generator Utility*. APGEN is an application generator used to generate application code onto the segment hierarchy.

MAE *Model Application Environment*. MAE is a set of libraries and modules that serves as an example of how to develop applications on the GTNP. It is not supported as a product; however, the licensee is provided with and may retain source code to the MAE.

GTC/GTAT *Gemini Test Controller/Acceptance Tests*. These are a set of acceptance tests for the customer to run.

PDS *Product Distribution Support*. A collection of SYSGEN command segments that support the distribution and installation of Gemini systems.

KIT *Kernel Initialization Tables*. A collection of all kernel parameter tables and initial process parameter tables that are delivered to customers as part of Gemini systems, as well as examples of parameter tables that are appropriate of use in configuring the GTNP TCB.

Note that all of the unevaluated software listed above is configuration managed by Gemini. Furthermore, MAE, APGEN, and GTAT/GTC are used by the Gemini functional security test package; as such, they are covered by RAMP. Both PDS and KIT may be used by a user to simplify the installation process; however, their use is not mandatory.

Appendix C
EPL Entry

[0394] (170 lines) McBride.CPE 11/21/94 1101.4 edt Mon Eval_Announcements
Subject: Gemini Computers, Inc., Gemini Trusted Network Processor A1 EPL Entry
Serial No. CSC-EPL-94/008

EVALUATED PRODUCT:	Gemini Trusted Network Processor (GTNP)
VENDOR:	Gemini Computers, Incorporated P. O. Box 222417 Carmel CA 93922 +1 408 373-8500 +1 408 373-5792 FAX
VERSIONS:	GTNP Version 1.01, consisting of: (a) GEMSOS Security Kernel Version 2.01, configurations 0007, 0029, 0031, 0032, 0107, 0126, 0127, 0128, or 0129. (b) GEMSOS Network Processor System Parent Version 1.00e. (c) GEMSOS Kernel Gate Library Version 1.03. (d) GEMSOS Multilevel Subject Intersegment Linkage Tool Version 1.00b. Running on either an evaluated Gemini Trusted Base or an evaluated Gemini PC Workstation Kit for selected PC compatibles.
EVALUATION DATE:	06 September 1994
OVERALL EVALUATION CLASS:	A1 Mandatory-Only Network Component

PRODUCT DESCRIPTION:

The Gemini Trusted Network Processor (GTNP) provides a Mandatory Network Trusted Computing Base (M-NTCB) for network components that implement a Mandatory Access Control (MAC) policy. In addition to providing multilevel security, the GTNP provides Data Encryption Standard (DES) encryption and

concurrent processing. The GTNP is designed to support integration with other technologies and products to build a variety of secure network interconnection and secure data-sharing components for multilevel secure (MLS) and/or multiple security level (MSL) heterogeneous distributed information systems. In these systems, the GTNP provides the trusted underlying foundation upon which applications and protocols are built. Taking an open-architecture approach, users of the GTNP are not restricted to a single application or suite of protocols; rather, applications and protocols may be developed to run on top of the structured base provided by the GTNP to support the specific network requirements or network component composition, without affecting the M-NTCB.

The GTNP is based on the Gemini Multiprocessing Secure Operating System (GEMSOS) Security Kernel (Kernel). The Kernel is a mandatory security reference monitor that enforces a lattice-based MAC policy for both secrecy (confidentiality) based on the Bell-LaPadula model, and integrity based on the Biba model. Support is provided for up to 16 secrecy levels, 64 secrecy categories, 16 integrity levels, and 32 integrity categories. The Kernel implements real time, priority-based scheduling to provide multiprogramming and multiprocessing to support concurrent computing including parallel and pipeline processing.

The GTNP uses DES encryption and cryptographic checksum mechanisms, together with a trusted key management algorithm and other controls to implement trusted distribution. It insures that critical software and hardware are controlled with a high level of assurance throughout their life-cycle.

The GTNP hardware includes a tightly coupled multiprocessing architecture that supports up to eight Intel 80x86 processor boards (a mixture of i486, i386, and 80286 processors), memory boards, and device interface boards on the IEEE standard 796 system bus, Multibus I. The PC bus supports single processor GTNP configurations. All evaluated configurations utilize the Gemini System Controller board, which provides the hardware DES encryption device and bus-arbitration circuitry.

The GTNP supports network interfaces for local area networks using Ethernet and RS-232, and X.25 wide area networks using High-Level Data Link Control (HDLC) protocol; a preallocated address interface (Multibus I); and a virtual machine monitor interface. GTNP Version 1.01 is designed to support single-level network protocols outside the TCB without affecting the evaluation rating of the GTNP.

The GTNP software architecture is implemented on the 80x86 multi-state machine consisting of four hierarchical hardware enforced privilege levels. The Kernel is in the highest privileged level (PL0). The GTNP uses the remaining three privilege levels (PL1, PL2, PL3) to implement eight hierarchical rings that can be used to separate processes into different

domains. The hierarchical architecture can be used to implement other Network Trusted Computing Base (NTCB) components defined in the TNI, enforcing different security and supporting policies.

Trusted GTNP software is implemented within the lower 5 (more privileged) rings and the Kernel. Applications software is implemented in the higher 3 (less privileged) rings to facilitate effective evaluation of new applications and minimize re-evaluation of the GTNP.

The GTNP supports two types of composite structures: the first is based on the NTCB paradigm of the TNI; the second is based on the TCB Subset paradigm of the TDI. Using these composite structures, enforcement of the overall network security policy may be allocated to the various NTCB components or TCB subsets, with the MAC policy enforcement allocated to the GTNP (and, if necessary, other M-components). Other policies can then be enforced by proper integration of other products with the GTNP.

Within the NTCB paradigm, a secure distributed system with a coherent Network Security Architecture and Design would be composed of multi-vendor network components which can be evaluated in one of two views: the Interconnected Accredited AIS view or the single Trusted System view. The GTNP has been used as the Mandatory Component together with NTCB components provided by other products for trusted security guards, firewalls, access controllers and key distribution centers applications.

Within the TCB Subset paradigm, the overall system TCB would be composed of a number of separately evaluated TCB subsets. The GTNP's hierarchical ring abstraction would be used to provide a layered foundation in which the system TCB is composed of separate trusted software components with distinct domains, i.e, each TCB subset occupies a different ring.

The GEMSOS Security Kernel has been shipped as a commercial product since 1985.

EVALUATION SUMMARY:

The security protection provided by the Gemini Trusted Network Processor, configured according to the most secure manner described in the Trusted Facility Manual, has been evaluated by the National Computer Security Center (NCSC) against the requirements specified by the Department of Defense Trusted Computer System Evaluation Criteria [DOD 5200.28-STD] dated December 1985, as interpreted by the Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria [NCSC-TG-005, Version-1], dated 31 July 1987. The GTNP has been evaluated against Appendix A of the Trusted Network Interpretation.

The National Security Agency (NSA) evaluation team has determined that the highest class at which the Gemini Trusted Network Processor satisfies all

of the specified requirements of the Criteria as interpreted by the Network Interpretation is as an A1 Mandatory-Only Network Component. As a result, the GTNP can potentially be incorporated into a network system that can meet the Trusted Network Interpretation (of the Trusted Computer System Evaluation Criteria) (TNI) part I requirements for class A1 and several of the TNI part II requirements.

A system that has been rated as being an A division system is characterized by the use of formal security verification methods to assure that the controls provided by the system can effectively protect classified or other sensitive information stored or processed by the system. The system architecture is that of a minimized security reference monitor. Extensive documentation is required to demonstrate that the Trusted Computing Base (TCB)* meets the security requirements in all aspects of design, development, and implementation.

M-Components are components that provide network support of the Mandatory Access Control (MAC) Policy as specified in the TNI. M-Components do not include the mechanisms necessary to completely support any of the other network policies (Discretionary Access Control, Identification and Authentication, and Audit) as defined in the interpretation.

Gemini Computers, Incorporated is participating in the Ratings Maintenance Phase (RAMP) Program for the Gemini Trusted Network Processor. Future changes to the evaluated system will continue to be reviewed via RAMP so as to maintain the rating of the system.

For a complete description of how the Gemini Trusted Network Processor satisfies each requirement of the Criteria, see Final Evaluation Report, Gemini Trusted Network Processor (Report No. CSC-EPL-94/008). The report should also be consulted for an exact list of the evaluated hardware components.

---

*: In the case of network component evaluations, this demonstration is for the Network TCB (NTCB) component partition.

---[0394]---

Appendix D

80286 Hardware Overview

This section presents a somewhat detailed overview of the 80286. For even more detail refer to the iAPX 286 Hardware Reference Manual [104] and the 80286 Operating Systems Writers Guide [107].

The 80286 is a general purpose microprocessor. It supports a 24-bit address bus and a 16-bit data bus, both internal and external.

D.1 Registers

The 80286 has fifteen 16-bit registers (**AX**, **BX**, **CX**, **DX**, **BP**, **SI**, **DI**, **SP**, **F**, **IP**, **MSW**, **CS**, **DS**, **SS**, **ES**), which may be grouped into four categories: general registers, base and index registers, status and control registers, and segment registers.

Memory Management Registers — **GDTR**, the global descriptor table register; **LDTR**, the local descriptor table register; **IDTR**, the interrupt descriptor table register, which points to a table of entry points for interrupt handlers; and **TR**, the task register, which defines the current task when multitasking.

General Registers — Eight general purpose registers. Of these, four may be used either as 16-bit registers, or may be split into pairs of individually-addressable 8-bit registers. The 8-bit registers are referenced by the byte (low or high) which they occupy in the respective 16-bit register. [**AX** (**AH** and **AL**), **BX** (**BH** and **BL**), **CX** (**CH** and **CL**), and **DX** (**DH** and **DL**)].

Base and Index Registers — The other four general purpose registers which may also be used for address offsetting in memory. They may contain base addresses or indices to specific locations within a segment. The addressing mode in use will determine the particular registers used for the calculation of an address. [Base registers: **BX** and **BP**; Index registers: **SI** and **DI**; Stack pointer **SP**].

Status and Control Registers — Three registers which maintain the current state of the processor. [**F**, flags; **IP**, instruction pointer; **MSW**, machine status word].

Segment Registers — Four segment registers which are used to select the segments of memory that are addressable for code, stack, and data. [**CS**, code segment selector; **DS**, data segment selector; **SS**, stack segment selector; **ES**, extra segment selector].

D.2 Modes of Operation

The 80286 supports two modes of operation: Real Address Mode and Protected Virtual Address Mode (PVAM). The Real Address Mode should not be used in protection-critical applications, because in that state any task running on the processor has full and unrestricted access to the entire 1 Mbyte address space. The PVAM, however, expands the address space to 16 Mbytes and provides protection for memory partitions within that address space.

The processor always starts operating in Real Address Mode. A single microprocessor instruction, LMSW (Load Machine Status Word), can switch the processor into PVAM, subsequently only a hardware reset can cause a switch back to Real Address Mode. Hence, the rest of this discussion will only be necessarily relevant to the PVAM of operation.

The 80286, in PVAM, supports a set of four hierarchical privilege levels, memory access mediation through a central mechanism, and task separation mechanisms.

The following terms are used throughout this discussion:

Alias	Alternate descriptor, for a segment with different segment attributes.
Descriptor	Structure used to define a memory segment.
Descriptor Table	Memory resident structure used to store descriptors.
Gate	Special descriptor, used in transferring control.
Interrupt	Break in normal task execution.
Privilege Level	Hierarchical domains of privilege.
Segment	Finest granularity of memory separation, described by descriptors.
Selector	Used to indicate a descriptor.
Task	Single thread of execution.
Trap (exception)	Similar to interrupt, occurs when instruction fails to complete normally.

D.3 Security Features

D.3.1 Segments

The 80286 microprocessor views physical memory as a collection of segments. Each segment can be from 1 byte to 64 Kbytes in size and can reside anywhere within the 16-Mbyte memory address space. Segments can also be defined to overlap each other. However, whenever a reference to a segment is made, the address calculated is interpreted as being within that segment.

D.3.2 Address Translation

Memory addressing is accomplished by the use of 32-bit pointers, each composed of a 16-bit selector field and a 16-bit offset field. In Real Mode, the 16-bit selector field contains the upper 16 bits of the 20-bit segment address, the lower four bits being zero; while the 16-bit offset field contains the lower 16 bits of the

20-bit offset address, with the upper four bits always zero. The 20-bit physical address is then calculated by adding together these two addresses.

Whereas the Real Mode selector field represents the high-order 16 bits of a real memory address, in Protected Mode it represents a 13-bit index into a memory-resident Segment Descriptor Table. The value contained within the specific table entry, called a Descriptor, contains the 24-bit segment base address. This base address is then added to the contents of the offset field of the pointer to result in the translation to the indicated physical memory address location.

D.3.3 Descriptors

A descriptor is used to refer to a data structure consisting of the set of attributes associated with each segment known to the 80286. Each segment can be accessed only through a descriptor.

The descriptor contains a descriptor privilege level (DPL), segment type, access, segment specific information (e.g., stack expansion direction), and base address and bounds of the segment.

The DPL assigns one of the four 80286 privilege levels (discussed later) to the given descriptor. Because a segment is only accessible through its descriptor, this privilege level can conceptually be associated with the segment as well. However, the fact that one descriptor can refer to a segment (at one privilege level) does not imply that another descriptor cannot refer to another segment (at another privilege level), occupying the same memory space.

The descriptor defines its associated segment as being one of the following types: data segment, code segment, special system segment, or gate.

A data segment may have the following access permissions: read-only or read/write. The descriptor also defines whether it will expand up or down in the memory address space. This is useful when a data segment is to be used for a stack.

A code segment may have the following access permissions: execute or read/execute. The descriptor also defines whether it will be a conforming code segment or not. A conforming code segment is one that acquires the privilege level of the calling task if the task has less privilege.

Special system segments are Task State Segments (TSS) and Local Descriptor Tables (LDT). Each will be described in more detail below.

Gates are special descriptors used for transferring control indirectly. There are four types of gate:

Call gate	A call gate contains the DPL, base address, and index for a code segment. The index is used to set up an entry point.
Trap gate	A trap gate is identical to call gate, except that they are used in the Interrupt Descriptor Table (IDT) to specify a trap service routine.
Interrupt gate	An interrupt gate is identical to a trap gate, except that when they are invoked, interrupts are automatically disabled.
Task gate	A task gate contains a descriptor referencing a TSS. Use of this gate cause a complete context switch.

All descriptors are physically located in memory resident tables, each a segment itself. The 80286 recognizes three types of descriptor table: LDT, Global Descriptor Table (GDT), and IDT. LDTs are used for task-local data; there can be virtually any number of these in memory, but only one associated with any given task. Multiple tasks may share a common LDT so that they may share a semiprivate (as opposed to completely global) data set. The GDT is used for system-wide, global data. There is one GDT defined to any given 80286. The IDT is used to store interrupt vectors in the form of gates. Up to 256 such gates may exist in this table and there can be only one such table known to the 80286 at any given instant.

D.3.4 Selectors

Segment selectors are used by tasks to select the descriptor to use in order to reference a segment. The selectors are composed of an index, descriptor table selector bit, and a Requested Privilege Level (RPL). The descriptor table selector bit specifies whether to use the GDT or LDT. The index specifies the descriptor within that table to use. The RPL indicates the privilege level of the access request.

An executing task has access to four segment selector registers: Code Segment (**CS**), Data Segment (**DS**), Stack Segment (**SS**), and Extra Segment (**ES**). Hence, a task has access to up to four segments at any given instant.

When a selector is loaded, the selected descriptor is checked to ensure that it exists and is well formed. Also, the segment is checked to ensure that it is present.

Instructions that load selectors into **DS** and **ES** must refer to a data segment descriptor or a readable code segment descriptor, and the privilege requirements must be met. Instructions that load a selector into **SS** must refer to a writable data descriptor.

Control transfer is accomplished when a selector is loaded into the **CS** by a control transfer operation. A transfer can occur only if the operation that loads the selector references the correct type of descriptor.

If an attempt is made to load an incorrect type of descriptor or if the privilege check fails, then a general-protection exception will occur. A general-protection exception indicates the occurrence of a violation to privilege rules or usage rules. An interrupt handler reads in an error code that is pushed onto the stack after the return address. This error code identifies the segment sector that is involved, while the return address identifies the instruction that caused the exception. Although most exceptions are restartable after the case for exception has been removed, a restart is generally not attempted for general-protection exceptions.

D.3.5 Privilege Levels

The 80286 supports four hierarchical levels of privilege (Privilege Levels), numbered zero (PL0) through three (PL3) in decreasing level of privilege. One such level (at a time) is associated with every task and descriptor, and hence with every segment, subject, and gate.

A set of privilege rules are enforced by the 80286 and are mandatory in nature. They can be summarized as follows:

- Data can be accessed only from the same or more privileged level.
- Code can be executed (called) only from the same or less privileged level.

After all checks, including privilege access checks and descriptor access checks, a selector may be loaded. Any further references to the same segment require further checks to ensure that the specific reference is still within the bounds of the segment and that no attempt to write to a read-only segment has been made.

Any violations will cause a trap to occur before any memory reference is made or any registers are modified. This ensures that the process state remains unchanged in case a restart may be possible.

D.3.6 Stacks

Stacks are referenced through the **SS** and **SP**, to provide an offset. The stack segment is merely a data segment, possibly configured to expand downward in memory address space.

When a user transfers control to a new (more privileged) privilege level, the **SS:SP** pair is placed on the stack at the new level. When a return from that level occurs, those values are restored at the previous level.

Each privilege level has its own stack that is determined by the Current Privilege Level (CPL) of the task and the TSS (described below), which contains stack segment information for PL0-PL2. There is no need for any stack information to be predefined in this manner for PL3, because of the fact that PL3 is the lowest privilege level and could not have been transferred into from a lower privileged level. Information can also be passed from lower privileged to higher privileged levels, but not the other way. If such is the case, the information will be placed onto the stack along with a counter indicating how much information was passed on the stack.

D.3.7 Tasks

A task, with respect to the 80286, is conceptually a single thread of execution, which is defined by a TSS. The TSS defines a save area for the 80286 registers and stacks for PL0-PL2. It also contains a selector for the task LDT and a selector referencing the calling task (only in the event of a task switching interrupt).

At any given instant, the 80286 Task Register is used to reference a descriptor referencing the currently-executing task's TSS and also indicating the CPL of the task by virtue of the descriptor's DPL.

During a task switch (transfer of control), the static portion (stack pointers and LDT selector) of the TSS remains unchanged, but the dynamic portion (the register save area) is automatically updated with the current register contents. These registers are restored automatically when the task regains control.

D.3.8 Control Transfers

Control transfers can be made intrasegment, intersegment, and interlevel.

Intrasegment control transfers can be done via CALL, JMP, and RET instructions. In the case of the CALL and JMP, only an offset is specified. This offset is used to calculate a new point of execution within the same segment.

Intersegment control transfers can be done via CALL, JMP, and RET instructions. In the case of the CALL and JMP, either a code segment and offset are specified directly, or a call gate may be specified. In this case,

the gate must be accessible (i.e., at the same or lower privilege level) and the code segment is assumed to be at the same level and, thus, accessible.

Interlevel control transfers may be done via CALL and RET instructions. In the case of the CALL, either a call or task gate must be referenced so that control can be transferred to the new level as dictated by the gate. In this case, the gate must be accessible (i.e., at the same or lower privilege level) and the code segment must also be accessible (i.e., at a greater privilege level). Hence, interlevel control transfers via the CALL instruction can be only to a more privileged level. The RET instruction allows control to transfer in the other direction, but only after a successful CALL has been completed.

D.3.9 Interrupts and Traps

Interrupts and traps are special cases of control transfers.

Interrupts may be internal or external and maskable or non-maskable. In the case of external hardware interrupts, they are independent of the currently executing task. The 80286 also provides the capability for software interrupts.

A trap, as opposed to an interrupt, is generated when an instruction fails to complete normally.

Tasks available to service interrupts and traps can be configured to execute in the interrupted task's context, or to do a complete task switch before servicing. The choice is determined by whether the interrupt/trap service routine is defined to be a trap, interrupt, or task gate.

Up to 256 interrupts and traps may be defined on the 80286; 32 of these are defined internally by Intel, the rest may be defined by the system. A gate corresponding to each interrupt/trap is contained within the IDT. The IDT is a variable length segment, but must be minimally large enough to support the 32 internally defined interrupts. It may then grow as needed by the system.

Just like with all other code segments, the interrupt/trap service code segment must be at least as privileged as the interrupted code segment's privilege level.

Use of trap or interrupt gates may cause the context of the interrupted routine to change, since they execute in the same context. However, use of a task gate leaves the interrupted task's context unchanged, since a complete context switch is made.

D.3.10 Aliasing

The term "aliasing" refers to the capability to create more than one descriptor referring segments that occupy (at least in part) the same memory address space. This allows a programmer to have access to a segment in multiple modes or even with multiple privilege levels.

Example: One descriptor indicates that read/write access is allowed to some data segment at PL0. Another descriptor indicates that execute access is allowed to some code segment (occupying the same memory address space) at PL1. In this instance, the actual code may be read and modified by PL0 tasks, while it could be executed by PL1 and lower privileged tasks.

This feature may be used dynamically during system operation in order to create and modify descriptor tables. In order to create new LDTs, some task must have write access to the GDT in order to build the

pointer. Thus, a descriptor referring to the GDT address space as a writable data segment must exist. From a security standpoint, it would be desired that this descriptor be protected as much as possible (e.g., reside in PL0). Also, when initially creating the LDT (e.g., specifying its descriptor entries) it must also be accessible as a writable data segment. However, once it is built, the descriptor providing this capability may be destroyed.

It should be apparent that great care must be taken to restrict modify access to the descriptor tables. Such access can be used to render the protection features of the 80286 inert.

This page intentionally left blank

Appendix E

i386 Hardware Overview

This section presents a somewhat detailed overview of the i386. For even more detail refer to the Intel386 DX Microprocessor Hardware Reference Manual [119].

The i386 is an advanced high performance microprocessor optimized for multitasking operating systems. It supports a 32-bit address bus and a 32-bit data bus.

The i386 is available in a variety of designations:

Intel386DX	Intel386 processor with 32-bit wide address bus, and 32-bit wide internal data pathways.
Intel386SX	Intel386 processor with 24-bit wide address bus, and 32-bit wide internal data pathways.

E.1 Registers

The system registers of the i386 fall into the following categories: General Registers, Segment Registers, Memory Management Registers, Control Registers, the **EFLAGS** Register, Debug Registers, and the Test Registers.

The eight General Registers include: **EAX**, **EBX**, **ECX**, **EDX**, **EBP**, **ESP**, **ESI**, and **EDI**. These 32-bit registers can be used as operands for logical and arithmetic operations and for address calculation (with the exception of **ESP** which cannot be used as an index operand). The low order word of each general register can be addressed as a single 16-bit unit by using the identifiers: **AX**, **BX**, **CX**, **DX**, **BP**, **SP**, **SI**, and **DI**. The general registers **AX**, **BX**, **CX**, and **DX** can be further divided for addressing purposes into two 8-bit registers. These 8-bit registers are referenced by the byte (high or low) which they occupy in the respective 16-bit register [**AX** (**AH** and **AL**), **BX** (**BH** and **BL**), **CX** (**CH** and **CL**), and **DX** (**DH** and **DL**)].

The six 16-bit Segment Registers are used to select the segments of memory that are addressable as code, stack, or data. They include: **CS** (code segment selector), **SS** (stack segment selector), **DS**, **ES**, **FS**, and **GS** (all data segment selectors).

Memory Management Registers	**GDTR**, the global descriptor table register; **LDTR**, the local descriptor table register; **IDTR**, the interrupt descriptor table register, which points to a table of entry points for interrupt handlers; and **TR**, the task register, which defines the current task when multitasking.
Control Registers	**CR0**, **CR2**, and **CR3**. **CR0** contains the flags which control or indicate status of the system as a whole, not of an individual

task. These status conditions include emulation, protection enable, paging, and extension type. **CR0** is the equivalent of the **MSW** register in the 80286, and can be accessed by instructions from the 80286 instruction set that interact with the **MSW**. **CR2** handles page faults when the Paging flag (PG) of **CR0** is set. It is used to store the linear address that triggered the page fault. **CR3** contains the address of the page table directory for the current task. As with **CR2**, this is used only when PG is set.

System Flags

EFLAGS contains flags that control such features as the I/O, maskable interrupts, and debugging. These flags are: IF, Interrupt Enable Flag; NT, Nested Task; RF, Resume (from debug exception) Flag; TF, Trap Flag (for debugging); and VM, Virtual 8086 Mode.

Debug Registers

These registers are used for implementing the advanced debugging abilities of the i386, such as the setting of breakpoints without the modification of code segments. They are 32 bits in length and may be accessed only at privilege level zero (PL0). **DR0**, **DR1**, **DR2**, and **DR3** are Debug Address Registers used to store the addresses of breakpoints. **DR4** and **DR5** are reserved by the processor. **DR6** is the Debug Status Register which is set by the processor when an enabled debug exception is detected. This allows the debugger to determine which debug conditions have occurred. **DR7** is the Debug Control Register, which is used to specify which actions cause breaks and to enable the Debug Address Registers locally or globally (when paging is used).

Test Registers

These registers are not a standard part of the i386 architecture. They are provided solely for confidence testing of the cache used for storing information from page tables.

E.2 Modes of Operation

The i386 supports three modes of operation: Real Address Mode, Protected Virtual Address Mode (PVAM), and Virtual 8086 (V86) Mode.

E.2.1 Real Mode

The processor always starts operating in Real Address Mode. The only way to leave Real Mode is to switch to PVAM. PVAM is entered when a move instruction sets the protection enable (PE) bit in **CR0**. To maintain compatibility with the 80286, the LMSW (Load Machine Status Word) instruction may also be executed to enter PVAM. Once in PVAM, Real Mode is then entered upon RESET or if software (in PL0) reloads the **CR0** register.

Real Mode, to a programmer, appears to be a fast 8086. The processor is always in Real Mode after RESET. Real Mode is typically used only for initialization. The Real Address Mode should not be used in protection

critical applications, because in that state any task running on the processor has full and unrestricted access to the entire 1 Mbyte address space. Because paging (see section on Address Translation) is not used in Real Mode, the linear address is equivalent to the physical address. PVAM, however, expands the logical address space to 64 Tbytes and provides protection for memory partitions within that address space.

E.2.2 Protected Mode

PVAM is the natural environment of the i386 processor. In PVAM, all features and instructions are available. The features of the i386 are a superset of those of the 80286; therefore, programs designed to be executed in PVAM on the 80286 generally execute without modification on the i386. The few differences which do exist primarily affect operating system code and can be easily remedied. For example, it is possible that the software for an 80286 relies on the fact that a base and offset combination which yields an address outside of the 16 Mbytes of available memory will wrap around. If this were run on the i386, though, its greater physical address space would result in the address falling in the 17th Mbyte. However, paging can be used on the i386 to simulate the behavior of the 80286.

The i386, in PVAM, supports a set of four hierarchical privilege levels, memory access mediation through a central mechanism, and task separation mechanisms.

E.2.3 Virtual 8086 Mode

Virtual 8086 (V86) Mode is dynamic in that the processor can rapidly alternate between V86 mode and PVAM. The CPU enters V86 mode from PVAM to execute an 8086 program, and then returns to PVAM to continue executing a native i386 program.

The i386 supports execution of one or more 8086 programs in an i386 environment by use of a V86 task. V86 tasks take advantage of the hardware support of multitasking that is offered by the PVAM. In V86 mode the i386 does not interpret selectors by referring to descriptors. Instead, it interprets linear addresses as would the 8086.

E.3 Security Features

E.3.1 Segments

The i386 microprocessor views physical memory as a collection of segments. Each segment can be from 1 byte to 4 Gbytes in size and can reside anywhere within the memory address space. The logical address space is composed of up to 64 Tbytes (4 Gbytes/segment $\times$ 16384 (16K) addressable segments). The processor maps the 64 Tbyte logical address space onto the 4 Gbyte physical address space (see section on Address Translation). Segments can also be defined to overlap each other. However, whenever a reference to a segment is made, the address calculated is interpreted as being within that segment.

E.3.2 Address Translation

Memory addressing in the i386 is of two forms: segment addressing or page addressing. Page addressing, which is optional, is built upon segment addressing.

Segment addressing is accomplished by the use of a 48-bit pointer, composed of a 16-bit selector field and a 32-bit offset field. The 16-bit selector field contains a bit that specifies the descriptor table (GDT or LDT) being referenced; 2 bits that denote the requested privilege level; and 13 bits that indicate the specific descriptor in the descriptor table. This descriptor contains the base address, which is then added to the 32-bit offset to yield the linear address in the desired segment.

Page translation is optional, being used only when the PG bit of **CR0** is set. A linear address (derived as described above) refers to a physical address by specifying a page table, a page within that table, and an offset within that page. The 32-bit linear address is divided into a 10-bit directory field (DIR), a 10-bit page field (PAGE), and a 12-bit offset field (OFFSET).

The physical address of the current page directory is stored in control register **CR3**. The DIR field of the linear address points to an entry in that directory. The contents of that entry indicate the starting address of a specific page table in memory. The PAGE field of the linear address points to an entry in the page table. This page table entry contains the page frame address, which is the first address of a page. This address is then added to the OFFSET field to produce the physical address desired.

E.3.3 Descriptors

A descriptor is used to refer to a data structure consisting of the set of attributes associated with each segment known to the i386. Each segment can be accessed only through a descriptor.

The descriptor contains a descriptor privilege level (DPL), segment type, access, segment specific information (e.g., stack expansion direction), and base address and bounds of the segment.

The DPL assigns one of the four i386 privilege levels (discussed later) to the given descriptor. Because a segment is only accessible through its descriptor, this privilege level can conceptually be associated with the segment as well. However, the fact that one descriptor can refer to a segment (at one privilege level) does not imply that another descriptor cannot refer to another segment (at another privilege level), occupying the same memory space.

The descriptor will define its associated segment as being one of the following types: data segment, code segment, special system segment, or gate.

A data segment may have the following access permissions: read-only or read/write. The descriptor also defines whether it will expand up or down in the memory address space. This is useful when a data segment is to be used for a stack.

A code segment may have the following access permissions: execute or read/execute. The descriptor also defines whether it will be a conforming code segment or not. A conforming code segment is one that acquires the privilege level of the calling task if the task has less privilege.

Special system segments are Task State Segments (TSS) and Local Descriptor Tables (LDT). Each will be described in more detail below.

Gates are special descriptors used for transferring control indirectly. There are four types of gate:

Call gate	A call gate contains the DPL, base address, and index for a code segment. The index is used to set up an entry point.
Trap gate	A trap gate is identical to call gate, except that they are used in the Interrupt Descriptor Table (IDT) to specify a trap service routine.
Interrupt gate	An interrupt gate is identical to a trap gate, except that when they are invoked, interrupts are automatically disabled.
Task gate	A task gate contains a descriptor referencing a TSS. Use of this gate cause a complete context switch.

All descriptors are physically located in memory resident tables, each a segment itself. The i386 recognizes three types of descriptor table: LDT, GDT, and IDT. LDTs are used for task local data there can be virtually any number of these in memory, but only one associated with any given task. Multiple tasks may share a common LDT so that they may share a semiprivate, as opposed to completely global, data set. The GDT is used for system-wide, global data. There is one GDT defined to any given i386. The IDT is used to store interrupt vectors in the form of gates. Up to 256 such gates may exist in this table and there can be only one such table known to the i386 at any given instant.

E.3.4 Selectors

Segment selectors are used by tasks to select the descriptor to use in order to reference a segment. The selectors are composed of an index, descriptor table selector bit, and a Requested Privilege Level (RPL). The descriptor table selector bit specifies whether to use the GDT or LDT. The index specifies the descriptor within that table to use. The RPL indicates the privilege level of the access request.

An executing task has access to six segment selector registers: Code Segment (**CS**), Stack Segment (**SS**), Data Segments (**DS**, **ES**, **FS**, and **GS**). Hence, a task has access to up to six segments at any given instant.

When a selector is loaded, the selected descriptor is checked to ensure that it exists and is well formed. Also, the segment is checked to ensure that it is present.

Instructions that load selectors into **DS**, **ES**, **FS**, and **GS** must refer to a data segment descriptor or a readable code segment descriptor, and the privilege requirements must be met. Instructions that load a selector into **SS** must refer to a writable data descriptor.

Control transfer is accomplished when a selector is loaded into the **CS** by a control transfer operation. A transfer can occur only if the operation that loads the selector references the correct type of descriptor.

If an attempt is made to load an incorrect type of descriptor or if the privilege check fails, then a general-protection exception will occur. A general-protection exception indicates the occurrence of a violation to privilege rules or usage rules. An interrupt handler reads in an error code that is pushed onto the stack after the return address. This error code identifies the segment sector that is involved, while the return address identifies the instruction that caused the exception. Although most exceptions are restartable after the case for exception has been removed, a restart is generally not attempted for general-protection exceptions.

E.3.5 Privilege Levels

The i386 supports four hierarchical levels of privilege (Privilege Levels), numbered zero (PL0) through three (PL3) in decreasing level of privilege. One such level (at a time) is associated with every task and descriptor, and hence with every segment, subject, and gate.

A set of privilege rules are enforced by the i386 and are mandatory in nature. They can be summarized as follows:

- Data can be accessed only from the same or more privileged level.
- Code can be executed (called) only from the same or less privileged level.

After all checks, including privilege access checks and descriptor access checks, a selector may be loaded. Any further references to the same segment require further checks to ensure that the specific reference is still within the bounds of the segment and that no attempt to write to a read-only segment has been made.

Any violations will cause a trap to occur before any memory reference is made or any registers are modified. This ensures that the process state remains unchanged in case a restart may be possible.

E.3.6 Stacks

Stacks are referenced through the **SS** and **SP**, to provide an offset. The stack segment is merely a data segment, possibly configured to expand downward in memory address space.

When a user transfers control to a new (more privileged) privilege level, the **SS**:**SP** pair is placed on the stack at the new level. When a return from that level occurs, those values are restored at the previous level.

Each privilege level has its own stack that is determined by the Current Privilege Level (CPL) of the task and the TSS (described below), which contains stack segment information for PL0-PL2. There is no need for any stack information to be predefined in this manner for PL3, because of the fact that PL3 is the lowest privilege level and could not have been transferred into from a lower privileged level. Information can also be passed from lower privileged to higher privileged levels, but not the other way. If such is the case, the information will be placed onto the stack along with a counter indicating how much information was passed on the stack.

E.3.7 Tasks

A task, with respect to the i386, is conceptually a single thread of execution, which is defined by a TSS. The TSS defines a save area for the i386 registers and stacks for privilege levels 0-2. It also contains a selector for the task LDT and a selector referencing the calling task (only in the event of a task switching interrupt).

At any given instant, the i386 Task Register is used to reference a descriptor referencing the currently-executing task's TSS and also indicating the CPL of the task by virtue of the descriptor's DPL.

During a task switch (transfer of control), the static portion (stack pointers and LDT selector) of the TSS remains unchanged, but the dynamic portion (the register save area) is automatically updated with the current register contents. These registers are restored automatically when the task regains control.

E.3.8 Control Transfers

Control transfers can be made intrasegment, intersegment, and interlevel.

Intrasegment control transfers can be done via CALL, JMP, and RET instructions. In the case of the CALL and JMP, only an offset is specified. This offset is used to calculate a new point of execution within the same segment.

Intersegment control transfers can be done via CALL, JMP, and RET instructions. In the case of the CALL and JMP, either a code segment and offset are specified directly, or a call gate may be specified. In this case, the gate must be accessible (i.e., at the same or lower privilege level) and the code segment is assumed to be at the same level and, thus, accessible.

Interlevel control transfers may be done via CALL and RET instructions. In the case of the CALL, either a call or task gate must be referenced so that control can be transferred to the new level as dictated by the gate. In this case, the gate must be accessible (i.e., at the same or lower privilege level) and the code segment must also be accessible (i.e., at a greater privilege level). Hence, interlevel control transfers via the CALL instruction can be only to a more privileged level. The RET instruction allows control to transfer in the other direction, but only after a successful CALL has been completed.

E.3.9 Interrupts and Traps

Interrupts and traps are special cases of control transfers.

Interrupts may be internal or external and maskable or unmaskable. In the case of external hardware interrupts, they are independent of the currently executing task. The i386 also provides the capability for software interrupts.

A trap, as opposed to an interrupt, is generated when an instruction fails to complete normally.

Tasks available to service interrupts and traps can be configured to execute in the interrupted task's context, or to do a complete task switch before servicing. The choice is determined by whether the interrupt/trap service routine is defined to be a trap, interrupt, or task gate.

Up to 256 interrupts and traps may be defined to the i386; 32 of these are defined internally by Intel, the rest may be defined by the system. A gate corresponding to each interrupt/trap is contained within the IDT. The IDT is a variable length segment, but must be minimally large enough to support the 32 internally defined interrupts. It may then grow as needed by the system.

Just like with all other code segments, the interrupt/trap service code segment must be at least as privileged as the interrupted code segment's privilege level.

Use of trap or interrupt gates may cause the context of the interrupted routine to change, since they execute in the same context. However, use of a task gate leaves the interrupted task's context unchanged, since a complete context switch is made.

E.3.10 Aliasing

The term "aliasing" refers to the capability to create more than one descriptor referring segments that occupy (at least in part) the same memory address space. This allows a programmer to have access to a segment in multiple modes or even with multiple privilege levels.

Example: One descriptor indicates that read/write access is allowed to some data segment at PL0. Another descriptor indicates that execute access is allowed to some code segment (occupying the same memory address space) at PL1. In this instance, the actual code may be read and modified by PL0 tasks, while it could be executed by PL1 and lower privileged tasks.

This feature may be used dynamically during system operation in order to create and modify descriptor tables. In order to create new LDTs, some task must have write access to the GDT in order to build the pointer. Thus, a descriptor referring to the GDT address space as a writable data segment must exist. From a security standpoint, it would be desired that this descriptor be protected as much as possible (e.g., reside in PL0). Also, when initially creating the LDT (e.g., specifying its descriptor entries) it must also be accessible as a writable data segment. However, once it is built, the descriptor providing this capability may be destroyed.

It should be apparent that great care must be taken to restrict modify access to the descriptor tables. Such access can be used to render the protection features of the i386 inert.

Appendix F

i486 Hardware Overview

This section presents a somewhat detailed overview of the i486. For even more detail refer to the Intel486 DX Microprocessor Hardware Reference Manual [120].

The i486 is an advanced high-performance microprocessor optimized for multitasking operating systems. It supports a 32-bit address bus and a 32-bit data bus. It uses Reduced Instruction Set Computer (RISC) design techniques to reduce the execution times of frequently used instructions to a single-cycle.

The i486 provide an on-chip 8-Kbyte unified code and data cache, with on-chip hardware to ensure cache consistency. With the exception of the SX model of the i486, the mathematical coprocessor floating point unit is also on-chip.

The i486 also provides a built-in self-test capability. This self-test extensively tests on-chip logic, cache memory, and the on-chip paging translation cache.

The i486 is available in a variety of designations:

Intel486DX/*nn* Intel486 processor with on-chip floating point (Intel487), running at a bus and clock speed of *nn* MHz.

Intel486DX2/*nn* Intel486 processor with on-chip floating point (Intel487), running at a bus speed of $\frac{nn}{2}$ MHz and an internal clock speed of *nn* MHz.

Intel486DX4/*nn* Intel486 processor with on-chip floating point (Intel487), running at a bus speed of $\frac{nn}{3}$ MHz and an internal clock speed of *nn* MHz.

Intel486SX/*nn* Integer-only Intel486 processor running at a bus and clock speed of *nn* MHz. An Intel487 math coprocessor is *not* on-chip and must be added separately.

Intel486SX2/*nn* Integer-only Intel486 processor running at a bus speed of $\frac{nn}{2}$ MHz and an internal clock speed of *nn* MHz. An Intel487 math coprocessor is *not* on-chip and must be separately added.

Intel486 processors may also be SL-enhanced, which means that they have been adapted to support the low-power SL technology.

F.1 Registers

The system registers of the i486 fall into the following categories: General Registers, Segment Registers, Memory Management Registers, Control Registers, the **EFLAGS** Register, Debug Registers, and the Test Registers.

The eight General Registers include: **EAX**, **EBX**, **ECX**, **EDX**, **EBP**, **ESP**, **ESI**, and **EDI**. These 32-bit registers can be used as operands for logical and arithmetic operations and for address calculation (with the exception of **ESP** that cannot be used as an index operand). The low order word of each general register can be addressed as a single 16-bit unit by using the identifiers: **AX**, **BX**, **CX**, **DX**, **BP**, **SP**, **SI**, and **DI**. The general registers **AX**, **BX**, **CX**, and **DX** can be further divided for addressing purposes into two 8-bit registers. These 8-bit registers are referenced by the byte (high or low) that they occupy in the respective 16-bit register [**AX** (**AH** and **AL**), **BX** (**BH** and **BL**), **CX** (**CH** and **CL**), and **DX** (**DH** and **DL**)].

The six 16-bit Segment Registers are used to select the segments of memory that are addressable as code, stack, or data. They include: **CS** (code segment selector), **SS** (stack segment selector), **DS**, **ES**, **FS**, and **GS** (all data segment selectors).

Memory Management Registers

GDTR, the global descriptor table register; **LDTR**, the local descriptor table register; **IDTR**, the interrupt descriptor table register, which points to a table of entry points for interrupt handlers; and **TR**, the task register, which defines the current task when multitasking.

Control Registers

CR0, **CR2**, and **CR3** (**CR1** is reserved). **CR0** contains the flags that control or indicate status of the system as a whole, not of an individual task. These status conditions include emulation, protection enable, paging, and extension type. **CR0** is the equivalent of the **MSW** register in the 80286 and can be accessed by instructions from the 80286 instruction set that interact with the **MSW**. **CR2** handles page faults when the Paging flag (PG) of **CR0** is set. It is used to store the linear address that triggered the page fault. **CR3** contains the address of the page table directory for the current task. As with **CR2**, this is used only when PG is set.

System Flags

EFLAGS contains flags that control such features as the I/O, maskable interrupts, and debugging. These flags are: IF, Interrupt Enable Flag; NT, Nested Task; RF, Resume (from debug exception) Flag; TF, Trap Flag (for debugging); and VM, Virtual 8086 Mode.

Debug Registers

These registers are used for implementing the advanced debugging abilities of the i486, such as the setting of breakpoints without the modification of code segments. They are 32 bits in length and may be accessed only at privilege level zero (PL0). **DR0**, **DR1**, **DR2**, and **DR3** are Debug Address Registers used to store the addresses of breakpoints. **DR4** and **DR5** are reserved by the processor. **DR6** is the Debug Status Register that is set by the processor when an enabled debug exception is detected. This allows the debugger to determine the debug conditions that have occurred. **DR7** is the Debug Control Register, which is used to specify the actions that result in breaks and to enable the Debug Address Registers locally or globally (when paging is used).

Test Registers	These registers are not a standard part of the i486 architecture. They are provided solely for confidence testing of the cache used for storing information from page tables.

F.2 Modes of Operation

The i486 supports three basic modes of operation: Real Address Mode, Protected Virtual Address Mode (PVAM), and Virtual 8086 (V86) Mode. In addition, i486 microprocessors that support the low-power SL technology also support System Management Mode.

F.2.1 Real Mode

The processor always powers-up in Real Address Mode. The only way to leave Real Mode is to switch to PVAM. PVAM is entered when a move instruction sets the protection enable (PE) bit in **CR0**. To maintain compatibility with the previous processors in the family, the LMSW (Load Machine Status Word) instruction may also be executed to enter PVAM. Once in PVAM, Real Mode is then entered upon RESET, or if software (in PL0) reloads the **CR0** register.

Real Mode, to a programmer, appears to be a fast 8086. The processor is always in Real Mode after power-up or RESET. Real Mode is typically used only for initialization. Real Mode should not be used in protection critical applications, because in that state any task running on the processor has full and unrestricted access to the entire 1 Mbyte address space. Because paging (see section on Address Translation) is not used in Real Mode, the linear address is equivalent to the physical address. PVAM, however, expands the logical address space to 64 Tbytes and provides protection for memory partitions within that address space.

F.2.2 Protected Mode

The i486, in PVAM, supports a set of four hierarchical privilege levels, memory access mediation through a central mechanism, and task separation mechanisms.

F.2.3 Virtual 8086 Mode

Virtual 8086 (V86) Mode is dynamic in that the processor can rapidly alternate between V86 mode and PVAM. The CPU enters V86 mode from PVAM to execute an 8086 program, and then returns to PVAM to continue executing a native i486 program.

The i486 supports execution of one or more 8086 programs in an i486 environment by use of a V86 task. V86 tasks take advantage of the hardware support of multitasking that is offered by the PVAM. In V86 mode the i486 does not interpret selectors by referring to descriptors. Instead, it interprets linear addresses as would the 8086.

F.2.4 System Management Mode

System Management Mode (SMM) was established to provide support for power-management code, and allows this code to run protected from, and transparent to, any other code running on the processor. A processor enters SMM via a non-maskable interrupt (System Management Interrupt) triggered by an external signal. SMM code is located in a separate and protected address space, and has complete addressability of all CPU memory and I/O space. The processor exits SMM when the Resume from System Management Mode (RSM) instruction is executed.

F.3 Security Features

F.3.1 Segments

The i486 microprocessor views physical memory as a collection of segments. Each segment can be from 1 byte to 4 Gbytes in size and can reside anywhere within the memory address space. The logical address space is composed of up to 64 Tbytes (4 Gbytes/segment $\times$ 16384 (16K) addressable segments). The processor maps the 64 Tbyte logical address space onto the 4 Gbyte physical address space (see section on Address Translation). Segments can also be defined to overlap each other. However, whenever a reference to a segment is made, the address calculated is interpreted as being within that segment.

F.3.2 Address Translation

Memory addressing in the i486 is of two forms: segment addressing or page addressing. Page addressing, which is optional, is built upon segment addressing.

Segment addressing is accomplished by the use of a 48-bit pointer, composed of a 16-bit selector field and a 32-bit offset field. The 16-bit selector field contains a bit that specifies the descriptor table (GDT or LDT) being referenced; 2 bits that denote the requested privilege level; and 13 bits that indicate the specific descriptor in the descriptor table. This descriptor contains the base address, which is then added to the 32-bit offset to yield the linear address in the desired segment.

Page translation is optional, being used only when the PG bit of **CR0** is set. A linear address (derived as described above) refers to a physical address by specifying a page table, a page within that table, and an offset within that page. The 32-bit linear address is divided into a 10-bit directory field (DIR), a 10-bit page field (PAGE), and a 12-bit offset field (OFFSET).

The physical address of the current page directory is stored in control register **CR3**. The DIR field of the linear address points to an entry in that directory. The contents of that entry indicate the starting address of a specific page table in memory. The PAGE field of the linear address points to an entry in the page table. This page table entry contains the page frame address, which is the first address of a page. This address is then added to the OFFSET field to produce the physical address desired.

F.3.3 Descriptors

A descriptor is used to refer to a data structure consisting of the set of attributes associated with each segment known to the i486. Each segment can be accessed only through a descriptor.

The descriptor contains a descriptor privilege level (DPL), segment type, access, segment specific information (e.g., stack expansion direction), and base address and bounds of the segment.

The DPL assigns one of the four i486 privilege levels (discussed later) to the given descriptor. Because a segment is only accessible through its descriptor, this privilege level can conceptually be associated with the segment as well. However, the fact that one descriptor can refer to a segment (at one privilege level) does not imply that another descriptor cannot refer to another segment (at another privilege level), occupying the same memory space.

The descriptor will define its associated segment as being one of the following types: data segment, code segment, special system segment, or gate.

A data segment may have the following access permissions: read-only or read/write. The descriptor also defines whether it will expand up or down in the memory address space. This is useful when a data segment is to be used for a stack.

A code segment may have the following access permissions: execute or read/execute. The descriptor also defines whether it will be a conforming code segment or not. A conforming code segment is one that acquires the privilege level of the calling task if the task has less privilege.

Special system segments are Task State Segments (TSS) and Local Descriptor Tables (LDT). Each will be described in more detail below.

Gates are special descriptors used for transferring control indirectly. There are four types of gate:

Call gate	A call gate contains the DPL, base address, and index for a code segment. The index is used to set up an entry point.
Trap gate	A trap gate is identical to call gate, except that they are used in the Interrupt Descriptor Table (IDT) to specify a trap service routine.
Interrupt gate	An interrupt gate is identical to a trap gate, except that when they are invoked, interrupts are automatically disabled.
Task gate	A task gate contains a descriptor referencing a TSS. Use of this gate cause a complete context switch.

All descriptors are physically located in memory resident tables, each a segment itself. The i486 recognizes three types of descriptor table: LDT, GDT, and IDT. LDTs are used for task local data there can be virtually any number of these in memory, but only one associated with any given task. Multiple tasks may share a common LDT so that they may share a semiprivate, as opposed to completely global, data set. The GDT is used for system-wide, global data. There is one GDT defined to any given i486. The IDT is used to store interrupt vectors in the form of gates. Up to 256 such gates may exist in this table and there can be only one such table known to the i486 at any given instant.

F.3.4 Selectors

Segment selectors are used by tasks to select the descriptor to use in order to reference a segment. The selectors are composed of an index, descriptor table selector bit, and a Requested Privilege Level (RPL). The descriptor table selector bit specifies whether to use the GDT or LDT. The index specifies the descriptor within that table to use. The RPL indicates the privilege level of the access request.

An executing task has access to six segment selector registers: Code Segment (**CS**), Stack Segment (**SS**), Data Segments (**DS**, **ES**, **FS**, and **GS**). Hence, a task has access to up to six segments at any given instant.

When a selector is loaded, the selected descriptor is checked to ensure that it exists and is well formed. Also, the segment is checked to ensure that it is present.

Instructions that load selectors into **DS**, **ES**, **FS**, and **GS** must refer to a data segment descriptor or a readable code segment descriptor, and the privilege requirements must be met. Instructions that load a selector into **SS** must refer to a writable data descriptor.

Control transfer is accomplished when a selector is loaded into the **CS** by a control transfer operation. A transfer can occur only if the operation that loads the selector references the correct type of descriptor.

If an attempt is made to load an incorrect type of descriptor or if the privilege check fails, then a general-protection exception will occur. A general-protection exception indicates the occurrence of a violation to privilege rules or usage rules. An interrupt handler reads in an error code that is pushed onto the stack after the return address. This error code identifies the segment sector that is involved, while the return address identifies the instruction that caused the exception. Although most exceptions are restartable after the case for exception has been removed, a restart is generally not attempted for general-protection exceptions.

F.3.5 Privilege Levels

The i486 supports four hierarchical levels of privilege (Privilege Levels), numbered zero through three in decreasing level of privilege. One such level (at a time) is associated with every task and descriptor, and hence with every segment, subject, and gate.

A set of privilege rules are enforced by the i486 and are mandatory in nature. They can be summarized as follows:

- Data can be accessed only from the same or more privileged level.
- Code can be executed (called) only from the same or less privileged level.

After all checks, including privilege access checks and descriptor access checks, a selector may be loaded. Any further references to the same segment require further checks to ensure that the specific reference is still within the bounds of the segment and that no attempt to write to a read-only segment has been made.

Any violations will cause a trap to occur before any memory reference is made or any registers are modified. This ensures that the process state remains unchanged in case a restart may be possible.

F.3.6 Stacks

Stacks are referenced through the **SS** and **SP**, to provide an offset. The stack segment is merely a data segment, possibly configured to expand downward in memory address space.

When a user transfers control to a new (more privileged) privilege level, the **SS**:**SP** pair is placed on the stack at the new level. When a return from that level occurs, those values are restored at the previous level.

Each privilege level has its own stack that is determined by the Current Privilege Level (CPL) of the task and the TSS (described below), which contains stack segment information for PL0-PL2. There is no need for any stack information to be predefined in this manner for PL3, because of the fact that PL3 is the lowest privilege level and could not have been transferred into from a lower privileged level. Information can also be passed from lower privileged to higher privileged levels, but not the other way. If such is the case, the information will be placed onto the stack along with a counter indicating how much information was passed on the stack.

F.3.7 Tasks

A task, with respect to the i486, is conceptually a single thread of execution, which is defined by a TSS. The TSS defines a save area for the i486 registers and stacks for PL0-PL2. It also contains a selector for the task LDT and a selector referencing the calling task (only in the event of a task switching interrupt).

At any given instant, the i486 Task Register is used to reference a descriptor referencing the currently-executing task's TSS and also indicating the CPL of the task by virtue of the descriptor's DPL.

During a task switch (transfer of control), the static portion (stack pointers and LDT selector) of the TSS remains unchanged, but the dynamic portion (the register save area) is automatically updated with the current register contents. These registers are restored automatically when the task regains control.

F.3.8 Control Transfers

Control transfers can be made intrasegment, intersegment, and interlevel.

Intrasegment control transfers can be done via CALL, JMP, and RET instructions. In the case of the CALL and JMP, only an offset is specified. This offset is used to calculate a new point of execution within the same segment.

Intersegment control transfers can be done via CALL, JMP, and RET instructions. In the case of the CALL and JMP, either a code segment and offset are specified directly, or a call gate may be specified. In this case, the gate must be accessible (i.e., at the same or lower privilege level) and the code segment is assumed to be at the same level and, thus, accessible.

Interlevel control transfers may be done via CALL and RET instructions. In the case of the CALL, either a call or task gate must be referenced so that control can be transferred to the new level as dictated by the gate. In this case, the gate must be accessible (i.e., at the same or lower privilege level) and the code segment must also be accessible (i.e., at a greater privilege level). Hence, interlevel control transfers via the CALL instruction can be only to a more privileged level. The RET instruction allows control to transfer in the other direction, but only after a successful CALL has been completed.

F.3.9 Interrupts and Traps

Interrupts and traps are special cases of control transfers.

Interrupts may be internal or external and maskable or unmaskable. In the case of external hardware interrupts, they are independent of the currently executing task. The i486 also provides the capability for software interrupts.

A trap, as opposed to an interrupt, is generated when an instruction fails to complete normally.

Tasks available to service interrupts and traps can be configured to execute in the interrupted task's context, or to do a complete task switch before servicing. The choice is determined by whether the interrupt/trap service routine is defined to be a trap, interrupt, or task gate.

Up to 256 interrupts and traps may be defined to the i486; 32 of these are defined internally by Intel, the rest may be defined by the system. A gate corresponding to each interrupt/trap is contained within the IDT. The IDT is a variable length segment, but must be minimally large enough to support the 32 internally defined interrupts. It may then grow as needed by the system.

Just like with all other code segments, the interrupt/trap service code segment must be at least as privileged as the interrupted code segment's privilege level.

Use of trap or interrupt gates may cause the context of the interrupted routine to change, since they execute in the same context. However, use of a task gate leaves the interrupted task's context unchanged, since a complete context switch is made.

F.3.10 Aliasing

The term "aliasing" refers to the capability to create more than one descriptor referring segments that occupy (at least in part) the same memory address space. This allows a programmer to have access to a segment in multiple modes or even with multiple privilege levels.

Example: One descriptor indicates that read/write access is allowed to some data segment at PL0. Another descriptor indicates that execute access is allowed to some code segment (occupying the same memory address space) at PL1. In this instance, the actual code may be read and modified by PL0 tasks, while it could be executed by PL1 and lower privileged tasks.

This feature may be used dynamically during system operation in order to create and modify descriptor tables. In order to create new LDTs, some task must have write access to the GDT in order to build the pointer. Thus, a descriptor referring to the GDT address space as a writable data segment must exist. From a security standpoint, it would be desired that this descriptor be protected as much as possible (e.g., reside in PL0). Also, when initially creating the LDT (e.g., specifying its descriptor entries) it must also be accessible as a writable data segment. However, once it is built, the descriptor providing this capability may be destroyed.

It should be apparent that great care must be taken to restrict modify access to the descriptor tables. Such access can be used to render the protection features of the i486 inert.

F.4 Differences Between The i386 And i486

Functionally, the i386 and i486 processors are practically identical. The differences are limited to new or modified control and status flags, along with several new instructions. The differences are summarized as follows:

- An i386 program, executing on the i486, can access the following new instructions available in real mode:
  - Application: BSWAP, CMPXCHG, and XADD.
  - System: INVD, INVLPG, and WPINVD.
- New Flag: The Alignment Check (AC) flag has been defined in the EFLAGS register. In conjunction with the alignment mask (AM) bit, it controls the generation of the alignment check exception.
- New Exception: In protected mode the i486 can produce the Alignment Check exception, which was previously undefined.
- New Test Registers: New test registers have been defined for testing of the cache.
- New Control Register Bits: Five new bits have been defined in **CR0** as shown below:
  - Numeric Error (NE)
  - Write Protect (WP)
  - Alignment Mask (AM)
  - Not Write-through (NW)
  - Cache Disable (CD)

  Two new bits have been defined in **CR3**:
  - Page-level cache disable (PCD)
  - Page-level write transparent (PWT)
- New Page Table entry bits: Two new bits have been defined in page table entries to control the caching of pages:
  - Page-level cache disable (PCD)
  - Page-level write transparent (PWT)
- ET Bit in **CR0**: On the i486, the processor extension type (ET) bit in **CR0** should always be set to one to indicate compatibility with Intel80387 protocols.
- Segment descriptor loads: On the i386, a locked read and write would be initiated to set the accessed bit of a segment descriptor during a load. On the i486, the locked read and write are generated only if the accessed bit is not already set.
- Initial register values differ after RESET: The i486 initializes some registers differently after a hardware reset.
- Prefetch queue: The prefetch queue on the i486 is 32 bytes, while the queue on the i386 is 16 bytes. Self-modifying code must be sure to flush the queue after modifications.

This page intentionally left blank

Appendix G
Acronyms

ABC	PC-AT Binary Clock
AC	Alternating Current; Alignment Check; Access Class
ACC	PC-AT Calendar Clock
ADP	Automated Data Processing
AM	Alignment Mask
APGEN	GEMSOS Applications Generator Utility
APP	PC-AT Parallel Printer Port
ASIC	Application Specific Integrated Circuit
ASIO	PC-AT serial I/O; Asynchronous Serial I/O
AT	Designation of IBM PC Model (Hardware); Alias Table (Software)
BIOS	Basic Input/Output System
BPRN	Bus Priority In
BREQ	Bus Request
CBC	Cipher Block Chaining
CBRQ	Common Bus Request
CCB	Configuration Control Board
CD	Cache Disable
CDCI	Computer Documentation Configuration Item
CDR	Critical Design Review
CI	Configuration Item
CM	Core Manager (Software); Configuration Management (Assurances)
CMM	Core Manager Module
CMOS	Complementary Metal-Oxide Semiconductor
COTS	Commercial "Off-The-Shelf"
CPL	Current Privilege Level
CPU	Central Processing Unit
CS	Code Segment
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
DAC	Discretionary Access Control
DAP	Design Analysis Phase
DC	Direct Current
DC37	Type of connector
DCE	Data Circuit-terminating Equipment
DCP	Data Ciphering Processor
DD	Disk Device
DED	Data Encryption Device
DES	Data Encryption Standard
DIR	Directory Field
DMA	Direct Memory Access
DOD	Department of Defense
DODCSC	Department of Defense Computer Security Center
DPL	Descriptor Privilege Level
DRAM	Dynamic Random-Access Memory

DR*n*	Debug address Register *n*
DS	Data Segment
DSD	Data Sealing Device
DSKGEN	Hard Disk Conditioning Utility
DTE	Data Terminal Equipment
DTLS	Descriptive Top Level Specification
DUD	Disk Unit Driver
DUM	Disk Unit Manager
DVM	Disk Volume Manager
DX	Designation for Intel Processor Series. For the i386, the DX designation indicates i386 chips with 32-bit data paths. For the i486, the DX designation indicates i486 processors with matching internal and external clock speeds and on-chip Intel487 numeric coprocessors.
DX2	Designation for i486 DX processors with clock-doubled internal clocks
DX4	Designation for i486 DX processors with clock-tripled internal clocks
ECC	Error Checking and Correcting
ECP	Engineering Change Proposal
EETH	Expandable Ethernet
EGA	Enhanced Graphics Adapter
EHDLC	Expandable HDLC
EPL	Evaluated Products List
EPROM	Erasable Programmable Read-Only Memory
ES	Extra Segment
ESDI	Enhanced Small Device Interface (an enhancement to the ST506 interface)
ESIO	Extended Serial I/O
ESLM	Encryption Support Library Module
ET	Extension Type
FCRB	Future Change Review Board
FDM	Formal Development Methodology
FDR	Final Design Review
FS	Extra Segment (i386 and i486)
FST	Functional Security Tests
FTLS	Formal Top-Level Specification
Gbyte	1,073,741,824 bytes
GDT	Global Descriptor Table
GDTR	Global Descriptor Table Register
GEMSOS	Gemini Multiprocessing Secure Operating System
GKI	GEMSOS Security Kernel Initialization
GLB	Greatest Lower Bound
GS	Extra Segment (i386 and i486)
GSC	Gemini System Controller
GSC-AT	Gemini System Controller - PC-AT
GT	GEMSOS Test
GTAT	GT Acceptance Test
GTC	GEMSOS Test Controller
GTNP	Gemini Trusted Network Processor
HDLC	High-level Data Link Control
HDS	Hardware Distribution Support

HDU	Hardware Distribution Utility
HIT	Half-Inch Tape
HP	Designation for a class of Gemini 80286 processors
HWCI	Hardware Configuration Item
I/O	Input/Output
iAPX	Intel Processor Designation
IBM	International Business Machines
ID	Identifier/Identification
IDM	Inner Device Manager
IEEE	Institute of Electrical and Electronic Engineers
IF	Interrupt Enable Flag
iLBX	Intel Local Bus Extension
IM	Interrupt Manager (Module)
IMM	Interrupt Manager Module
IOPB	Input/Output Parameter Block
IOPL	Input/Output Privilege Level
IP	Instruction Pointer; Internet Protocol
IPAR	Initial Product Assessment Report
IPC	Interprocess Communication
IPPT	Initial Process Parameter Table
IPR	Initial Parameter Record
IRS	Interface Requirements Specification
IR*n*	Hardware Interrupt Level *n*
iSBC	Intel Single Board Computer
iSBX	Intel Single Board Extension
ISL	Intersegment Linkage
ITC	Inner Traffic Controller
ITP	Interactive Theorem Prover
IV	Initialization Vector
JEDEC	Designation for a 26-pin IC socket
KB	PC-AT Keyboard
Kbyte	1,024 bytes
KD	Kernel Device (Layer)
KGL	GEMSOS Kernel Gate Library
KIT	Kernel Initialization Tables
KLT	Kernel Loader Table
KM	Key Management
KMS	Kernel Metering Support
KPT	Kernel Parameter Table
KSTACK	Kernel Stack
LAN	Local Area Network
LAPB	Link Access Procedure B
LDT	Local Descriptor Table
LED	Light-Emitting Diode
LHU	Logical Hardware Unit
LMSW	Load Machine Status Word
LSIO	Local Serial I/O
LUB	Least Upper Bound
LVN	Logical Volume Number
M-NTCB	Mandatory Network Trusted Computing Base
MAC	Mandatory Access Control
MAE	Model Application Environment
MBINTR*n*	Multibus interrupt *n*
Mbyte	1,048,576 bytes

MHz	Megahertz
MIPS	Million Instructions per Second
MIT	GEMSOS Multilevel Subjects Intersegment Linkage Tool
MLS	Multilevel Secure; GEMSOS Multilevel Subjects
MM	Memory Manager
MPIC	Master Programmable Interrupt Controller
MPSC	Multiple Protocol Serial Controller
MSA	Mandatory Security Administrator
MSM	Message-stream Modification
MSW	Machine Status Word
NCB	Network Controller Base
NCSC	National Computer Security Center
NDSM	Non-Discretionary Security Manager
NE	Numeric Error
NMI	Non-Maskable Interrupt
NPCT	GTNP Configuration Tools
NPSP	GEMSOS Network Processor System Parent
NPX	Numeric Processor Extension
NSA	National Security Agency
NSAD	Network Security Architecture and Design
NT	Nested Task Flag
NTCB	Network Trusted Computing Base
NW	Not Write-through
OEM	Original Equipment Manufacturers
OUI	Original User Installation
PAI	Preallocated Address Interface
PAL	Programmable Array Logic
PC	Personal Computer
PC-AT	Part of kit name for PC/AT
PC/AT	Personal Computer/AT
PCD	Page-level cache disable
PDR	Preliminary Design Review
PDS	Product Distribution Support
PE	Protection Enable bit
PG	Paging Flag
PIC	Programmable Interrupt Controller
PIT	Programmable Interrupt Timer
PL	Privilege Level
PLDN	Process Local Device Number
PLSN	Process Local Segment Number
PM	Process Manager
PROM	Programmable Read Only Memory
PTR	Preliminary Technical Review
PVAM	Protected Virtual Address Mode
PWT	Page-level write transparent
QA	Quality Assurance
QIC	Quarter-Inch Cartridge
R/W	Read/Write
RAM	Random Access Memory
RAMP	Ratings Maintenance Phase
RBn	Ring Bracket n

RCO	Responsible Corporate Officer
RF	Resume (from debug) Flag
RISC	Reduced Instruction Set Computer
RM	Ratings Maintenance
RMR	Ratings Maintenance Report
ROM	Read-Only Memory
RPL	Requested Privilege Level
RTC	Real-Time Clock
RVG	Random Value Generator
R*n*	Ring *n*
SA-Team	Security Analysis Team
SBC	Single Board Computer
SCC	Serial Communications Controller
SDR	System Discrepancy Report
SFUG	Security Features Users Guide
SG	Intersegment Linkage Layer
SIO	Serial I/O
SL	System Library; Intel designation for low-power technology
SM	Segment Manager
SMU	System Maintenance Utility
SP	Designation for a class of Gemini i386 processors; Stack Pointer (80*x*86)
SPBU	System PROM Bootstrap Utility
SPIC	Slave Programmable Interrupt Controller
SRAM	Static Random Access Memory
SRM	Shared Resource Matrix
SS	Stack Segment
SSIO	i386 Local Serial I/O
SSM	Secondary Storage Manager
ST	Streaming Tape
ST506	Seagate Technology 506 Interface. Also known as ST412.
SX	Intel designation for either an i386 processor with a 24-bit data path, or a i486 processor with no math coprocessor.
SX2	Designation for i486 SX processors with clock-doubled internal clocks
SYSGEN	GEMSOS System Generation Utility
Tbyte	1,099,511,627,776 bytes
TCB	Trusted Computing Base
TCSEC	Trusted Computer System Evaluation Criteria
TDI	Trusted Database Interpretation
TEMPEST	Not An Acronym
TF	Trap Flag
TFM	Trusted Facility Manual
TNI	Trusted Network Interpretation (of the Trusted Computer System Evaluation Criteria)
TPEP	Trusted Product Evaluation Program
TPOC	Technical Point of Contact
TRB	Technical Review Board
TSS	Task State Segment
UDM	Upper Device Manager
UI	User Interface
UNIX	Not an acronym

UP	Designation for a class of Gemini i486 processors
UTC	Upper Traffic Controller
VAP	Vendor Assistance Phase
VAT	Volume Alias Table
VD	Volume Device
VM	Virtual 8086 Mode Flag (Hardware); Virtual Memory
VP	Virtual Processor
VSA	Vendor Security Analyst
VTOC	Volume Table of Contents
VWG	Verification Working Group
WP	Write Protect

Appendix H

Bibliography and References

The following notations are used with respect to Gemini Computers, Incorporated documentation: PUBLIC, GEMINI PROPIN INTERNAL and GEMINI PROPIN CUSTOMER. GEMINI PROPIN INTERNAL indicates that the document is not released outside of Gemini Computers, Incorporated, and is for use internal to Gemini Computers, Incorporated. GEMINI PROPIN CUSTOMER indicates that the document is used both internally within Gemini as well as being releasable to Gemini customers. In both cases, the document is proprietary to Gemini Computers, Incorporated. Only when the PUBLIC indication is provided is a Gemini document publically releasable.

[1] American Micronics Inc. (AMI). *AMI Elephant-12 Board for IBM and Compatibles.* Irvine, CA.

[2] Bell, D. E., and La Padula, L. J. Computer Security Model: Unified Exposition and Multics Interpretation. Technical Report ESD-TR-75-306, The MITRE Corporation, Bedford, MA, June 1975.

[3] Biba, K. J. Integrity Considerations for Secure Computer Systems. Technical Report ESD-TR-76-372, The MITRE Corporation, Bedford, MA, April 1977.

[4] Central Data Corporation. *CD21/2032 High Density EPROM/RAM Board.* Champaign, IL, December 1986. Manual Order# CD91/2032-HMAN.

[5] Ciprico Inc. *Tapemaster 1000 Product Specification*, April 1987. 21014500A. (1/2″ magnetic tape adaptor).

[6] Eckmann, S. T. "Ina Flo: The FDM Flow Tool". *Proceedings of the 10th National Computer Security Conference* (September 1987), 175–182.

[7] Fellows, J., et al. The Architecture of a Distributed Trusted Computing Base. *Proceedings of 10th National Computer Security Conference* (September 1987), 68.

[8] Gambel, D. Panel, Gemini Developers: Facts, Myths, and War Stories. *Proceedings of The Fifth Aerospace Computer Security Applications Conference* (December 1989), 110.

[9] Gemini Computers Incorporated. *Gemini System Controller (GSC-AT) for PC/AT Systems, Hardware Specification.* Carmel, CA, January 1988. GSC0A-0101. GEMINI PROPIN INTERNAL. [Hardware Documentation].

[10] Gemini Computers Incorporated. *Gemini Trusted and Concurrent Processing Computers.* Carmel, CA, 1988. NDN. PUBLIC. [Design Documentation].

[11] Gemini Computers Incorporated. *GEMSOS Kernel Formal Top Level Specification (FTLS) Proofs.* Carmel, CA, September 1988. KER00-FTP01-0000. GEMINI PROPIN INTERNAL. [Formal Documentation].

[12] GEMINI COMPUTERS INCORPORATED. *Software Development Specification for GEMSOS Kernel Gate.* Carmel, CA, February 1989. KG000-SPB00-0101. GEMINI PROPIN INTERNAL. [Testing].

[13] GEMINI COMPUTERS INCORPORATED. *Hardware/Software Configuration Note.* Carmel, CA, December 1990. NDN. GEMINI PROPIN INTERNAL. [Evaluated Configuration].

[14] GEMINI COMPUTERS INCORPORATED. *Software Development Specification for Multilevel Subjects Intersegment Linkage Tool.* Carmel, CA, October 1990. MIT00-SPB01-0101. GEMINI PROPIN INTERNAL. [Design Documentation].

[15] GEMINI COMPUTERS INCORPORATED. *Software Development Specification for the Kernel Gate Library.* Carmel, CA, October 1990. KGL00-SPB00-0101A. GEMINI PROPIN INTERNAL. [Design Documentation].

[16] GEMINI COMPUTERS INCORPORATED. *Software Product Specification for GEMSOS Multilevel Subjects (CSCI MLS00).* Carmel, CA, October 1990. MLS00-SPC00-0101. GEMINI PROPIN INTERNAL. [Design Documentation].

[17] GEMINI COMPUTERS INCORPORATED. *Software Product Specification for Multilevel Subjects Intersegment Linkage Tool.* Carmel, CA, October 1990. MIT00-SPC00-0101. GEMINI PROPIN INTERNAL. [Design Documentation].

[18] GEMINI COMPUTERS INCORPORATED. *Software Product Specification for the Kernel Gate Library.* Carmel, CA, October 1990. KGL00-SPC00-0101A. GEMINI PROPIN INTERNAL. [Design Documentation].

[19] GEMINI COMPUTERS INCORPORATED. *Gemini Secure PC-AT Workstation Kit, Installation Manual, Volume I.* Carmel, CA, March 1991. PCK01-IM001-0000. GEMINI PROPIN CUSTOMER. [Hardware Documentation].

[20] GEMINI COMPUTERS INCORPORATED. *Gemini Secure PC-AT Workstation Kit, Installation Manual, Volume II, IBM PC/AT Models: 5170, 4459.* Carmel, CA, March 1991. PCK01-IM002-0100. GEMINI PROPIN CUSTOMER. [Hardware Documentation].

[21] GEMINI COMPUTERS INCORPORATED. *Gemini Secure PC-AT Workstation Kit, Installation Manual, Volume II, Zenith Z-248 Model ZBF-s337-BK.* Carmel, CA, March 1991. PCK01-IM002-0200. GEMINI PROPIN CUSTOMER. [Hardware Documentation].

[22] GEMINI COMPUTERS INCORPORATED. *GT Memory Monitor User Reference Manual.* Carmel, CA, October 1991. GT000-UM002-0001. GEMINI PROPIN CUSTOMER. [Testing].

[23] GEMINI COMPUTERS INCORPORATED. *GT Segment Monitor User Reference Manual.* Carmel, CA, December 1991. SMON0-UM001-0001. GEMINI PROPIN CUSTOMER. [Testing].

[24] GEMINI COMPUTERS INCORPORATED. *Kernel Call Monitor User Manual.* Carmel, CA, December 1991. GTAT0-UM019-0000. GEMINI PROPIN CUSTOMER. [Testing].

[25] GEMINI COMPUTERS INCORPORATED. *Software Development Specification for the GEMSOS Network Processor System Parent (CSCI NPSP0).* Carmel, CA, October 1991. NPSP0-SPB00-0101. GEMINI PROPIN INTERNAL. [Design Documentation].

[26] GEMINI COMPUTERS INCORPORATED. *System Specification for Gemini Trusted Network Processor.* Carmel, CA, March 1991. GTN00-SPA00-0301. Appendix IV: GEMSOS TCB: Philosophy and Design of Protection. GEMINI PROPIN INTERNAL. [Design Documentation].

[27] GEMINI COMPUTERS INCORPORATED. *System Specification for Gemini Trusted Network Processor.* Carmel, CA, March 1991. GTN00-SPA00-0201. Appendix III: GEMSOS TCB: Test Plan. GEMINI PROPIN INTERNAL. [Testing Documentation].

[28] GEMINI COMPUTERS INCORPORATED. *System Specification for Gemini Trusted Network Processor.* Carmel, CA, March 1991. GTN00-SPA00-0401. Appendix V: GEMSOS TCB Verification Plan. GEMINI PROPIN INTERNAL. [Formal Documentation].

[29] GEMINI COMPUTERS INCORPORATED. *Volume Monitor User Manual.* Carmel, CA, October 1991. VMON0-UM001-0001. GEMINI PROPIN CUSTOMER. [Testing].

[30] GEMINI COMPUTERS INCORPORATED. *Appendices to the Software Development Specification for GEMSOS Security Kernel (CSCI KER00).* Carmel, CA, December 1992. KER00-SPB00-0201D. [Design Documentation]. Includes:[1]

Appendix	Title	Number	In Eval Conf?
A2	Ethernet Device	GTN00-SFGP3-A021	No
A4	HDLC Device	GTN00-SFGP3-A041	No
A7	System PROM Device	GTN00-SFGP3-A070	Yes
A11	Local Serial I/O Device	GTN00-SFGP3-A111A	Yes
A12	Extended Serial I/O Device	GTN00-SFGP3-A121	Yes
A13	Disk Devices	GTN00-SFGP3-A131	Yes
A18	PC/AT Serial I/O Device	GTN00-SFGP3-A181	Yes
A19	PC/AT Keyboard Device	GTN00-SFGP3-A191	Yes
A20	PC/AT Enhanced Graphics Adapter Device	GTN00-SFGP3-A201	Yes
A21	Streaming Tape Device	GTN00-SFGP3-A211	Yes
A22	Volume Device	GTN00-SFGP3-A222B	Yes
A23	PC/AT Parallel Printer Device	GTN00-SFGP3-A231	Yes
A24	PC/AT Calendar Clock Device	GTN00-SFGP3-A241	Yes
A25	Half-Inch Tape Device	GTN00-SFGP3-A251	Yes
A26	Real-time Clock Device	GTN00-SFGP3-A261A	Yes
A27	PC/AT Binary Clock Device	GTN00-SFGP3-A271	Yes
A28	Key Management Device	GTN00-SFGP3-A283	Yes
A29	Data Encryption Device	GTN00-SFGP3-A292	Yes
A30	Gemini 386 Local Serial I/O Device	GTN00-SFGP3-A301	Yes
A32	Hardware Distribution Support Device	GTN00-SFGP3-A323	Yes
A33	Disk Unit Device	GTN00-SFGP3-A331	Yes
A34	IBM Channel-Type 1 Device	GTN00-SFGP3-A342	No
A35	Data Sealing Devices	GTN00-SFGP3-A352	Yes
A36	Random Value Generator Device	GTN00-SFGP3-A361	Yes
A37	Kernel Metering Support Device	GTN00-SFGP3-A371	Yes
A38	Expandable Ethernet Device	GTN00-SFGP3-A381	Yes
A39	Expandable HDLC Device	GTN00-SFGP3-A391A	Yes

GEMINI PROPIN CUSTOMER.

[1] The appendices cited are all for the Pascal Language. Equivalent versions are available for the C language with document numbers of the form GTN00-SFGC3-*nnnn*.

[31] GEMINI COMPUTERS INCORPORATED. *Device Test Monitor User Manuals.* Carmel, CA, July 1992. GEMINI PROPIN CUSTOMER. [Testing]. This is a set of 18 user manuals that describe the test monitors for various devices, all with document numbers of the form GTAT0-UM*nnn-mmmm*: Ethernet (UM003), HDLC (UM004), LSIO (UM005), ESIO (UM006), ASIO (UM007), PC/AT Keyboard (UM008), PC/AT EGA (UM009), Streaming Tape (UM010), Volume Device (UM011), PC/AT Parallel Printer Port (UM012), PC/AT Calendar Clock (UM013), RTC (UM014), ABC (UM015), DED (UM016), SSIO (UM017), IC1 (UM018), EETH (UM020), and EHDLC (UM021).

[32] GEMINI COMPUTERS INCORPORATED. *Gemini Trusted Multiple Microcomputer System Products: Commercial/OEM Price List.* Carmel, CA, October 1992. NDN. PUBLIC. [Evaluated Configuration].

[33] GEMINI COMPUTERS INCORPORATED. *Gemini Trusted Network Processor, Security Features User's Guide, Volume III, GEMSOS Device Appendices: Complete Appendices Set.* Carmel, CA, December 1992. GTN00-SFGP3-*nnnn*.[2]. The last part of the number (*nnnn*) is indicative of the configuration number (e.g., 0073A for PC/AT configuration 007, 0290 for Multibus configuration 029. 0013A is the complete set. The contents of these documents are drawn from KER00-SPB00-0201D[30]). GEMINI PROPIN CUSTOMER. [SFUG].

[34] GEMINI COMPUTERS INCORPORATED. *GEMSOS Model Application Environment User Manual [Pascal].* Carmel, CA, May 1992. MAE00-UMP01-0002A. GEMINI PROPIN CUSTOMER. [Test Scaffolding].

[35] GEMINI COMPUTERS INCORPORATED. *GTNP Formal Security Policy Model Proofs.* Carmel, CA, April 1992. GTN00-MDP01-0000. GEMINI PROPIN INTERNAL. [Formal Documentation].

[36] GEMINI COMPUTERS INCORPORATED. *Notes on GTNP Volume Allocation for Gemini Systems in GTNP Version 0.0.* Carmel, CA, August 1992. NDN. GEMINI PROPIN INTERNAL.

[37] GEMINI COMPUTERS INCORPORATED. *NPSP FTLS Proofs.* Carmel, CA, April 1992. NPSP0-FTP01-0000. GEMINI PROPIN INTERNAL. [Formal Documentation].

[38] GEMINI COMPUTERS INCORPORATED. *NPSP Functional Security Test Description.* Carmel, CA, April 1992. NPSP1 Version 1.0. GEMINI PROPIN INTERNAL. [Testing].

[39] GEMINI COMPUTERS INCORPORATED. *Original User Installation Manual.* Carmel, CA, December 1992. OUI00-UM001-0000. GEMINI PROPIN CUSTOMER. [TFM].

[40] GEMINI COMPUTERS INCORPORATED. *System Specification for Gemini Trusted Network Processor.* Carmel, CA, April 1992. GTN00-SPA00-0501A. Appendix VI: GTNP Formal Security Policy Model. GEMINI PROPIN INTERNAL. [Formal Documentation].

[41] GEMINI COMPUTERS INCORPORATED. *APGEN User Manual, GEMSOS Application Generator Utility.* Carmel, CA, December 1993. APG00-UM001-0003. GEMINI PROPIN CUSTOMER. [Testing].

[42] GEMINI COMPUTERS INCORPORATED. *Deficiency Summary for GEMSOS Security Kernel, Version 2.00.* Carmel, CA, July 1993. NDN. GEMINI PROPIN CUSTOMER.

[43] GEMINI COMPUTERS INCORPORATED. *DSKGEN User Manual, Hard Disk Conditioning Utility.* Carmel, CA, November 1993. DGN00-UM001-0004. GEMINI PROPIN CUSTOMER. [TFM].

[2] Pascal Presentation. C Language Presentation is GTN00-SFGC3-*nnnn*.

[44] Gemini Computers Incorporated. *Gemini System Hardware Manual, Volume 1.* Carmel, CA, December 1993. GSB00-HM001-0000. GEMINI PROPIN CUSTOMER. [Hardware].

[45] Gemini Computers Incorporated. *Gemini System Hardware Manual, Volume 2, Model 26.* Carmel, CA, December 1993. GSB02-HM001-0000. GEMINI PROPIN CUSTOMER. [Hardware].

[46] Gemini Computers Incorporated. *Gemini System Hardware Manual, Volume 2, Models 6, 9, 12, and 15.* Carmel, CA, December 1993. GSB01-HM001-0000. GEMINI PROPIN CUSTOMER. [Hardware].

[47] Gemini Computers Incorporated. *Gemini Test Controller User Manual.* Carmel, CA, September 1993. GTC00-UM001-0000A. GEMINI PROPIN CUSTOMER. [Testing].

[48] Gemini Computers Incorporated. *Gemini Trusted Network Processor, Security Features User's Guide, Volume II, Programmer's Guide to the GEMSOS Security Kernel Interface.* Carmel, CA, September 1993. GTN00-SFGP2-0004B.[3] GEMINI PROPIN CUSTOMER. [SFUG].

[49] Gemini Computers Incorporated. *Gemini Trusted Network Processor, Security Features User's Guide, Volume I, Introduction to the GEMSOS Security Kernel.* Carmel, CA, September 1993. GTN00-SFG01-0005A. GEMINI PROPIN CUSTOMER. [SFUG].

[50] Gemini Computers Incorporated. *Gemini Trusted Network Processor, Security Features User's Guide, Volume 4, Language Independent Interface for the GEMSOS Security Kernel.* Carmel, CA, September 1993. GTN00-SFG04-0001A. GEMINI PROPIN CUSTOMER. [SFUG].

[51] Gemini Computers Incorporated. *GEMSOS Test Application Environment User Reference Manual.* Carmel, CA, September 1993. GT000-UM001-0001B. GEMINI PROPIN CUSTOMER. [Testing].

[52] Gemini Computers Incorporated. *GT Acceptance Test Documentation: Test Description Reference Manual, Volume 1.* Carmel, CA, October 1993. GTAT0-RM001-0100. GEMINI PROPIN CUSTOMER. [Testing].

[53] Gemini Computers Incorporated. *Interface Requirements Specification for GEMSOS Multilevel Subjects (CSCI MLS00).* Carmel, CA, May 1993. MLS00-IRS01-0101A. GEMINI PROPIN INTERNAL. [Design Documentation].

[54] Gemini Computers Incorporated. *Interface Requirements Specification for the GEMSOS Security Kernel (CSCI KER00).* Carmel, CA, June 1993. KER00-IRS00-0101B. GEMINI PROPIN INTERNAL. [Design Documentation].

[55] Gemini Computers Incorporated. *Kernel Initialization Tables User Manual, Volume 1.* Carmel, CA, September 1993. KIT00-UM001-0101. GEMINI PROPIN CUSTOMER. [Initialization].

[56] Gemini Computers Incorporated. *Kernel Initialization Tables User Manual, Volume 2.* Carmel, CA, September 1993. KIT00-UM001-0201. GEMINI PROPIN CUSTOMER. [Initialization].

[57] Gemini Computers Incorporated. *Parameters for GEMSOS Security Kernel Generation.* Carmel, CA, June 1993. KER00-UM101-0003. GEMINI PROPIN CUSTOMER. [TFM].

[58] Gemini Computers Incorporated. *Product Distribution Support User Manual Volume 1.* Carmel, CA, September 1993. PDS00-UM001-0101. GEMINI PROPIN CUSTOMER.

[3] Pascal Presentation. C Language Presentation is GTN00-SFGC2-0004.

[59] GEMINI COMPUTERS INCORPORATED. *Product Distribution Support User Manual Volume 2.* Carmel, CA, September 1993. PDS00-UM001-0201. GEMINI PROPIN CUSTOMER.

[60] GEMINI COMPUTERS INCORPORATED. *Software Development Specification for GEMSOS Security Kernel Initialization (GKI00).* Carmel, CA, May 1993. GKI00-SPB00-0101C. GEMINI PROPIN INTERNAL. [Design Documentation].

[61] GEMINI COMPUTERS INCORPORATED. *Software Development Specification for GEMSOS Security Kernel (CSCI KER00).* Carmel, CA, July 1993. KER00-SPB00-0101F. GEMINI PROPIN INTERNAL. [Design Documentation].

[62] GEMINI COMPUTERS INCORPORATED. *Software Development Specification for Gemsos Security Kernel (CSCI KER00), GEMSOS Kernel Formal Top Level Specification (FTLS).* Carmel, CA, December 1993. KER00-SPB0-0301B. GEMINI PROPIN INTERNAL. [Formal Documentation].

[63] GEMINI COMPUTERS INCORPORATED. *Software Development Specification for Kernel Initialization Tables.* Carmel, CA, January 1993. KIT00-SPB00-0101. GEMINI PROPIN INTERNAL. [Initialization].

[64] GEMINI COMPUTERS INCORPORATED. *Software Development Specification for Product Distribution Support.* Carmel, CA, March 1993. PDS00-SPB00-0101A. GEMINI PROPIN INTERNAL.

[65] GEMINI COMPUTERS INCORPORATED. *Software Product Specification for Gemsos Security Kernel (CSCI KER00).* Carmel, CA, December 1993. KER00-SPC00-0101C. GEMINI PROPIN INTERNAL. [Design Documentation].

[66] GEMINI COMPUTERS INCORPORATED. *Software Product Specification for Gemsos Security Kernel (CSCI KER00), Appendix VII: Covert Storage Channel Analysis for the GEMSOS Kernel.* Carmel, CA, April 1993. KER00-SPC00-0301. GEMINI PROPIN INTERNAL. [Formal Documentation].

[67] GEMINI COMPUTERS INCORPORATED. *Software Product Specification for Gemsos Security Kernel (CSCI KER00), Appendix VIII: Covert Timing Channel Analysis for the GEMSOS Kernel.* Carmel, CA, November 1993. KER00-SPC00-0401. GEMINI PROPIN INTERNAL. [Formal Documentation].

[68] GEMINI COMPUTERS INCORPORATED. *Software Product Specification for GEMSOS Security Kernel Initialization (GKI00).* Carmel, CA, December 1993. GKI00-SPC00-0101C. GEMINI PROPIN INTERNAL. [Design Documentation].

[69] GEMINI COMPUTERS INCORPORATED. *Software Product Specification for Kernel Initialization Tables.* Carmel, CA, June 1993. KIT00-SPC00-0101A. GEMINI PROPIN INTERNAL. [Initialization].

[70] GEMINI COMPUTERS INCORPORATED. *Software Product Specification for Kernel Initialization Tables, Appendix I, Detailed Product Specification.* Carmel, CA, June 1993. KIT00-SPC00-0201. GEMINI PROPIN INTERNAL. [Initialization].

[71] GEMINI COMPUTERS INCORPORATED. *Software Product Specification for Product Distribution Support, Appendix I, Detailed Product Specification.* Carmel, CA, September 1993. PDS00-SPC00-0201A. GEMINI PROPIN INTERNAL.

[72] GEMINI COMPUTERS INCORPORATED. *Software Product Specification for Product Distribution Support, Volume 1.* Carmel, CA, September 1993. PDS00-SPC00-0101B. GEMINI PROPIN INTERNAL.

[73] Gemini Computers Incorporated. *Software Product Specification for the GEMSOS Security Kernel, Appendices X, XI, and XIII, Kernel Code Correspondence Data (KER00).* Carmel, CA, September 1993. KER00-SPC00-0501. Replaces GEM00-CC001-0000. GEMINI PROPIN INTERNAL. [Formal Documentation].

[74] Gemini Computers Incorporated. *Standard User Interface Package User Reference Manual.* Carmel, CA, March 1993. UI000-UM001-0002. GEMINI PROPIN CUSTOMER. [Testing].

[75] Gemini Computers Incorporated. *SYSGEN User Manual, System Generation Utility.* Carmel, CA, November 1993. SGN00-UM001-0004. GEMINI PROPIN CUSTOMER. [TFM].

[76] Gemini Computers Incorporated. *Test Plan to Test Mapping Document.* Carmel, CA, August 1993. Contained in [83]. [Testing].

[77] Gemini Computers Incorporated. *Deficiency Summary for GEMSOS Security Kernel, Version 2.00a.* Carmel, CA, January 1994. NDN. GEMINI PROPIN CUSTOMER.

[78] Gemini Computers Incorporated. *Deficiency Summary for the Gemini Trusted Network Processor Version 1.00 and System Generation Tools Product Version 1.00.* Carmel, CA, January 1994. NDN. GEMINI PROPIN CUSTOMER.

[79] Gemini Computers Incorporated. *Gemini Configuration Management Plan.* Carmel, CA, March 1994. GCI00-CMP01-0002. GEMINI PROPIN INTERNAL. [Configuration Management].

[80] Gemini Computers Incorporated. *GEMSOS Hardware Distribution Utility Reference Manual.* Carmel, CA, January 1994. HDU00-RM0001-0001. GEMINI PROPIN CUSTOMER. [TFM].

[81] Gemini Computers Incorporated. *GEMSOS Mandatory Security Administrator Reference Manual.* Carmel, CA, June 1994. MAC00-RM001-0003A. GEMINI PROPIN CUSTOMER. [TFM].

[82] Gemini Computers Incorporated. *GEMSOS Security Features User's Guide, Volume 4, User Guide to the GEMSOS Trusted Path.* Carmel, CA, January 1994. GEM00-SFG04-0002. GEMINI PROPIN CUSTOMER. [TFM].

[83] Gemini Computers Incorporated. *GT Acceptance Test Documentation, Test Description Reference Manual, Volume 2, Special Tests.* Carmel, CA, January 1994. GTAT0-RM001-0200. GEMINI PROPIN INTERNAL. [Testing].

[84] Gemini Computers Incorporated. *GTNP Configuration Tool Operator Reference Manual.* Carmel, CA, January 1994. NPCT0-UM001-0004. GEMINI PROPIN CUSTOMER. [TFM].

[85] Gemini Computers Incorporated. *GTNP Mandatory Security Administrator Reference Manual.* Carmel, CA, January 1994. NPCT0-RM001-0003. GEMINI PROPIN CUSTOMER. [TFM].

[86] Gemini Computers Incorporated. *Guide to Generating the Gemini Trusted Network Processor from Source Code.* Carmel, CA, January 1994. GTN00-TFM02-0001. [TFM].

[87] Gemini Computers Incorporated. *Guide to Trusted Facility Management for the Gemini Trusted Network Processor.* Carmel, CA, June 1994. GTN00-TFM01-0005. GEMINI PROPIN CUSTOMER. [TFM].

[88] Gemini Computers Incorporated. *Hardware Development Specification for Gemini System Controller for Multibus I Systems.* Carmel, CA, January 1994. GSC01-SPB00-0101. GEMINI PROPIN INTERNAL. [Hardware Documentation].

[89] GEMINI COMPUTERS INCORPORATED. *Manufacturing Operating Procedure: Software Development Standards.* Carmel, CA, April 1994. GOP:ENG-0002-02. GEMINI PROPIN INTERNAL. [Design Documentation].

[90] GEMINI COMPUTERS, INCORPORATED. *Product Creation Book for the Gemini Trusted Network Product.* Carmel, CA, August 1994. GCI00-IM101-0101 (Domestic); -0201 (Export). GEMINI PROPIN CUSTOMER. [TFM].

[91] GEMINI COMPUTERS, INCORPORATED. *Product Creation Book for the System Generation Tools Product.* Carmel, CA, August 1994. GCI00-IM102-0101. GEMINI PROPIN CUSTOMER. [TFM].

[92] GEMINI COMPUTERS INCORPORATED. *Product Installation Manual.* Carmel, CA, July 1994. GCI00-IM001-0002. GEMINI PROPIN CUSTOMER. [TFM].

[93] GEMINI COMPUTERS, INCORPORATED. *Product Update Book for the Gemini Trusted Network Product.* Carmel, CA, August 1994. GCI00-IM201-0100 (Domestic); -0200 (Export). GEMINI PROPIN CUSTOMER. [TFM].

[94] GEMINI COMPUTERS, INCORPORATED. *Product Update Book for the System Generation Tools Product.* Carmel, CA, August 1994. GCI00-IM202-0100. GEMINI PROPIN CUSTOMER. [TFM].

[95] GEMINI COMPUTERS INCORPORATED. *System Maintenance Utility User Manual.* Carmel, CA, May 1994. SMU00-UM0001-0002. GEMINI PROPIN CUSTOMER. [TFM].

[96] GEMINI COMPUTERS INCORPORATED. *System PROM Bootstrap Utility User Manual.* Carmel, CA, June 1994. SPBU0-UM001-0001A. GEMINI PROPIN CUSTOMER. [TFM].

[97] GEMINI COMPUTERS INCORPORATED. *System Specification for Gemini Trusted Network Processor.* Carmel, CA, February 1994. GTN00-SPA00-0101B. Includes Appendix I, "Gemini Hardware Base" and Appendix II, "Network Security Architecture and Design. GEMINI PROPIN INTERNAL. [Design Documentation].

[98] GEMINI COMPUTERS INCORPORATED. *System Specification for the Gemini Trusted Network Processor, Appendix VIII, GTNP Rating Maintenance Plan.* Carmel, CA, February 1994. GTN00-SPA00-0701A. GEMINI PROPIN INTERNAL. [Configuration Management].

[99] GEMINI COMPUTERS INCORPORATED. PUBLIC. *System Overview: Gemini Trusted Multiple Microcomputer Base (Version 0).* Carmel, CA, May 1984. Revision 2.0. GEM00-OVS01-0000. [Design Documentation].

[100] IBM CORPORATION. *IBM PC/AT Technical Reference Manual Change Pages for the 8MHz PC/AT (submodel 339).* Boca Raton, FL. Order# S229-9608-00, Part# 6280099.

[101] IBM CORPORATION. *IBM PC/AT Technical Reference Manual.* Boca Raton, FL, September 1985. Order# S229-9611-00, Part# 6280070.

[102] IBM CORPORATION. *IBM Technical Reference: Options and Adapters (includes Enhanced Graphics Adaptor).* Boca Raton, FL, March 1986. Order# SS34-0007-00, Part# 6280131.

[103] IBM CORPORATION. *IBM Technical Reference: Options and Adapters (includes Serial/Parallel Adaptor).* Boca Raton, FL, March 1986. Order# SS34-0010-00, Part# 6280134.

[104] INTEL CORPORATION. *iAPX 286 Hardware Reference Manual.* Santa Clara, CA, 1983. Order# 210760-001.

[105] INTEL CORPORATION. *iSBC® 028CX/056CX/012CX/010CX/020CX (CX-Series) RAM Boards Hardware Reference Manual.* Santa Clara, CA, 1984. Order# 145158-003.

[106] INTEL CORPORATION. *iSBC® 286/12 Hardware Reference Manual.* Santa Clara, CA, 1985. Order# 147533-001.

[107] INTEL CORPORATION. *80286 Operating Systems Writers Guide.* Santa Clara, CA, 1986. Order# 121960-002.

[108] INTEL CORPORATION. *iSBC® 386/21/22/24/28 Single Board Computer Hardware Reference Manual.* Santa Clara, CA, 1986. Order# 149094-001. Change Notice 1 – Order# 451626-001. Change Notice 2 – Order# 455539-001.

[109] INTEL CORPORATION. *80286 and 80287 Programmer's Reference Manual.* Santa Clara, CA, 1987. Order# 210498-005.

[110] INTEL CORPORATION. *The Installation Guide for the Above™ Board Plus 8 and Plus 8 I/O.* Santa Clara, CA, 1987. Order# PCEO 301913-004.

[111] INTEL CORPORATION. *The Installation Guide for the Above™ Board Plus and Plus I/O.* Santa Clara, CA, 1987. Order# PCEO 301661-007.

[112] INTEL CORPORATION. *Installing the Above™ Board 286 and Above™ Board PS/286.* Santa Clara, CA, 1987. Order# PCEO 300805-002B.

[113] INTEL CORPORATION. *Installing the Above™ Board PS/AT.* Santa Clara, CA, 1987. Order# PCEO 300469-001B.

[114] INTEL CORPORATION. *iSBC® 188/56 Advanced Communications Computer Hardware Reference Manual.* Santa Clara, CA, 1987. Order# 148209-002.

[115] INTEL CORPORATION. *iSBC® 386/31/32/34/38 Single Board Computer Hardware Reference Manual.* Santa Clara, CA, 1987. Order# 453652-001. Change Notice – Order# 505142-001.

[116] INTEL CORPORATION. *387 DX User's Manual/Programmer's Reference.* Santa Clara, CA, 1989. Order# 231917-002.

[117] INTEL CORPORATION. *80186/188, 80C186/C188 Hardware Reference Manual.* Santa Clara, CA, 1989. Order# 270788-001.

[118] INTEL CORPORATION. *iSBC® 186/51 IEEE 802.3/Ethernet Communication Computer Hardware Reference Manual.* Santa Clara, CA, October 1989. Order# 122330-002.

[119] INTEL CORPORATION. *386™ DX Microprocessor Hardware Reference Manual.* Santa Clara, CA, 1990. Order# 231732-004.

[120] INTEL CORPORATION. *i486™ Microprocessor Hardware Reference Manual.* Santa Clara, CA, 1990. Order# 240552-001.

[121] INTEL CORPORATION. *Intel 8-/16-Bit LAN Computer Users Manual.* Santa Clara, CA, 1991. Order # 296852-001. Includes information on 82586 IEEE 802.3 Ethernet Board.

[122] INTEL CORPORATION. *Intel486™ Microprocessor Family Programmer's Reference Manual.* Santa Clara, CA, 1992. Order# 240486-002, ISBN 1-55512-159-4.

[123] INTEL CORPORATION. *iSBC® 486/12 Series Single Board Computers Hardware Reference Manual.* Santa Clara, CA, September 1992. Order# 507914-003. Revision 003.

[124] INTEL CORPORATION. *Intel Connectivity.* Santa Clara, CA, 1993. Order# 231658-008. Includes information on 82586 IEEE 802.3 Ethernet Board and 8273 Programmable HDLC/SDLC Controller.

[125] INTERPHASE CORPORATION. *Storager™ High-Performance Multibus® Disk/Tape Controller User's Guide (Storager "I" and "II").* Dallas, TX, April 1989. UG-0580-000-040.

[126] INTERPHASE CORPORATION. *Storager™ III High-Performance Multibus® I ESDI/Floppy/Tape Controller User's Guide.* Dallas, TX, November 1989. UG-1050-000-XOA.

[127] JANSEN, P. *Using Type Extension to Organize Virtual Memory Mechanisms.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, August 1976. MIT/LCS/TR-167.

[128] KEMMERER, R. A. "A Practical Approach to Identifying Storage and Timing Channels". *Proceedings of the 1982 Symposium on Security and Privacy* (April 1982), 66–73.

[129] KENNEDY COMPANY (A DIVISION OF SHUGART CORPORATION). *Model 9600A/9650A Tape Drive Installation and Operation Manual.* Monrovia, CA, February 1988. 93-09600-997.

[130] LEVIN, T., AND PADILLA, S. Covert Storage Channel Analysis: A Worked Example. *Proceedings of the 13th National Computer Security Conference* (October 1990).

[131] LEVIN, T., PADILLA, S., AND IRVINE, C. A Formal Model for UNIX Setuid. *Proceedings of 1989 Symposium on Security and Privacy* (May 1989).

[132] LEVIN, T., TAO, A., THOMPSON, M., AND SCHELL, R. Modeling of GEMSOS I/O Devices. Technical Note GCI-91-01-01, Gemini Computers Incorporated, Carmel, CA, April 1991. GEMINI PROPIN INTERNAL.

[133] LUNT, T., DENNING, D., SCHELL, R., ET AL. The SeaView Security Model. *IEEE Trans. Softw. Eng. 16*, 6 (June 1990).

[134] MAXTOR CORPORATION. *XT-1000 Specification and OEM Manual.* San Jose, CA, 1985. Part# 101101 Rev H.

[135] MILLEN, J. K. Finite-State Noiseless Covert Channels. In *Proceedings of The Computer Security Foundations Workshop II* (June 1989), The IEEE Computer Society Press.

[136] NATIONAL COMPUTER SECURITY CENTER. *Department of Defense Trusted Computer System Evaluation Criteria.* Linthicum, MD, December 1985. DoD 5200.28-STD.

[137] NATIONAL COMPUTER SECURITY CENTER. *Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria.* Linthicum, MD, July 1987. NCSC-TG-005 Version 1.

[138] NATIONAL COMPUTER SECURITY CENTER. *Computer Security Subsystem Interpretation of the Trusted Computer System Evaluation Criteria.* Linthicum, MD, September 1988. NCSC-TG-009 Version 1.

[139] NATIONAL COMPUTER SECURITY CENTER. *Trusted Database Interpretation of the Trusted Computer System Evaluation Criteria.* Linthicum, MD, 1991.

[140] National Institute of Standards and Technology. Data Encryption Standard. FIPS PUB 46, National Technical Information Service, Springfield, VA, 1977.

[141] National Security Agency. B2 RAMP Requirements. Transaction `[0268]`, `Eval_Announcements` forum, `dockmaster.ncsc.mil`, September 1992. (posted by `Menk.CPE`).

[142] Newbury Data Recording Limited. *1000 Range Disc Drives User Manual.* Staines, Middlesex, ENGLAND, 1985. 55510064-B.

[143] Parnas, D. L. A Technique for Software Module Specification with Examples. *Commun. ACM 15*, 5 (May 1972), 330–336.

[144] Parnas, D. L. On the Criteria to be Used in Decomposing Systems into Modules. *Commun. ACM* (December 1972), 1053–1058.

[145] Price, W. R. *Implications of a Virtual Memory Mechanism for Implementing Protection in a Family of Operating Systems.* PhD thesis, Carnegie-Mellon University, June 1973.

[146] Reed, D. P. Processor Multiplexing in a Layered Operating System. Tech. Rep. ESD-TR-78-151, MIT/LCS/TR-164, Massachusetts Institute of Technology, Cambridge, MA, June 1976.

[147] Reed, D. P., and Kanodia, R. K. Synchronization with Eventcounts and Sequencers. *Commun. ACM 22*, 2 (February 1979), 115–124.

[148] Rushby, J., and Randell, B. A Distributed Secure System. *Computer 16*, 7 (July 1983), 55–67.

[149] Scheid, J., Anderson, S., Martin, R., and Holtzberg, S. The Ina-Jo™ Specification Language Reference Manual – Release 1. TM 6021/001/02, System Development Corporation, Santa Monica, CA, 1986. This company is currently known as Unisys Defense Systems and is located in McLean, VA.

[150] Schell, R. R., and Tao, T. F. Microcomputer-Based Trusted Systems for Communication and Workstation Applications. *7th DoD/NBS Computer Security Initiative Conference* (September 1984), 277.

[151] Schell, R. R., Tao, T. F., and Heckman, M. Designing the GEMSOS Security Kernel for Security and Performance. *Proceedings of the 8th National Computer Security Conference* (September 1985), 108.

[152] Schell, R. R., Thompson, M. F., and Shockley, W. R. A Network of Trusted Systems. *Proceedings of the AIAA/ASIS/IEEE Third Aerospace Computer Security Conference* (December 1987), 140.

[153] Schroeder, M. D., Clark, D. D., and Saltzer, J. H. The Multics Kernel Design Project. *Proceedings of the Sixth ACM Symposium on Operating Systems Principles* (November 1977), 43.

[154] Seagate Technology. *ESDI Specification.* Scotts Valley, CA. Publication# 77738076.

[155] Seagate Technology. *ST4026/4038/4051 Product Manual.* Scotts Valley, CA, February 1987. Order# 36038-001, Revision A.

[156] Seagate Technology. *Wren 3 ST2106E (94216-106) Product Manual.* Scotts Valley, CA, 1990. Publication# 77765276-E. (100 Mbyte fixed-disk drive).

[157] SEAGATE TECHNOLOGY. *Wren 6 ST2182E (94246-182), ST2383E (94246-383) Product Manual.* Scotts Valley, CA, 1990. Publication# 77765369-C. (350 Mbyte fixed-disk drive).

[158] SEIKO EPSON CORPORATION. *5.25 Inch Floppy Disk Drive SD-580L.*

[159] SEIKO EPSON CORPORATION. *5.25 Inch Floppy Disk Drive, 50-621L-532, -533, Rev 1*, October 1989. 621L-E06-REV.1.

[160] SEIKO EPSON CORPORATION. *5.25 Inch Mini-Floppy Disk Drive, SD-680L-530, -531, -537*, October 1989. 680L-409-REV.1.

[161] SHANNON, C. E., AND WEAVER, W. *The Mathematical Theory of Communications.* The University of Illinois Press, Urbana, IL, 1964.

[162] SHOCKLEY, W. R. Implementing the Clark/Wilson Integrity Policy Using Current Technology. *Proceedings of the 11th National Computer Security Conference* (October 1988), 29.

[163] SHOCKLEY, W. R., AND SCHELL, R. R. TCB Subsets for Incremental Evaluation. *Proceedings of the AIAA/ASIS/IEEE Third Aerospace Computer Security Conference* (December 1987), 131.

[164] SHOCKLEY, W. R., TAO, T. F., AND THOMPSON, M. F. An Overview of the GEMSOS Class A1 Technology and Application Experience. *Proceedings of 11th National Computer Security Conference* (October 1988), 238.

[165] SIBERT, O., PORRAS, P., AND LINDELL, R. The Intel 80x86 Processor Architecture: Pitfalls for Secure Systems. *Submitted to 1995 Symposium on Security and Privacy* (May 1995).

[166] TANDBERG DATA A/S. *TDC 3610/3630/3650 Reference Manual.* Oslo, NORWAY, June 1988. 411387 Rev. 1. (Streaming Tape Cartridge).

[167] TAO, T. F. Four High Assurance Multilevel Secure Systems Using Commercial Off-the-Shelf Gemini Trusted Computer Products. *Proceedings of the Sixth Annual Symposium and Technical Displays on Physical and Electronic Security (AFCEA)* (1990), A5–1.

[168] THOMPSON, M., SCHELL, R. R., TAO, A., AND LEVIN, T. Introduction to the Gemini Trusted Network Processor. *Proceedings of the 13th National Computer Security Conference* (October 1990), 211.

[169] TURLEY, J. L. *Advanced 80386 Programming Techniques.* Osborne McGraw-Hill, Berkeley, CA, 1988. ISBN 0-07-881342-5.

[170] WEISSMAN, C. BLACKER: Security for the DDN. Examples of A1 Security Engineering Trades. *Proceedings of the 1992 IEEE Symposium on Security and Privacy* (May 1992), 286–292. Presented in May 1988; Published in May 1992.

[171] ZENITH DATA SYSTEMS/GROUPE BULL. *80286 Desktop Computers Technical Reference Manual.* St. Joseph, MI, 1988. Part No. 595-3951-01.

[172] ZILOG CORPORATION. *Z8000™ Family Data Book.* Campbell, CA, November 1988. Order# 00-2488-01. (Data Ciphering Processor).