Security Supplement

to the

Software Communications Architecture Specification

JTRS-5000 SEC
V2.2.1
April 30, 2004

Prepared by
Joint Tactical Radio System (JTRS) Joint Program Office

Revision Summary

1.0	Initial Release
1.1	Incorporate approved Change Proposals, numbers 382, 385, 482, 483, 484, 494
2.2.1	Document numbering change for consistency with SCA main document numbering. Incorporate approved Change Proposals, numbers CP SCA-CCM 112, 113, 114, 115, 116, 36, 77

Table of Contents

1 INTRODUCTION. 1-1
1.1 Scope. 1-1
1.2 Background. 1-1
1.2.1 Radio Security. 1-1
1.2.1.1 Traditional Secure Environment. 1-1
1.2.1.2 Embedded Cryptography. 1-2
1.2.1.3 Networked Radios. 1-3
1.3 JTRS Architecture. 1-4

2 REFERENCES. 2-1
2.1 Documents. 2-1
2.2 Definitions of Terms. 2-1
2.2.1 Definition of Algorithm 2-1
2.2.2 Definitions of User Entities. 2-1
2.2.3 Definitions of Levels of Cryptography. 2-2

3 JTRS ARCHITECTURE OVERVIEW. 3-1
3.1 System Description. 3-1
3.2 Hardware. 3-3
3.3 Software. 3-4
3.3.1 Operating Environment. 3-4
3.3.1.1 Operating System. 3-4
3.3.1.2 Board Support Packages and Hardware Drivers. 3-4
3.3.1.3 Middleware. 3-4
3.3.1.4 Core Framework (CF). 3-5
3.3.2 Application Software. 3-5
3.3.2.1 System Applications. 3-5
3.3.2.1.1 Core System Applications. 3-6
3.3.2.1.2 Implementation Dependent Systems Applications. 3-6
3.3.2.2 Communicator Waveforms and Applications. 3-6
3.3.2.3 Radio Boot and Run Time Environment. 3-7
3.4 Radio Operating Policy and Doctrine (No Security). 3-7
3.5 JTRS Infrastructure. 3-8
3.5.1 Hardware Maintenance/Replacement. 3-8
3.5.2 Software Composition and Waveforms. 3-8
3.5.3 Key Management Infrastructure. 3-11
3.5.4 Mission Planning/Execution. 3-12
3.5.5 Radio Control/Status Bypass. 3-13
3.5.5.1 Radio Control Bypass Discussion. 3-13
3.6 SCA Impacts. 3-18
3.6.1 Operating System. 3-18
*3.6.1.1 FileSystem*s Discussion. 3-18
3.6.1.2 Approach. 3-19
3.6.2 Middleware. 3-19
3.6.2.1 Discussion. 3-19
3.6.2.2 Approach. 3-19
3.6.3 Core Framework. 3-20
3.6.3.1 Discussion. 3-20
3.6.3.2 Approach. 3-20
3.6.4 Guards. 3-22
3.6.4.1 Discussion. 3-23
3.6.4.2 Approach. 3-23

4 JTRS SECURITY ARCHITECTURE REQUIREMENTS. 4-1
4.1 Summary. 4-1
4.1.1 Security Boundaries. 4-4
4.1.1.1 Cryptographic Boundary. 4-4
4.1.1.2 INFOSEC Boundary. 4-4
4.1.1.3 Equipment Level Boundary. 4-4
4.1.2 Overall Security Architecture Requirements. 4-5
4.2 Cryptographic Subsystem Requirements. 4-5
4.2.1 General Security Requirements. 4-6
4.2.2 CS/S Initialization, Operation, and Termination. 4-7
4.2.2.1 CS/S Initialization, Operation, and Termination Discussion. 4-7
4.2.2.2 CS/S Initialization, Operation, and Termination Requirements. 4-8
4.2.2.2.1 CS/S Boot Requirements. 4-8
4.2.2.2.2 CS/S Application Instantiation Requirements. 4-8
4.2.2.2.3 CS/S Run-Time Requirements. 4-8
4.2.2.2.4 CS/S Normal Termination (Channel Teardown). 4-9
4.2.2.2.5 CS/S Abnormal Termination Requirements. 4-9
4.2.3 Keystream Functions. 4-9
4.2.3.1 Keystream Discussion. 4-9
4.2.3.2 Keystream Requirements. 4-10
4.2.3.2.1 Encrypt Requirements. 4-10
4.2.3.2.2 Decrypt Requirements. 4-11
4.2.3.2.3 TRANSEC Requirements. 4-11
4.2.4 Identification and Authentication Functions. 4-11
4.2.4.1 Identification and Authentication Discussion. 4-11
4.2.4.2 Identification and Authentication Requirements. 4-12
4.2.5 Integrity Function. 4-12
4.2.5.1 Integrity Discussion. 4-12
4.2.5.2 Integrity Requirements. 4-12
4.2.6 Security Management Functions. 4-12
4.2.6.1 Key Management Functions. 4-13
4.2.6.1.1 Key Management Discussion. 4-13
4.2.6.1.2 Key Management Requirements. 4-14
4.2.6.2 Algorithm Management Functions. 4-17
4.2.6.2.1 Algorithm Management Discussion. 4-17
4.2.6.2.2 Algorithm Management Requirements. 4-17
4.2.6.3 Security Policy Management Functions. 4-18
4.2.6.3.1 Security Policy Management Discussion. 4-18
4.2.6.3.2 Security Policy Management Requirements. 4-19
4.2.6.4 Security Critical Software Discussion. 4-19
4.2.6.5 Security Critical Software Requirements. 4-19
4.2.7 Cryptographic Bypass. 4-20
4.2.7.1 Communicator Information Bypass. 4-21
4.2.7.1.1 Communicator Information Bypass Discussion. 4-21
4.2.7.1.2 Communicator Information Bypass Requirements. 4-21
4.2.7.2 Radio Control/Status Bypass. 4-21
4.2.7.2.1 Radio Control/Status Bypass Requirements. 4-22
4.2.8 Cryptographic Control and Status Functions. 4-23
4.2.8.1 Cryptographic Control and Status Discussion. 4-23
4.2.8.2 Cryptographic Control and Status Requirements. 4-23
4.3 Requirements Within The INFOSEC Boundary. 4-23
4.3.1 Background Information. 4-23
4.3.1.1 INFOSEC Boundary Discussion. 4-24
4.3.1.2 INFOSEC Boundary Requirements. 4-25
4.3.2 Process Separation. 4-25

4.3.2.1 Process Separation Discussion. 4-25
4.3.2.2 Process Separation Requirements. 4-25
4.3.3 Discretionary Access Control. 4-26
4.3.3.1 Discretionary Access Control Discussion. 4-26
4.3.3.2 Requirements. 4-26
4.3.4 Identification and Authentication. 4-26
4.3.4.1 Identification and Authentication Discussion. 4-26
4.3.4.2 Requirements. 4-26
4.3.5 Object Reuse. 4-27
4.3.5.1 Object Reuse Discussion. 4-27
4.3.5.2 Object Reuse Requirements. 4-27
4.3.6 System Integrity. 4-27
4.3.6.1 System Integrity Discussion. 4-27
4.3.6.2 System Integrity Requirements. 4-27
4.3.7 Core Systems Applications. 4-27
4.3.7.1 Security API. 4-28
4.3.7.1.1 Security API Discussion. 4-28
4.3.7.1.2 Security API Requirements. 4-36
4.3.7.2 HMI Security Policy Enforcement. 4-36
4.3.7.2.1 HMI Security Policy Enforcement Discussion. 4-37
4.3.7.2.2 HMI Security Policy Enforcement Requirements. 4-38
4.3.7.3 Installer. 4-38
4.3.7.3.1 Installer Discussion. 4-38
4.3.7.3.2 Installer Requirements. 4-39
4.3.7.4 Audit. 4-39
4.3.7.4.1 Audit Discussion. 4-40
4.3.7.4.2 Audit Requirements. 4-40
4.3.7.5 Cross-Banding. 4-41
4.3.7.5.1 Cross-Banding Discussion. 4-41
4.3.7.5.2 Cross-Banding Requirements. 4-41
4.4 Equipment Level Boundary Functions. 4-41
4.4.1 Tempest. 4-42
4.4.1.1 TEMPEST Requirements. 4-42
4.4.2 Tamper. 4-42
4.4.2.1 Tamper Requirement. 4-42
4.4.3 Electrical Protection. 4-42
4.4.3.1 Electrical Protection Requirement. 4-42
4.5 JTRS Security Policy. 4-42
4.5.1 Security Policy Discussion. 4-43
4.5.2 Security Policy Requirements. 4-45
4.6 SCA Behavior Impacts. 4-45
4.6.1 Key and Waveform Selection. 4-45
4.6.1.1 Key Selection Requirements 4-45
4.6.2 Software Installation. 4-45
4.6.2.1 Software Installation Requirements. 4-46
4.6.3 Power-up Sequence. 4-46
4.6.3.1 Power-up Sequence Requirements. 4-46
4.6.4 Instantiation. 4-46
4.6.4.1 Instantiation Requirement. 4-46
4.6.5 Keep-Alive Ping. 4-46
4.6.5.1 Keep-Alive Ping Requirements. 4-46
4.6.6 Operating System. 4-46
4.6.6.1 Operating System Requirements. 4-46

5 GLOSSARY AND DEFINITIONS. 5-1

6 ACRONYMS AND ABBREVIATIONS. 6-1

APPENDIX A FUNCTIONAL SECURITY REQUIREMENTS FOR JTRS

ATTACHMENT 1 SECURITY APPLICATION PROGRAM INTERFACE SERVICE DEFINITION

List of Figures

Figure 1-1. Secure Channel Configuration 1-1
Figure 1-2. Cryptography Embedded in Radio 1-2
Figure 1-3. Networked Radio 1-3
Figure 1-4. JTRS Radio Architecture 1-4
Figure 3-1. JTRS Software Communications Architecture 3-2
Figure 3-2. JTRS Architectural Functional Layers 3-2
Figure 3-3. JTRS Run Time Waveforms Elements 3-10
Figure 3-4. Waveform Elements 3-11
Figure 3-5. JTR Mission Download 3-12
Figure 3-6. CORBA Stack Diagram 3-13
Figure 3-7. Local Object Communication 3-14
Figure 3-8. Communications Model with ORB Bypassed 3-15
Figure 3-9. Bypass Example Using a Proxy Mechanism 3-15
Figure 3-10. Bypass Using a Security API Approach 3-16
Figure 3-11. Alternative Bypass Approach without Proxy 3-16
Figure 3-12. Bypass Considerations Using ESIOP 3-17
Figure 3-13. Examples of Guards as a Resource 3-24
Figure 3-14. ApplyLabel Use Case 3-25
Figure 3-15. Processing Data from the Guard Bus 3-25
Figure 3-16. Example of a Port Specific Guard 3-26
Figure 3-17. Example of a Security Device with a Guard 3-28
Figure 4-1. JTRS Functional Security Allocation 4-1
Figure 4-2. Basic JTRS Security Architecture 4-2
Figure 4-3. JTRS Security Boundaries 4-3
Figure 4-4. Cryptographic Boundary Functions (The Cryptographic Subsystem) 4-7
Figure 4-5. Types of Keystream Utilization 4-10
Figure 4-6. Cryptographic Bypass Types 4-20
Figure 4-7. Security Guard and Monitor Supplements 4-24
Figure 4-8. MAC Adapters w/o Security API 4-29
Figure 4-9. MAC Adapters Using Security APIs 4-30
Figure 4-10. JTRS Security Service Groups 4-31
Figure 4-11. JTRS Security Device 4-34
Figure 4-12. HMI Security Policy Enforcement 4-38

List of Tables

Table 4-1 Cross-Reference of Services and Primitives 4-32
Table 4-2. Security Policy and Enforcement 4-44
Table 6-1. Acronyms and Abbreviations 6-1

1 INTRODUCTION.

1.1 SCOPE.

The purpose of this document is to delineate the government security requirements for JTRS.

The remainder of Section 1 provides some historical background of how communication systems and security have been accomplished. Section 2 provides a list of applicable documents that have been used to generate the Security Supplement. Section 3 is an explanation of the Software Communications Architecture (SCA) from a security viewpoint. Section 3 does not contain requirements. Section 4 provides the security requirements at an architectural level to build a Joint Tactical Radio System (JTRS) radio. Section 4 has been written around the concept of "security in layers". Sufficient requirements have been included in Section 4 to build, at a minimum, a system high radio with type separation. Implementers are cautioned to understand the requirements and the potential assurance level that can be associated with the requirement implementation. It is the requirement's implementation and the associated assurance levels that will determine if a radio will satisfy a 'system high' only radio requirement or a radio that will satisfy 'Multiple Security Level' radio requirement. Section 5 provides the glossary and definitions. Section 6 contains an acronym and abbreviations that have been used throughout the supplement. There are two appendices to this supplement. The Appendix A is a functional requirements matrix that was used to highlight the various functional requirements that security would need to provide when included in a JTRS radio. Attachment 1 delineates the Security Application Program Interfaces (APIs) that will be required for a JTRS-compatible security implementation.

1.2 BACKGROUND.

1.2.1 Radio Security.

The JTRS Architecture presents a number of challenges for the implementation of high-grade security. These challenges are based on a number of characteristics of the architecture that represent departures from more traditional means of achieving high security systems. This introduction places the security architecture within the context of more traditional secure systems, and new security measures that will be required to protect classified data in the JTRS environment.

1.2.1.1 Traditional Secure Environment.

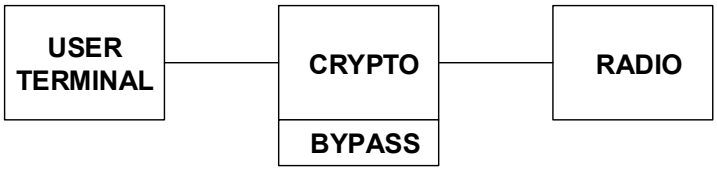


Figure 1-1. Secure Channel Configuration

Figure 1-1 shows a traditional secure radio environment that is characterized by discrete "boxes" performing discrete functions. The security of the system is achieved primarily by the following:

- Physical access control to user terminal (terminal a teletype or voice access handset)
- Hard wired connections for each radio channel
- High-grade hardware cryptography in a discrete box
- Very limited bypass capability typically within communicator channel or manual ancillary device
- System high application

The basis for security is physical separation and well defined insertion of purely hardware based cryptographic protection for communications security (COMSEC) and transmission security (TRANSEC) protection.

1.2.1.2 Embedded Cryptography.

With the advent of embedded cryptography, the cryptographic function began to merge with the radio function to achieve economies of size, weight, and power. Figure 1-2 shows embedded cryptographic equipment combining a cryptographic chip into the radio enclosure.

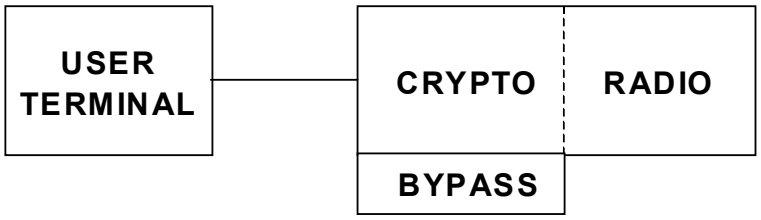


Figure 1-2. Cryptography Embedded in Radio

The characteristics of this security solution were not radically changed from the previous box level scenario with the exception that a RED/BLACK boundary was now drawn inside of the radio equipment. The characteristics were:

- Physical access control to user terminal (terminal a teletype or voice access handset)
- Hard wired internal connections for each radio channel
- High-grade hardware cryptography within the radio box
- Very limited bypass capability typically within communicator channel or manual ancillary device
- System High application
- Limited RED user applications processing

1.2.1.3 Networked Radios.

Computer networking increased the need for "wireless" networks that could extend packet protocols over radio channels for improved interconnection of separated network communicators and networks. A single communicator or wired network of communicators could thus be placed into a remote network or networks.

Figure 1-3 shows a Local Area Network (LAN) connected to a secure radio device for remote interconnection to other LANs, thus creating wide area networks in a wireless domain. The radio becomes the network extension in the packet protocol based system on both the local communicator and remote network sides. This configuration creates changes in the implementation of security that involve computer security (COMPUSEC), and network security, in addition to the more traditional cryptographic protection. The characteristics of the system now are:

- Access control to network and radio services governed by software, not physical, methods
- Hard wired internal connections or computer bus for the single radio channel configuration
- High-grade hardware cryptography within the radio box
- Bypass requirement increased to handle protocols and network information
- Separation of data classification and types performed by network - radio System High
- Multiple access methods for the communicator networks (e.g., wireless LAN)
- Interconnection of networks at multiple communicator sites

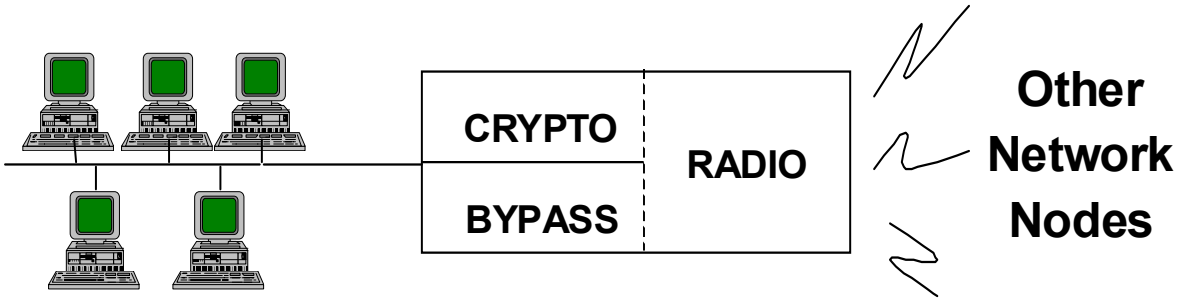


Figure 1-3. Networked Radio

The increased connectivity provided by the networked radios introduces new vulnerabilities to such attacks as virus dissemination, password theft (and malicious use). A security price is paid for the increased connectivity, just as with terrestrial networks. There is still very little communicator data processing performed within the radio. Commercial software components are not used within the crypto function, and not typically used within the radio function.

1.3 JTRS ARCHITECTURE.

The JTRS architecture, as simplified in figure 1-4 combines the embedded cryptography COMSEC/TRANSEC capabilities with added RED side radio functions (e.g., networking capability), multiple simultaneous channel operation within a single system, and introduces several new characteristics that are critical from a security point of view.

These security relevant characteristics in the JTRS system are:

- Multi-channel/multi-communicator radio operation.
- Access control to network and radio services governed by software; multiple communicators can share a physical connection
- Virtual internal connections for each radio communicator port and radio channel
- Single RED bus architecture (typical, but not mandated by SCA)
- High-grade programmable cryptography embedded within the radio box
- Bypass requirement further increased to handle internal radio control
- Radio functions programmable for all processes
- Use of commercial software products [Operating System (OS) and Object Request Broker (ORB)]

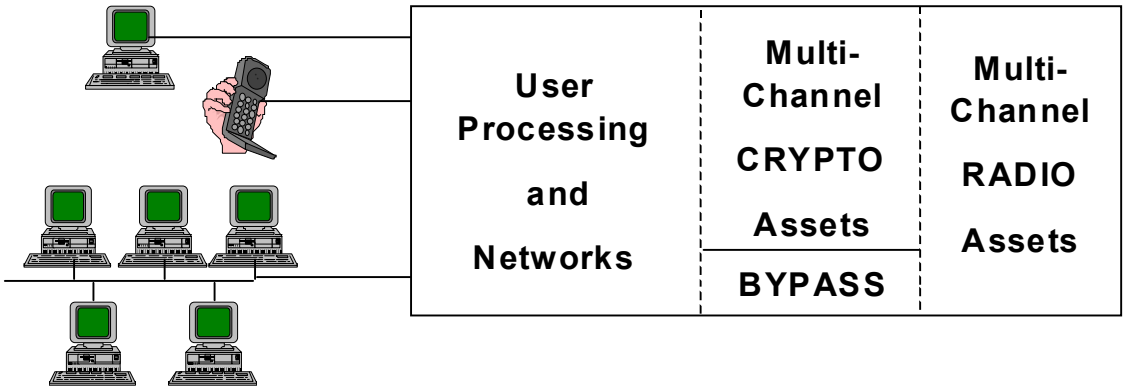


Figure 1-4. JTRS Radio Architecture

In summary, many of the characteristics associated with computer security are now part of the radio system.

The single RED bus is not mandated by the SCA (an SCA-compliant implementation could include multiple RED buses to achieve separation), but it represents a worst case scenario from a security point of view in terms of ability to separate users and various types of systems information.

These following characteristics make the security more difficult to implement under the JTRS architecture:

- Maintaining assured separation of data and user paths (e.g., access control, need to know) in a common processing environment that involves shared elements such as a single RED bus, virtual path establishment via the ORB, single processor, and single cryptographic subsystem.
- Controlling the information that is bypassed from the RED to the BLACK side of the system.
- Maintaining the integrity of software (applications and operating environment) during transport to the radio, installation into the radio, and during operation within the radio.

The JTRS Operational Requirements Document (ORD) specifies System High operation with Multiple Independent Levels of Security (MILS) as a goal. The MILS goal means that communicators at different levels of classification can be connected to the radio and use radio resources simultaneously. The architecture will not preclude the ability to meet this goal.

2 REFERENCES.

2.1 DOCUMENTS.

- JTRS-5000SCA, "Software Communications Architecture Specification (SCA)", Version 2.2.1, April 30, 2004.
- MSRC-5000SRD, "SCA Support and Rationale Document (SRD)", Version 2.0, December 15, 2000
- ON481180, EKMS-308, "Key Material Transfer Protocols", Revision B dated 20 January 1999.
- ON481180 (Secret) Unified Information Security (INFOSEC) Criteria (UIC) Evaluation Criteria for Cryptographic Implementation baseline dated October 1998
- Common Criteria & Methodology for Security Evaluation, Version 2.1, dated August 1999.
- National Information Systems Security (INFOSEC) Glossary, National Security telecommunications Information Systems Security Instruction (NSTISSI) 4009
- Operational Requirements Document (ORD) for Joint Tactical Radio, dated 23 March 1998
- Common Object Request Broker Architecture (CORBA) Specification, Version 2.2

2.2 DEFINITIONS OF TERMS.

This document attempts to maintain the use of terminology in the SCA in the terms specified in that document. Certain terminology used in discussing security does not match with the usage in the SCA. When conflicts in terminology occur, this document will clarify the sense of the word or term as used herein. Several security terminology definitions will be useful toward better understanding of the discussion in this security supplement paper.

2.2.1 Definition of Algorithm

This document makes various references to algorithms. In the strictest sense, an algorithm is a building block for a "cryptographic equipment application", also known as a crypto equipment application or CEA. A CEA is an application contained within or loadable to the CS/S, which contains or uses a cryptographic algorithm. Throughout this document a reference to an algorithm, a crypto algorithm or a cryptographic algorithm is usually a reference to a CEA that can be used in the establishment of a crypto or TRANSEC channel."

2.2.2 Definitions of User Entities.

This security supplement paper discusses a hierarchy of personnel who interact with a Joint Tactical Radio (JTR). The functions of the entities may well be combined in many mission scenarios, but the separation is used to help determine the forms of access control and privileges required for the JTRS security architecture. These entities are "user types" as follows:

- Administrator - entity that configures the radio via software downloads, waveform instantiation, and establishment of administrative privileges. Manages general audits.
- Communicator - entity that uses the radio communications capabilities of the JTR.
- Maintainer - entity that has open box access for repair and maintenance of equipment. Has access to maintenance logs.
- Operator - entity that configures the radio for specific missions by entering frequency and other mission specific parameters.
- Security Officer - entity that enters keys, cryptographic algorithms, security policies, and other security relevant information into the radio. Manages security audits.

Note that there is not necessarily a one-to-one correspondence of communicator channels and radio channels. In situations such as Demand Assign Multiple Access (DAMA), multiple communicator ports are connected through multiple cryptographic channels to a single radio channel. In the case where a LAN or other network is connected to the JTR via Ethernet or similar packetized connection, multiple communicators are connected to a single communicator port.

When the term "user" is stated, the context is one where multiple user types are or can be involved.

Additional definition for security related terminology is provided in Section 5 of this document.

2.2.3 Definitions of Levels of Cryptography.

This document makes reference to Type 1 cryptography. National Security Agency (NSA) works in partnership with the National Institute of Standards and Technology (NIST) to maintain a set of cryptographic algorithms that are suitable to applications across a wide range of communicator needs. NSA defines cryptographic algorithms in 4 "types" according to the evaluated strength or origin of the algorithms. These types are:

Type 1 - Certified by NSA for classified information protection

Type 2 - Certified by NSA for Unclassified For Official Use Only (FOUO)

Type 3 - Certified by NIST for general applications for unclassified information

Type 4 - Algorithms produced by industry or other nations (no Government certification)

Because of the programmable nature of the JTR, any level of algorithms might be implemented within the equipment. Concurrency of operation in a multi-channel environment is a security policy issue.

3 JTRS ARCHITECTURE OVERVIEW.

This section describes the JTRS Software Communications Architecture (SCA) environment from which this security supplement is drawn. The emphasis is on the aspects of the architecture that affect JTR security.

3.1 SYSTEM DESCRIPTION.

The JTRS SCA, as presented in the SCA Specification, defines a composition of hardware, software (operating environment and processes), and interfaces to provide a platform for the download and execution of software driven functionality. The SCA Specification is intended for commercial acceptance as well as military application, so security functions (other than an INFOSEC subsystem) are not explicitly addressed in the SCA. For this reason, the security requirements are expressed largely in terms of applications that operate within the radio, and cryptographic subsystems which use both hardware and security applications software. The architecture does not detail the run-time operation of a JTRS implementation.

The JTRS SCA is shown in figure 3-1, using the standard graphic used to summarize the components of system. The figure shows the allocation of functions within the architecture and the methods used to interconnect the functions; it does not define the allocation of software applications. The use of a commercial OS, a commercial ORB, and a single RED bus in a multi-channel/multi-communicator configuration are significant for the security requirements in Section 4 of this document. The heart of the operability of the radio architecture is the Core Framework Service implementation of the SCA. A Core Framework (CF) that allocates communicator and systems software applications and functions to processing capabilities within the radio, and forms the interconnections among the applications and functions. Taken collectively, the OS, ORB, and CF are referred to as the operating environment (OE). The figure depicts a single "logical software bus via CORBA" that actually requires separation for proper security functionality in RED and BLACK environments.

The SCA alone is not sufficient in specifying the software required to achieve an operating radio. Additional software applications are necessary to operate over the SCA OE. The layering of the applications over the OE and hardware is shown in figure 3-2. The figure is not meant to represent the actual interconnections within a JTR, but rather to show the layers of functionality required to support radio operation. The applications above the OS and ORB include systems applications and utilities that are present for any communicator application (e.g., waveform) to operate within the JTR. Above this layer are the communication systems applications and waveforms that constitute the interoperable radio applications of the system. Above communications system applications layer are the data entry and download functions required to insert radio parameters for specific radio mission operations (e.g., frequencies, keys, net addresses). The point of the figure is that an operational radio requires numerous software elements, API, and operator defined parameters to properly perform its communications function.

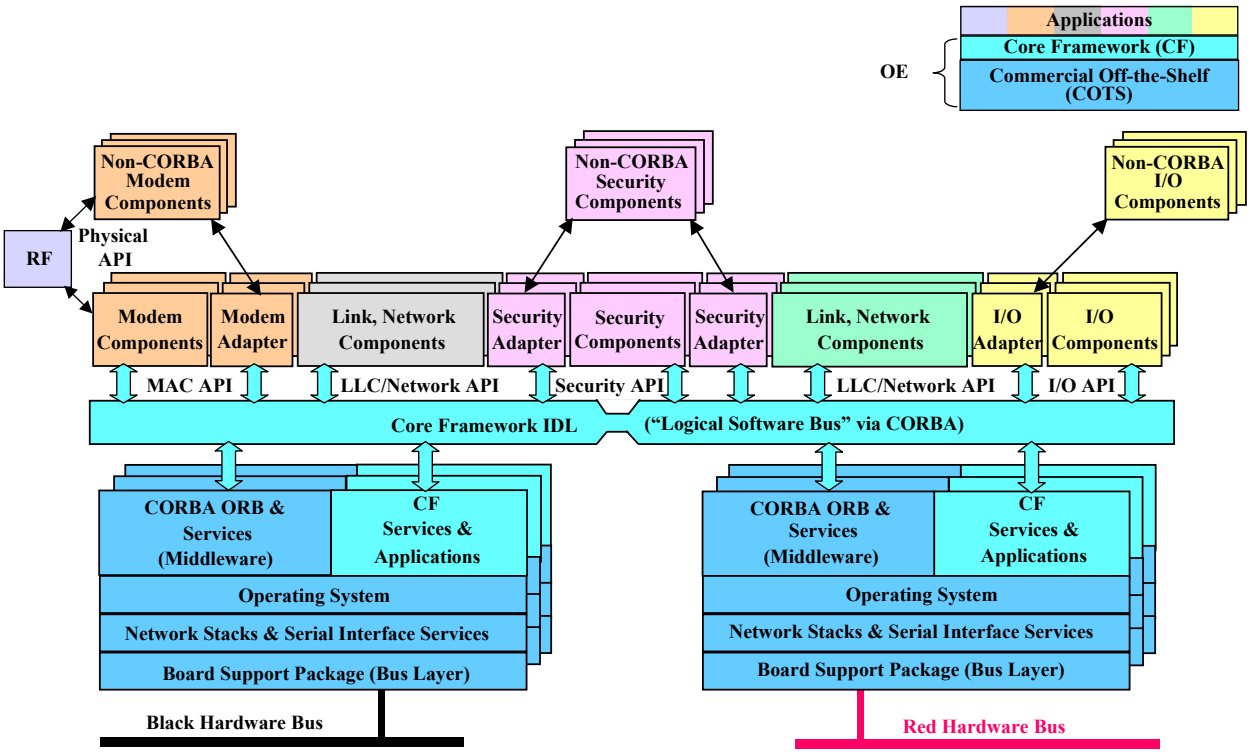


Figure 3-1. JTRS Software Communications Architecture

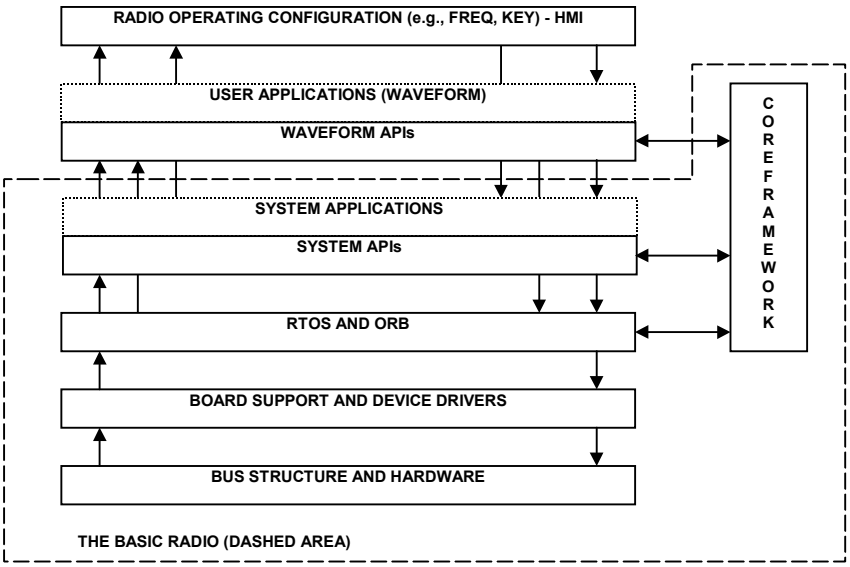


Figure 3-2. JTRS Architectural Functional Layers

In figure 3-1, the blocks for "Security Components" and "Security Adapters" are depicted as the separators of the RED and BLACK hardware buses, as well as the separation mechanism for RED and BLACK OS and ORB components. The CF is shown as spanning across the block called Security Component. The term "Security Component" should not give the impression that

all information security requirements are encompassed within this functional block. In fact, security requirements exist in RED and BLACK processing modules as well. Therefore, in this security supplement, the "Security Components" block in the figure is redefined as the "Cryptographic Subsystem (CS/S)", to permit more accurate definition of its functions.

For a detailed description the architecture, see the JTRS SCA Specification and its associated Support and Rationale Document (SRD).

3.2 HARDWARE.

The JTRS Architecture is composed of a set of devices that provide the processing capabilities within the software radio architecture, and the hardware components that support the devices.

Processing devices and memory are allocated among the application types shown in figure 3-1. The architecture does not provide specific direction on the allocation of applications to devices. At a minimum, it would appear that there would be one RED processor, one Crypto processor, and one BLACK processor for a secure radio implementation, although no processing allocation is specified in the SCA. The actual structure of processing applications is left to the implementation. The basic hardware architecture provides a single RED and a single BLACK hardware bus (but does not preclude multiple buses) assembled with one or more processor boards and required memory on each bus. The JTR will require some form of non-volatile storage to contain software programs, keys, and other information requiring retention. The architecture does not specify allocation of non-volatile storage to RED or BLACK components.

The Human-Machine Interface (HMI) (i.e., radio operator/communicator control) is not allocated in the SCA. Although host applications are allocated to the RED side of SCA processing, location of the HMI function on the BLACK side appears to be feasible. Depending on the form factor of the radio implementation, the HMI can be integral to the equipment, or provided via an external interface in a separate processing device. As an extension of the HMI, the JTRS ORD also calls for a remote capability for operation and control of the radio at a threshold range of 1.5 kilometers (approximately 1 mile).

The power supply characteristics are mentioned as part of the architecture because power conditioning is required for RED/BLACK isolation, and the physical partitioning of functions is performed according to power and unintended emanations (i.e., TEMPEST) characteristics. The hardware housing, packaging, and cooling are significant in that commercial parts are to be used to the maximum extent possible, but requirements for TEMPEST and Tamper protection, and military environments, will be met.

3.3 SOFTWARE.

The JTRS software environment is composed of two basic software types, OE and applications that are further subdivided as follows:

- Operating Environment
  - Operating System
  - Board Support Packages and Hardware Drivers
  - Middleware (Object Request Broker)
  - Core Framework (CF)
- Software Applications
  - Core Systems Applications
- Implementation Dependent Systems Applications
  - Communicator Waveform Applications

3.3.1 Operating Environment.

3.3.1.1 Operating System.

The OS provides the first abstraction of the software application from the hardware. This means that the OS manages and controls the hardware interfaces for the software applications. The OS provides various services for the middleware and applications within the JTRS, including:

- Process management
- Process creation
- Inter-process communication
- Timing services
- Input/output management
- Scheduling management (for real time OS)
- *FileSystem*s

3.3.1.2 Board Support Packages and Hardware Drivers.

Board support packages (BSP) and hardware drivers are a class of software and firmware that provide interfaces between the OS and electronics boards and computer peripheral devices. BSPs and drivers vary according to the boards and backplanes used, the peripherals used, and the OS itself. The functions are transparent to software applications at higher-levels. The software and firmware are either provided by the commercial vendors of the boards and equipment, or are written by the radio developer in cases of specialized hardware.

3.3.1.3 Middleware.

The middleware allows for the transparent exchange of information across multiple processing nodes. The middleware for the JTRS is the CORBA compliant ORB. The ORB provides flexible flow of information across the software bus, thus abstracting the applications from the specifics of the operating system and lower layers by producing standard interfaces for software applications.

3.3.1.4 Core Framework (CF).

The CF provides for application instantiation, tear down, and basic application services such as file management. CF entities include:

- *DomainManager*
- *Device*
- Domain Profile
- *ApplicationFactory*
- *ResourceFactory* (optional)
- *FileSystem*
- *FileManager*

The current version of the SCA contains a complete list of Core Framework interfaces and their descriptions. The CF is the critical core systems application that establishes the allocation of processing resources and interconnection of software components within the JTR.

3.3.2 Application Software.

The SCA does not differentiate among the types of software applications that will exist in JTRS radios. There are software applications that service the functions of the radio (herein called "systems applications"), and applications that service the functions of the radio communicator (herein called "communicator applications" and "waveforms").

3.3.2.1 System Applications.

Systems applications are those software applications that are integral to the function of the radio and are delivered with the radio, or are applications that can service a number of waveforms. Systems applications are of two types:

- Core Systems Applications
- Implementation Dependant Applications

Core systems applications are delivered with the radio as an integral part of the radio (e.g., HMI, software download function). Implementation Dependent systems applications are operated with more than one waveform. That is, the Implementation Dependant systems application can service more than one waveform, but is not delivered with the waveform (e.g., user access control, audit). The Implementation Dependant systems applications can be part of the core radio or delivered via download as are waveforms.

Since this document deals with radios in a military environment, the cryptographic software functions are considered to be part of the core systems applications, with the assumption that cryptographic security services will be integral to the function of the radio.

3.3.2.1.1 Core System Applications.

Core systems applications are those that will be provided with the radio as a resident function to support user applications. Examples of core system applications that are part of or have an impact on the security architecture are:

- Cryptographic software and Key Management software
- COMSEC devices (for device control)
- Audit Consumers
- Download Utilities
- HMI
- Administrator, Operator, Security Operator, and Maintainer HMI security policy enforcement
- Installer Application
- Security Policy Enforcement (application instantiation, inter-object communications)
- Virus protection

It is necessary to provide these types of functions as part of the basic radio to properly satisfy ORD requirements in various applications. In software terminology, many of the core systems applications could be referred to as "utilities", that is, programs that support the overall operation of the radio, and can be applied as part of various functions.

3.3.2.1.2 Implementation Dependent Systems Applications.

Implementation dependent systems applications are applications developed according to specific items of a radio procurement functional specification. That is, the requirements that drive the specification may be specific to certain environments and missions. Implementation dependent system applications that impact the security architecture include:

- COMSEC devices (for application specific components - e.g., tokens)
- Network manager
- Waveform independent key transfer mechanism Over The Air Transfer (OTAT)
- Crossbanding

3.3.2.2 Communicator Waveforms and Applications.

Communicator applications are made up of waveforms (e.g., Link-16 software package), and other applications required by communicators to satisfy specific needs as called in the functional specification for a specific radio procurement. The communicator applications would be considered as downloaded (installed) items of software as opposed to core systems applications that are delivered as part of the radio.

3.3.2.3 Radio Boot and Run Time Environment.

The SCA defines a method for the installation and instantiation of software. The term "instantiate" means to take a software application (e.g., waveform) from storage, to allocate it to processing elements, and to interconnect the software objects to provide radio functionality. The instantiation process relies on the CF. The CF and Cryptographic Subsystem (CS/S) itself are instantiated as part of the radio boot (cold start) process. Thus, the sequence of operations involves:

- Boot operating system
- Instantiate CF via boot
- Instantiate ORB via boot
- Instantiate CS/S via CS/S internal boot
- Instantiate applications via CF
- Run Applications
- Tear down applications via CF
- Shut down radio

The "run applications" aspect of the sequence is particularly interesting in that the CF provides little in the way of run time facilities. The CF is largely dormant during actual radio operation. *FileManager* is a CF application that continues to be utilized while the radio is in operational use. There are numerous security-related functions that are required during radio run time. The security processes are related to assuring the proper establishment and tear down of each operating state, the proper operation of hardware, the allocation and erasure of memory and storage, and monitoring of functions during run time to assure that proper configurations and interconnections are maintained. Use cases have been developed to show the details of these operations from a security perspective.

3.4 RADIO OPERATING POLICY AND DOCTRINE (NO SECURITY).

JTRS will operate under different sets of policies and operational doctrines depending on mission requirements. Many of the policies will be personnel and procedure related, but some will be contained within the radios themselves or as part of the download software packages. Policy and doctrine will vary according to which waveforms and applications are operative in a radio at a given time. Examples of internal radio policies are varied, but could include:

- Restrictions on crossbanding of waveforms
- Restrictions on waveforms that can simultaneously operate (e.g., spectrum concerns)
- Limitations on communicator/operator/administrator access to radio control and status
- Remote control functions permitted
- Acceptance of software packages for download

Legacy radios tended to have a fixed operating policy, that is, the set of rules under which the radio operates. Functions were not modifiable. Legacy radios tended to support a single waveform, and a single thread of that waveform. The policy was hard-coded in hardware so it could not be reprogrammed. Operating systems were customized if they existed at all. Most waveform processing was performed in hardware. JTRS is opening an environment where operating policy will be dynamic, and will be controlled, at least in part, by software loaded into the radio as part of waveform or system applications.

3.5 JTRS INFRASTRUCTURE.

This section describes elements of the JTRS systems and operational requirements that are external to the equipment or effect its configuration in the field. Although these aspects of the JTRS are external to the radio, they can impact the architecture and the interaction of a JTRS-compliant radio with its environment.

3.5.1 Hardware Maintenance/Replacement.

Except for the small form factors of the radio, hardware is anticipated to be modular with backplane buses and replaceable standard Commercial Off-the Shelf (COTS) cards (or custom cards for specialized functions). Maintenance is assumed to be by line replaceable units (LRUs) at the organizational level. Cards will require more than functional compatibility for software to be operable, since the processors are the same or OE-compatible for existing software downloads to function. Different replacement processors could require different software compilations. Thus, version control of hardware components is required to assure that software packages will maintain compatibility with the radios.

Field changes to hardware may effect the Tamper and TEMPEST characteristics of the equipment. Integrity of filtering mechanisms in power supplies and consistency in the types of boards and processors used be maintained.

The JTRS will be required to identify software downloaded to it in order to determine if the software is suitable to the platform. It is anticipated that the radio will be required to identify itself and its capabilities to the downloading system and vice versa, so that hardware and software can be properly matched. This need implies that a specific interface be provided in the JTR for communication between the JTR and a download system (yet to be developed) that consolidates mission planning, software, and operating parameters for the radio in specific missions. A communicator interface may be used for the download function, but from a security point of view, maximized separation of user data and radio internal information is desirable, so that a separate hardware control port would be desirable.

3.5.2 Software Composition and Waveforms.

The question, "What is in a waveform?" has numerous answers and definitions depending on applications and hardware platforms. This paragraph presents a strawman definition of a "runtime waveform" (RTW). For the purposes of the discussion, the RTW is the consolidated set of downloaded communications applications software, the systems and utility applications required for the functioning of the communications capability, the Input/Output (I/O) requirements for communicator interface, and the mission specific parameters that are loaded by an operator or user (e.g., frequencies and keys). From a security perspective, the waveform parameters and the utility functions presented as systems/communicator applications are of interest, since security policies for processing requirements, interconnections, and access controls will be established by those waveform segments.

The software profile, provided as part of the software waveform download package, provides information to the CF for instantiation requirements.

For the operation of a waveform or set of waveforms, configuration information is drawn from a variety of sources including:

- Information included as part of the CF (e.g., the interconnect configuration)
- Information included with the waveform software to be downloaded (e.g., software profile)
- Information entered via the HMI (e.g., channel I/O, and frequencies)
- Key management tags
- Mission download packages (e.g., mission information)

The typical waveform definition is for the downloaded software package, but the RTW defines a run time configuration that consolidates all software elements required for waveform operation. A "waveform" can be defined at three points in radio operation:

Download and Install - Separate download of crypto software, application software, systems utilities, and system security applications.

Instantiate - Combine elements above to form full communicator application.

Run - Incorporate HMI and externally entered network parameters for specific network, frequency, key, and time of day (TOD).

The full content of the waveform is "built" as the waveform package is assembled for use. The elements from which a run time waveform is built are:

Crypto Software - Chip/Module/Subsystem Boot, Algorithms, Key Handling, Internal Channel Set-up, Alarms, Control, Internal Security Policy

Communicator Application - Communicator Channel Specific Software to Include Modem, Coding, Interleaving, Synchronization, Data Formatting, Voice Processing, I/O Port Definition

Utility - A Systems Application That Can Serve Multiple Waveforms (e.g., Download Software, Crossband, HMI, Remote Control, Access Control, Audit)

Strap/Mode Settings - Specific Operating Mode Settings for Software That Can Be Run in Multiple Modes [e.g., Interleaver, Code Constraint Length, key generator (KG) Motion, Sync Modes]

HMI Input - Specific selections/settings for a given network/connection on a given radio date

A sample of the construction requirements for a complete waveform, and the confidentiality, authentication, and integrity measures required is provided in figure 3-3. The figure indicates that various forms of encryption, cryptographic authentication, and cryptographic integrity check are required to provide suitable protection for JTR software installations.

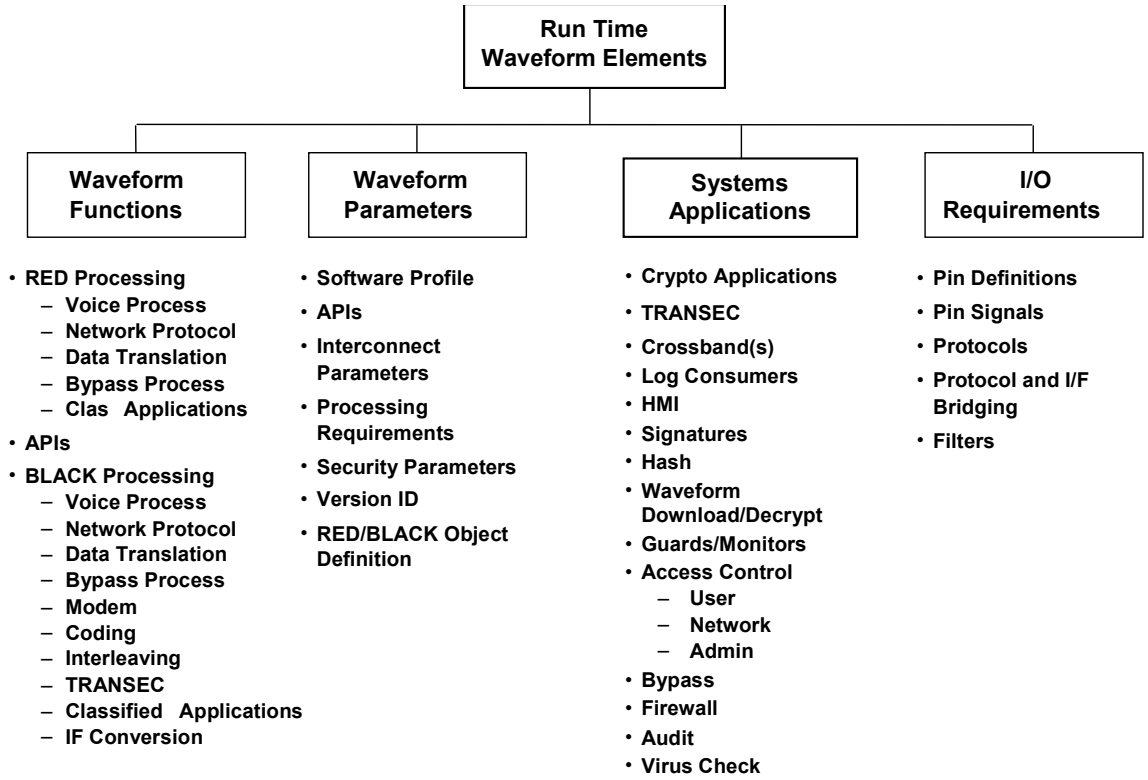


Figure 3-3. JTRS Run Time Waveforms Elements

Figure 3-4 shows a nesting of functions to be applied to waveforms for security services. For instance, the JOSEKI application is an encryption function that provides confidentiality for classified crypto algorithms. The signature function provides for authentication of the software packages as to source. The hash functions provide for integrity of the software. These protection elements become an integral part of the waveforms to provide assurance of delivery of software as intended. As described in Section 4, these protection functions will be applied to assure both the download of correct software and the instantiation of correct software. The figure also indicates that software identification being applied as part of the waveform so that software types and modification levels can be properly distributed and tracked. In some cases, mandatory modifications to software may be required. To satisfy these cases, the status of software versions in fielded in JTRS will need to be tracked so that modifications can be properly delivered.

Crypto Segment

(((Algorithm(s) $_{JOSEKI}$) + (Crypto Support Software)

+ (Algo Strap Options)+(ID))$_{(NSA\ Sign)Hash}$)

Application Segment

((Application) + (Application Profile) + (Software ID) + (Strap Settings/Options) $_{Government\ Sign}$)

Systems Segment

((Utility) + (Utility Profile) + (Software ID) + (Strap Settings/Options) $_{Government\ Sign}$)

User Input

(Key, Frequencies, Network Parameters, Access)

Figure 3-4. Waveform Elements

3.5.3 Key Management Infrastructure.

The key management function has a special role in the security of the JTR because of the criticality of key protection and accounting in a multi-channel, multi-algorithm processing environment. The use of software cryptographic algorithms and BLACK keying/key storage methods further expands the number of keys and key types required.

When a common fill interface is utilized in the radio, the JTR internal interfaces will require the ability to route keys to assigned storage and use locations. Transfer of keys to cryptographic, TRANSEC, and Global Positioning System (GPS) applications, may require that keys be passed from the cryptographic function into the BLACK side modem or GPS function. The radio will require the ability to read Electronic Key Management System (EKMS) generated key tags, and to provide an operator/security officer capability to enter key tag information as keys are loaded, transferred, and used.

The types of keys and key related material has grown beyond the need for traffic keys for communications traffic. BLACK and Benign keying techniques require the use of key encryption keys and asymmetric key development systems (e.g., Firefly). Improved key protection has resulted from techniques developed under the EKMS. Additional requirements for improved authentication of users in networked systems have expanded keying requirements in the areas of Public Key Infrastructure (PKI). A unified program for the management of all keying related functions has been established as the Key Management Infrastructure (KMI)

initiative. The initiative basically combines both EKMS and DoD PKI capabilities. Depending on required applications, the JTR will make use of all KMI services for confidentiality of user traffic, TRANSEC protection, protection of keying material, distribution and accounting of keying material, and authentication of users via PKI certificates and credentials. A unified approach to the varied keying requirements will be necessary to take full advantage of the versatility the JTR platform.

3.5.4 Mission Planning/Execution.

For the JTRS to operate as a radio, multiple types of information will be required to support the basic internal software downloads to configure for specific missions. Such information can be delivered via the HMI, or can be part of a consolidated download package delivered to the radio in its operational environment as shown in figure 3-5.

The JTRS multi-waveform, open system architecture introduces new complexities to the operating policy and security policy. The policy requirements will change according to mission. Future applications may require automated tools to properly and securely download the varied information types required for JTR operation. Some automated radio support systems exist for mission planning and parameter downloads, but these systems have not been integrated.

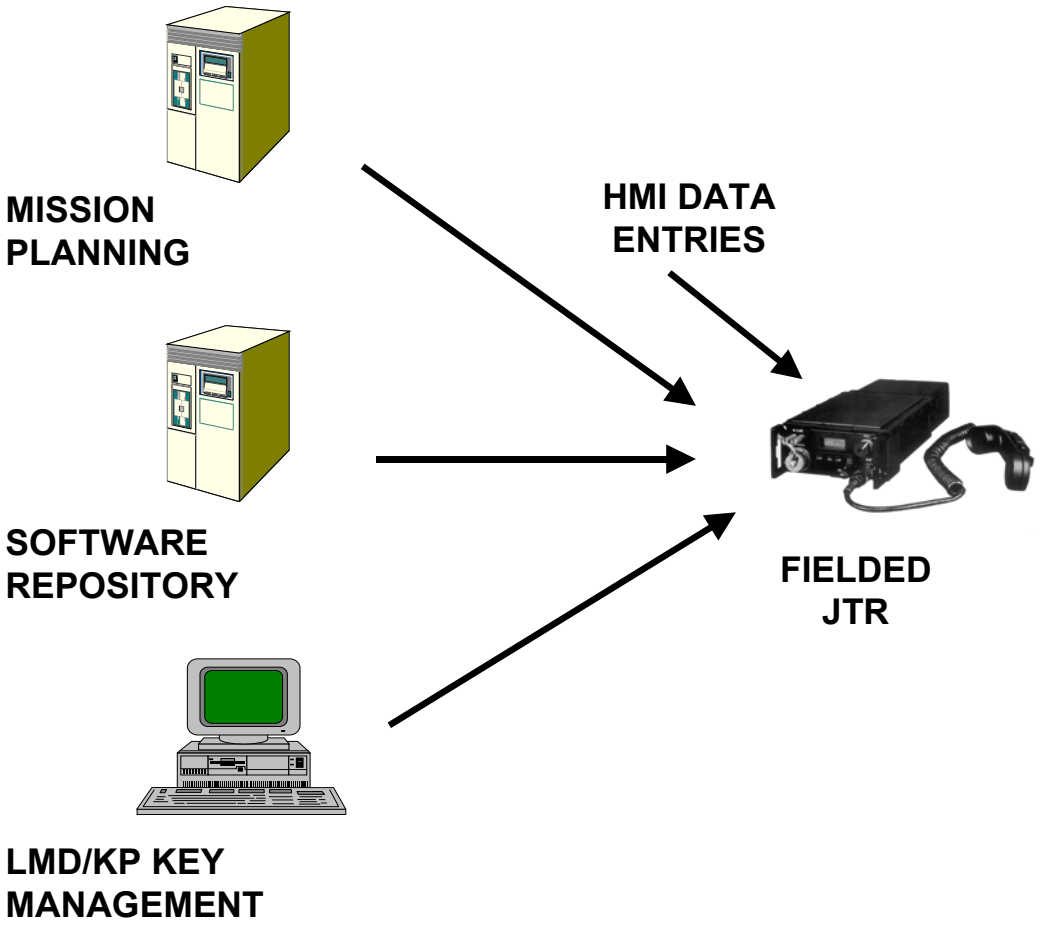


Figure 3-5. JTR Mission Download

3.5.5 Radio Control/Status Bypass.

It is possible to architecturally view the bypass mechanism at different levels for JTR. Because of the operating environment, the CS/S needs to act like a bridge between the red side ORB and the black side ORB. It is necessary to assume this viewpoint because the remote references from the red side to the black side can be accomplished with two methodologies. The first one uses a naming service to obtain the network address of the registered object. The RED side object would then get the remote object reference from the naming service and perform the remote object call. The second method uses 'stringified Inter Object Reference (IOR)' to make remote references. The bypass mechanism needs to accommodate both approaches.

The amount of information presented to a bypass channel will be minimized. Several mechanisms can be employed to reduce bypass information throughput. The preferred method is the use of look-up tables to index the possible types (e.g., object names or destination addresses) for messages passing from RED to BLACK. Such a table could convert a full object call or Internet Protocol (IP) address to a numeric value (e.g., 8 or 16 bits). The table value is reconstituted on the BLACK side for message delivery. This technique relies on the knowledge of the instantiated objects within the radio and their interconnections. The tables are built at run time based on information provided by the CF *ApplicationFactory*.

3.5.5.1 Radio Control Bypass Discussion.

The JTR architecture uses a CORBA-compliant ORB as the internal transport mechanism. CORBA messages are built to the format as shown in Figure 3-6. The figure provides an example of the protocol flow since the SCA does not require an IP protocol as shown. The ORB requires the Stub (client interface) that is shown in the stack. The complement of the stub is the skeleton (server interface). The stub and skeleton can be considered to be ORB level APIs, defining the information interchange. The skeleton contains the requested objects Interface Definition Language (IDL) that the requesting object needs to exchange data. (Complete information can be found in *The CORBA Architecture and Specification.*)

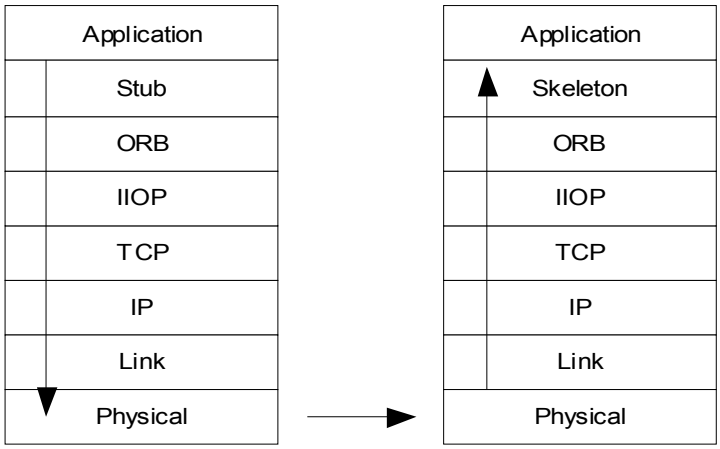


Figure 3-6. CORBA Stack Diagram

The important points about this stack are:

- Order of message building
- Message encapsulation as it moves through the stack.

If Object A and Object B are not local to each other, the Object A message will proceed down the stack to the physical layer, i.e. a bus; go across the bus to the remote Object B's physical processing site and up the stack to Object B. As the message proceeds up the stack, the encapsulation layers are removed. If the Objects are local to each other, i.e. on the same processor, the messages proceed in one of two methods as shown in figure 3-7. If Object A is requesting service from Object B that is resident (i.e. in the same memory space), the Object A call proceeds to the ORB layer and returns up the stack. If Object B is not within the same memory space with Object A, Object A's request proceeds to the inter-ORB communication methodology. For this example, the IIOP is the Inter-ORB Communication methodology.

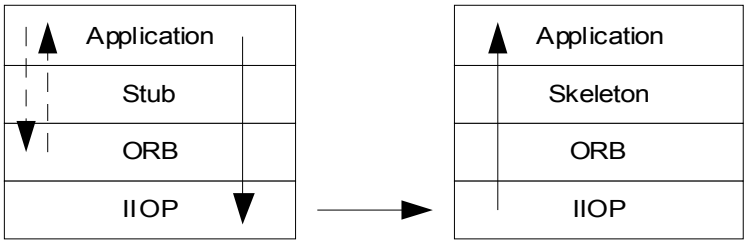


Figure 3-7. Local Object Communication

IIOP is the common layer that is permitted by CORBA. (The local objects are not aware of the exact transport mechanism.) There is another option permitted in the SCA if real-time communication between objects cannot be met using the standard CORBA communication mechanism. This mechanism is shown in figure 3-8.

This example uses TCP/IP as the underlying communication mechanism. The SCA currently permits <u>any</u> mechanism that is justified within the SCA guidelines. The bypass with this type of mechanism is occurring under CORBA. (I.e., CORBA does not know of the communication.) The consequence of this type of bypass is that the connection between applications is also unknown to the *DomainManager* and the *ApplicationFactory*. (In a standard SCA-compliant architecture, the *ApplicationFactory* makes the object connections.) For bypass to work properly, it should be able to examine and validate that two applications are permitted to exchange information. With the preceding as background information, the discussion now focuses on the bypass function.

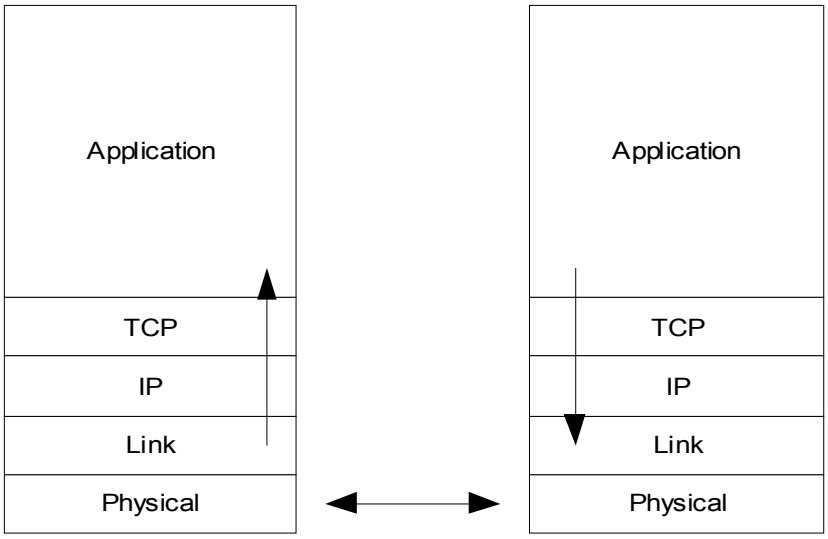


Figure 3-8. Communications Model with ORB Bypassed

The first type of bypass to be discussed is an SCA compliant JTRS radio using CORBA. The example shown in figure 3-9 illustrates a *DomainManager* working through a Modem *Device*.

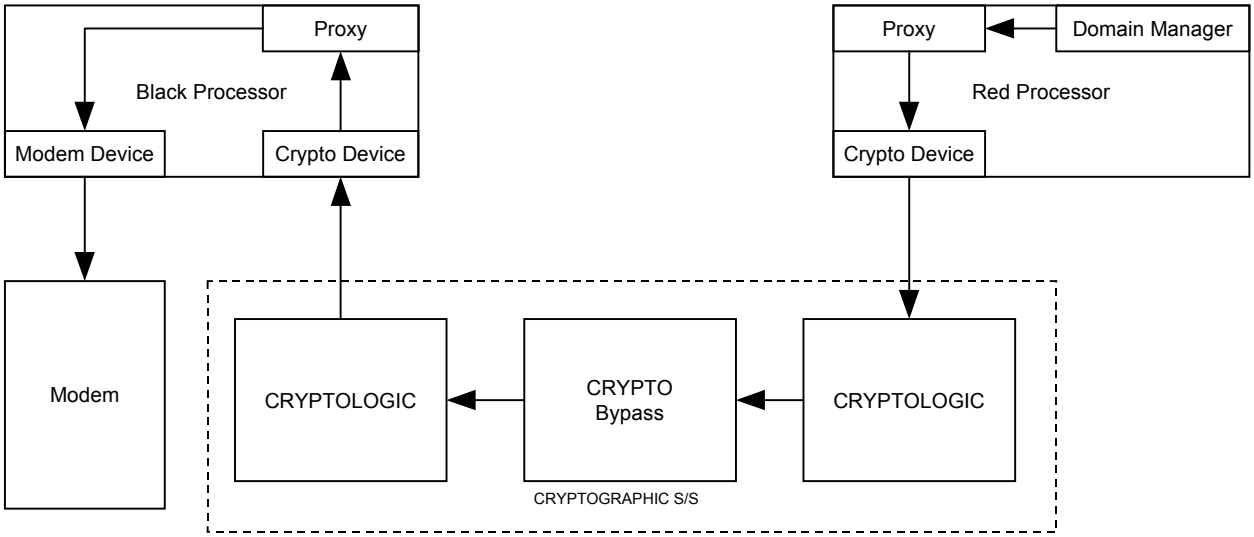


Figure 3-9. Bypass Example Using a Proxy Mechanism

This type of message requires the CS/S to bypass the Red General Purpose Processor (GPP) *DomainManager* message to the Black GPP *DeviceManager*. It also highlights the use of a proxy mechanism to isolate the application from the specific message translation that is required.

An alternate methodology to a proxy is to use a specific Security API for bypass. An example of this mechanism is shown in figure 3-10.

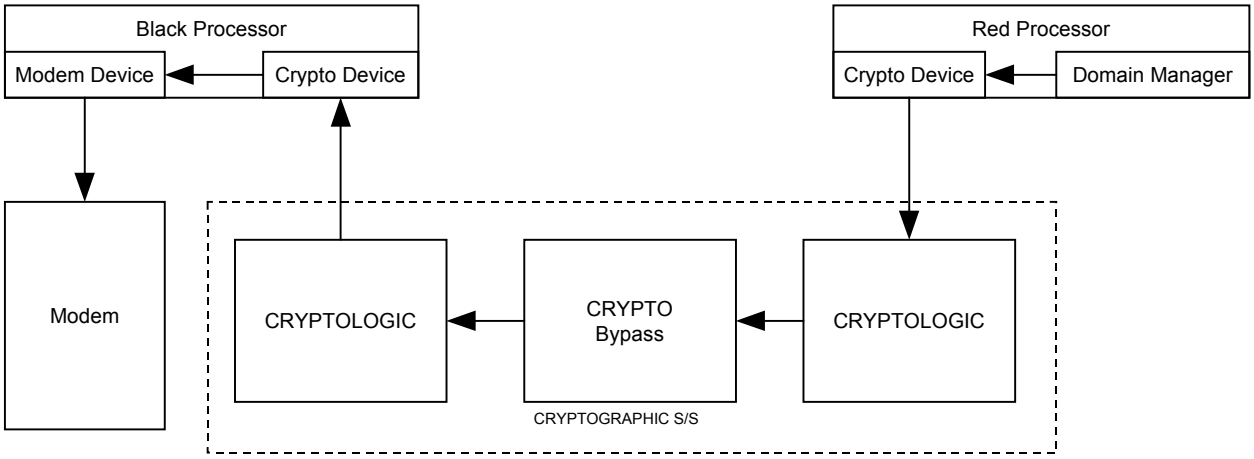


Figure 3-10. Bypass Using a Security API Approach

An example on how the message would proceed through the software protocol stacks without using a proxy is shown in figure 3-11. This representation highlights the stack flow. It also shows that even if the proxy mechanism is used, the method used to validate the bypass message does not change. (The proxy software runs as an application on both the RED and BLACK GPPs.)

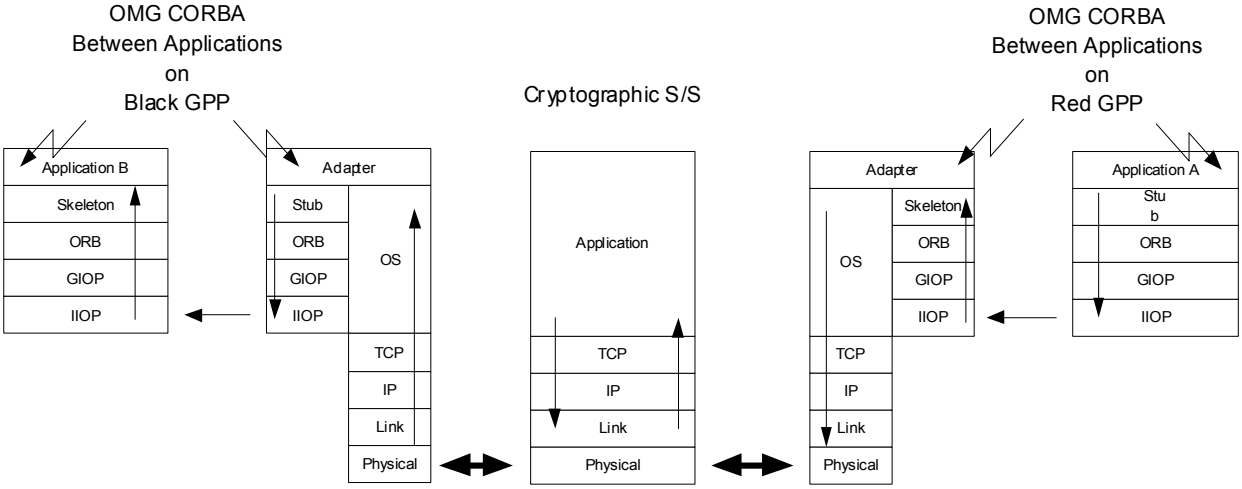


Figure 3-11. Alternative Bypass Approach without Proxy

This stack representation has some assumptions:

1. A non-CORBA-compliant CS/S is present.
2. The GPPs are using adapters to translate the CORBA compliant message into a non-CORBA message type specific to the CS/S.
3. The underlying transport layer is TCP/IP based.
4. The algorithm that examines the message to be bypassed executes as an application in the crypto subsystem.
5. Client Object and Adapter are executing in separate memory space.

Item 4 listed above is very important. It means that the lower layers of protocol have been removed. The application only has to check the original message. It is not concerned with covert channels hiding in unused bits because those bits have been stripped off and discarded. Once the bypass message has been validated, the other layers are rebuilt from a trusted mechanism within the cryptographic boundary.

Another type of bypass that needs to be considered involves Environment-Specific Inter-ORB Protocols (ESIOPs). This protocol is allowed under the Object Management Group (OMG) CORBA specification, and can be used in specific circumstances. As it implies, it is private protocol that can be used to support an existing infrastructure. However, the particular ESIOP will conform to the specifications in Chapter 12.1 of *The CORBA Architecture and Specification.* The impact to Bypass is shown in figure 3-12. As with the previous examples, the transport encapsulation needs to be stripped from the message prior to being examined by the bypass algorithm. The ORB as shown in the CS/S provides a protocol stripping function only. (It does not imply CORBA compatibility.)

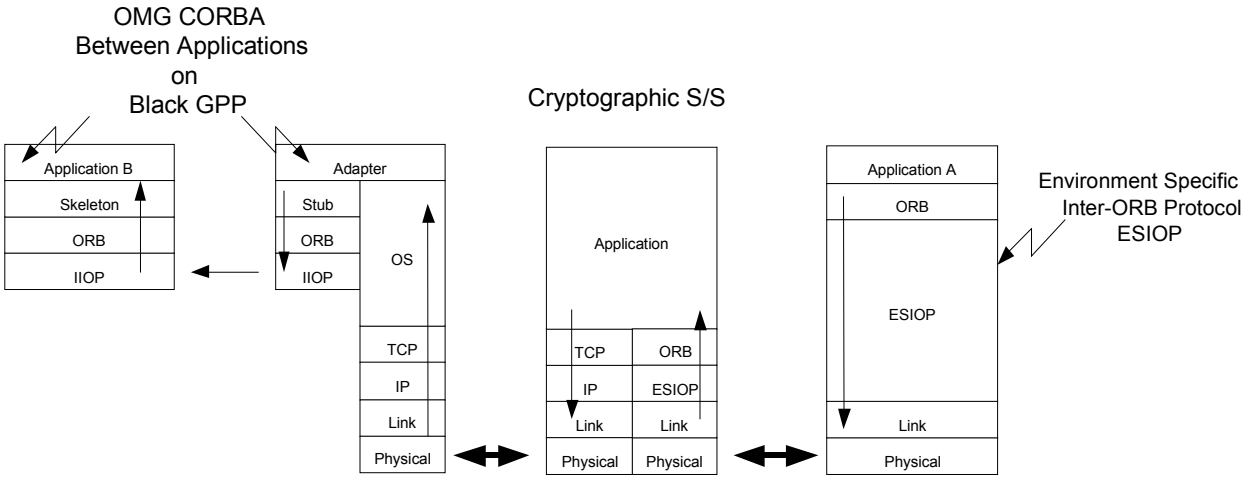


Figure 3-12. Bypass Considerations Using ESIOP

The last type of bypass to be examined would be the use of a transport function that operates below the CORBA level. Using figure 3-7 as a reference, it can be seen that the process does not complicate the bypass method as long as the connection method uses the same transport

mechanism as CORBA. The issue to be dealt with is the application to application connections that are permitted and how this information gets to the algorithm validating the bypass message. A bypass mechanism can be readily implemented in the JTRS radio. The use of alternative connection methods for real-time traffic control can be accommodated as long as they use the same hardware transport mechanism that is used by CORBA for any given radio implementation.

3.6 SCA IMPACTS.

The following sections discuss security impacts on the SCA. Security Requirements as outlined apply to the Operating Environment and the Guard. Implementation will decide how these requirements are satisfied. All requirements can be implemented in the OE and software guards. All requirements can be implemented using hardware guards. This section provides the developer with relevant discussions on various implementations and should only be used as guidance.

3.6.1 Operating System.

The OS provides the first abstraction of the software application from the hardware. That is, the OS handles the specifics of interfaces to the hardware elements. The OS provides various services for the middleware and applications within the JTRS, including:

- Process and/or protected space creation
- Process and/or protected space management
- Inter-process communication
- Timing services
- Input/output management
- Scheduling management (for real time OS)
- Thread creation and management

3.6.1.1 *FileSystem*s Discussion.

A key element of the security enforcement for an OS is the OS kernel. The kernel is defined as the OS code subset that has unrestricted (supervisory) access to OS management functions. The OS code outside of the kernel space does not have the capability to update/modify the Memory Management functions. For purposes of assurance and assurance evaluation, an OS design that provides as small software kernel (on the order of 10K lines of code or less) is considered evaluatable.

The operating system provides the lowest layer of security enforcement. The security related responsibility of the operating system layer is to enforce a policy of *information flow* and *data isolation* among the objects under control of the operating system. It is the responsibility of the operating system-level security policy is to enable the next higher-level of security functions to enforce its security policy. Of particular importance is the protection of higher-level access control information as well as application critical data from being compromised or altered.

The operating system also enforces the object reuse requirement. Object reuse refers to the reassignment of assets from one process to another. This means that processes do not have access to the information contained in other processes to which they should not have access. More simply stated, program and data storage need to be erased before a new application is

instantiated, or a context change is made where common processing or memory facilities are used.

3.6.1.2 Approach.

The operating system process management and creation functions create process separation and data isolation using capabilities such as a hardware Memory Management Unit (MMU). The MMU enforces the process separation and data isolation while higher-level applications are active in the processor. The policy of data isolation is to prevent critical information belonging to the higher-level entities (i.e., middleware and applications) from being mixed or compromised.

The information flow policy of the operating system is required so that unauthorized bypass of security functions at the next higher-level can be prevented. The information flow policy is enforced by the inter-process communications functionality within the kernel. The inter-process communications functionality provides access tables restricting communications between unique process spaces. Inter-process communications function enforces the information flow policy at the process level.

3.6.2 Middleware.

The middleware allows for the transparent exchange of information across multiple processing nodes. The middleware for the JTRS is the CORBA ORB. The ORB provides flexible flow of information across the CORBA software bus while providing a common interface to higher-level applications. The applications are abstracted from the functions of the operating system.

3.6.2.1 Discussion.

The next high layer of security policy enforcement (i.e., above the OS) comes from the middleware level. The CORBA ORB in figure 3-1 shows the middleware functionality. This layer of security implementation requires the services of information flow and data isolation from the OS kernel layer in order for it to be able to enforce its own (middleware) security policy. Application-specific security services need to be addressed at the application level. If the application specific security services migrate down to the middleware, the system becomes unsupportable and unevaluatable. The open-architecture CORBA middleware discussed in the SCA creates a security concern in that information flow between objects could become unrestricted.

3.6.2.2 Approach.

The middleware security approach is to control message traffic between objects by restricting access to object references within the system. As an example, each object, upon instantiation, registers with the ORB Naming Service by performing a bind() operation. This Naming Service may reside within the *DomainManager* process space. Only entities within the *DomainManager* process space shall be permitted to perform resolve() and list() operations, effectively protecting object references from being discovered by unauthorized entities, and preventing unwanted message traffic across the CORBA bus. The access to object references is realized by providing protected processing space. Thus, unauthorized connections cannot be made by objects after the application is instantiated.

A logical place to allocate middleware security policy enforcement mechanisms is the CORBA ORB. Current ORB technology places the ORB within the application process space. By allocating the middleware security policy enforcement mechanisms to the ORB, the security policy enforcement mechanism is subject to tamper by untrusted application software. For this reason, the system high middleware security policy enforcement mechanisms are performed by the *DomainManager*, *ApplicationFactory*, and Naming Service. See the Core Framework security discussion in paragraph 4.3.1.14.3.3.1, which describes a middleware security approach.

3.6.3 Core Framework.

The Core Framework provides for application instantiation, teardown, and basic application services such as file management. Core Framework entities include:

- *DomainManager*
- *Device*
- *LoadableDevice*
- *ExecutableDevice*
- Domain Profile
- *ApplicationFactory*
- *ResourceFactory* (optional)

- *FileSystem*
- *FileManager*

See the current version of the SCA for a complete list of Core Framework components and their descriptions.

3.6.3.1 Discussion.

To have confidence in the correct functioning of the CFand the applications that it instantiates, mechanisms are provided to assure that:

- The correct processes occur within an application (e.g., the correct waveform and systems applications are instantiated).
- The inter-object messages are flowing between the correct objects (information flow security policy).
- Objects of low or no assurance do not instantiate, teardown, or modify an application.
- File permissions are enforced on all read, write, and execution of files.
- CF boots correctly with software integrity (implementation dependent).
- The audit records are understandable (implementation dependent).
- The audit record is protected from unauthorized changes or reading. (The CosLwLog *LogProducer* interface is being considered to support the audit function.)

3.6.3.2 Approach.

Correct operation of the CF and its functions are assured by:

- Testing the correctness of the instantiation of the CF itself
- Testing the correctness of the waveforms and applications that the CF instantiates
- Testing the correct operation of waveforms and applications as they operate
- Testing the correctness of the waveform tear down mechanism in the CF

The architecture addresses the concern that only the correct processes occur within an application by requiring that the CF itself, and all application software installed in any given JTR be authenticated when downloaded into the JTR, and be subject to an integrity check - both functions being performed using cryptographic tests for source authentication and software integrity. To verify that the CF has been properly instantiated, known answer tests in the form of test waveforms can be exercised at CF instantiation to verify test waveform operability. When a waveform is instantiated, Software Profile comparisons are made via an application to determine violations of the waveform and radio security policy tables prior to waveform instantiation. A table is then maintained of the waveform interconnections, as instantiated, so that further monitor tests can be performed to determine that the interconnection of objects has not changed. Such test applications takes the form of process monitors under the guard and monitor definitions shown in figure 4-7. Tear down is assured by the requirement to erase program and data memory under the rules of object reuse as part of the tear down process.

An enhancement to the Core Framework that is required for security is the creation of protected memory. Within the protected memory space, the following elements are contained:

- *DomainManager*
- Domain Profile
- *ApplicationFactory*
- Naming Service

The Naming Service interface exposed to objects outside the *DomainManager* process space is restricted to the bind() and unbind() functions. Inside the *DomainManager* process space, the full Naming Service interface is made available so the *DomainManager*/Application Factories may perform the resolve() and list() functions. This effectively restricts access to object references outside the *DomainManager* process space. The *DomainManager*/Application Factories are then able to enforce the middleware information flow security policy by passing object references to establish message flow within an application.

Tables that are maintained with the *DomainManager* in support of security include:

- Installed devices
- Instantiated devices
- Individual device capabilities
- Device capability allocations
- Installed applications (Software Profiles)
- Instantiated Application Details

The installed devices are a list of the various devices installed within the JTR. The instantiated devices are lists of devices that have been instantiated within the system, presumably at system startup or boot. The device profiles will contain the device capabilities. The device capability allocations are the tables maintained with the CF *DomainManager,* which identify the allocations of device capabilities to application instances. The installed applications are a list of

the applications installed on the JTR with pointers to the application Software Profiles. (Instantiated application details are tables built by the *DomainManager*/*ApplicationFactory* as the application is instantiated.) These tables include:

- Device capabilities allocated to the application
- Application component allocations
- Application component object references
- Application component ports
- Interconnections (message paths) between application components (wiring diagram)

In addition, the security manager (within the cryptographic boundary) maintains the master lists of the following tables. (The *DomainManager* may maintain copies or subsets of these tables as well.)

- Cryptographic algorithm holdings
- Cryptographic key holdings

Any cryptographic key or algorithm information maintained in the *DomainManager* should remain unclassified in its information content.

The application instantiation security policy is contained in the Software Profiles for each application. The application Software Profile identifies resources requiring data or process isolation, and describes the allowed information flow between objects. The *DomainManager* and *ApplicationFactory* set up the middleware data isolation and information flow security policy. The *ApplicationFactory* allocates device capabilities required for each application upon instantiation, directs the individual *Devices* to create instances of the necessary objects, and establishes the allowed information flow between objects. It is required that a compilation of established connections be developed and maintained.

Devices perform a similar function at the discrete processor or device level. *Devices* create processes that run on a local node, create inter-process communication paths, and establish any potential operating system services.

When *DomainManager*s, *ApplicationFactories*, and Devices are used to establish and enforce data isolation and information flow policy within the JTR, they undergo security evaluation. Further, the *FileSystem* and *FileManager* allow access to files throughout the JTR. When the *FileSystem* and *FileManager* are used to enforce read, write, and execute security policy on all files within the JTR based on user/object identity, they will undergo security evaluation.

Any function supporting audit functionality will have access control capability to restrict object access to audit information.

3.6.4 Guards.

The Security elements of the JTRS Architecture provide required Security Services in a multi-channel environment, where the channels may be at different Security Levels. The JTRS ORD requires System-High operation with eventual transition to MILS and multiple levels of security (MLS). System-High operation presents operational difficulties since it precludes concurrent operation of unclassified and secret channels. Thus, different Security Levels should be considered from the beginning. This section describes the use of guards as elements of the architecture that support this approach.

3.6.4.1 Discussion.

The architecture provides for a Trusted Path between the information source and encryption *Resource*. The Trusted Path guarantees source authentication, channel separation, and data integrity of the information-bearing data stream at some level of trust, specified by the Security Policy. The Trusted Path provides the required security services under the Integrity/ Authentication and Enforce Policy requirements of that figure. The Trusted Path can be implemented by a variety of hardware devices and/or software components such as physical separation, services of Trusted Operating Systems, security services of CORBA, cryptographic signatures, and point-to-point encryption. An implementer selects the elements that are appropriate to the specific JTRS and implement them with a level of trust such that the layered effect supports the trust required by the system's Security Policy

Traditional radio systems with embedded security have only one Security Policy that describes how the radio would be used for a particular application. This doctrine also describes any restrictions or specific operational procedures that have to be used. The JTRS requires a Security Policy that contains the specific user domain information including doctrine for each waveform application in the implementation. This Security Policy will need to contain the relevant security information for a particular waveform. An example of this information would be what cryptographic algorithm is to be used with the instantiated waveform. This composite policy is the Security Policy for the JTRS implementation.

3.6.4.2 Approach.

The creation and use of a trusted path can be realized by several means. One example is implementing physically separate hardware paths. A Trusted OS could be used throughout the JTRS to provide the proper level of trust to separate traffic channels. Another example is to use a security guard to isolate traffic channel data. An example of how guards can be implemented in the SCA as security resources is shown in figure 3-13. As such, one can abstract a security guard bus that permits only guards to release data to the guard bus.

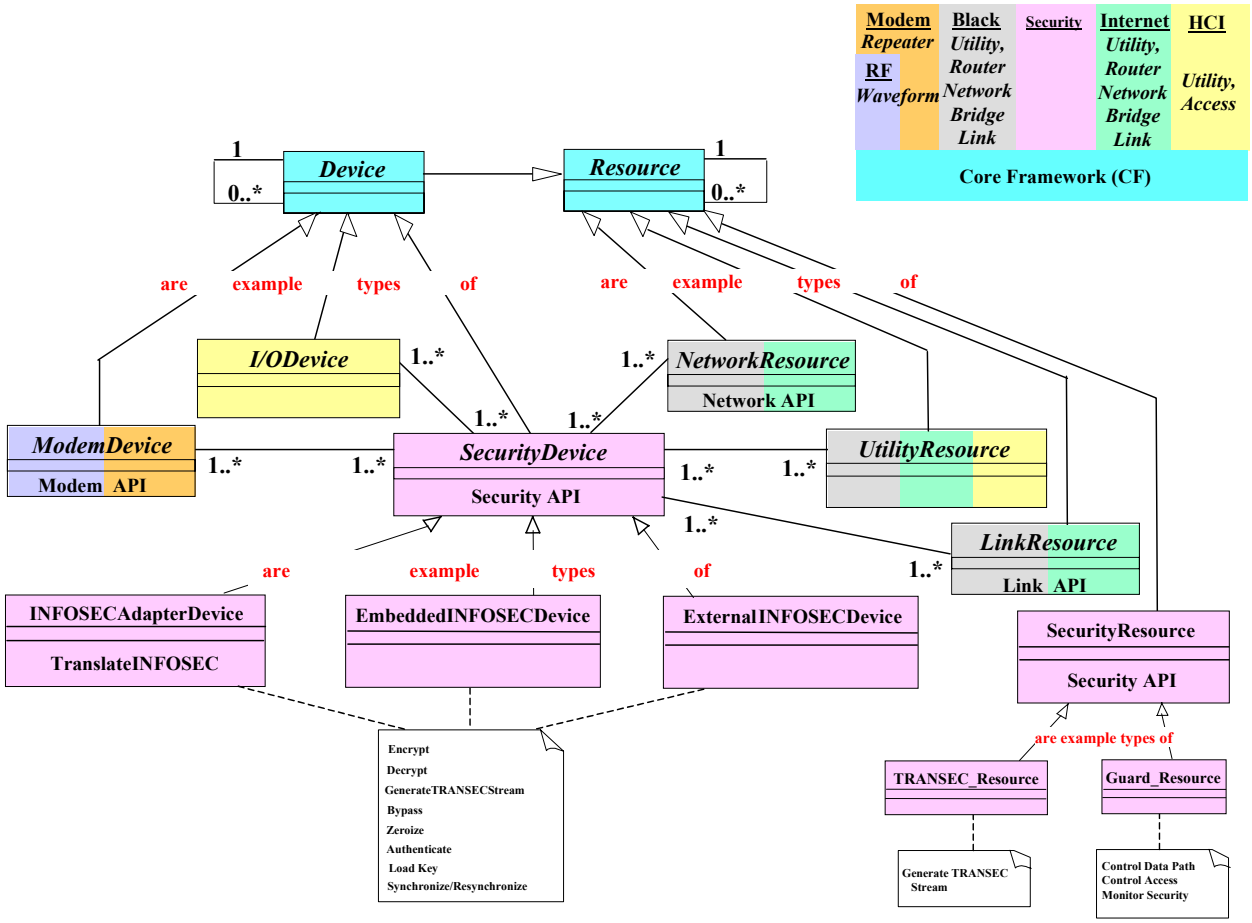


Figure 3-13. Examples of Guards as a Resource

Conceptually, the security guard has two functions. The first function is to uniquely manipulate data through a mathematical process. This process can change the data or add to the original data a label. The second function is to validate the guard bus data such that:

- The data was not changed in any way.
- The data that is trying to pass through the guard is valid for that guard.
- During a waveform instantiation, producer and consumer Ports are created to move data. Ports will be uniquely created for each channel that is associated with each waveform. Security Guards can be created and assigned with one and only one channel. A consequence of the above guard definition is that Channel A's guard cannot pass data through Channel B's guard.

Figure 3-14 illustrates a simple use case diagram for a guard that applies labels to the data.

The diagram shows that once data has passed from a port to the ApplyLabel use case, data and its label appear on the guard bus that was discussed earlier. Once on the guard bus, only guards can validate labels and allow data to move through the system.

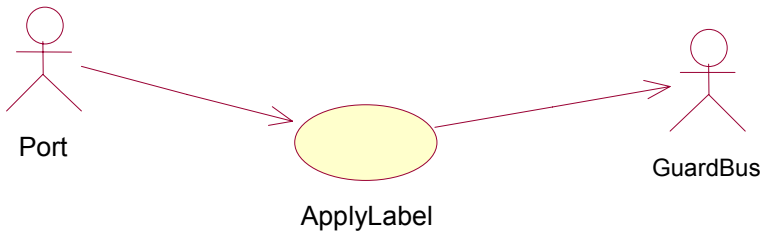


Figure 3-14. ApplyLabel Use Case

Figure 3-15 illustrates a second use case showing how the data is processed from the guard bus.

The ValidateLabel use case checks that the accompanying label is correct, strips the label and releases the data for Encryption or RedDataProcessing use cases. For the RedDataProcessing use case, the label needs to be recalculated and applied before the data can move onto the GuardBus.

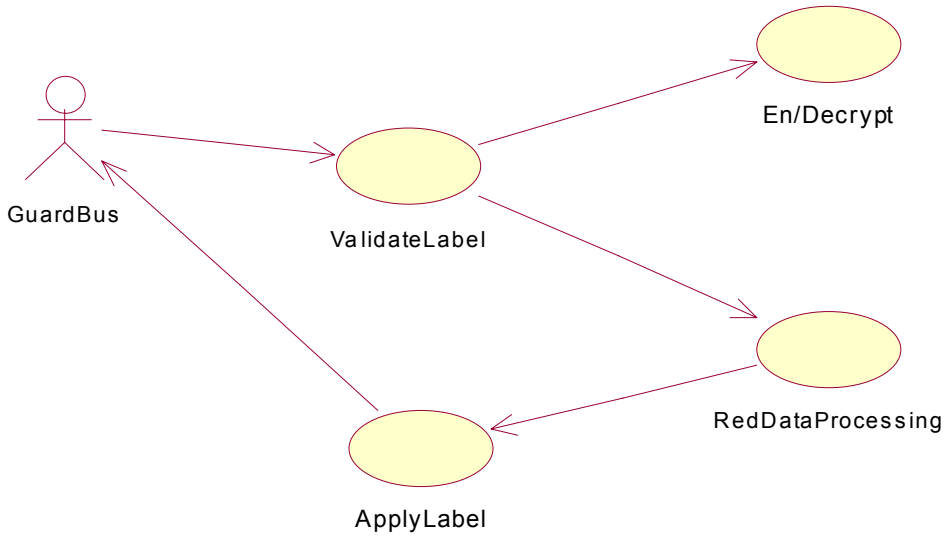


Figure 3-15. Processing Data from the Guard Bus

Multiple types of guards can be created. One implementation is with the creation of a *Port* for a waveform instantiation. The class diagram in figure 3-16 depicts how this guard can be created.

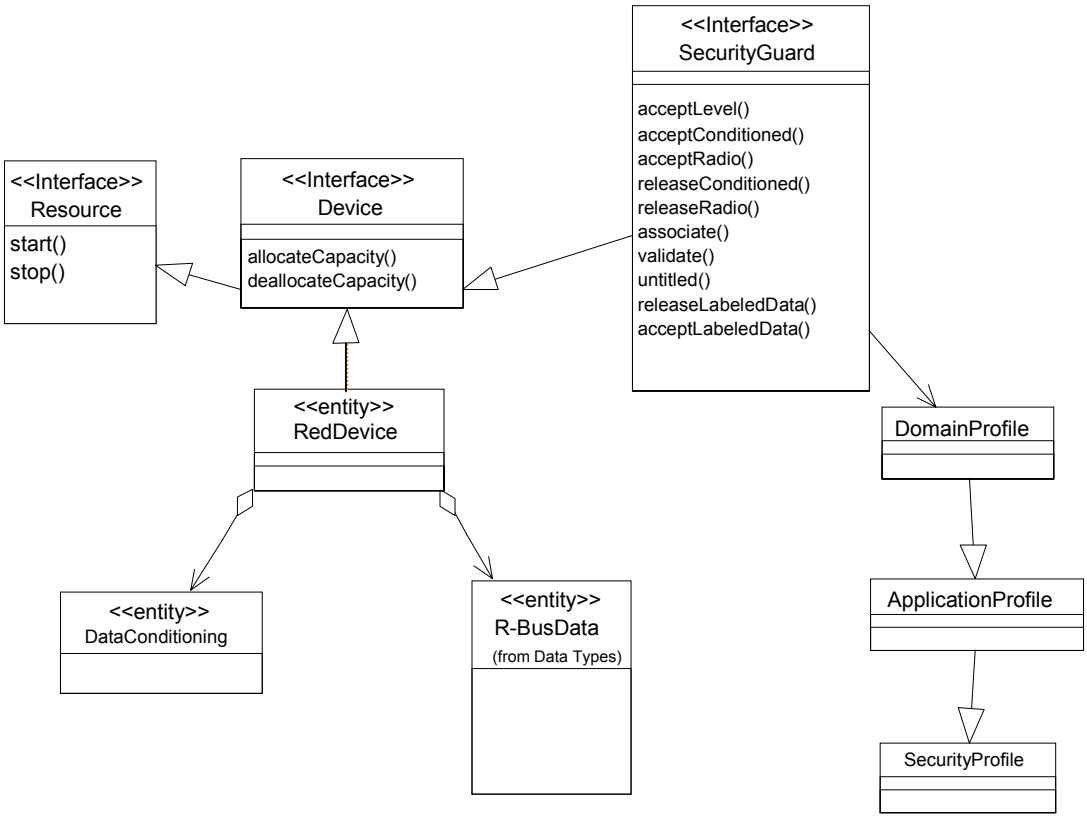
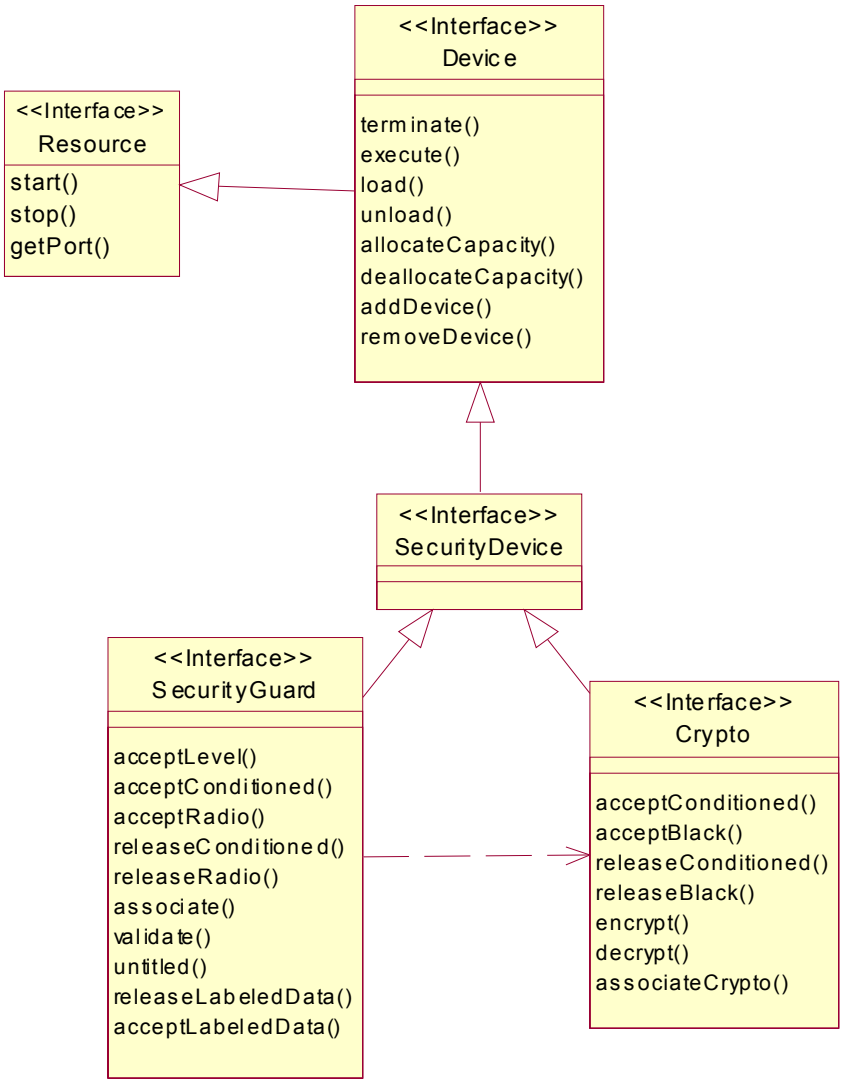


Figure 3-16. Example of a Port Specific Guard

The second type of guard can be created with the security device, see figure 3-17. This guard does not use the supplied configuration file. It independently builds the guard with the channel information.



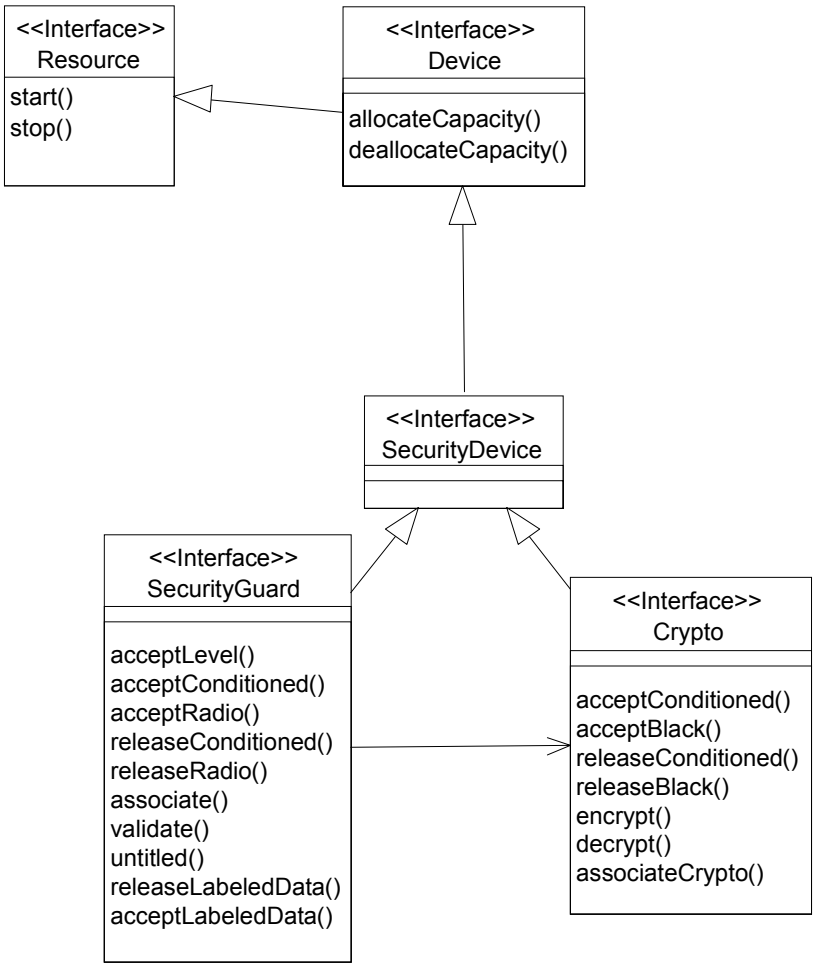


Figure 3-17. Example of a Security Device with a Guard

4 JTRS SECURITY ARCHITECTURE REQUIREMENTS.

4.1 SUMMARY.

This section takes each aspect of the JTRS Architecture as defined in the SCA and provides discussion in terms of security requirements. The initial discussions explain boundaries and how these boundaries apply to the JTRS radio. Security requirements are delineated in Sections 4.2 (cryptographic subsystem requirements) and 4.3 (computer security requirements).

The security architecture divides the JTR into three parts that provide:

- RED Processes and functions
- Cryptographic processes and functions
- BLACK processes and functions

As shown in figure 4-1 and figure 4-2, applications operate in the RED and BLACK processing sides of the radio while the cryptographic function maintains the separation of RED and BLACK information in the system. In the specified system high environment, the cryptographic functions require higher security assurance than either the RED or BLACK processes. Therefore the architecture dictates a cryptographic subsystem that is highly isolated from the software and processes that configure and control the RED and BLACK processes. The security architecture provides for a specialized bypass function that permits communication between the RED and BLACK sides of the radio for JTR internal control and status functions. The security architecture is an overlay on the SCA. The security architecture makes use of the facilities of the SCA, particularly the CF to the maximum extent possible requisite with required security.

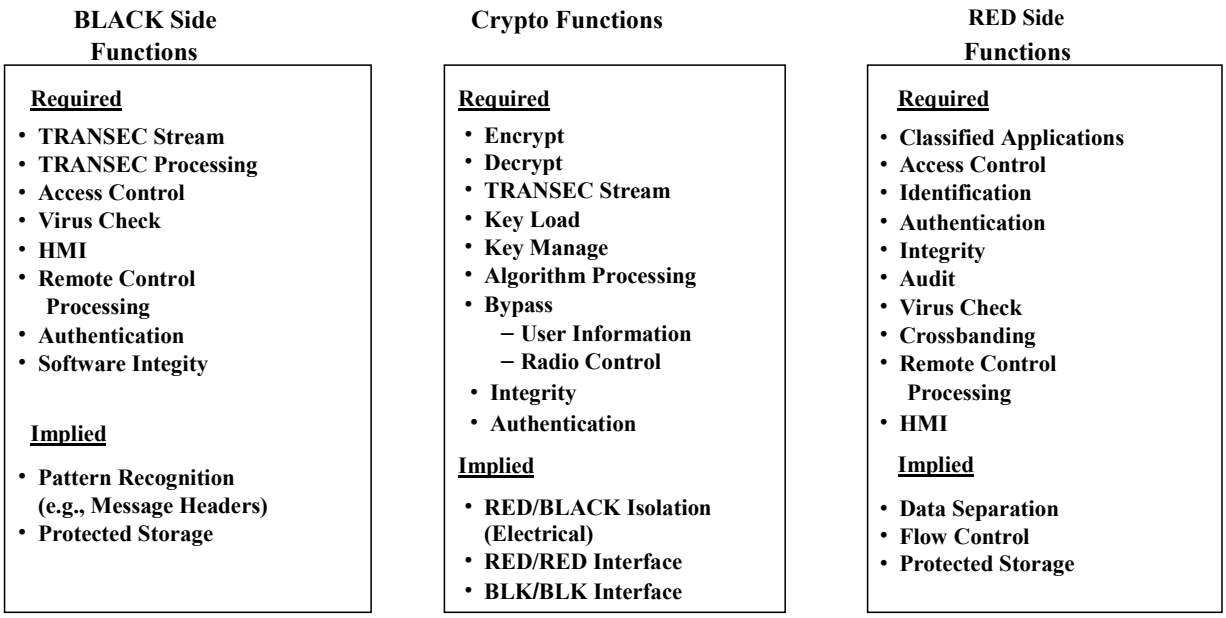


Figure 4-1. JTRS Functional Security Allocation

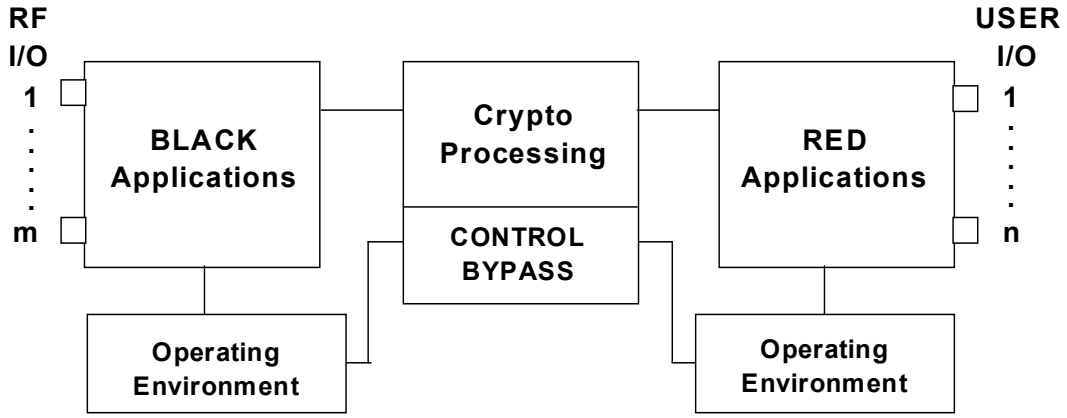


Figure 4-2. Basic JTRS Security Architecture

For legacy hardware, implementations of security functions such as encryption/decryption and key management), security boundaries were drawn physically to contain a set of hardware components that were security critical. For JTRs, security boundaries are drawn functionally rather than physically. The Cryptographic Subsystem physically contains all the high assurance security functions shown in figure 4-1. The software nature of implementation leads us to the functional partitioning since security related processes can be performed in different processors.

The implied functions become the drivers in the structure of the security architecture. These implied functions set the common basis that spans a set of functional requirements. For instance, functions such as access control, authentication, and multi-communicator operation all imply that certain data will be protected from unauthorized access, and will be isolated from other data elements.

Note that several functions shown in figure 4-1 are redundant among the three function sets. The redundancy is based on the level of assurance required for a given function. For example, certain TRANSEC waveforms operate totally within the modem function, while other TRANSEC key streams are generated using cryptographic functions within the cryptographic boundary.

A basic presentation of the JTRS architecture for secure radios is shown in figure 4-2. The architecture has RED applications processing, crypto applications processing, and BLACK applications processing. The control element is not shown. The SCA does not allocate the control element to a specific location. The presence of a bypass for radio control and status is a direct result of the need for OE and applications elements to communicate with each other in unencrypted form across the cryptographic function.

Figure 4-2 depicts the use of the cryptographic subsystem with its bypass function to control the transfer of both communicator and control/status information from the RED to the BLACK side of the JTR. There are obviously details of the SCA and the JTRS ORD requirements that are not captured in figure 4-2. For example, the diagram does not show any of the software allocations or hardware device structures, or address the remote control requirements. However, the figure does provide a basis to allocate security elements, and to define the means used to insert security into the SCA, and to evaluate the levels of security provided.

The JTRS Security Architecture involves:

- Cryptographic functions (crypto processing and bypass)
- RED and BLACK processing functions
- Equipment level functions (not shown in figure 4-2)

Figure 4-3 shows a set of blocks (broken line patterns) that functionally provide boundaries for different types of radio processing and security mechanisms. The boundaries define different methods by which security assurance will be achieved. Radio operating requirements (e.g., system high, compartmented) also strongly drive specific security assurance requirements.

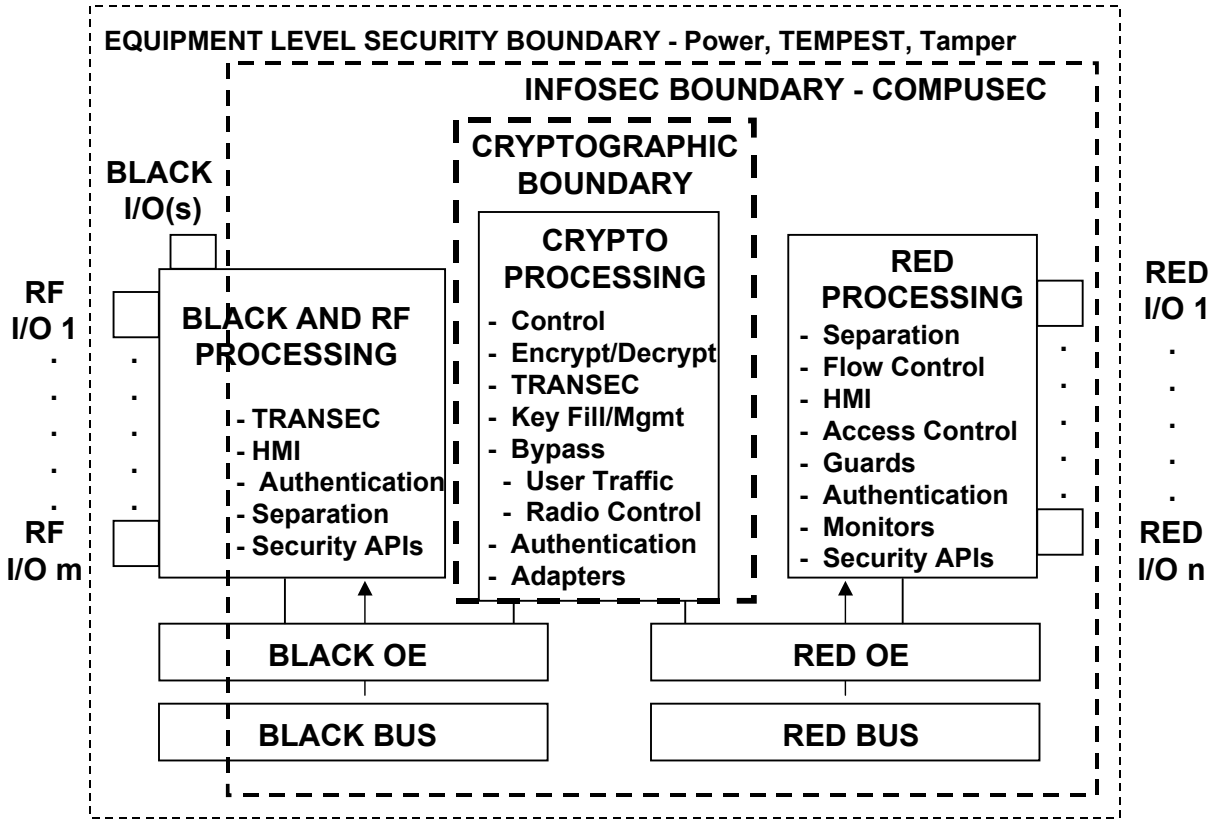


Figure 4-3. JTRS Security Boundaries

From a security design and evaluation point of view, the crypto processing boundary would be treated in a "traditional" manner using high assurance design and evaluation methods, as well as NSA approved modules, algorithms, and protection mechanisms. The protection of communicator information is fully dependent on the quality of the cryptography, the protection of cryptographic keys, and the control of cryptographic and bypass functions to prevent inadvertent or malicious output of classified communicator information.

The highly integrated capabilities of a JTR, and the software nature of its implementation require that the JTRS Security Architecture include hardware and software elements that reside in RED and BLACK processing that are not part of legacy cryptographic equipment (whose only function is cryptographic protection). The requirements for protection and separation of

information within the RED (and possibly the BLACK for functions such as authentication, integrity and TRANSEC) processing functions will be addressed using COMPUSEC design and evaluation principles.

As an adjunct to the security architecture presented in the JTRS Security Supplement, it is also necessary to define sets of control parameters that define what security enforcement is required for a given radio configuration. Taken together, these control parameters define a *security policy* for the radio. The security policy is an aggregate of information from different sources to include the radio OE and ORB, the downloaded waveforms, and operator/security officer entered parameters. The security of the JTR is thus tied into the supporting infrastructure for the equipment. The integration of the security policy into the proper components of the JTRS architecture is key to successful implementation of robust systems security.

4.1.1 Security Boundaries.

The following paragraphs discuss the functional content of the security boundaries shown in figure 4-3.

4.1.1.1 Cryptographic Boundary.

The security architecture defines a cryptographic boundary of functions, software, and hardware that are tightly physically and logically coupled to one another in start-up and operation. The crypto functions are not instantiated by the Core Framework, but rather are separately booted into operation at JTR start-up. The mechanisms that configure the basic cryptographic functions are self-contained within the cryptographic boundary. The OE of the JTR is used to establish communications paths to the borders of the cryptographic boundary. These boundary interfaces provide data, control, and status paths into and out of the cryptographic boundary.

Requirements are provided in Section 4.2 of this document.

4.1.1.2 INFOSEC Boundary.

The INFOSEC boundary defines functions that use computer security technologies rather than cryptographic technologies. It is comprised of all types of processing, storage, and software interconnections performed within the RED and BLACK processing areas. The important security elements of the software processing environments include:

- Ability to control access to privileged information
- Ability to maintain accurate inter-process/inter-object flow of information
- Ability to assure authenticity and integrity of software
- Ability to provide a HMI
- Ability to provide an Application Program Interface (API)

Requirements are provided in Section 4.3 of this document.

4.1.1.3 Equipment Level Boundary.

The equipment level security functions are those that span across the "housekeeping" functions of the JTR and equipment level protection. These functions are Tamper protection and TEMPEST protection against compromising emanations, which includes power supply filtering for electrical RED/BLACK isolation and power failure detection.

4.1.2 Overall Security Architecture Requirements.

1. The Security Architecture shall permit Secret System High traffic.

4.2 CRYPTOGRAPHIC SUBSYSTEM REQUIREMENTS.

The Cryptographic Boundary encapsulates a set of security critical functions that provide RED/BLACK separation within the system high JTR. This set of security critical functions collectively constitutes the CS/S. The CS/S performs multiple functions that provide RED/BLACK separation including encryption/decryption, controlled bypass of communicator and control information, and electrical isolation. The encrypt/decrypt functions also incorporate the supporting functions to include; key and algorithm management, cryptographic channel instantiation, and cryptographic control. CS/S requirements can vary according to user functional specification, but the security architecture herein provides the capability to satisfy a broad range of functional security requirements.

This section provides a discussion and then requirements for the cryptographic elements of the JTRS security architecture. It defines a self contained CS/S that is instantiated independently from the OE, and has restricted interfaces to the rest of the JTRS architecture elements. The CS/S interfaces to the rest of the JTRS functions at the Cryptographic Boundary. The CS/S defined herein applies for Type 1 cryptography for the protection of DoD classified information in tactical environments.

The cryptographic functions are composed of the following elements:

- Initialization, Operation, and Termination
  - Boot
  - Instantiation
  - Run-Time
  - Normal Termination (Channel/Teardown)
  - Abnormal Termination
- RED/BLACK Isolation
- Keystream Functions
  - Encrypt
  - Decrypt
  - TRANSEC
  - Identification and Authentication
  - Integrity
- Security Management Functions
  - Key Management Functions
  - Algorithm Management Functions
  - Security Policy and Enforcement
- Cryptographic Bypass
  - Communicator Data
  - Radio Control/Status
- Cryptographic Control and Status
- Cryptographic Interfaces

The primary functions, within the CS/S, are:

- Cryptographic Keystream generation capability
  - Encryption and decryption of communicator information
  - TRANSEC functions
  - Generation and validation of high-grade signatures
  - High-grade integrity checks
- Controlled bypass of communicator and radio information

All of the other mechanisms within the boundary are used to service and protect the keystream generation capability and to enforce security policies for data separation and access. For proper operation, the CS/S is placed into a known good operating state prior to running cryptographic functions. Therefore, the initialization, instantiation, and teardown of cryptographic functions are critical to the security provided by the CS/S.

Figure 4-4 provides a functional block diagram of the CS/S showing the allocation of the previously listed functions. All CS/S interfaces to external devices within the JTR are at the Cryptographic Boundary.

4.2.1 General Security Requirements.

For the implementation of JTRs, the following overall security requirements apply:

1. The JTR shall use only NSA certified cryptographic chips and modules for Type 1 security functions.
2. The JTR shall implement cryptographic algorithms for specific applications as determined and specified by NSA.
3. The JTR shall be implemented in accordance with Unified INFOSEC Criteria (UIC) requirements for high security cryptographic applications.
4. The JTR shall be implemented according to the system requirements of the UIC to include TEMPEST, tamper protection, and power requirements.
5. Performance of ancillary cryptographic functions (e.g., algorithm decryption, authentication) shall not reduce the number of cryptographic channels available for communicator data operations.

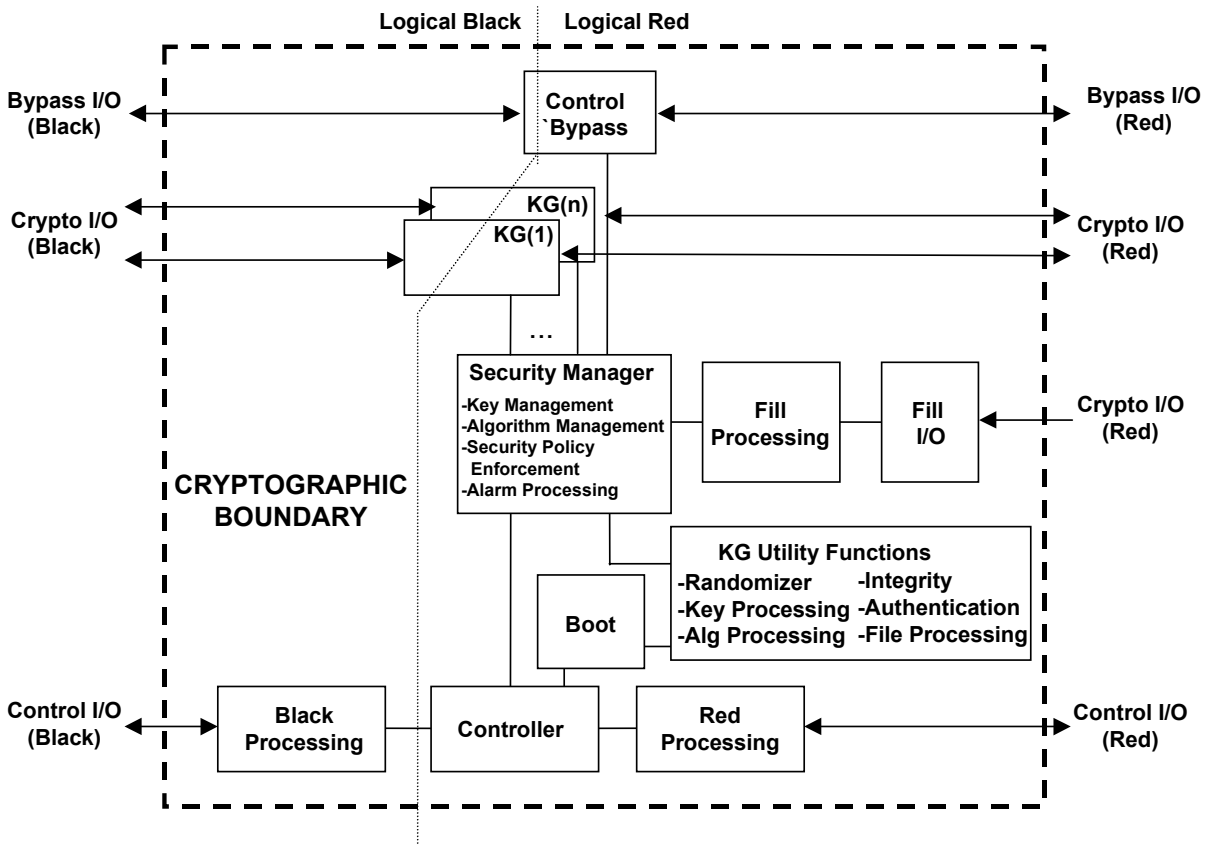


Figure 4-4. Cryptographic Boundary Functions (The Cryptographic Subsystem)

4.2.2 CS/S Initialization, Operation, and Termination.

This section deals with the start-up, use, and closure of the functions within the CS/S. The CS/S security posture is enhanced by the degree of independence the CS/S maintains from the operations of the OE and user applications.

4.2.2.1 CS/S Initialization, Operation, and Termination Discussion.

The CS/S is isolated from the services of the SCA. The CS/S performs a start-up boot independent of the OE boot and CF instantiation. It communicates its status and accepts requests/commands via the APIs or adapters to the overall radio control functions in the OE and HMI as an Audit Producer and Consumer (see paragraph 4.3.7 for audit discussion).

The references to CS/S instantiation below handle the instantiation of the CS/S itself as part of internal boot, while initialization refers to the instantiation of algorithms and communicator channels within the cryptographic boundary in response to waveform requirements and HMI instructions. In the same manner that the CF is instantiated before communicator waveforms can be instantiated, the CS/S core functions are instantiated before algorithms and communicator channel paths can be instantiated.

The reference to abnormal channel termination below indicates that a failure of some type occurred in the JTR or in the CS/S that would cause the communicator channel to shut down, or an unknown operating state of the radio to occur.

4.2.2.2 CS/S Initialization, Operation, and Termination Requirements.

The start-up is composed of CS/S boot and function initialization. The run-time requirements involve the stability and security of waveform operations during actual processing. Terminations may either be normal teardown of functions, or abnormal termination due to some type of error or failure.

4.2.2.2.1 CS/S Boot Requirements.

1. The CS/S shall utilize a boot function contained within the Cryptographic Boundary.
2. The CS/S shall perform validity checks of data stored on non-volatile memory.
3. The CS/S shall perform internal health tests to assure proper interconnection of objects and functionality.
4. The CS/S shall instantiate the cryptographic algorithm decryption capability prior to waveform instantiation.
5. The CS/S shall instantiate the key fill capability prior to waveform instantiation.
6. The CS/S shall instantiate the key decryption capability prior to waveform instantiation.
7. The CS/S shall instantiate the randomization capability prior to waveform instantiation.

4.2.2.2.2 CS/S Application Instantiation Requirements.

1. The CS/S shall perform validity checks on the cryptographic algorithm requirements (e.g., algorithm type and mode sets) provided by the CF from the waveform software profile.
2. The CS/S shall decrypt and instantiate algorithms as part of the JTR waveform instantiation process.
3. The CS/S shall perform internal tests of instantiated algorithms prior to waveform operation.
4. The CS/S shall receive interconnection files for instantiated waveforms from the *DomainManager*.
5. The CS/S shall accept requests for key instantiation to support a specific waveform and net application.
6. The CS/S shall verify that there are no violations in matching classification levels of algorithms, waveforms, and keys during instantiation.
7. The CS/S shall provide random bit stream generation to support cryptographic initialization functions.

4.2.2.2.3 CS/S Run-Time Requirements.

1. The CS/S shall perform periodic tests to assure that instantiated operations are maintained.
2. The CS/S shall provide status outputs to authenticated requests as required.

4.2.2.2.4 CS/S Normal Termination (Channel Teardown).

1. The CS/S shall erase all cryptographic data memory upon cryptographic channel teardown.
2. The CS/S shall zeroize the cryptographic channel key(s) upon channel teardown.
3. The CS/S shall erase the cryptographic channel crypto algorithm upon user channel teardown.
4. The CS/S shall signal channel teardown complete on status output to Audit.

4.2.2.2.5 CS/S Abnormal Termination Requirements.

1. In the event of internal failure, the CS/S shall execute internal security policy for termination or suspension of operation.
2. The CS/S shall monitor external alarm and warning conditions, and execute actions according to security policy requirements.
3. The CS/S shall provide failure status to Audit, if possible.
4. The CS/S shall not execute automatic retries to recover from alarm conditions.
5. The CS/S shall accept requests for reset/retry from authenticated sources to recover from alarm conditions.

4.2.3 Keystream Functions.

Figure 4-5 shows examples of the different uses of keystream and algorithmic functions to satisfy security requirements. The functions are discussed in the following paragraphs.

4.2.3.1 Keystream Discussion.

NSA requires that the Type 1 algorithms (See paragraph 2.2.3) produced to protect DoD classified information be implemented and operated in NSA certified cryptographic modules and subsystems. The encryption and decryption of communicator traffic is the traditional role of the cryptographic function to provide protection for information broadcast over unprotected transmission media. The transfer of information in this scenario is encrypt RED and decrypt BLACK.

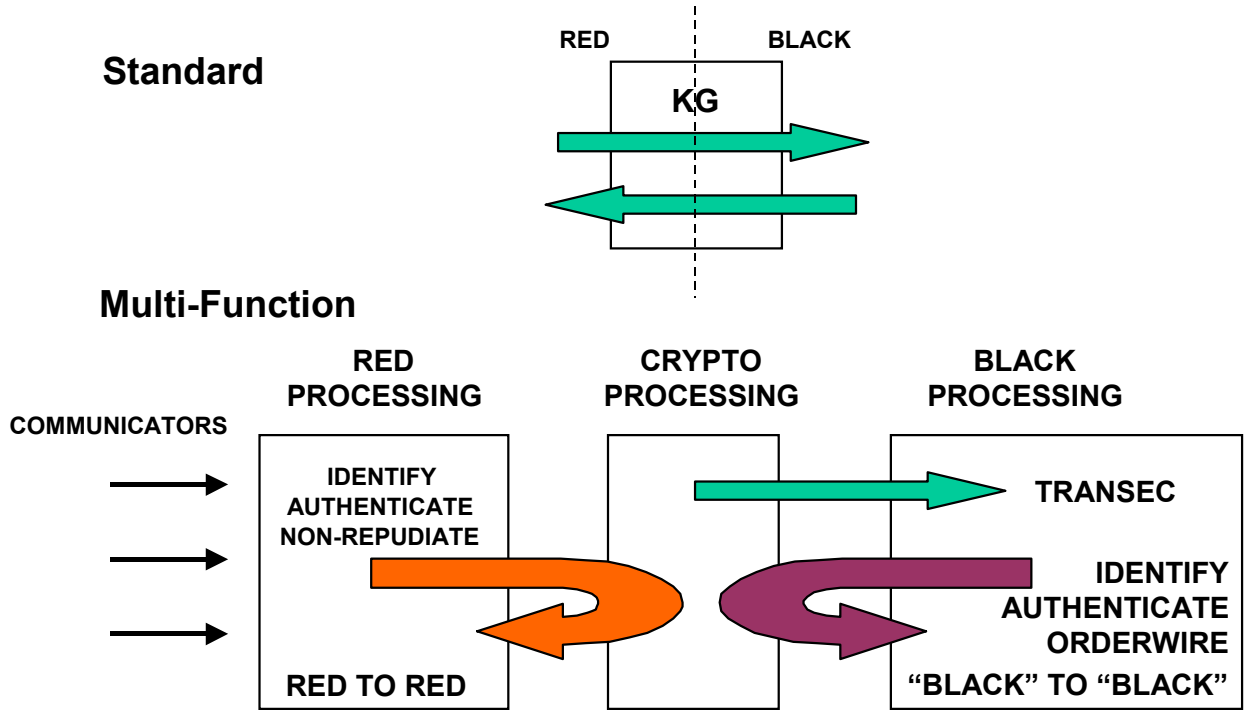


Figure 4-5. Types of Keystream Utilization

Most high-grade TRANSEC keystream generation systems (i.e., Type 1) are generated internal to the cryptographic boundary. However, TRANSEC modes can be internal or external to the cryptographic boundary at the discretion of the designer. When keystream generation is external to the crypto boundary, the issue of key fill for the function is raised.

4.2.3.2 Keystream Requirements.

Addressed generically, the term "keystream" refers to the generation of bits according to the rules of motion of a cryptographic algorithm.

4.2.3.2.1 Encrypt Requirements.

1. The CS/S shall provide Type 1 encryption capabilities for protection of classified information.
2. When required for a specific application, the CS/S shall encrypt software files for JTR internal storage.
3. When required for a specific application, the CS/S shall encrypt communicator data for JTR internal storage.
4. The CS/S shall encrypt keying material for JTR internal storage.
5. The CS/S shall be able to use different encryption algorithms.
6. When required for specific applications and configurations, the CS/S shall simultaneously encrypt communicator data using different cryptographic applications.

7. When required for specific applications, the CS/S shall encrypt RED side data and return the processed output to the RED side.
8. When required for specific applications, the CS/S shall encrypt BLACK side data and return the processed output to the BLACK side.

4.2.3.2.2 Decrypt Requirements.

1. The CS/S shall provide Type 1 capabilities for decryption of classified information.
2. When required for a specific application, the CS/S shall decrypt software files from JTR internal storage.
3. When required for a specific application, the CS/S shall decrypt communicator data from JTR internal storage.
4. The CS/S shall decrypt keying material from JTR internal storage.
5. The CS/S shall be able to use different decryption algorithms.
6. When required for specific applications and configurations, the CS/S shall simultaneously decrypt communicator data using different cryptographic applications.
7. When required for specific applications, the CS/S shall decrypt RED side data and return the processed output to the RED side.
8. When required for specific applications, the CS/S shall decrypt BLACK side data and return the processed output to the BLACK side.

4.2.3.2.3 TRANSEC Requirements.

1. The CS/S shall provide keystream to JTR processing functions external to the CS/S for use in TRANSEC waveform development.
2. When required for a specific application, the CS/S shall use time-of-day (TOD) functions for keystream generation.
3. The TRANSEC keystream shall be used immediately after generation, as specified as in the UIC.
4. The TRANSEC keystream shall be distinct from the encrypt and decrypt keystreams.

4.2.4 Identification and Authentication Functions.

4.2.4.1 Identification and Authentication Discussion.

Identification and Authentication (I&A) functions for high-grade access control use cryptographic mechanisms as part of the protection function. There are several additional requirements that are introduced. First is the need to have a security policy that determines approval for access (including a protected access control list and an enforcement mechanism). Second, access control can be required for control of three different sources:

- Radio channel input for BLACK side remote access
- Communicator port input for RED side access
- Control interface for communicator/operator/administrator functions

The differing sources and controls required dictate that a cryptographic function will be able to perform an I&A operation on either the RED or BLACK side of the radio, and return a result to the same side (RED or BLACK) of the radio depending on the required enforcement mechanism. Figure 4-5 shows an example of this type of RED/RED and BLACK/BLACK processing.

4.2.4.2 Identification and Authentication Requirements.

1. The CS/S shall perform cryptographically based authentication for RED messages.
2. The CS/S shall perform cryptographically based authentication for BLACK messages.
3. The CS/S shall return results of authentication processing to the message originating object.
4. The CS/S shall provide Digital Signature Standard (DSS) cryptographic processing.
5. The CS/S shall store DSS certificates.
6. The CS/S shall generate DSS authentication messages.

4.2.5 Integrity Function.

The integrity function verifies the accuracy of information. In the JTRS context, integrity is applied to assure that software and keys to be loaded into and used by the JTR are not modified in transport to or storage within the JTR. In certain circumstances, integrity checks can also be applicable to communicator data.

4.2.5.1 Integrity Discussion.

Integrity can be verified cryptographically or non-cryptographically depending on the criticality of information to be protected, or the degree of access that outside entities might have to the information. For security critical software that may be transported for delivery through unprotected channels, a cryptographically based integrity technique is preferred.

4.2.5.2 Integrity Requirements.

1. The CS/S shall perform cryptographically based authentication for RED messages.
2. The CS/S shall perform cryptographically based authentication for BLACK messages.
3. The CS/S shall return results of integrity processing to the originating object.
4. The CS/S shall provide Secure Hash Algorithm (SHA-1) processing.
5. The CS/S shall generate SHA-1 based integrity checks.
6. The CS/S shall generate standard integrity checks [e.g., parity, cyclic redundancy check (CRC)] for data.

4.2.6 Security Management Functions.

The traditional key management role performed within cryptographic systems is expanded to include handling of software algorithms and security policies for multi-channel systems where functions can be changed.

4.2.6.1 Key Management Functions.

4.2.6.1.1 Key Management Discussion.

The key management functions are the most security critical set of functions in the JTR since the compromise of keys can compromise the information on a full network of communicator systems. A local security failure can compromise a single device or message, whereas compromise of a key compromises the full network. The primary key management functions are:

- Key Receipt and Identification
- Key Storage
- Key Allocation and Use
- Key Zeroization
- Key Accounting
- Rekey

For the purposes of this security supplement, cryptographic keys, certificates, credentials, and Firefly vectors are treated as keys.

Keys received from distribution channels are be in accordance with EKMS standards for form factor, format, and tagging. The DS-101 protocol will be used for fill of keys (Data DS-100-13) format) and algorithms into a JTR. For files of keys loaded into black non-volatile memory, keys need to be in the Tag 100-1 issue format. JTR system keys [e.g., key encryption keys (KEKs), JOSEKI keys, File encryption keys, crypto ignition keys (CIKs)] that provide security for JTRS will be filled at relatively long intervals using benign techniques. Relatively frequent routine distribution of keys, algorithms, and other encrypted JTR configuration data will be in BLACK form.

The goal of effective key management and utilization is to keep all keys in BLACK (encrypted form) to the maximum extent possible. The EKMS provides facilities for the transfer of BLACK (encrypted) keys and for Benign Fill.

Primary storage of keys is in BLACK form, encrypted by a key encryption key. BLACK keys can be stored in any available memory or storage, volatile or non-volatile, as long the tagging information is protected and bound to the keys. That is, each tag is tightly coupled to the key that it identifies. When keys are in use in RED form, they should exist only in the working registers of the cryptographic function.

Key selection for use is provided from the HMI or via a mission download instruction loaded into the JTR. The actual allocation and transfer of keys in RED form into cryptographic working storage is performed within the cryptographic boundary except for TRANSEC and GPS functions, where keys may be passed external to the crypto boundary in unencrypted form. It is noted that the key loading process is largely independent of waveform instantiation, although key selection for a channel does occur as part of waveform instantiation, or after the waveform is instantiated, but prior to actual run-time operation.

The only assured method for zeroization of keys is a manual, operator/communicator activated function that eliminates all RED keys. Software methods for zeroization can provide a back-up capability if the request is cryptographically authenticated (authenticated using cryptographic methods in the CS/S, such as DSS), as would be the case on a remote zeroization or an over-the-

air zeroization (OTAZ) request scenario. BLACK keys in non-volatile storage should be erased if power is available in a zeroization, but in cases of power failure, such a function may not be possible. In all cases, the key encryption key that decrypts the BLACK keys is zeroized. There are also numerous cases in which keys can be selectively zeroized or erased. Primary among the cases is the need to destroy keys that have been superceded on a net, or updated. Given the different types of keys that could be present in a JTR (e.g., key encryption, algorithm encryption, communications, certificates, vectors) systems engineering efforts will be required to determine the actual sequence of events for zeroization and erasure.

For legacy systems, accounting for transfer, use and destruction of keys is largely manual and is paper intensive. The AN-CYZ-10 Data Transfer Device (DTD) used to load keys for cryptographic equipment provides an audit function, but does not yet provide a convenient means for transfer of audit information for analysis. For keys in electronic form, the accounting for keys becomes more difficult without a physical form factor (e.g., key tape). The accounting function is further complicated by the potentially large number of BLACK keys that might be stored in a JTR. It is apparent that a communicator/operator of a JTR needs to have access to the listing of keys stored within his/her equipment and that the display of key related information is unclassified. The key management function within the crypto boundary has to be able to provide status on the keys loaded, used, stored, and destroyed within the equipment, and the information has to be restricted in access to authorized individuals (or internal radio functions) using access control mechanisms, and protected memory.

The simplest case of rekey is the loading of a new key using EKMS methods. However, other methods involving over-the-air rekey (OTAR) or OTAT requires consideration. Presently, OTAR functions are conducted as waveform specific actions. It appears reasonable for the JTRS community to develop a waveform independent over-the-air key transfer and over-the-air zeroize mechanisms as system application. Additionally, designers will need to consider waveform specific non-standard keying requirements on a case-by-case basis.

4.2.6.1.2 Key Management Requirements.

It is anticipated that all key management requirements will be satisfied within the cryptographic boundary with the exception of HMI based command and status requests, and possibly the storage of keys in BLACK form. For the purpose of these requirements, asymmetric vector sets for development of keys and authenticators are considered to be keys themselves.

4.2.6.1.2.1 Key Receipt and Identification Requirements.

1. The JTR shall accept BLACK keys using the DS-101 protocol.
2. The JTR shall accept RED keys using the DS-101 protocol.
3. The JTR shall accept DS-100-1 key tagging information.
4. The JTR shall be capable of binding key tagging and ID information to a key.
5. The JTR shall accept RED keys using the DS-102 protocol.
6. The JTR shall accept encrypted keys using the DS-102 protocol.
7. The JTR shall permit the operator/security officer to enter key identification information.

8. Key identification information (Tag 100-1 or optional text, e.g., "text ID") for stored keys shall be available to an authenticated user.
9. In a multi-channel radio environment, the CS/S shall accept black key load while other channels are operational (for local fill).
10. The JTR shall accept Data 100-1 key fill data information.
11. The JTR shall have a benign fill capability for JTR System keys (i.e., not TEKs).

4.2.6.1.2.2 Key Storage Requirements.

1. Key storage shall be in BLACK form.
2. Classified keys in RED form shall not exist outside of the Cryptographic Boundary.
3. Keys shall be bound to their respective identification information in storage.
4. The CS/S shall encrypt developed keys for storage.
5. The CS/S shall encrypt updated keys for storage.

4.2.6.1.2.3 Key Allocation and Use Requirements.

1. Keys shall be requested for use according to their tag or ID information.
2. The CS/S shall not permit keys to be instantiated for an improper application. For the purposes of Requirement Number 2 in this paragraph, an “improper application” of a key is:
   a. Using the wrong classification key
   b. Using the wrong key type (e.g., TEK, KEK, TSK)
   c. Mismatching a key with an algorithm (e.g., a KGV-11 key with a KG-84 algorithm)
   d. Using the wrong compartment (e.g., US, NATO, Allied).
3. Keys shall be tested for integrity at instantiation.
4. The CS/S shall permit Unclassified keys to be passed outside of the Cryptographic Boundary for specific purposes (e.g., BLACK processor generated TRANSEC, GPS).
5. The CS/S shall use Tamper detection keys, when required by a specific application.
6. The CS/S shall use split keys, when required by a specific application.
7. The CS/S shall provide token [e.g., crypto ignition key (CIK)] split generation and processing, when required by a specific application.
8. When required for a specific application, the token/CIK function shall disable the classified processing capabilities of the CS/S upon token removal.
9. The CIK function shall declassify the equipment (a minimum of Controlled Cryptographic Item) by removing RED keys and cryptographic algorithms.
10. When required for specific applications, the CS/S shall develop keys via Firefly processing.

4.2.6.1.2.4 Key Zeroization Requirements.

1. The JTR shall provide a hardware mechanism for a user to zeroize all RED keys.
2. The JTR shall provide a software mechanism for a user to zeroize all RED keys.
3. The JTR shall provide a hardware mechanism for a user to zeroize all active key splits.
4. The JTR shall provide a software mechanism for a user to zeroize all active key splits.
5. A capability shall be provided for a user to selectively erase BLACK and RED keys.
6. Remote key zeroization commands shall be cryptographically authenticated.
7. Local HMI software-based zeroization commands shall be authenticated.
8. Legacy key update functions shall erase the prior version of the key .
9. BLACK keys shall be erased as part of zeroization if power is available.
10. When required for a specific application, the JTR shall generate OTAZ messages.
11. When required for a specific application, the JTR shall receive, authenticate, and process and respond to OTAZ messages.
12. OTAZ messages shall be cryptographically authenticated.

4.2.6.1.2.5 Key Accounting Requirements.

1. The CS/S shall maintain an identification file of cryptographic key holdings correlated with waveform, algorithm, and network management key data.
2. When audit is invoked, the CS/S shall maintain an identification file of keys loaded and keys zeroized or erased.
3. The CS/S shall provide information to authorized users and to the *DomainManager* as to which keys have expired.

4.2.6.1.2.6 Rekey Requirements.

1. The JTR shall accept keying messages over a radio (BLACK) channel using an OTAT function.
2. When required for a specific application, the CS/S shall develop OTAR messages for legacy interoperability.
3. When required for a specific application, the CS/S shall accept OTAR messages for legacy interoperability.
4. When required for a specific application, the CS/S shall perform legacy key update functions.
5. When required for a specific application, the CS/S shall provide key development using asymmetric vector techniques.
6. OTAR messages shall be cryptographically authenticated.
7. OTAT messages shall be cryptographically authenticated.
8. The CS/S shall not generate keys for external communications use.

9. When required for a specific application, the CS/S shall generate key splits to update token-based functions.

4.2.6.2 Algorithm Management Functions.

Cryptographic algorithms in software form represent the state of the art cryptographic implementation within the Cryptographic Boundary. The discussion and requirements below apply to both legacy and new cryptographic algorithms and the mode settings that apply to them.

4.2.6.2.1 Algorithm Management Discussion.

The layers of the algorithm management process are similar to those exercised for key management. One item to be emphasized. In the event that an algorithm is replaced under a mandatory modification, the holders (users) of the algorithm will be tracked so that the modification can reach all users. This tracking can actually be applied to a broader set of software applications where a method of tracking and assured replacement of software may be necessary.

4.2.6.2.2 Algorithm Management Requirements.

4.2.6.2.2.1 Algorithm Receipt and Identification Requirements.

1. All classified cryptographic algorithms shall be decrypted using the JOSEKI-1 algorithm.
2. CS/S shall only accept cryptographic algorithms/algorithm packages signed by NSA.
3. The CS/S shall perform authentication and integrity checks of received algorithms or algorithm packages.
4. The CS/S shall report the results of integrity checks as auditable events.
5. The CS/S shall not accept versions of a cryptographic algorithm or algorithm package that are older than the currently installed version.
6. Cryptographic algorithms or algorithm packages shall be identified as to type and version.
7. The JTR shall provide for operator/security officer entry of acceptable versions of cryptographic algorithms or algorithm packages.
8. The CS/S shall accept BLACK key and cryptographic downloads using the security API.

4.2.6.2.2.2 Algorithm Storage Requirements.

1. Cryptographic algorithms or algorithm packages shall be downloaded separate from other JTR non-security software.
2. The JTR shall accept cryptographic algorithms or cryptographic algorithm packages downloaded as software files using the DS-101 protocol.
3. Cryptographic algorithms shall be stored in BLACK form.
4. Cryptographic algorithms shall be bound to their version and type identification in storage.

4.2.6.2.2.3 Cryptographic Algorithm Allocation and Use Requirements.

1. The CS/S shall accept requests for cryptographic algorithm instantiation based on waveform software profile requirements.
2. If the cryptographic algorithm is constructed with configuration options, the CS/S shall accept control requests for cryptographic algorithm options based on waveform software profile requirements.
3. Instantiated cryptographic algorithms shall be tested for correct function prior to operational use.
4. Instantiated cryptographic algorithms shall be erased at cryptographic channel teardown.
5. In a multi-channel radio environment, the CS/S shall accept cryptographic algorithm instantiation while other channels are operational.

4.2.6.2.2.4 Algorithm Erasure Requirements.

1. Old cryptographic algorithm versions shall be erased by a manual action when new algorithm versions are downloaded and verified as to authentication and integrity.
2. Requirements for Zeroization of decrypt key(s) for cryptographic algorithms concurrent with traffic keys shall be design, mission, and environment dependent.

4.2.6.2.2.5 Algorithm Accounting Requirements.

1. Algorithm download events shall be stored as part of audit log functions.
2. Audit files for algorithm activities shall be access restricted to authenticated users or automated audit data recipients.

4.2.6.3 Security Policy Management Functions.

A security policy is a set of rules which are tested for compliance and then enforced by a given security function. In aggregate, the security policy is the overall set of security rules that are observed for proper secure operation of a JTR. Elements exist both inside the CS/S and within the INFOSEC Boundary.

4.2.6.3.1 Security Policy Management Discussion.

The security policy originates from multiple sources, e.g., waveform software profiles, administrator/security officer entry, mission load instructions, embedded in boot functions. Ultimately an allocation of security policy enforcement based on criticality will be required, since all aspects of security enforcement cannot be centralized. For example, a communicator access control function could not be reasonably executed within a cryptographic boundary if cryptographically based authentication is not used. Elements of the policy execution are relegated to lower levels of security implementation including individual functions for items such as access controllers, guards, and monitors.

Security policy internal to the CS/S is composed of multiple elements:

- Alarm conditions and responses
- Permitted communications with objects external to the CS/S
- Permitted information paths internal to the CS/S
- Permitted interconnection of algorithms to waveforms
- Permitted levels of classification and key use

4.2.6.3.2 Security Policy Management Requirements.

1. Policy rules for cryptographic bypass shall be protected either by: storage within the CS/S or using access control with integrity checks.
2. The CS/S shall accept waveform security policy elements from software profiles at waveform instantiation.
3. The CS/S shall examine requested cryptographic keys for consistency of tag information (e.g., classification).
4. The CS/S shall accept user security policy elements from authenticated HMI communications.
5. The CS/S shall provide security policy status upon authenticated request.
6. The CS/S internal security policy tables shall not be accessible to JTR objects external to the CS/S without authentication.
7. The CS/S shall enforce the internal security policy for security critical items prior to reporting status to the OE or HMI.

4.2.6.4 Security Critical Software Discussion.

For the purposes of this discussion and the requirements below, all software within the Cryptographic Boundary is considered security critical.

4.2.6.5 Security Critical Software Requirements.

1. The CS/S shall decrypt Security Critical Software files from JTR internal storage.
2. The CS/S shall encrypt Security Critical Software files for JTR internal storage.
3. Security critical software within the CS/S shall be subject to detailed software evaluation.
4. Security Critical Software that implements a cryptographic algorithm and supporting functions for keying, instantiation, and control shall be configuration controlled after evaluation.
5. Only software that is configuration controlled shall be implemented into JTRS-compliant equipment for security critical functions.
6. The CS/S shall perform integrity checks on Security Critical Software programs and files prior to instantiation or use.

4.2.7 Cryptographic Bypass.

This section discusses the RED to BLACK and BLACK to RED bypass mechanism as it applies to the JTRS architecture. JTRS requirements are for a multi-channel radio equipment that operates in a SECRET system high configuration. The security architecture includes an embedded CS/S that provides functionality required for RED/BLACK bypass of communicator and radio control/status information.

An extended discussion of the SCA implications of the bypass function is presented in Section 3 of this document. Two types of bypass are discussed:

- Communicator information bypass
- Radio Control/Status bypass

Figure 4-6 presents the different types of bypass functions with A, B, and C in the figure representing communicator information, and D representing radio control information bypass which also includes inter process communications for waveform installation and instantiation. The reference to B, C, and D system high requirements at the bottom of the figure apply to a multi-channel/multi-communicator JTRS environment. A is a violation of system high operation if another classified channel is operating concurrently.

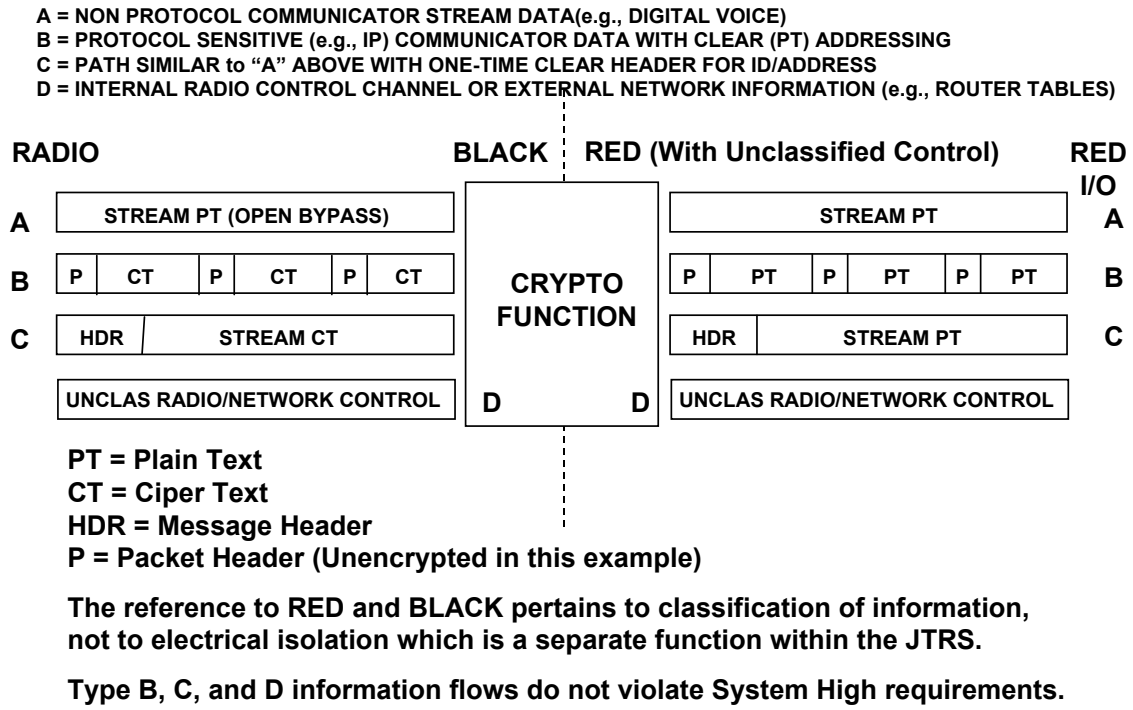


Figure 4-6. Cryptographic Bypass Types

The JTRS radio requires different types of bypass. Bypasses are required for initial start-up such that the *DomainManager* can orchestrate and complete a power up sequence. After this power up sequence, the *DomainManager* will instantiate, upon request, a waveform that an operator has requested. This instantiation requires additional communication across the RED/BLACK boundary. After the waveform instantiation, various waveform traffic control messages will be required to proceed between the RED/BLACK processors. Some of these messages are considered time critical to the waveform. In these cases, the implementation considers the latency of the bypass control process and attempt to find mechanisms that minimize the amount of control/status information to be examined. SCA APIs have been proposed that concatenate time critical control or timing information with the applicable communicator data.

4.2.7.1 Communicator Information Bypass.

The communicator information bypass is typically of very restricted size and fixed format. The information bypassed can be considered to be in-band with the user information itself.

4.2.7.1.1 Communicator Information Bypass Discussion.

The communicator information bypass functions are unique to specific waveforms to be implemented in the JTR. Each waveform is programmed and profiled in such a manner that the required bypass functions are defined as to bit rate and content to permit monitoring of the correctness of the bypass function. Bypass characteristics that are monitored include length of bypass message, content of message, and format of message to be bypassed. Figure 4-6 provides examples of the communicator bypass formats.

4.2.7.1.2 Communicator Information Bypass Requirements.

1. The *ApplicationFactory* shall provide the application specific requirement of the bypass security policy to the CS/S control interface.
2. The CS/S shall control the bypass function according to the security policy.
3. The CS/S shall provide a mechanism to terminate communication on the cryptographic channel if the bypass policy is violated.

4.2.7.2 Radio Control/Status Bypass.

For proper radio operation, internal radio control type messages are required to pass through the CS/S from the RED processing (e.g., a RED GPP) to the BLACK processing area (e.g., a BLACK GPP). These pass-through messages require the CS/S to provide a bypass mode. This bypass mode allows the message to proceed from the RED GPP to the BLACK GPP unaltered. This message will be examined by a bypass mechanism. The requirements for the bypass mechanism are provided at the end of this section.

The bypass control mechanism is required to check content and length in the bypassed information. If the lower protocol layers are removed, the bypass control is only required to validate the bypass message itself. If the lower layers of the transport mechanism are present, the algorithm has to check the protocol bits that are included with that layer. As a result, the lower the algorithm is in the stack, the harder the bypass control mechanism is to implement or the bypass policy to be loaded becomes implementation dependent. Therefore, the bypass mechanism can architecturally exist at the application layer. All transport encapsulations are

thus removed. Regardless of bypass placement, the bypass mechanism will address the requirements below.

4.2.7.2.1 Radio Control/Status Bypass Requirements.

1. Only the crypto subsystem shall provide the control/status bypass function between the RED and BLACK side of the JTR.
2. The control/status bypass parameters shall be associated with the application requesting the bypass.
3. The control/status bypass mechanism(s) shall accept bypass policy either on a per-application basis or create a separate bypass mechanism per policy.
4. The system control control/status bypass mechanism shall be distinct from waveform control/status bypass mechanism.
5. The control/status bypass mechanism shall be non-bypassable.
6. The control/status bypass mechanism shall be always invoked.
7. The control/status bypass mechanism shall be provided tamper protection.
8. Covert channel analysis shall be performed on the bypass channel.
9. The control/status bypass mechanism shall output all violations of bypass security policy to the Audit Function.
10. The bypass mechanism shall block messages that violate the bypass security policy.
11. Bypass messages that violate the bypass policy shall not be transmitted.
12. If policy violation exceeds a policy-determined threshold, the bypass channel shall be closed, resulting in termination of the operation of the affected channel.
13. When required for a specific application, the control/status bypass mechanism algorithm shall check for valid connections between objects.
14. When required for a specific application, the control/status bypass mechanism algorithm shall check for valid format of bypass messages.
15. When required for a specific application, the control/status bypass mechanism algorithm shall check for valid length of bypass message.
16. When required for a specific application, the control/status bypass mechanism algorithm shall check for valid frequency of bypass messages for a given bypass mechanism.
17. The bypass algorithm requirements shall be contained in the bypass policy that is downloaded to the bypass mechanism.
18. The bypass policy shall be protected either by storage in the cryptographic boundary or by using access controls and integrity checks.
19. The format of the policy that is downloaded shall be eXtensible Mark-up Language (XML).

4.2.8 Cryptographic Control and Status Functions.

This section is not to be confused with the radio internal control bypass functions of the CS/S discussed in the previous section. The cryptographic control and status functions in this section provide the means for the CS/S to communicate with the rest of the radio functions for channel establishment and operation.

4.2.8.1 Cryptographic Control and Status Discussion.

Although the cryptographic function set is highly isolated from the rest of the radio functions, the CS/S will communicate with the RED and BLACK sides via APIs or adapters for control and status information. Commands for the CS/S can be generated via the HMI, application program, systems program, and/or core framework. The CS/S security critical commands are executed internal to the CS/S. Status reporting and CS/S internal holdings (e.g., key holdings, algorithm holdings) are restricted to administrators or internal radio entities with need to know authentication. The security policy determines which control and status information originating from the CS/S is privileged.

4.2.8.2 Cryptographic Control and Status Requirements.

1. The CS/S shall accept control messages only from proper sources external to the CS/S.
2. The CS/S shall accept requests only from the proper sources external to the CS/S.
3. The CS/S shall accept files from the OE that define the waveform interconnections as instantiated by the CF
4. The CS/S shall internally store files that define the waveform interconnections as instantiated by the CF.
5. When required for a specific application, the CS/S will perform health checks during run time operation of a waveform that verifies the accuracy of the instantiated waveform interconnections.
6. The CS/S shall provide status messages and alarms to the Audit function.
7. The CS/S shall be a Consumer for alarms generated within the JTR.
8. The CS/S shall maintain a security policy to determine responses to alarms generated internal to the CS/S.
9. The CS/S shall maintain a security policy to determine responses to alarms generated external to the CS/S.

4.3 REQUIREMENTS WITHIN THE INFOSEC BOUNDARY.

As previously stated, the INFOSEC Boundary constitutes security functions that are outside of the Cryptographic Boundary.

4.3.1 Background Information.

The JTRS Security Supplement calls for the use of COMPUSEC requirements and protection mechanisms to govern the implementation of security within the INFOSEC boundary as defined in figure 4-3. There are two basic ways in which COMPUSEC functions can be implemented:

- Develop independent monitor and guard functions to assess correct operation of software code and to control access.
- Utilize combinations of hardware and OE software code that executes on the Red or Black side and is capable of the required level of assurance.

4.3.1.1 INFOSEC Boundary Discussion.

The required level of assurance varies with the use of the radio (e.g., system high, concurrent U.S. and NATO processing) and application (types of waveforms that will operate concurrently). The guard and monitor functions examine data or processes respectively. As shown in figure 4-7, the guard is an in-band function through which target data passes for examination of some form. The monitor examines results of another process to see if expected outputs are achieved.

An access control mechanism is an example of a guard. A watchdog timer is an example of a monitor. In both cases, it is important that the guard or monitor is functionally independent of the process being examined, and that the memory, which stores policy rules, is protected.

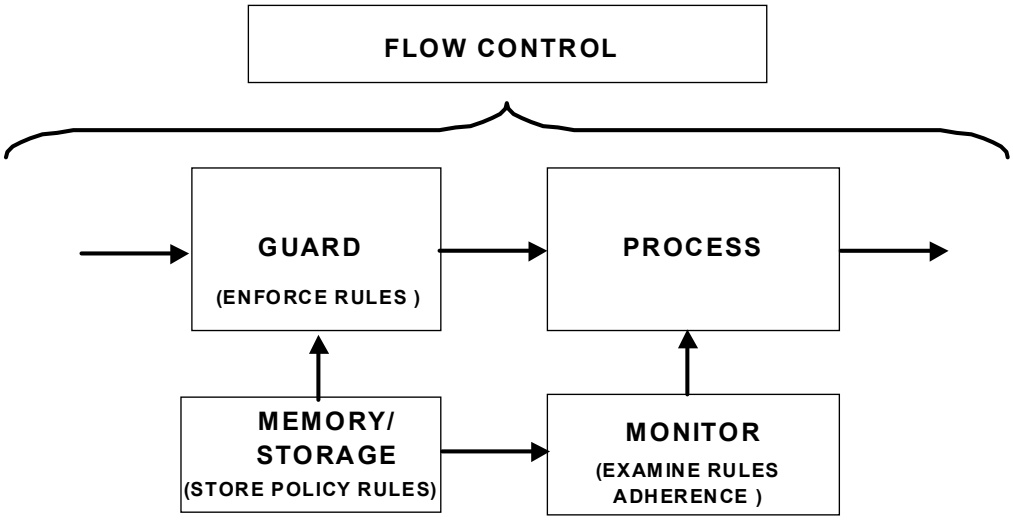


Figure 4-7. Security Guard and Monitor Supplements

The following is a discussion of COMPUSEC protection mechanisms using the OE security. The JTRS software is composed of three distinct layers. The three software layers from a security perspective are:

- Operating System
- Middleware (in the JTRS case, an ORB)
- Application Software

The security of software elements will be addressed under a security policy that governs the interaction of the software layers, the transfer of information and access control of the stored data. Security policy will have a description that is both understandable to the software entities and an enforcement mechanism that the policy works with.

In general, the attributes related to security policy enforcement mechanisms are:

- Non-bypassable
- Always invoked
- Prohibits unauthorized modification
- Evaluatable

Each layer of software (i.e., OS, ORB, and applications) provides security policy enforcement appropriate at its own layer. The operating system creates separate process spaces, provides application support functionality (e.g., timing services), and provides secure transfer of control between processes. The middleware security policy enforcement provides for secure inter-object message flow. The application security policy enforcement provides application specific security functions such as firewalls or network application flow control.

Each layer is responsible for enabling the next higher layer's ability to provide its security services. The middleware relies on the kernel to provide process separation between objects and secure transfer of control between processes. The applications rely on the middleware to provide secure message passing between objects within the application. Each layer ensures that the next higher-level can enforce its own specific security policy. The amount of assurance that is attributed to each layer depends on numerous factors including kernel implementation, software size, and specific elements of the product involved (e.g., MMU capability in an OS). As a result, any assurance measures cannot be absolutely determined.

4.3.1.2 INFOSEC Boundary Requirements.

The general requirements placed upon the INFOSEC Boundary are stated in section 4.1.2.

10.

4.3.2 Process Separation.

4.3.2.1 Process Separation Discussion.

Implicit with Process Separation is Data Isolation. The argument revolves around the fact that processes that execute out of a common process space have access to each process data space. If different data spaces need to be isolated, the processes that are associated with each data space also need to be separated.

The process separation mechanism may exist solely within JTR hardware. An example of a process separation implementation is a MMU in association with an appropriate operating system. A hardware MMU will be required to operate in a MILS environment.

4.3.2.2 Process Separation Requirements.

A JTRS implementation shall provide process separation:

1. When processing classified/sensitive data
2. When performing security critical functions
3. Between OS, CF, and application(s)
4. The process separation mechanism shall provide controlled methods of transfer from one process to another.

5. The process separation shall be implemented by a combination of hardware and/or software.
6. The software portion of the process separation implementation shall exist within the operating system.
7. The process separation mechanism 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).
8. The process separation mechanism shall isolate the resources to be protected so that they are subject to the access control and auditing requirements.

4.3.3 Discretionary Access Control.

4.3.3.1 Discretionary Access Control Discussion.

User access to files and instantiated applications access to data will be restricted.

4.3.3.2 Requirements.

1. An entity shall define and control access between named users and named objects.
2. The enforcement mechanism shall allow users to specify and control sharing of those objects by named individuals.
3. The enforcement mechanism shall provide controls to limit propagation of access rights.
4. The access control mechanism shall, either by explicit user action or by default, provide that objects are protected from unauthorized access.
5. Access controls shall be capable of including or excluding access to a single user.
6. Access permission to an object by users not already possessing access permission shall only be assigned by authorized users.
7. The access control mechanism shall exist in a domain separate from user applications that protects it from external interference or tampering (e.g., by modification of its code or data structures). Resources controlled by the entity may be a defined subset of the subjects and objects in the processing system.
8. The access control mechanism shall isolate the resources to be protected so that they are subject to the access control and auditing requirements.

4.3.4 Identification and Authentication.

4.3.4.1 Identification and Authentication Discussion.

Various levels of users have access to the JTRS radio. These users are described in Section 2.2.2 Definition of User Entities.

4.3.4.2 Requirements.

1. An entity shall require users to identify themselves to it before performing any other actions that the entity is expected to mediate.

2. Furthermore, the entity shall use a protected mechanism (e.g., passwords) to authenticate the user's identity.
3. The entity shall protect authentication data so that any unauthorized user cannot access it.
4. The entity shall be able to enforce individual accountability by providing the capability to uniquely identify each individual system user.
5. The entity shall provide the capability of associating this identity with all auditable actions taken by that individual.

4.3.5 Object Reuse.

4.3.5.1 Object Reuse Discussion.

The intent of these requirements is to stop RED data from being accessed from a newly instantiated application that happens to be using previously allocated memory space for an application of a different classification or a different type definition.

4.3.5.2 Object Reuse Requirements.

1. 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 entity's pool of unused storage objects.
2. The JTR shall clear all memory used to store classified traffic prior to instantiating lower classifications of system high.
3. The JTR shall clear all memory used to store classified traffic prior to instantiating channels of different system high type classification.

4.3.6 System Integrity.

4.3.6.1 System Integrity Discussion.

A user has the ability to take the JTRS radio off-line and initiate self-test to determine full functionality of equipment.

4.3.6.2 System Integrity Requirements.

1. Hardware and/or software test requirements are not part of this document. The requirements for hardware and/or software tests will be established by individual procurements.

4.3.7 Core Systems Applications.

Systems Applications-Core (SAC) are those that are provided with the radio as a resident function to support secure user applications. The core system applications that impact the security architecture include:

- Cryptographic Services
- Administrator, Operator, Security Operator, and Maintainer HMI security policy enforcement mechanisms
- Installer (software download) Application

- Security Policy Enforcement (application instantiation, inter-object communications) Audit

4.3.7.1 Security API.

4.3.7.1.1 Security API Discussion.

This section describes the CORBA-capable software object residing on a CORBA capable processor that interfaces with the cryptographic entity that instantiates the security mechanism. The actual mechanisms within the cryptographic boundary were discussed in detail in Section 4.2. These "cryptographic services" as expressed in the APIs, may each be unique, or may incorporate various combinations of these services.

If a non-CORBA-capable CS/S is used in the JTRS radio, an adapter is required to make the transition from a CORBA-compliant interface to a non-CORBA-compliant interface. In other words, adapters have a CORBA interface and a non-CORBA interface. Additionally, adapters are unique per CS/S implementation since the non-CORBA interface contains vendor specific communication between the CS/S and the adapter.

The current API methodology requires an instantiation of an adapter for non-real time radio control. As channels are instantiated, additional adapters will be instantiated for each channel. These channel-specific adapters are required because of waveform-specific content. The CORBA side of the adapter uses the Security APIs as the method to request cryptographic services that the CS/S provides to the JTRS radio. Security APIs are included in Attachment 1.

The channel-specific adapters have been implemented to streamline traffic data flow. As a consequence of the real-time data flow concerns and the adopted API methodology, the adapters that are used to interface with the CS/S have specialized characteristics. These characteristics accommodate a standard Security Interface without introducing a Security API into the non-security API framework, a method of implementing the layer that performs the security functions is defined.

One solution would be to create Red and Black adapters for the layer that abstracts the RED/BLACK boundary. Each adapter presents the API that is defined for that layer.

Figure 4-8 shows an example of an application where the encrypt/decrypt is performed between the Logical Link Control (LLC) and Media Access Control (MAC) layers. This implementation example interfaces directly to the Security device. Keep in mind, that the MAC API implementation is used as an example. This architecture applies between any two waveform components.

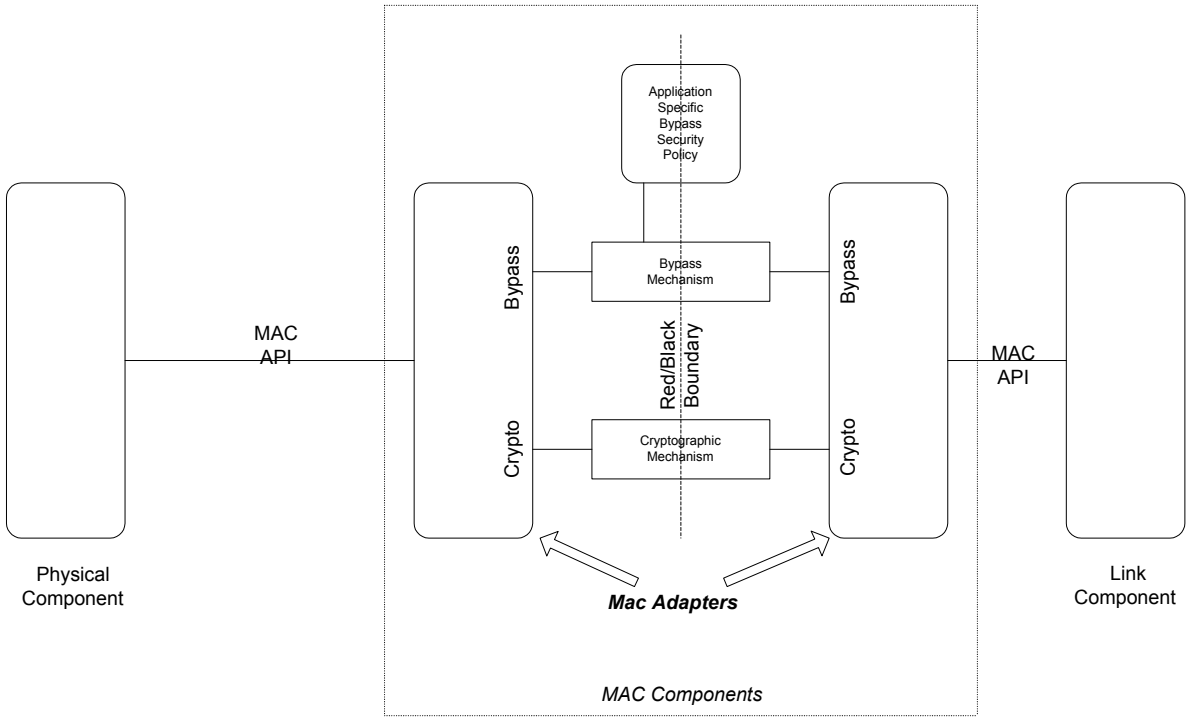


Figure 4-8. MAC Adapters w/o Security API

The Red and Black adapters communicate with the cryptographic subsystem through a vendor unique interface. These adapters identify to the cryptographic subsystem the channel identity, the data to be encrypted/decrypted and the information to bypass the encryption/decryption function. The bypass policy for the waveform application is used by the bypass enforcement mechanism to determine whether the information designated for bypass should be allowed through.

The adapters are created when the waveform application is created. The channel identity of the adapter is established when the adapter is configured.

Figure 4-8 does not have a defined security API. This requires the adapter to interface directly to the specific security device, which may affect portability of applications. The adapters are specific to the Crypto implementation. This will require adapters to be written for each device for each specific waveform.

A second solution would be to embed a security API within the Adapters. Figure 4-9 illustrates a Red-side security adapter with two different components. As an example a MAC Adapter and the Security API**.** The security API is supplied with the crypto device. The waveform implementer provides the MAC adapter that interfaces to the security API. The Security API provides the abstraction of the details of the specific Crypto device, therefore only one MAC adapter is required for an application, regardless of the type of Crypto device being used.

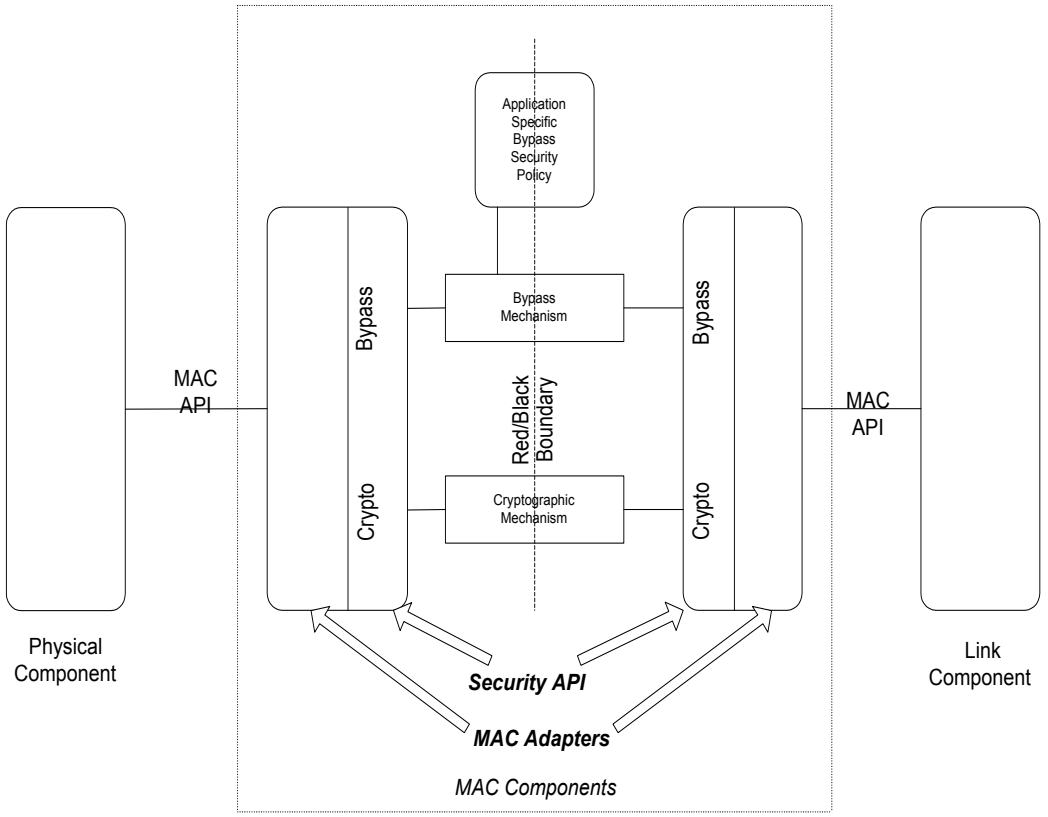


Figure 4-9. MAC Adapters Using Security APIs

The entirety of the "cryptographic service" can logically be represented as composed of service groups. The Unified Modeling Language (UML) package diagram figure 4-10 depicts the JTRS Security Service and its service groups as packages. Each of these groups represents a functional area of security that directly or indirectly supports secure JTR operation. Each functional group contains one or more related services. The service groups also provide naming scope for services within different groups that are related. An example of this is a security management service. Several of the service groups contain a management service. The general behavior of this service is the same across certain groups. What differentiates the specific behavior is the type of element being managed, which is identified by the service group (e.g. Key).

Figure 4-10. JTRS Security Service Groups

The individual primitives that may flow between the Service User and Service Provider define each service within a service group. The services and primitives are tabulated in Table 4-1.

Table 4-1 Cross-Reference of Services and Primitives

Service Group	Service	Primitives
Security	Management	ZEROIZE_ALL
Fill	Port	FILL_PORT_CONFIGURE, FILL_PORT_ENABLE, FILL_PORT_DISABLE, FILL_PORT_LOAD
	Port User	FILL_ PORT_SIGNAL_ASSIGN_ID, FILL_ PORT_SIGNAL_LOAD, FILL_ PORT_SIGNAL_CONNECT,
	Bus	FILL_BUS_LOAD
	Management	FILL_ZEROIZE, FILL_ZEROIZE_ALL, FILL_GET_IDS, FILL_EXPIRY
Algorithm	Management	ALG_ZEROIZE, ALG_ZEROIZE_ALL, ALG_GET_IDS, ALG_EXPIRY
Certificate	Management	CERT_ZEROIZE, CERT_ZEROIZE_ALL, CERT_GET_IDS, CERT_EXPIRY
Crypto	Control	CRYPT_CREATE_CHAN, CRYPT_DESTROY_CHAN, CRYPT_GET_CHAN_CONFIG, CRYPT_START_CHAN, CRYPT_STOP_CHAN, CRYPT_RESET_CHAN, CRYPT_RESET
	Encrypt/Decrypt	CRYPT_ENCRYPT, CRYPT_DECRYPT, CRYPT_ENCRYPT_WITH_ ID, CRYPT_DECRYPT_WITH_ ID, CRYPT_TRANSFORM_REQ, CRYPT_TRANSFORM_REQ_WITH_ID
Key	Management	KEY_ZEROIZE, KEY_ZEROIZE_ALL, KEY_GET_IDS, KEY_EXPIRY, KEY_UPDATE, KEY_GET_UPDATE_COUNT, KEY_STORE_KEY

Service Group	Service	Primitives
TRANSEC	Control	TRAN_CREATE_CHAN, TRAN_GET_CHAN_CONFIG, TRAN_DESTROY_CHAN
	Key Stream	TRAN_GEN_KEY_STREAM, TRAN_GEN_NEXT_KEY_STREAM
	Management	TRAN_ZEROIZE, TRAN_ZEROIZE_ALL, TRAN_GET_IDS, TRAN_EXPIRY, TRAN_STORE , TRAN_GET_FILL
Policy	Management	POL_ZEROIZE, POL_ZEROIZE_ALL, POL_GET_IDS, POL_EXPIRY, POL_GET_POLICY
Integrity and Authentication	Control	IA_CREATE_CONTEXT, IA_DESTROY_CONTEXT
	Digital Signatures	IA_SIGN_FILE, IA_VERIFY_FILE, IA_HASH, IA_SIGN_HASH, IA_VERIFY_HASH
Alarm	User	ALARM_SIGNAL
Time	Management	TIME_SET_TOD, TIME_GET_TOD, TIME_SET_DATE, TIME_GET_DATE
GPS	Management	GPS_ZEROIZE, GPS_ZEROIZE_ALL, GPS_GET_IDS, GPS_EXPIRY, GPS_STORE , GPS_GET_FILL

Figure 4-11 shows an SCA component which is a CF::*Device*. The device is a logical representation of a cryptographic subsystem. The device has several ports. Each port supporting a security service. For example, the Key Management Service is at one port while the Crypto Control Service is at another. Each of these ports has an identifier. When a Security Service User needs to gain access to a service, it invokes the getPort operation on the security device with the port identifier as input. The *getPort* operation returns the object reference of the service provider which can then be passed to the service user through the CF::*Port*::*connectPort* operation. The service user can then invoke the primitives that comprise the service. The

following sections briefly discuss the services. Details and supporting information for each is in Attachment 1.

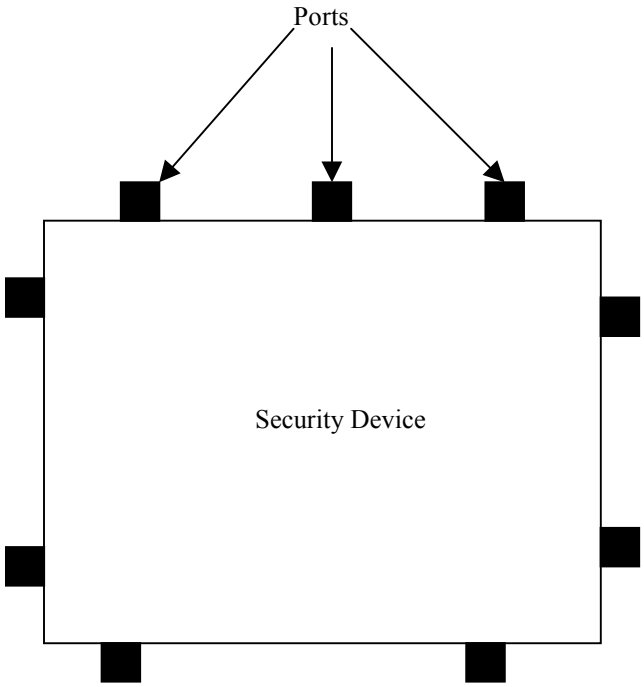


Figure 4-11. JTRS Security Device

4.3.7.1.1.1 Security.

Security at the top level has one service, a management service. As illustrated in Attachment 1, a Service User invoking the ZEROIZE_ALL primitive of the Security Management Service zeroizes all elements of fill information. It is equivalent to invoking the individual zeroize all primitives of the Algorithm, Certificate, Key, Policy and TRANSEC Management Services.

4.3.7.1.1.2 Fill.

The fill services defined in the security API consist of services to get fill information into a JTR and to manage that information using the Port and Port User services. Fill includes a Port service implemented by the security service, and a Port User service implemented by the user (e.g., the HMI software). The Bus Service allows the Service User to fill the radio from a file resident on an SCA compliant *FileSystem*.

Fill Management Service provides a set of primitives that are common across a set of management services. The Fill Management Service is inherited, specialized, and extended by other services. It is in these other services where the implementation will reside.

4.3.7.1.1.3 Algorithms.

Algorithms encompass both encryption and classified TRANSEC algorithms. Algorithms require only one service, a management service. The Crypto Control Service and TRANSEC Control Service instantiate traffic and key stream generation channels with the algorithms managed by the algorithm management service. The logical separation of the service that manages algorithms and the services that use algorithms imposes no such separation in the

implementation. The Algorithm Management Service is a specialization of the Fill Management Service with no additional primitives.

4.3.7.1.1.4 Certificate.

The Integrity and Authentication Service uses certificates for generating and verifying Digital Signatures. In addition, they may be used for key exchanges such as Firefly. Certificates require only one service, a management service. The Certificate Management Service is a specialization of the Fill Management Service with one additional primitive.

4.3.7.1.1.5 Crypto.

Cryptographic (COMSEC) functionality is encompassed in the Crypto Control and Encrypt/Decrypt services.

The Crytpo Control Service covers channel creation, destruction, starting, stopping, resetting, and registration for crypto alarm notification.

The Encrypt/Decrypt services provide for encryption and decryption of data using a given channel created by the Crypto Control Service.

4.3.7.1.1.6 Key.

Keys require only one service, a management service. Keys in this context are persistent keys which require storage and are not to be confused with session keys which are generated in a Firefly exchange for example. The Crypto Control Service instantiates traffic channels with the keys managed by the key management service. The logical separation of the service that manages keys and the services that use keys imposes no such separation in the implementation.

The Key Management Service is a specialization of the Fill Management Service with three additional primitives.

4.3.7.1.1.7 TRANSEC.

TRANSEC requires three services, a management service for managing stored TRANSEC information, a control service for creating and destroying key stream generation channels and a key stream provider service for providing the actual key stream. TRANSEC in this context is persistent information used to generate TRANSEC cover. The Key Stream provider service provides the actual key stream data to a waveform. The logical separation of the TRANSEC management service that manages TRANSEC information and the services that use TRANSEC information imposes no such separation in the implementation.

The TRANSEC Control Service instantiates and destroys classified key stream generation channels with the TRANSEC information managed by the TRANSEC management service.

The Key Stream Service provides generated classified key stream data from a channel instantiated by the TRANSEC Control Service.

The TRANSEC Management Service is a specialization of the Fill Management Service with two additional primitives.

4.3.7.1.1.8 Policy.

Policies in the context of the JTRS Security Service API are information used to control the behavior of the JTRS Security Enforcement mechanisms. The number and content of the

policies in any given JTRS platform will vary according to the platform configuration and the number and type of waveform applications loaded on it. Policies are used to parameterize crypto bypass behavior, access control to objects and files, and audit behavior. Figure 3-32 of Attachment 1 illustrates how bypass policies are used. The Control Bypass Guard enforces System and Waveform configuration and control bypass policies that are accessed from a Policy store. The Bypass policies contain information that the guard uses in its enforcement mechanism to either allow or disallow information to flow from red to black. The Header Bypass Guard is similar except that it performs its enforcement function at real time data rates and on header information that is associated with packets of data.

The Policy Management Service is a specialization of the Fill Management Service with one additional primitive.

4.3.7.1.1.9 Integrity and Authentication.

Integrity and Authentication encompasses verification of the identity of the source of information (authentication) and verification that the information has not been changed (integrity). Certificates are used to generate the Integrity and Authentication context.

4.3.7.1.1.10 Alarm.

The Security Service divides alarms into two components, an audit record and an alarm indicator. The audit record is modeled after the ITU X.736 standard. The CosLwLog *LogProducer* interface is used to log the audit record.

4.3.7.1.1.11 Time.

Some Security Service implementations require management of time. The Security Service API defines a Time Management Service for this purpose.

4.3.7.1.1.12 GPS.

The GPS Management Service is a specialization of the TRANSEC Management Service with no additional primitives.

4.3.7.1.2 Security API Requirements.

1. Access to cryptographic interfaces shall be restricted only to objects with the appropriate authorization (permissions).
2. Security APIs shall be implemented in accordance with Attachment A:Security Application Program Interface Service Definition.

   .
3. Security APIs shall be used to control/access all Cryptographic Services by other software objects within the JTR Set. This requirement is not intended to prevent the implementation of legacy cryptographic interfaces such as Crypto Ignition Key and DS-101 fill port on the cryptographic subsystem.
4. The Security APIs shall use the Cryptographic Service Building Blocks.

4.3.7.2 HMI Security Policy Enforcement.

The Human-Computer Interface (HCI) includes the standard radio control interface called a Human-Machine Interface (HMI) and the communicator channel interface (the actual

communicator external data I/O port). For the purposes of this discussion and requirements, the definition is the HMI, and not the broader HCI. In application-specific cases where the communicator I/O port is used for control, the requirements also apply to that interface. The use of the communicator I/O ports for radio control is not the preferred method of JTR control.

4.3.7.2.1 HMI Security Policy Enforcement Discussion.

The HMI security policy enforcement mechanism (guard) is much like other application-level, security policy enforcement mechanisms within the JTRS. See Figure 4-12. The HMI security policy enforcement guard relies on a protected process space to protect object references for a system high implementation. The intent of the HMI security policy enforcement guard is to restrict an Operator/Administrator/Security Operator/Maintainer access to system and communicator information, and to functionality within the JTRS. The authentication portion of the HMI security policy depends on the mission and application for its use.

The elements of HMI security policy enforcement mechanism (a guard function) include:

- Access control per "user type" (See Section 2.2.)
- Access privilege
- Remote control (1.5 km as specified in the ORD)
- Zeroize mechanism

The access control applies to both "user type" and other entities and objects (e.g., remote zeroize command) that may have the opportunity to affect radio operability if not subject to access restriction.

Unrestricted access within the JTR undermines the ability to enforce security policy within the system. Concerns raised by failure to restrict access include:

- Commonality of internal and external IP addressing causing access to internal objects
- Unauthorized modification of instantiated waveform or application
- Unauthorized access to information
- Unauthorized modification of information
- Intrusion due to remote control transport mechanism
- Adequacy of software vice hardware zeroize mechanism

In addition, the transport mechanism used to communicate with the external users is a security concern. If a common implemented transport mechanism were used for internal and external messages, it would imply that the same security policy is used for all external users of the JTRS. Some implementations may have authentication requirements placed on the host interface in addition to, or instead of, communicator interfaces authentication mechanisms.

The HMI security policy enforcement mechanism (guard) is much like other application-level, security policy enforcement mechanisms within the JTRS. See figure 4-12. The HMI security policy enforcement guard relies on a protected process space to protect object references for a system high implementation. The intent of the HMI security policy enforcement guard is to restrict an Operator/Administrator/Security Operator/Maintainer access to system and communicator information, and to functionality within the JTRS.

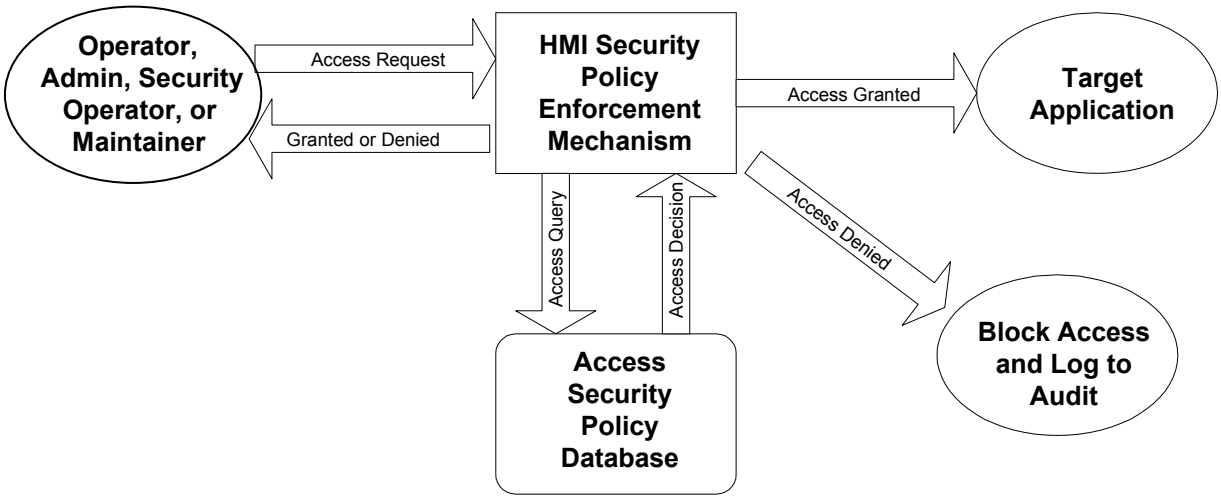


Figure 4-12. HMI Security Policy Enforcement

4.3.7.2.2 HMI Security Policy Enforcement Requirements.

1. The HMI transport utilized shall be independent of other transports that are used to communicate between internal JTR objects.
2. Internal message protocols stacks and host interface message protocol stacks shall reside in separate process space.
3. The security policy associated with the HMI security policy enforcement guard shall be tailored specifically for operator, administrator, security operator, maintainer, or communicator (as a systems application or SAP).
4. The security policy shall be loaded during the application instantiation.
5. A unique instantiation of the security policy enforcement mechanism shall be used for each host interface.
6. The associated security policy shall identify an individual's legitimate permissions within the system.
7. If high assurance host authentication is required (e.g., remote control requirement), the HMI security policy enforcement mechanism shall use the Security API for the Integrity/Authentication cryptographic service.
8.

4.3.7.3 Installer.

4.3.7.3.1 Installer Discussion.

The installer application is used to load new software into the system or update existing software. Installer is a generic name given to utility applications used for installing and uninstalling devices and components within the system.

The approach to assuring that only valid software is downloaded into a JTR involves the use of confidentiality, authentication, and integrity mechanisms:

- Confidentiality (encryption for classified applications in transport)
- Understanding where the software originated (authentication of origin via digital signature)
- Understanding whether the software has been modified since originated (integrity of data using a cryptographic hash function)
- Audit of software downloads (internal systems application using Log and log consumers)
- Storage of data internally in such a way that the data integrity can be checked at a later time (e.g., use of a cyclic redundancy check or other checkword)
- Storing of classified files in encrypted form (encryption for internal storage)

4.3.7.3.2 Installer Requirements.

1. The CF shall be checked for file integrity prior to instantiation.
2. Cryptographic Services of the JTR shall be active prior to the ability of the CF to instantiate waveforms.
3. The installer application shall provide confidentiality by using decryption, if necessary.
4. The downloaded software shall be checked to ensure data integrity has been maintained during the download process.
5. The downloaded software shall be authenticated to verify it has originated from an approved source.
6. The downloaded software shall be placed into storage (typically non-volatile BLACK storage).
7. The installer software shall replace the digital signature on downloaded files with an integrity check (e.g., CRC) after the digital signature has been verified and prior to placement on accessible storage..
8. The integrity and authentication of the download application software shall beverified using Type 1 cryptography.
9. The installer application shall provide the capability to install Security Policies files.
10. The installer shall use the Integrity and Authentication services of the JTRS Security API for the purpose of authenticating digitally signed files and checking their integrity.
11. The installer shall use the Crypto services of the JTRS Security API for the purpose of encrypting and decrypting downloaded files.

4.3.7.4 Audit.

The term audit is defined for security purposes as a mechanism that permits transfer of information concerning security relevant events from a reporting audit producer to a receiving audit consumer. Some audit events are required for immediate action (e.g., alarm or access control failure). Audit events that are stored for reporting and forensic purposes are contained in

an audit log. A means is needed to download audit logs from the JTR to administrators and security officers.

4.3.7.4.1 Audit Discussion.

The audit function provides a set of records that documents evidence to aid in tracing from original transactions forward to related records and reports, and/or backwards from records and reports to their component source transactions.

The primary security concerns related to audit are the collection of proper relevant information and the ability to act upon the information, both in real time within the radio for critical events, and on information logged and delivered to security and administrative personnel. Not all audit events are security related since audit files are also applicable to maintenance and configuration management actions, among others. Any audit collection capability is limited by the processing and storage capabilities of the target JTR, as well as the ability to collect, transport and store the information for analysis. Types of events suitable for security audit operations are:

- Cryptographic software and key loads
- Exceptions to bypass and access control policy
- Cryptographic alarms

The ability to set priorities for audit events and actions that might be taken is necessary. It is noted that the audit function is not part of any policy enforcement function. The audit feeds applications that can act on information delivered. The implementation of Audit is dependent upon the operational environment and the equipment being used for that mission.

4.3.7.4.2 Audit Requirements.

1. The audit function shall operate in conjunction with alarm and exception handling within the radio.
2. The audit information shall be passed via the Log to log consumers that have access restrictions for the types of events that each Consumer can collect.
3. An administrator or security officer shall be able to:
   a. Select the types of events that the audit function will collect and report
   b. Determine user access to the various event types.
4. The audit mechanism shall:
   a. Permit selection/entry of audit events by Administrators/Security Officers
   b. Perform both event and rate driven audit functions
   c. The audit mechanism shall collect information on violations of the CS/S bypass security policy.
   d. The audit mechanism shall collect information on violations of access control security policy.
   e. Provide a delivery mechanism for logged audit records to audit personnel (infrastructure requirement)

5. The audit application shall provide for a hierarchy of audit types to support alarm and exception handling.
6. Users and protected applications shall be able to set the hierarchy of events and thresholds on auditable events to take appropriate responsive action (such as system shutdown) if an event exceeds its threshold.

4.3.7.5 Cross-Banding.

4.3.7.5.1 Cross-Banding Discussion.

Cross-banding is an example of an implementation dependent systems application. It provides the ability to receive information on one waveform and retransmit the information on another waveform. The cross-banding function may be at various entry points in the waveform (e.g. BLACK modem or RED-side processor).

The primary security concerns related to cross-banding are:

- Only correct applications can be cross-banded (i.e., the cross-banding is consistent with the cross-banding security policy)
- Only objects or applications at the same security level can cross-banded
- Ability to recognize the waveform Software Profile parameters that determine whether cross-banding is permitted

4.3.7.5.2 Cross-Banding Requirements.

1. The proper interface points within the objects of the two waveforms that are cross-banded shall be defined either within the cross-banding systems application or within the two waveform applications profiles.
2. The cross-banding application shall ensure that the cross-banding is consistent with the security policy of the affected waveforms and applications.
3. Prior to instantiation, the cross-banding application shall request a validation of security levels.
4. Cross-banding shall only occur between applications of the same security levels.

4.4 EQUIPMENT LEVEL BOUNDARY FUNCTIONS.

The equipment level boundary functions involve mechanisms that encompass the overall protection of the radio, both physically and electronically. The major equipment level protections are:

- TEMPEST protection
- Tamper protection
- Electrical protection

4.4.1 Tempest.

TEMPEST concerns and solutions are similar to those for legacy equipment that conduct or emanate electrical signals. However, there are additional test scenarios considered for TEMPEST protection. For example, in multi-channel equipment, one radio channel may still be operating while another channel is having key loaded. Similarly, concurrent operation of multiple channels may introduce new TEMPEST concerns. TEMPEST plans will consider a full set of operating scenarios that may be encountered by a fielded JTR under mission conditions.

4.4.1.1 TEMPEST Requirements.

The requirements for TEMPEST will be established by individual procurements and can be found in the Unified INFOSEC Criteria (UIC).

5.

4.4.2 Tamper.

Legacy Tamper protection mechanisms (e.g., unique fasteners, counter sinking of surfaces) will require revision to maintain proper protection under requirements to use commercial hardware and packaging. Maintenance operations may require that equipment continue to function while the enclosure is open due to availability requirements for multi-channel radios. Availability requirements may limit the universality (protection of the full equipment enclosure) of tamper measures.

4.4.2.1 Tamper Requirement.

The requirements for TAMPER will be established by individual procurements and can be found in the Unified INFOSEC Criteria (UIC).

6.

4.4.3 Electrical Protection.

The JTRS radio needs to incorporate specific circuit designs that minimize unintentional compromising emanations.

4.4.3.1 Electrical Protection Requirement.

The guidelines/requirements will be established by the individual procurements and can be found in the Unified INFOSEC Criteria (UIC).

4.5 JTRS SECURITY POLICY.

The security policy and enforcement can be described in three elements:

- The actual security policy (requirements) requiring enforcement
- The embodiment of security policy in a form useful to the equipment (often in table, database, or embedded in hardware)
- The actual enforcement mechanism

4.5.1 Security Policy Discussion.

The existence of the Security Policy within the JTRS radio has two distinct entities. There is the chosen mechanism as one piece and the parameters that mechanism uses as the other. The security policy mechanism for the JTRS exists as a HW element or as executable SW. These mechanisms are always enabled and their parameters are provided in a file whose protocol is in XML.

The JTR architecture uses distributed security policy enforcement. Table 4-2 shows security policy elements; where the security policy elements are embodied within the JTR architecture; and which elements of the JTR architecture can be used to enforce that element of the security policy. The first column lists the different types of policies contained within the JTR. The second column lists methods that can be used to embody that security policy in the JTR. The third column lists possible security enforcement mechanisms that can be used to enforce the security policy within the JTR.

Table 4-2. Security Policy and Enforcement

Security Policy Element	Policy Embodiment	Enforcement Mechanism(s)
Cryptographic Boundary:		
KG configuration/Algorithm usage	Application specific crypto adapter	Crypto Adapter
Key usage policy	Software Profile/Application Configuration file	Security Manager
Key/Algorithm download policy	Fixed—hardware (HW) or SW implementation in Security Manager	Security Manager
Key distribution, Internal	Software Profile limited by fixed HW or SW implementation in Security Manager	Security Manager
System bypass policy (radio control/status bypass elements)	System bypass policy file	Bypass Guard
System bypass shutoff requirements	System bypass policy file	Bypass Guard
Application bypass policy	Application bypass policy file (application dependent)	Bypass Guard
Application bypass shutoff requirements	Application bypass policy file	Bypass Guard
CS/S alarm response security policy	Alarm policy file	*Domain Manager*/Security Manager
Termination and suspension policy	Termination and suspension policy	*Domain Manager*/Security Manager
INFOSEC Boundary:		
Process separation	Fixed: MMU	Operating System/MMU
Inter-Process Communication	Software Component	Operating System-IPC
File permissions	P/O file structure	CF *FileSystem*/Manager
Object instantiation	Software Profile	*DomainManager*/*ApplicationFactory*
CORBA inter-object communication	Software Profile	*DomainManager*/*ApplicationFactory*
Application Classification	Software Profile	*DomainManager*/*ApplicationFactory*
Device Classification	Software Profile	*DomainManager*/*ApplicationFactory*
Software version	Software Profile	Installer Application
Configuration	Software Profile	Installer Application
Audit access policy	Audit access policy file	Audit Logger
Audit recording policy	Audit recording policy file	Audit Application
Software download policy	Fixed policy—embedded in Installer Application	Installer Application
Admin access policy	Admin access policy file	HMI Guard
Operator access policy	Operator access policy file	HMI Guard
Security operator access policy	Security operator access policy file	HMI Guard
Maintainer access policy	Maintainer access policy file	HMI Guard
Communicator access policy (application dependent)	Communicator access policy file (application dependent)	HMI Guard
Integrity/Authentication Policy	Fixed policy or PKI-type policy; password policy embedded in application	Application component enforced (PKI-type components in Cryptographic boundary)
Network Security Policy	Network security policy file (application specific)	Network application component (this may be a crossbanding application, an IP router, network firewall, etc.)
Equipment Level Security Boundary:		
Tamper/TEMPEST	Built into chassis/hardware	System/hardware
Power	Built into hardware	System/hardware

4.5.2 Security Policy Requirements.

1. XML files shall be derived from the relevant IDL. (IDL provides the interface requirements to the enforcement mechanism. The XML will provide the required parameter inputs for the interface.)
2. NSA shall digitally sign all Security Policy XML files.
3. An XML parser shall be used to derive the required information for all policies that are used within the JTRS radio.
4. The XML parser shall calculate and append to the parsed policy an integrity checkword. (e.g. 32-bit CRC, parity.)
5. Policy file shall be integrity checked at instantiation.

4.6 SCA BEHAVIOR IMPACTS.

4.6.1 Key and Waveform Selection.

This behavior change will stop the user from completing a channel instantiation request at the wrong classification level for system high applications. See Paragraph 4.2.6.1.2.3 for key usage requirements.

4.6.1.1 Key Selection Requirements

1. Prior to the operation of a cryptographic channel, the *ApplicationFactory* shall send any key requests to the CS/S. A key request may be "empty" if session keys are to be generated by the waveform.
2. The CS/S shall verify that the selected keys for encryption/decryption of user data traffic (i.e. red to black, black to red) on a channel are at the same classification level as the other keys that are currently being used for encryption/decryption of user data traffic for system high applications.
3. The CS/S shall verify that the selected keys for encryption/decryption of information other than user/data traffic (e.g. TRANSEC, DAMA Orderwire, file encryption/decryption) are at the same classification level or below that of keys that are currently being used for encryption/decryption of user data traffic.
4. For system high applications, the first user traffic key instantiated into the CS/S shall be used to determine the classification and use of the radio.
5. The CS/S shall either acknowledge and process a key selection request or deny a key selection request with negative acknowledgement to the user.

4.6.2 Software Installation.

As part of the install behavior, the CS/S will validate Digital Signatures prior to non-volatile storage. Need a method that insures bogus SW does not get loaded into the JTRS radio.

4.6.2.1 Software Installation Requirements.

1. All software files to be installed in the JTR shall have digital signatures attached.
2. The CS/S shall validate signitures prior to storage.

4.6.3 Power-up Sequence.

During a Power-up sequence, *DomainManager* expects to have free access across the red/black boundary of the CS/S. CS/S bypass will be validated prior to allowing bypass traffic.

4.6.3.1 Power-up Sequence Requirements.

1. A test shall be executed that validates the correct operation of the red and black portions of the bypass individually prior to a cooperative red/black bypass test.
2. A cooperative test shall execute during boot across the CS/S that tests the bypss health from the red to black.

4.6.4 Instantiation.

All waveform SW needs to validate its integrity prior to instantiation.

4.6.4.1 Instantiation Requirement.

1. All waveform SW shall have an integrity check prior to instantiation.

4.6.5 Keep-Alive Ping.

To validate that the JTRS radio is functioning during idle times, a keep-alive ping between the Red GPP and the CS/S. A similar ping needs to exist between the Black GPP and the CS/S.

4.6.5.1 Keep-Alive Ping Requirements.

1. The Keep-Alive Ping message shall exists as part of the Security APIs.
2. This message shall be sent periodically between the Red GPP and the CS/S.
3. This message shall be sent periodically between the Black GPP and the CS/S.
4. Failure to receive this message from either the Red or Black GPPs shall cause to the Cryptographic to enter into an alarm state.

4.6.6 Operating System.

The Core Framework is started by an OS action. Some OSs have login capability to assist a developer during integration. These logins should be disabled in a JTRS radio.

4.6.6.1 Operating System Requirements.

1. Operating Systems with login capability shall have logins disabled.
2. The OS invocation method shall be a NSA digitally signed script or an equivalent assured method.

The script or assured method shall be validated prior to instantiation.

5 GLOSSARY AND DEFINITIONS.

Administrator – Entity that configures the radio via software downloads, waveform instantiations, and establishment of privileges. This entity manages general audits.

BLACK – Descriptive of unclassified information. The information can either be unclassified and/or encrypted.

Classified – Classified refers to information or equipment that has been determined to require protection against unauthorized disclosure and is marked either confidential, secret, or top secret to indicate its classified status.

Communicator – Entity that uses the radio communications capabilities of the JTR.

Communicator Application - Communicator Channel Specific Software to Include Modem, Coding, Interleaving, Synchronization, Data Formatting, Voice Processing, I/O Port Definition

Communicator Port – A local communicator connection for the use of JTR over-the-air communications services.

Crypto Software - Chip/Module/Subsystem Boot, Algorithms, Key Handling, Internal Channel Set-up, Alarms, Control, Internal Security Policy

Cryptographic Channel – A communications path within the cryptographic boundary that uses an algorithm and a key. The channel can be encrypt and decrypt, (full duplex, simplex, half duplex broadcast) or TRANSEC. The encrypt and decrypt functions can be for communicator data, authentication, file encryption, key encryption, or other cryptographic uses.

Domain – The unique context (e.g., access control parameters) in which a program is operating; in effect, the set of objects that a subject has the ability to access. See process and subject.

Download (Install) - Separate download of crypto software, application software, systems utilities, system security applications

Entity – A non-specific subject whose role must be deduced from the context out of which it is taken. Use of this word should be avoided whenever possible.

Erase, Erasure, or Deletion – Terms used for clearing algorithms, BLACK keys, programs, and/or user data memory or storage.

External Interface - External interference refers to untrusted objects outside a security enforcement mechanism being capable of altering the enforcement of its given security policy. Security enforcement mechanisms must be protected from external interference. See tampering.

Guard (Security Guard) – An in-band security mechanism. Information flows through the mechanism for testing and security enforcement. The guard has the capability to pass (accept) or not pass (reject) information flowing through it based on information characteristics such as rate, size, address, or content according to the security policy for that guard. Corrective actions are built into a guard function.

HMI Input - Specific selections/settings for a given network/connection on a given radio date.

Instantiate - Combine elements above to form full communicator application

Key Development – The use of a mathematical process to produce the same key at two ends of a communications connection. Key Development is typically performed using asymmetric cryptographic methods such as done in PKI or FIREFLY.

Key Generation – The production of a key from a random bit source.

Maintainer – Entity that has open box access for repair and maintenance of equipment. Has access to maintenance logs.

Monitor (Security Monitor) – An out-of-band security mechanism. Information does not flow through a monitor. The monitor has the capability to observe the correct or incorrect operation of another function, to provide indications of failure, and to respond to failures (e.g., command shutdown of channel). Corrective actions are commanded by the monitor function; it does not execute the corrective action.

Necessary (Necessity of Function) – The software code performs only the required function.

Operator – Entity that configures the radio for specific missions by entering frequency and other mission specific parameters.

Process – A program in execution. See domain and subject.

Radio Channel – A radio frequency (RF) connection to the JTR for transmission and reception.

RED – Descriptive of classified information.

Resource - A software component that implements the SCA defined Resource base application interface. The Resource interface provides an API for interaction between a Resource and the Core Framework and for interconnection of Resources. All visible SCA-conformant components of a user application must implement the Resource interface. Note that a logical Device is derived from Resource and must also implement the Resource interface.

Run - Incorporate HMI and externally entered network parameters for specific network, frequency, key, and date, TOD

Security Critical – Applies to functions, storage, applications, hardware, or software that perform functions that provide or support the logical RED/BLACK separation in a system (i.e., Electrical, Encrypt/Decrypt, Bypass).

Security Critical Function - Any logic (hardware or software) which performs, controls, alarms, checks, or influences the cryptologic, or any of the following functions or operations, is defined to be a Critical Function: Alarm and Alarm Check, Crypto Control, Randomization, Initialization (COMSEC as well as COMPUSEC), Key Generation, Key Handling and Storage, Key Updating, Zeroization, Access Control (COMSEC as well as COMPUSEC), Authentication (COMSEC as well as COMPUSEC), Bypass (COMSEC as well as COMPUSEC), Crypto Sync, Lock and Unlock, Split Variables, Secondary Keys, Spoof Protection, Transec, Information Flow and Management, Data Isolation and Separation, any other function that enforces or manages a system or component level security policy, and any other function whose transient or induced failure may cause security to be compromised.

Security Officer – Entity that enters keys, algorithms, security policies, and other security relevant information into the radio. Manages security audits.

Security Policy – A set of protection rules which are enforced by a given function or functions. In aggregate, a security policy is the overall set of security rules that are observed for proper secure operation of a JTR.

Sensitive Information - Information, the loss, misuse, or unauthorized access to or modification of, which could adversely affect the national interest or the conduct of federal programs, or the privacy to which individuals are entitled under 5 U.S.C. Section 552a (the Privacy Act), but that has not been specifically authorized under criteria established by an Executive Order or an Act of Congress to be kept classified in the interest of national defense or foreign policy. (Systems that are not national security systems, but contain sensitive information, are to be protected in accordance with the requirements of the Computer Security Act of 1987 (P.L. 100-235).)

Strap/Mode Settings - Specific Operating Mode Settings for Software That Can Be Run in Multiple Modes (e.g., Interleaver, Code Constraint Length, KG Motion, Sync Modes)

Subject - An active entity, generally in the form of a person, process, or device, that causes information to flow among objects or changes the system state. Technically, a process/domain pair.

Sufficient (Sufficiency of Function) – The software code accurately performs the required function.

Tampering – Unauthorized modification altering the proper functioning of INFOSEC equipment.

Utility - A Systems Application That Can Serve Multiple Waveforms (e.g., Download Software, Crossband, HMI, Remote Control, Access Control, Audit)

Zeroize – Term indicates erasure of a RED key, and is reserved for that function only.

The full content of the waveform is “built” as the waveform package is assembled for use. The elements from which a run time waveform is built are:

6 ACRONYMS AND ABBREVIATIONS.

The following table provides a listing of the acronyms and abbreviations used in this document.

Table 6-1. Acronyms and Abbreviations

Acronym/Abbreviation	Definition
API	Application Program Interface
BSP	Board Support Package
CC	Common Criteria
CF	Core Framework
CIK	Crypto Ignition Key
COMPUSEC	Computer Security
COMSEC	Communications Security
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off-the-Shelf
CS/S	Cryptographic Subsystem
CT	Cipher Text
DAMA	Demand Assign Multiple Access
DoD	Department of Defense
DSS	Digital Signature Standard
DTD	Data Transfer Device
EAL	Evaluated Assurance Level
EKMS	Electronic Key Management System
ESIOP	Environment-Specific Inter-ORB Protocol
FOUO	For Official Use Only
GIOP	General Inter-ORB Protocol
GPP	General Purpose Processor
GPS	Global Positioning System
HCI	Human Computer Interface
HDR	Header
HMI	Human-Machine Interface
HW	Hardware
I&A	Identification and Authentication
I/O	Input/Output
ID	Implementation Dependant
IDL	Interface Definition Language
IIOP	Internet Inter-ORB Protocol

Table 6-1. Acronyms and Abbreviations - Continued

Acronym/Abbreviation	Definition
INFOSEC	Information Security
IOR	Interoperable Object Reference
IP	Internet Protocol
JTRS	Joint Tactical Radio System
KG	Key Generator
KMI	Key Management Infrastructure
KP	Key Processor
LAN	Local Area Network
LLC	Logical Link Control
LMD	Local Management Device
LRU	Line Replaceable Unit
MAC	Medium Access Control
MILS	Multiple Independent Levels of Security
MLS	Multiple Levels of Security
MMU	Memory Management Unit
NIST	National Institute of Standards and Technology
NSA	National Security Agency
NSTISSI	National Security Telecommunications Information Systems Security Instruction
OE	Operating Environment
OMG	Object Management Group
ORB	Object Request Broker
ORD	Operational Requirements Document
OS	Operating System
OTAR	Over-the-Air Rekey
OTAT	Over The Air Transfer
OTAZ	Over-the-Air Zeroization
P	Packet
PKI	Public Key Infrastructure
PT	Plain Text
RF	Radio Frequency
RTOS	Real Time Operating System
S/S	Subsystem
SAC	Systems Applications-Core
SAP	Systems Application
SCA	Software Communications Architecture

Table 6-1. Acronyms and Abbreviations - Continued

Acronym/Abbreviation	Definition
SHA	Secure Hash Algorithm
SRD	Support and Rationale Document
SW	Software
TCP	Transport Control Protocol
TOD	Time of Day
TRANSEC	Transmission Security
UIC	Unified INFOSEC Criteria
UML	Unified Modeling Language
XML	eXtensible Mark-up Language