MILS Compliant Software Architecture for Satellites

HJ Herpel, M. Kerep, G. Montano
Airbus DS GmbH
Claude-Dornier-Str. 1
D-88090 Immenstaad
hans-juergen.herpel@airbus.com
Phone: +49 7545 82482

K. Eckstein, M. Schön,
ESA ESTEC
Kepplerlaan 1
NL-2201 AZ Noordwijk
knut.eckstein@esa.int

A. Krutak
SYSGO AG
Am Pfaffenstein 14
D-55270 Klein-Winternheim
andrey.krutak@sysgo.com

Abstract— **Future satellite missions like Earth Observation, Telecommunication or any other kind are likely to be exposed to various threats aiming at exploiting vulnerabilities of the involved systems and communications. Moreover, the growing complexity of systems coupled with more ambitious types of operational scenarios imply increased security vulnerabilities in the future. In the paper we will describe an architecture and software elements to ensure high level of security on-board a spacecraft. First the threats to the Security Partition Communication Controller (SPCC) will be addressed including the identification of specific vulnerabilities to the SPCC. Furthermore, appropriate security objectives and security requirements are identified to be counter the identified threats. The security evaluation of the SPCC will be done in accordance to the Common Criteria (CC). The Software Elements for SPCC has been implemented on flight representative hardware which consists of two major elements: the I/O board and the SPCC board. The SPCC board provides the interfaces with ground while the I/O board interfaces with typical spacecraft equipment busses. Both boards are physically interconnected by a high speed spacewire (SpW) link.**

TABLE OF CONTENTS

ABBREVIATIONS

AOCS	Attitude and Orbit Control
API	Application Programming Interface
CCSDS	Consultative Committee for Space Data Systems
CDHS	Core Data Handling System
ECSS	European Cooperation For Space Standardization
FDIR	Failure Detection, Isolation and Recovery
KARS	Kontroller fuer autonome Raumfahrtsysteme
OBC	On-board Computer
PUS	Packet Utilization Standard
SSS	Software System Specification
TC	TeleCommand
TM	Telemetry
TRL	Technology Readiness Level
TSP	Time and Space Partitioning
XML	Extended Markup Language
IOH	IO Handler
RTOS	Real-Time Operating System
OSAL	Operating System Abstraction Layer
RDC	Remote Data Concentrator
IOM	Input Output Module
ASIC	Application Specific Integrated Ciruit
FPGA	Field Programmable Gate Array
RIU	Remote Interface Unit
S/C	SpaceCraft
NASA	National Aeronautics and Space Administration
IPC	Inter Process Communication
SOW	Statement of Work
OBC-SA	On-Board Computer System Architecture
SPCC	Software Elements for Security - Partition Communication Controller
ESA	European Space Agency
BSP	Board Support Package
SLOC	Source Lines of Code
TOE	Target Of Evaluation
PTM	Payload Telemetry
BTM	Bus Telemetry
CC	Common Criteria
TSF	Target of Evaluation Security Functionality
EAL	Evaluation Assurance Level
IDE	Integrated Development Environment
MMU	Memory Management Unit
MDPA	MultiDSP/uProcessor Architecture
MILS	Multiple Independent Levels of Safety and Security
I/O	Input/Output
TT&C	Tracking, Telemetry and Command
RF	Radio Frequency
RTU	Remote Terminal Unit

1. INTRODUCTION

Current satellite missions that require security are protected through their Telemetry (TM)/TeleCommand (TC) gateways. Their on-board architecture is specifically designed and evaluated for security, e.g. resources are duplicated such that on-board data is physically separated into individual classification domains. A common example of this approach is the implementation of TM/TC crypto services in dedicated hardware devices. As a consequence security introduces an overhead in terms of mass, volume and power for spacecraft design. A higher level of integrating software applications by means of secure partitioning can reduce these penalties while incorporating the advantages of software to security functions typically implemented in hardware (Application Specific Integrated Ciruit (ASIC) or Field Programmable Gate Array (FPGA) based crypto units).

2. USE CASE EARTH OBSERVATION

The requirements for the implementation of the Software Elements for Security - Partition Communication Controller (SPCC) were derived from typical space use cases like earth observation, telecommunication and hosted payloads [5]. In

this paper we describe only one use case from earth observation.

The Pleiades and SPOT missions are both responsible for collecting earth images, storing and forwarding to the ground. The first SPOT mission launched was in 1986, with more satellites and upgrades improving both the resolution and responsiveness of the network. The Pleiades spacecraft provide the ability to monitor the same point on the earth at 45 minute intervals due to the spacecraft being 180 degrees apart in the same orbit. The data acquired is used for:

- Regular monitoring
- Environmental monitoring
- Disaster monitoring

SPOT scenes cover a geographical area of 60km x 60km, whilst the Pleiades images cover areas up to 100km x 100km. The image data is stored on board in a 90 Gbit memory, and passed to the ground via a 100Mbps data down-link.

Reference Architecture

We describe here the system architecture of a typical earth-imaging spacecraft. This is broadly representative of the typical architecture of many Earth Observation spacecraft systems. Two classes of TM/TC links are present:

- Platform TM/TC for spacecraft command and monitoring
- Payload TM/High speed telemetry from a payload

The functional elements of the Earth Observation satellite mission are shown in Figure 1, which shows the reference system to be considered. The elements shown as relevant to the current threat assessment include the space segment and the Radio Frequency (RF) communication links. The Ground Segment links are excluded in this document, as it is assumed that a similar threat analysis would be carried out for ground systems.

The space segment consists of two sections: the platform and the payload.

Spacecraft Platform—The platform is controlled and monitored by the Platform Operator using a Control Facility on the ground. For the purpose of the study, we can consider the platform to be comprised of three main elements:

- Tracking, Telemetry and Command (TT&C) Subsystem - Receives TC and transmits Bus Telemetry (BTM) to and from the Control Station
- Control and Data Handling Subsystem - Handles and distributes commands and data throughout the Platform and Payload
- Platform Subsystems - Perform Platform Operations (e.g. Attitude and Orbit Control (AOCS))

An on-board data-handling network is used to distribute TC and TM throughout the spacecraft.

Spacecraft Payload—The operation and requirements of the payload depend upon the specific mission. All command and control of the payload are performed via the spacecraft platform. This is the case on the majority of Earth Observation missions, and we use it as a template for our analysis of missions of this type. The User Facilities receive the imaging payload data, via the payload data-handling network interfaced to spacecraft payload telemetry transponder. In this case, the payload subsystem is responsible for:

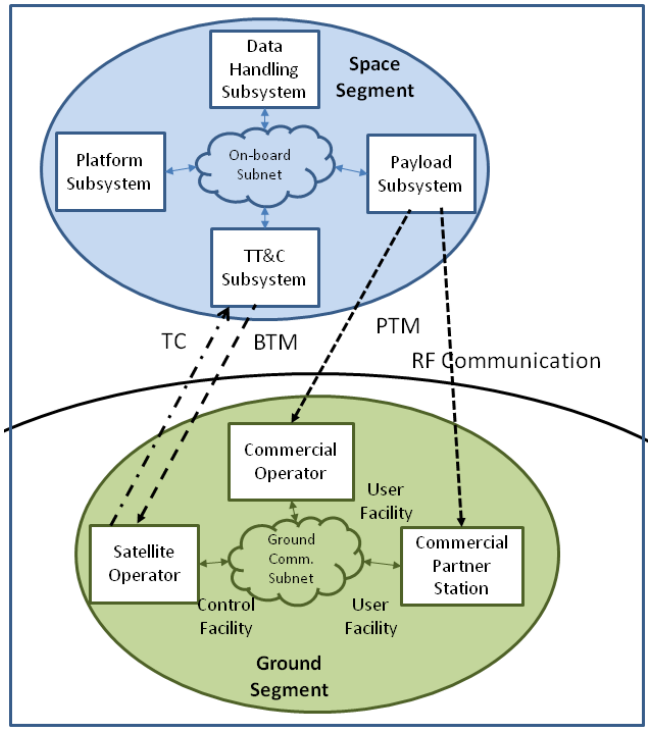


Figure 1. Functional elements of Space and Ground Segments

- Transmitting high-speed Payload Telemetry (PTM) or mission data to the User Facility
- Communication links: Reception of telecommands relevant for payload send from Control Station to SpaceCraft (S/C) platform (TC) and then forwarded via on-board subnet, Payload health telemetry (BTM) forwarded to S/C platform and from there to Control Station

Reference security architecture

Within the ground segment, TM/TC security services are provided by a TC/BTM Crypto Unit, and the protection of the imaging data is provided by a separate PTM Crypto Unit (Figure 2).

The image data in the Memory Management Unit (MMU) consists of data for a number of end-users. Encryption services are provided by a on-board dedicated hardware unit, where one section protects the low speed TC and BTM flows. The PTM encryption section provides high-speed encryption protection for all users' imaging data (PTM). There is a need to keep imaging data for different customers apart, due to a variety of reasons. For example there may be a commercial reason in that only one customer has ordered a specific image; or there may be an operational separation need, such as that identified on another earth observation mission, Earthcare, where 2% of the data is used for internal consumption only, and 26% and 72% of the data is distributed amongst two different user groups / levels.

The spacecraft receives Consultative Committee for Space Data Systems (CCSDS) compliant telecommand and emits telemetry, therefore the following appropriate security services are identified:

- Telecommand authenticity, confidentiality, integrity.
- Telecommand replay protection.

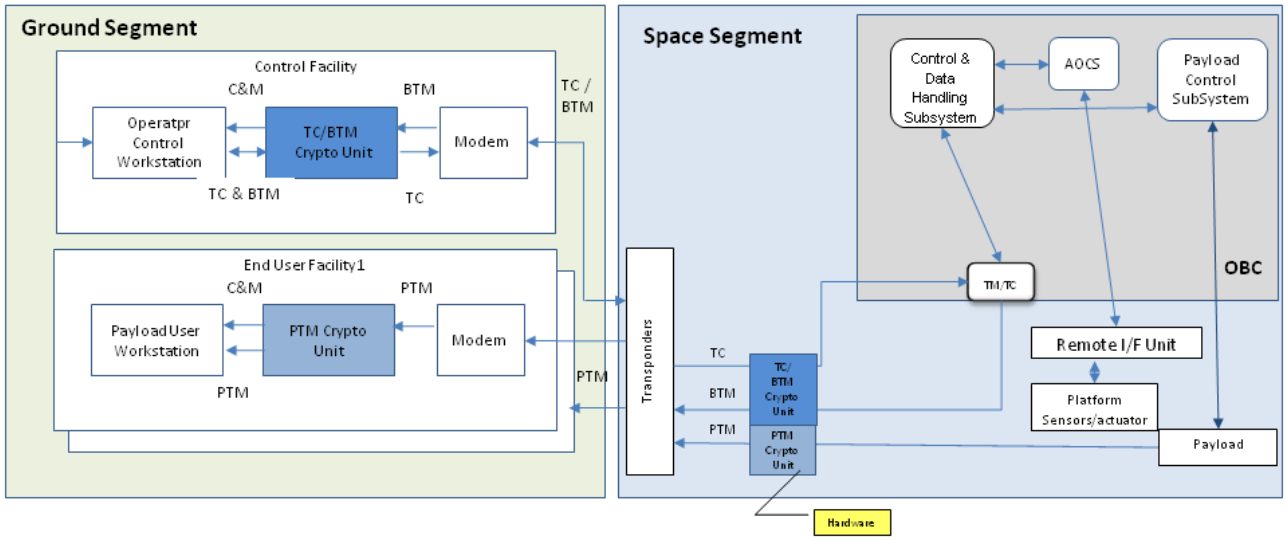


Figure 2. Block Diagram of Crypto Flight Unit and the Ground Communication Links

- Telemetry confidentiality.
- Key management for platform TT&C services.
- Image confidentiality protection.
- (Image integrity is provided, but not required as a security service).
- Key management for image data, i.e. selecting the appropriate key to protect the data for the appropriate end-user.

Due to the high data rates, a hardware solution for the Image Security is currently unavoidable. The high-speed PTM encryption function requires a key management which is tightly associated with the capture and storage of the images, such that the appropriate keys are used for the appropriate images. Depending on the mission needs, the TM/TC Security is implemented in either hardware or software. Government missions typically require hardware implementation to address crypto algorithm integrity and key material confidentiality concerns. Also, since the TC/BTM crypto function is part of the vital spacecraft command chain, hardware solutions may be required for reliability reasons.

3. Basic Concepts

Software partitioning and message communication

Higher levels of integration can be achieved by executing independent software modules isolated within partitions on the same computing node [13]. Time and Space Partitioning (TSP) enables the separation of concerns such as safety, performance and schedulability between functionally independent software applications to contain and/or isolate faults and reduce the effort of the software integration, verification and validation process.

A separation kernel is responsible for enforcing the temporal and spatial isolation of the partitions, where a partition is a dedicated set of computing resources (memory, CPU time, IO) allocated to a specific application. The kernel executes in supervisor mode while the applications within the partitions typically execute in user mode [1].

[1]TSP supports the concept of privileged partitions where applications execute in supervisor mode with higher access rights to kernel services and hardware resources

Partitions are able to only utilise message-based Inter Process Communication (IPC) [2].

Adding Security to the TSP Software Architecture

Originally a safety-motivated concept, time and space partitioning of software applications brings an increased capability of dependability and integrity in the avionics system since it prevents access to a given resource by a non-authorized application, either by technical fault, operator error or environmental effects.

As such TSP is complementary to the Multiple Independent Levels of Safety and Security (MILS) concept [18], where the original focus is on separation of security concerns such as confidentiality or privacy.

Combining the two approaches, to which we will refer to as secure partitioning, flight software applications can then be hosted on a single platform with guaranteed non-interference, resilience against malicious actions, while still maintaining the mission safety needs. Security mechanisms can therefore be added to the flight software to ensure the protection of the integrity, confidentiality and availability of the software system. A minimal trusted computing base can be identified, the secure separation kernel, which enforces the security properties, as shown in Figure 3.

Secure partitioning will protect the integrity, availability and confidentiality of data and programs within each partition and the separation kernel. However data that is communicated via IPC across the partition boundaries cannot have all of its security properties assured by the secure separation kernel. In order to maintain a minimum trusted computing base, i.e. keep the functionality of the kernel to the smallest possible, formally verifiable set of functionality, it only controls the setup of channels between partitions. The kernel should therefore not contain code governing content of the data communicated via IPC. A safe and secure design employing only a secure separation kernel in practice will have to limit channels to exist between partitions of the same security classification and safety criticality levels. This is the result of the

[2]IPC is defined as communication between two or more partitions executing either on the same processor/core or on different processors/cores

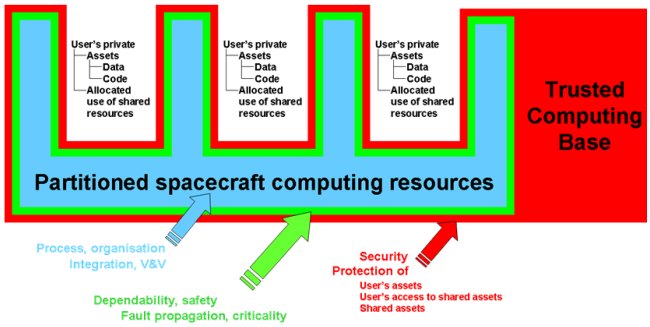


Figure 3. Security and Partitioning

combination of the multi-level security Bell-LaPadula ruleset *no read up; no write down* for maintaining confidentiality when communicating between elements of different security classification and the Biba ruleset *no read down; no write up* for communication between elements of different integrity criticality [3] [4].

Secure Partitioning as such does not prohibit reconfiguring the set of established communication channels. The reconfiguration has to be done in a secure fashion, e.g. switching between a fixed set of different configurations, which all need to adhere to the restrictions outlined in the paragraph above. The key point here is that the secure partitioning kernel can and does govern the existence of channels, but indeed it does not govern the content.

Therefore, a set additional software components is needed to provide mechanisms for data to be safely and securely transferred between applications that reside in partitions of different classification or criticality levels and to the external system, i.e. devices. This set of additional components we term *Secure Partition Communication Controller* (SPCC). It includes mechanisms such as routing (SPCC-R), encryption (SPCC-E), content checking (SPCC-CC) for data transfers to partitions of lower confidentiality classification or partitions of higher integrity criticality as well as mechanisms implementing efficient, safe and secure access to multiple communication channels (Input Output Module (IOM)).

Adding security to the TSP development process

A process and role description has been defined for all stakeholders involved in developing a partitioned software system for spacecraft avionics. A detailed description can be found in [12].

With TSP, security can be incorporated either within the application itself by the Application Supplier, or at the System Integrator level. In the former case, the application supplier must be aware of the confidentiality levels of all applications and devices their application is communicating with. They can then filter or downgrade their messages appropriately. This would create a dependency between applications meaning that a modification in the confidentiality level of a destination application could induce a change to the source application. In addition, this close binding between applications would reduce the opportunity for re-use of applications.

The alternative approach where security is handled under the responsibility of the system integrator requires messages to be adapted while they are being exchanged between partitions of different criticality and classification. The source application would be unaware that its messages have been modified, and the destination application would receive a message that it is appropriate for its confidentiality and integrity level. This concept of adjusting the integrity and confidentiality of messages passed between different entities is well discussed in the Introduction section of [22]. Therefore, the system integrator (not the application partition developer) configures the channel connections as well as security mechanisms such as redaction or encryption within a processor and the channel connections between a processor and components external to the processor.

4. Definition of Analogous SPCC-based Security Architecture

By analogy to the security architecture presented in the previous sections, we consider a system where the hardware Cryptographic Flight Unit is partially replaced by a software-based Security Partition Communication Controller (SPCC). In addition to implementing the cryptographic functions for the low-speed TC/BTM flows, software security functions can also be used to manage complex key management, thereby reducing implementation complexity for the remaining high-speed PTM security functionality. In addition, an I/O module is foreseen to separate the cyclic and event driven data aquisition/distribution from the TSP-based SPCC.

Furthermore, the SPCC data redaction functionality introduced earlier shall be employed. In order to provide a realistic scenario, sample applications shall be able to transmit messages of mixed confidentiality, i.e., a confidentiality, or a clearance level shall be allocated to each parameters in the message, instead of to the overall. This shall mean that parameters in a message can be of different confidentiality levels so the message needs to be filtered by the SPCC at parameter level. Consequently applications receive messages containing parameters appropriate for their clearance level and parameters containing higher classified data are censored.

In the reference use case described before, one or more of the spacecraft payload imaging equipments is capable of capturing image data that has sensitivities pertaining to a specific user, or group of users. The resulting data must be inaccessible to other applications. As the Image Security is implemented in a hardware solution, it is assumed that it is being managed by a software application which is running on the SPCC-based processor. The communications between the application and the hardware security function (which would be a peripheral on the payload spacewire interface) must be kept separate from other pieces of hardware and software, and it must not be possible for other pieces of hardware and software to communicate or control the Image Security hardware function.

The flight software security module (SPCC) operates as multiple software modules within separate partitions of the Onboard Computer which is equipped with a secure separation kernel. Separate modules would provide encryption and decryption capabilities for satellite TC, BTM and support the external security aspects of the PTM hardware encyption device. The respective modules of the SPCC would loaded with initial keys, with operational keys being delivered to the equipment over-the-air (OTAR). It is operated within a CCSDS compatible system, using a Security Layer within the communication protocol to enable the decryption and encryption capabilities. No fundamental changes would be necessary to the architecture of the ground security elements.

The following features are required:

- Telecommand decryption and authentication capability for all telecommands received by the spacecraft.
- Telecommand replay protection for all telecommands received by the spacecraft.
- Telemetry encryption capability for all telemetry passed through the SPCC, including security audit data generated internally.
- Key delivery to as well as general management such as scheduling of high-speed payload data encryption hardware for various end-users or groups depending on the customer.
- Capability to securely change and update of key material throughout the operational life of the SPCC.
- Secure internal control to ensure correct operation of the SPCC, the IO Module and the secure separation kernel throughout its operational life.

The end to end security architecture (Figure 5) considered replaces the On-board Computer (OBC) in Figure 2 with the OBC shown in Figure 4 and remove the TC/BTM section of the on-board crypto units. The purpose of using the on-board security architecture containing SPCC would be to:

- Allow more efficient implementation of the on board TT&C crypto unit, by reducing the need for separate hardware instantiation of resources for on board computer and security unit
- Securely implement the on board computer, restricting the required security evaluation / validation to the modules: SPCC-E, SPCC-CC, secure separation Kernel, I/O module. The standard flight software functions like AOCS, data-handling would not need to be evaluated.
- Isolate flight software functions operated by different roles into separate secure partitions, to increase e.g. confidentiality of event-driven payload planning vs routine platform operations.
- Likewise, a hosted payload's data management functions and TC/BTM routing could be isolated without requiring additional computer hardware.

In order to be able to recover from unforseen circumstances (e.g. data corruptions), and prevent spacecraft lock-outs, the on-board security architecture containing SPCC would:

- Carry initial / fall-back key material, state and configuration vectors in non-volatile memory only accessible by the partition dedicated to each particular security function.
- Detect data corruptions which would result in lock-out, and recover autonomously.
- Provide protection mechanisms to ensure that these recovery techniques cannot be initiated maliciously (e.g. a hosted application hogging the processor resulting in a system time-out and fall-back).
- Provide security functions with a strength of implementation comparable to a standalone hardware/software implementation.

The spacecraft TT&C link is critical to the successful operation of the spacecraft, and loss of the link results in loss of the mission. Processors and software are historically less trusted for system critical applications than purely hardware solutions, because of their inherent complexity, caused in part by the huge number of interfaces required to provide the desired functionality. Secure time ans space partitioning offers mechanisms to increase robustness which go beyond traditional approaches to increase the reliability of software such as the installation of hardware monitors and watchdogs,

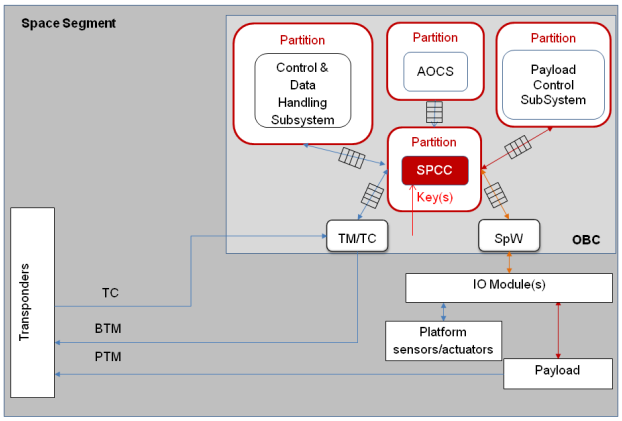


Figure 4. OBC Architecture

or the parallel development of the same application by two completely independent teams / companies.

The additional benefit of TSP lies in the ability to offer partitions as highly isolated compartments with very low internal complexity, into which singular critical functions can be implemented. With sufficient maturity of the secure separation kernel and the ability to reallocate such small, critical partitions to different cores of a high-reliability, multi-core CPU, this study assumes that critical functions like TC decoders or TC/TM crypto units, which today are implemented in a complex, hot redundant ASIC or FPGA either external or internal to the traditional OBC, can achieve similar reliability.

Description of the major Elements

Core Module (SPCC-R)—The SPCC-R shall be functionally similar to a router for inter-partition communication within a Time and Space Partitioned (TSP) system, as illustrated below. It enables authorised transmission of data between applications of different integrity and/or confidentiality by assuring that such data in transition is first forwarded to a SPCC-CC Content Checking function. The SPCC-R is the only element in entire architecture that has the ability to route communication between the partitions of differing confidentiality/integrity levels, because partitions of the same level are permitted to communicate directly via IPC by the secure partitioning kernel. All other partitions have to communicate via the SPCC-R (see Figure 4). As a consequence, the partitions can be unaware of the other partitions in the system because they are only aware of the SPCC-R. This encourages portability of applications. The desired properties of the SPCC-R are listed below:

- The SPCC-R must contain the routing table for all inter-partition communication between differing confidentiality and/or integrity levels so that it can correctly forward messages from a source partition to the destination partition, while guaranteeing that intermittend message redaction is performed by the appropriate SPCC-CC instance.
- In addition to regulating data flows between partitions, SPCC-R shall be able to route data flows towards other physical entities by forwarding them to physical communication interfaces, either directly via a local device driver, or indirectly via a channel to a partition implementing a device driver in isolation, or via offloading these external

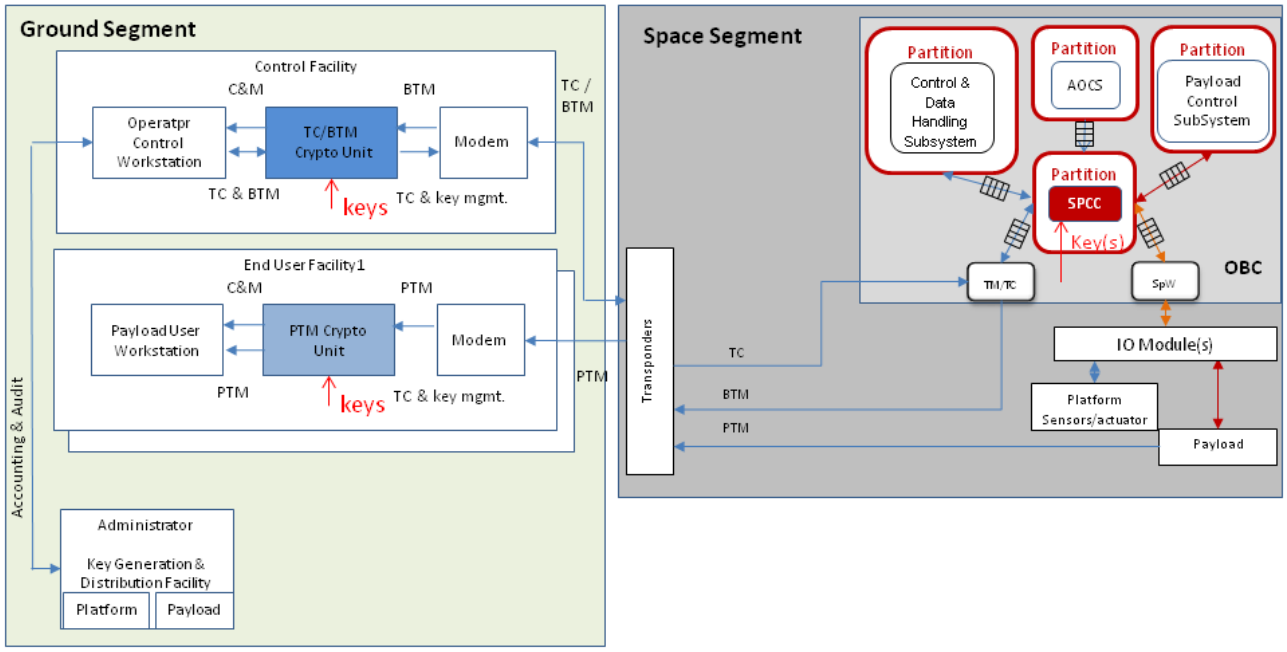


Figure 5. End to End Security Architecture between SPCC and the Ground Communication Links

communications to an IO Module.

- The routing mechanism shall allow priority channels to be used to reduce the time taken to transfer messages.
- All communication between applications shall be based on Packet Utilization Standard (PUS) [9], or similar , data protocols. This shall allow the application suppliers to deliver an application complete with Spacecraft Database. The System Integrator shall then be able to add confidentiality and integrity properties to the TM/TC parameters in Spacecraft Database. The SPCC-CC guard mechanism shall then use these confidentiality and integrity characteristics to determine what actions (i.e. enforce integrity or downgrading confidentiality) needed to be performed on the messages.
- The routing table can be updated in-flight. This allows the SPCC-R to be reconfigurable without affecting the underlying configuration of the separation kernel. The separation kernel specifies the allowed communication channels between partitions. Thereby the overall security and integrity protection responsibilities are split between the SPK which governs the fundamental security and integrity classification of each partition and can enable direct communication between equal levels, and the SPCC-R which regulates filtered communications required between levels.
  - Like for a secure separation kernel, each individual configuration variant must have been assessed to conform with security and safety requirements.
  - Like a secure separation kernel, SPCC-R reconfiguration may only be triggered by Failure Detection, Isolation and Recovery (FDIR) or authorized commands.
  - The SPCC-R implementation must be independent of the separation kernel technology.
- A hierarchical error handling strategy shall be used to resolve anomalies locally within the SPCC. If the SPCC-R as a master module cannot resolve its own errors, or those of SPCC-CC or SPCC-E, then the error is passed to the next higher level of spacecraft FDIR.

Content Checker (SPCC-CC)—The SPCC-CC shall ensure that confidentiality and integrity is preserved by guarding all data exchanges while being transparent to the application(s).

- SPCC-CC shall be executable in a separate partition from SPCC-R, for maximum separation of security concerns.
- The SPCC-CC shall implement a mechanism to downgrade the confidentiality of data communicated from a high confidential to an application of lower confidentiality.
- The SPCC-CC shall implement a mechanism to enforce data integrity for messages passed between a low criticality application to a higher criticality application.

Encryption/Decryption Service (SPCC-E)—

- SPCC-E shall be executable in a separate partition from SPCC-R, to achieve a strength of implementation similar to a dedicated crypto device.
- Multiple instances of SPCC-E shall co-exist, providing cryptographic services for different data flows owned by different operator roles, e.g. platform TC and BTM vs telecommands for SPCC-R to change configuration vs telecommands directed at SPCC-E to e.g. distribute new keys
- Each of these instances shall be able able to store its own initalisation key material, so that the secure separation kernel's memory access control mechanisms shall provide "natural" isolation for this sensitive data.

Input/Output Module—In order to reduce the overhead of servicing Input/Output (I/O), e.g. polling the devices on the software partitions, the aeronautical domain has adopted dedicated IO handling hardware modules that are additional hardware computing modules to the main processor platform. These IO modules buffer the data from the communications network and make the data available to the partitions via SPCC-R. This can remove the need for specific IO gateway partitions thereby improving the performance on a partitioned system. The IOM can be an additional module within the onboard computer or a dedicated data collector similar to the Remote Interface Unit (RIU)/Remote Terminal Unit (RTU)/ Remote Data Concentrator (RDC) units used on spacecrafts or aircrafts.

The IOM shall take responsibility for servicing IO away

from the partitions. Access to the IOM should be restricted and one approach to ensure security is that the access is only via the SPCC. This would mean that all IO traffic is routed via the SPCC-R to and from the IOM and the other partitions. The SPCC would then use its routing table to exchange the IO traffic between the IOM Port and the source/destination application Ports. The IOM would be responsible for performing any data formatting in order to prepare the message for transmission along the specific IO bus, e.g. apply SpaceWire headers or copy the data into the correct MIL1553 message or reading/setting data directly from/to digital or analog channels.

In a system with minimal or no confidentiality concerns, applications could communicate directly with IOMs without the need for the SPCC. This would enable prompt and timely access to the IO traffic within the applications own time partition.

5. THREAT ASSESSMENT

The Common Criteria (CC) Evaluation is an international standard (ISO/IEC 15408) for computer security certification. It is currently in version 3.1 revision 4 [2]. Common Criteria is a framework in which computer system users can specify their security functional and assurance requirements (SFRs and SARs respectively) through the use of Protection Profiles (PPs), vendors can then implement and/or make claims about the security attributes of their products, and testing laboratories can evaluate the products to determine if they actually meet the claims. In other words, adherence to Common Criteria provides assurance that the process of specification, implementation and evaluation of a computer security product has been conducted in a rigorous, standard and repeatable manner at a level that is commensurate with the target environment for use.

The CC provide a framework for assessing a security system, and specifically provides a set of areas that should be considered such as Security Audit, Communication, Cryptographic Support, User Data Protection, Identification and Authentication, Security Management, Privacy, Protection of Target of Target of Evaluation Security Functionality (TSF), Resource Utilisation, Target Of Evaluation (TOE) Access, Trusted Path/Channels. Within each area, a number of functions and requirements are identified within CC. All the applicable requirements and functions for flight software are considered in this document.

The Evaluation Assurance Level (EAL) (EAL1 through EAL7) of a system is a numerical grade assigned following the completion of a Common Criteria security evaluation (see Table 1). The increasing assurance levels reflect added assurance requirements that must be met to achieve Common Criteria certification.

Table 1. EAL Definition

Level	Description
EAL 1	Functionally Tested
EAL 2	Structurally Tested
EAL 3	Methodically Tested and Checked
EAL 4	Methodically Designed, Tested, and Reviewed
EAL 5	Methodically Designed, Tested, and Reviewed
EAL 6	Semi-formally Verified Design and Tested
EAL 7	Formally Verified Design and Tested

The CC defines security functional components to be considered for the development of secure systems in CC part 2. Part 3 of CC identifies the assurance requirements that the secure system must meet for the purpose of CC evaluation.

A threat assessment is used to determine the threats that are relevant to the current reference system as discussed and to attempt to quantify those threats. The threat assessment considers the threats to the reference system when there are no cryptographic capabilities present, in order to determine what threats a cryptographic processor is required to protect against. Additional threats created by the presence and implementation of a cryptographic processor within the system are considered separately [7].

The first step in the threat assessment is to identify the relevant threats. This has been done by using the generic threat list provided by [1], to determine those threats that are relevant to the system, referred to as the specific threats. A list of the generic threats and the resulting specific applicable threats to the reference missions from Earth Observation and Telecom is presented in [7].

The TOE is then defined (see Section 6), to indicate the boundary and contents of the security equipment being analysed and evaluated in this case, the components enforcing security on board the spacecraft computer. The generic threats are then detailed according to [2] to identify the specific threats that must be countered by the TOE.

The identified specific threats are next mapped to a set of Security Objectives, which will prevent the threat from occurring if they are met.

Finally, we map a set of Security Requirements for implementation on the Target Of Evaluation, which will meet those Security Objectives [5].

6. RESULTS

Target of Evaluation

The TOE has to be defined, to indicate the boundary and contents of the security equipment being analysed and evaluated in this case, the components enforcing security on board the spacecraft computer. The generic threats are then detailed according to [2] to identify the specific threats that must be countered by the Target of Evaluation. We then map the identified specific threats to a set of Security Objectives, which will prevent the threat from occurring if they are met. Finally, we map a set of Security Requirements for implementation on the Target Of Evaluation, which will meet those Security Objectives.

The Security Target for the use of SPCC to implement a Cryptographic Module comprises a set of flight software security modules (SPCC-R and SPCC-E as introduced above) designed to provide encryption and decryption capabilities for satellite TC, BTM and a custom module to support the management of the high-speed PTM hardware encryption function. [6].

TOE identification: SPCC Cryptographic Processor + Separation Kernel
Assurance Level: EAL4.
Key Words: Satellite Security, Telecommand Decryption, Telemetry Encryption, OTAR.

In Common Criteria terms the security target is Part 2 extended and Part 3 Conformant. The TOE is developed using security functional components as defined in the Common Criteria version 2.2 [2] part 2 and one explicitly stated functional requirement, with the assurance components as identified in part 3 of [2] for an assurance level of EAL4.

Threats to be countered by TOE

The TOE is required to assure the confidentiality and integrity of telecommands and telemetry on a satellite control path in order to support the availability of the control functions. This section briefly describes the threats that have been identified and which must be protected against, either by the TOE or the security environment in which it operates. The value of the assets to be protected depends upon the equipments which use the control path and the particular services that they provide. In the case of SPCC, the TOE is used in the direct control path to protect the spacecraft platform and SPCC payload control capabilities and its capacity to provide flexible communications services. The general threats to be countered are unauthorised access to the satellite control functions to either

- Cause denial of service
- Use the spacecraft against the wishes of the Operator
- Gain information that the appropriate User does not wish the aggressor to have

The specific threats that have been identified and need to be countered by the TOE are listed in [7]. Some of the threats are illustrated in Figure 6.

Validation Results

In section 6 the EAL to be achieved was defined to be four. This means, the software need to be methodically designed, tested and reviewed.

The software has been developed according to a tailored version of the European Cooperation For Space Standardization (ECSS) Software Engineering standard [10]. The amount of source code that has been developed during the study is less than 6000 Source Lines of Code (SLOC). The software has been fully tested,i.e. 148 unit test cases, 46 system test cases and 16 acceptance test cases covering specific threats [15].

The statement coverage figures are 93% for the SPCC component itself and 100% for the processing services (SPCC-E and SPCC-CC). The decision coverage is 94% for SPCC, 97% for SPCC-E and 100% for SPCC-CC. The requirements coverage is 100% for 114 software requirements and 98 user requirements. Approx. 80% of the requirements are verified by testing. The remaining 20% are verified either through review or inspection. The software has been validated on Leon4 platforms with equipments connected via typical interfaces used on-board a satellite like SpaceWire and MIL1553. The Leon4 is considered as next generation microprocessor for space applications.

Considering these facts and the National Aeronautics and Space Administration (NASA) Technology Readiness Level (TRL) definition [20] the SPCC software developed during the European Space Agency (ESA) study has achieved a TRL of five: *System/subsystem/component validation in relevant environment: Thorough testing of prototyping in representative environment. Basic technology elements integrated with reasonably realistic supporting elements. Prototyping implementations conform to target environment and interfaces.*

The performance figures obtained during testing with the LEON4 board running at 200MHz are summarized hereafter:

IO overhead (i.e. time of execution of void IO call) is 0.12ms, the avg. time it takes for a PUS packet to go from one application to another via SPCC is 1.5ms, time of data transfer between two SpW ports via the PikeOS SpW driver is between 0.4ms and 0.43ms, depending on packet size (0 - 122 Byte) and maximum data transfer rate is approx. 2340kiB/s.

From the requirments given in [16] (1 OBC, 4 payloads, 10 TC and 20 TM packets per element, max. packet length 1024 bytes) a typical data rate of approx. 1230kiB/s is needed. Thus, the demonstrator achieved the objectives.

7. THE DEMONSTRATOR

Overview

A demonstrator (Figure 11) has been developed to further investigate and improve the security on-board a spacecraft. As defined in [13] it consists of two modules:

- the OBC and
- the IOM

The OBC directly interface with the ground station (usually via the TM/TC frontend) while the IOM interfaces with the equipment busses like MIL1553 and SpaceWire. Both modules are directly interconnected via a SpaceWire link.

As shown in Figure 7 each module consists of:

- Application Layer with
  - Example applications of different confidentiality levels
  - SPCC components
- Execution Platform with
  - basic system and component support services typically provided by the on-board data handling software
  - separation kernel
  - hardware (computer board)

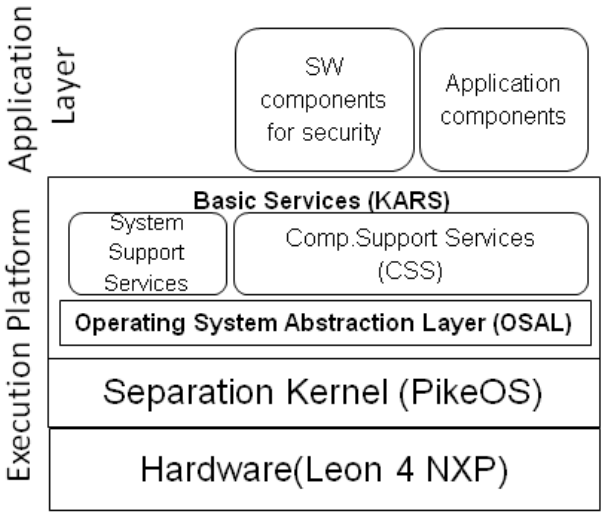


Figure 7. Hardware and Software Layers of the Demonstrator

In addition, system integrators and operators have to be supported in

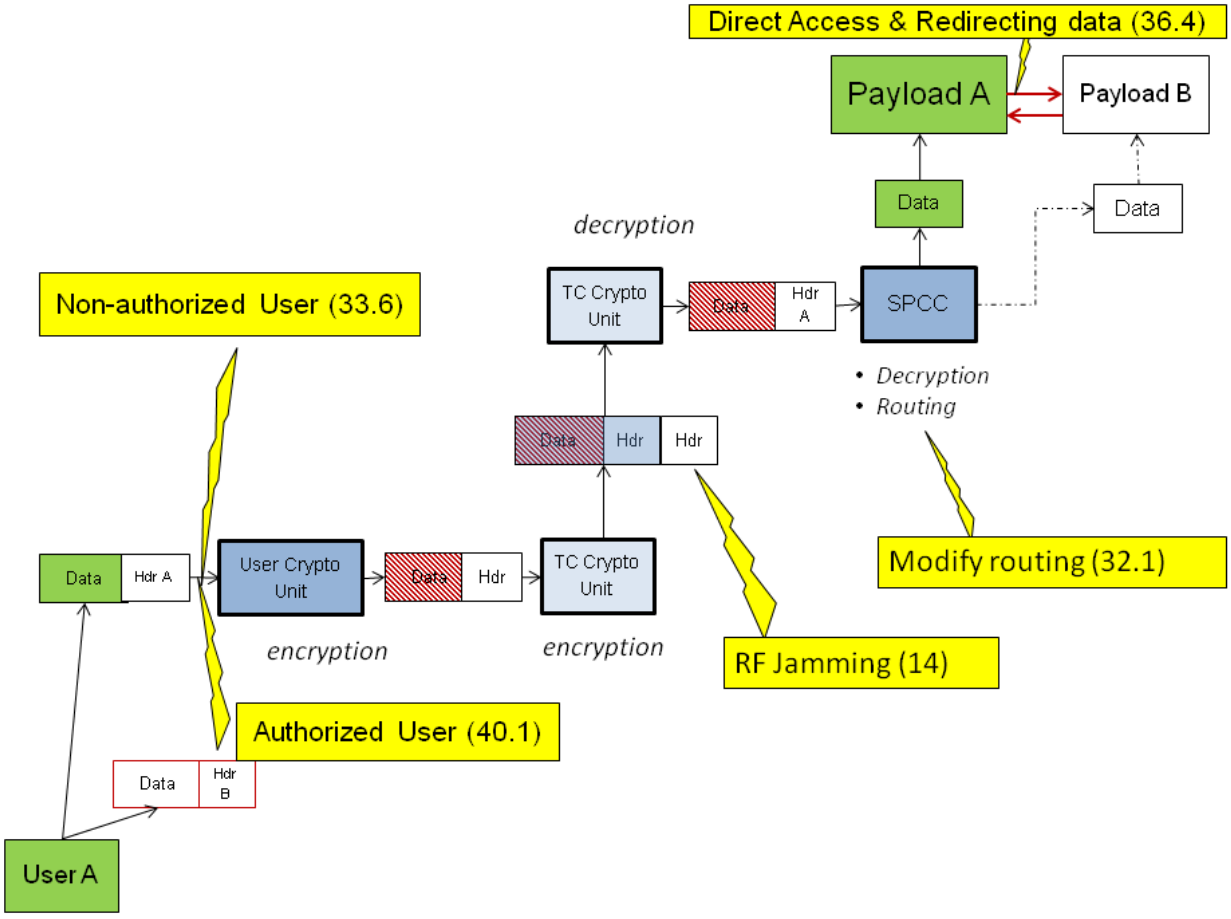


Figure 6. Specific threats to be encountered by the TOE

1. assigning applications to resource partitions,
2. defining the physical communication channels between resource partitions,
3. defining logical communication channels between applications of the same or different confidentiality levels.
4. placing resource partitions in time partitions,
5. generating the software image to be executed

The functionality for items 1,2,4 and 5 is provided by the tool environment of the selected separation kernel, i.e. CODEO for PikeOS.

For item 3 two additional tools have been developed in the frame of this study:

1. to syntactically check the definition of the logical communication channels and their characteristics
2. to check the defined logical channels wrt. the physical inter-partition communication channels provided by the separation kernel which were defined by the system integrator

The basic functionality of the toolset is described in the following section.

Toolset

Overview—According to the Statement of Work (SOW) [13] two tools were identified to support the SPCC generation process with the following capabilities:

1. Generator: generation of the SPCC sourecode using Extended Markup Language (XML) input files that specify the Routing Table and clearance and integrity actions to be performed on the message data.
2. Verifier: Checks the correct generation of the SPCC vs XML files. It could include automatically generated test cases that verify the SPCC Routing Table and filtering actions.

The Toolset is intended to be used in the following (or similar) scenario:

1. System integrator prepares the routing tables and target system configuration/integration project
2. The system configuration is validated
3. Routing tables are compiled (converted from XML to MSCF)
4. The target system image is built, including the compiled routing tables
5. The target system is deployed

When SPCC needs to be reconfigured on the fly, the following has to be done:

1. The new routing table is compiled
2. The routing table is sent to SPCC via PUS packets

Typically, the Toolset will likely be integrated into the system build scripts (Makefiles), so that the routing tables are recompiled when modified, and the target system configuration is

validated at every system compilation.

The following two tools have been developed in the frame of the study (see Figure 8):

1. Routing table compiler (spcc-rtc)
2. System configuration validator (spcc-validate)

These tools can be called any time will produce outputs solely depending on the input files. The flow to generate valid routing tables is shown in Figure 8. More details can be obtained from [19].

A third tool has been developed (PUS Commander) in order to provide an easy to use interface to the SPPC sending telecommands to the SPCC and receiving telemetry data from the SPCC.

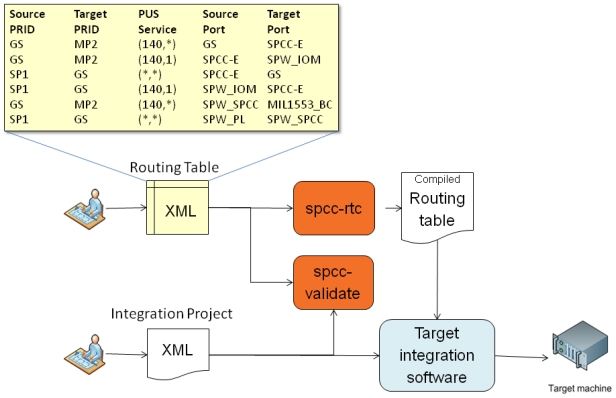


Figure 8. **Routing Table design flow**

Execution Platform

Several options were investigated for the execution platform in order to provide an optimal solution within the given time and cost frame. This includes the hardware, the separation kernel and the basic on-board software functionality.

Hardware—For the hardware several options were discussed:

- MultiDSP/uProcessor Architecture (MDPA) or SCOC3 processor boards (Airbus OBC products)
- On-Board Computer System Architecture (OBC-SA) High Reliable Processing Board based on LEON4 ASIC (developed in the frame of an DLR contract)
- ESA Next Generation Microprocessor based on LEON4 (Cobham Gaisler evaluation board)

Finally, the GR-CPCI-LEON4-N2X Quad-Core LEON4 Next Generation Microprocessor Evaluation Board from Gaisler [21] was selected (see Figure 9), mainly because of the following reasons:

- The quad core CPU provides sufficient processing power for software encryption
- The LEON4 comes with an MMU, i.e. hardware support for space partitioning
- Stability of the board
- Availability of the hardware at reasonable costs
- Reusability of the hardware for other studies on multicore architectures

In order to limit the effort for developing software for different execution platforms it was decided to use the same hardware for both modules.

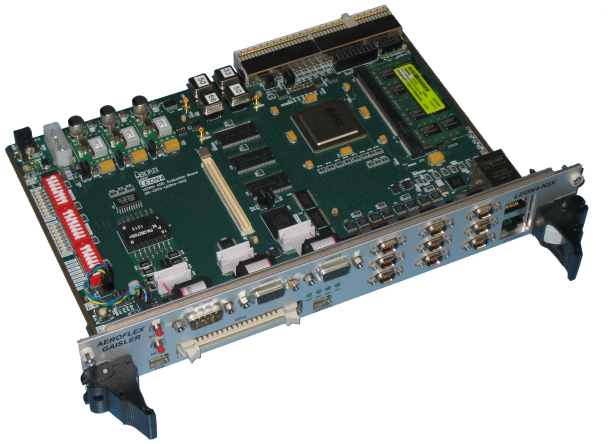


Figure 9. GR-CPCI-LEON4-N2X Board

Separation Kernel—Basically, there are two solutions available for the separation kernel:

- Pure Hypervisor, e.g. xTratum or AIR
- Virtualization platform incl. Real-Time Operating System (RTOS), e.g. PikeOS, VxWorks653

We selected PikeOS, mainly because of the following reasons:

- PikeOS includes a separation kernel and a mirco kernel
- PikeOS supports ARINC 653 compliant inter-partition communication channels
- Board support package for LEON4 board is available
- Kontroller fuer autonome Raumfahrtsysteme (KARS) software runs out of the box on PikeOS

Basic concepts of PikeOS—The concept of PikeOS combines real-time operating system (acrtos), virtualization platform, and Eclipse based integrated development environment (Integrated Development Environment (IDE)) for embedded systems. The PikeOS real time operating system has been developed for safety-critical and security-critical applications with certification needs in the fields of Aerospace & Defense, Automotive & Transportation, Industrial Automation & Medical, Network Infrastructures and Consumer Electronics. Functionality to provide a safe and secure environment for critical applications is a vital part of the PikeOS core components. Safety and security aspects have been accounted in the PikeOS design. The PikeOS real time operating system uses a modular approach as shown in Figure 10 to enable system architectures tailored to individual projects needs and to support an incremental certification process.

The main features and responsibilities of PikeOS are:

- Hardware abstraction (processor and platform)
- First level exception and interrupt processing
- Address space management
- Thread management and scheduling
- Communication and synchronization
- Resource management and protection with respect to resources and processing time

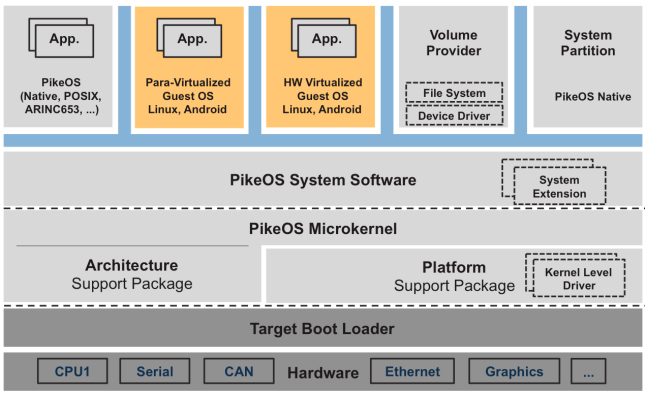


Figure 10. PikeOS Architecture

- Health monitoring
- Inter-partition communication
- I/O device abstraction and access control
- File system support

Most of the features listed above are implemented by the generic PikeOS Core. The PikeOS core consists of the PikeOS Micro kernel and the PikeOS System Software (PSSW).The generic part is, at source code level, independent from the CPU architecture, and, at object code level, independent from the underlying hardware platform.

Basic Services— For basic system and component support services three options were investigated:

1. Writing basic functionality from scratch
2. Reuse and adaptation an Airbus inhouse product called Core Data Handling System (CDHS)
3. Using KARS which has been developed by Airbus within the frame of an DLR contract.

The third option was selected as KARS is a highly modular software that can be easily configured for the purpose of the study. KARS has been develeoped for usage on a separation kernel and provides all basic PUS services.

Application - Software Components for Security

In the following only those components are briefly described that were specifically developed for the purpose of the ESA study. A detailed description of all components can be found in [17].

SPCC-R— The Software Elements for Security - Partition Communication Controller (SPCC-R) protects against unauthorized transmission of data between applications of different integrity and/or confidentiality. The SPCC-R routes all communication between the partitions of differing confidentiality/integrity levels. Partitions of the same level can communicate directly via ARINC 653 queueing ports whereas all other partitions have to perform authorized communications via the SPCC-R, whose routing table setup will enforce the necessery processing services required to make these communications across security or safety domain boundaries compliant with applicable policies.

In addition to allowing intra-partition communication between hosted payloads, SPCC-R also indirectly enables communication of the partitions with the ground station. SPCC-R serves as a router inside a software system (e.g. On-Board Computer), providing optional processing security services including content en/decryption and filtering of the communication. The main purpose is to control the inter-component communication according to the configuration provided by the system operator. SPCC uses the KARS framework for the code to be independent on the host OS but it routes PUS packets [9].

Processing Services (SPCC-E, SPCC-CC)— In case processing of message contents is required (like encryption or content validation), the message will be forwarded to dedicated processing modules. Forwarding rules for message content processing are also defined within the routing table of SPCC-R. SPCC-E component implements PUS packet encryption/decryption service. SPCC-CC component implements a demonstration of PUS packet contents modification and validation.

Sample Applications— APP1 is a specific application for development and testing purposes on the SPCC node. It implements dedicated PUS Services which allows testing of SPCC functionality as well as benchmarking.

The PUSAPP component is pure KARS component which accepts any PUS packets over dedicated queuing ports (via the IO Handler (IOH)) and responds by TM(1,1) and TM(1,7) if the TC Flags requires this.

Demonstrator Configuration

Figure 11 shows the hardware elements of the demonstrator. The SpaceWire RMAP Responder (SRR) simulates SpaceWire equipments and provides a bridge to connect the PUS Commander through an ethernet connection to the OBC node. The PUC Commander injects PUS commands into the system and reads telemetry data from the OBC node. Both, the OBC and IOM node are connected to the SRR as well as directly connected to eachother via SpaceWire. The IOM node has an additional interconnection to the MIL1553 equipment simulator.

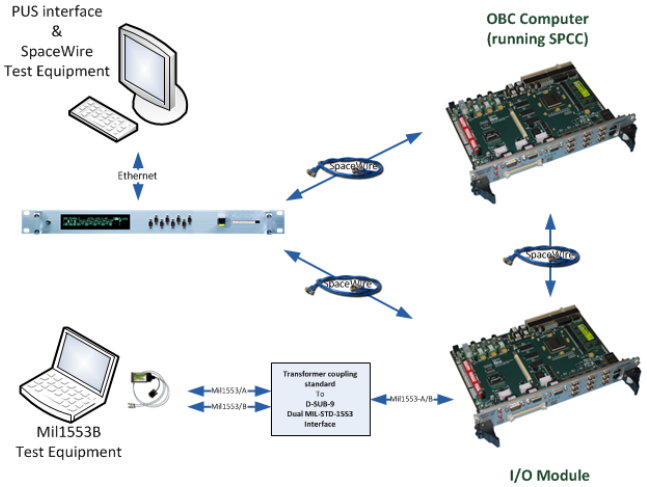


Figure 11. Demonstrator Hardware Setup

The OBC hardware platform in this setup hosts the following software elements (see also Figure 12):

- Separation kernel incl. Board Support Package (BSP) and driver for SpaceWire: PikeOS
- Space specific Software System Specification (SSS) implementing most PUS services (based on KARS [14])
- Component support services (CSS, tasking, ...) incl. the

Operating System Abstraction Layer (OSAL)
- Secure Partitioning Communication Controller (SPCC-R)
- Encryption Service (SPCC-E)
- Content Checking Service (SPCC-CC)
- Sample Applications (APP1, PUSAPP)
- Equipment Handler for SpaceWire links between OBC and IOM and for the TM/TC link to the simulated ground station

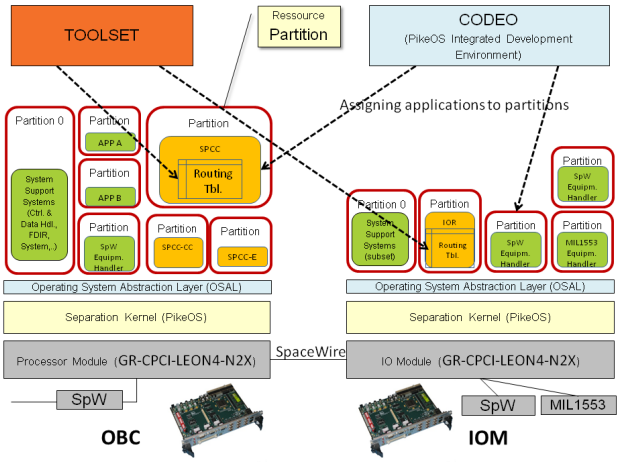


Figure 12. Demonstrator Node Configurations

The IOM node hosts a specific instance of the SPCC-R with a IOM specific routing table called IOR together with the equipment handlers for SpaceWire [11] and MIL1553 [8] incl. the corresponding drivers (Figure 13).

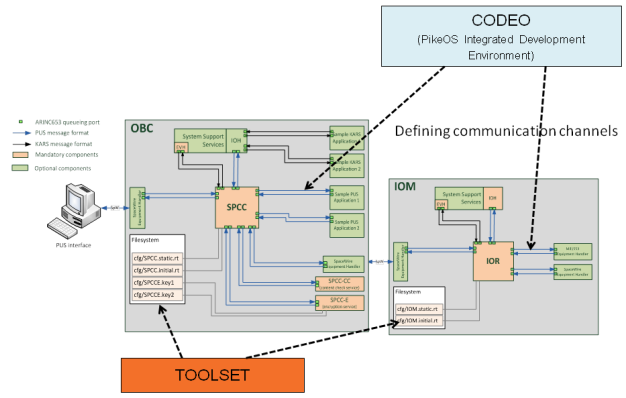


Figure 13. Demonstrator Interconnection Scheme

8. Conclusions

In this report we presented a software architecture adding security features to the on-board computer of satellites. This is achieved by using Time and Space Partitioning with a separation kernel and re-usable software components providing enhanced message routing and en-/decryption capabilities.

The separation kernel ensures that applications remain within the defined partition boundary. Communication paths between applications are defined by the system integrator based on the security level for each application. The mechanisms used are ARINC 653 compliant message queues. Thus, the source application would be unaware that its messages have been checked, and the destination application would receive a message that it is appropriate for its confidentiality level. These communication channels are defined when the correspondig integration project is created. As this information is compiled into the software, it is not possible to establish additional channels during run-time. Thus, bypassing the defined channels or reading memory areas of other partitions is not possible. In the architecture we have defined, the communication between applications of different confidentiality levels goes through a specific router - the Secure Partioning Communication Controller (SPCC). This router checks the validity of the message and ensures that only those information is passed to another application or device that fulfills the security rules specified in the routing table. In addition to the SPCC we have introduced an IOM. This module takes away the responsibility for servicing IO from the partitions. Access to the IOM is restricted and one approach to ensure security is that the access is only via the SPCC. This means that all I/O traffic is routed via the SPCC to and from the IOM and the other partitions. The SPCC uses its routing table to exchange the IO traffic between the IOM Port and the source/destination application ports. The IOM is responsible for performing any data formatting in order to prepare the message for transmission along the specific IO bus, e.g. apply SpaceWire headers or copy the data into the correct 1553 message or reading/setting data directly from/to digital or analog channels.

The architecture presented is based on the KARS framework developed in the frame of an DLR contract aiming at missions with a high level of autonomy. This framework provides a full implementation of the PUS services (called System Support Services) and provides an Application Programming Interface (API) called *Component Support Services* to application developers to reduce their development effort. The underlaying OSAL is available for a number of operating systems, e.g. PikeOS, VxWorks and Linux. Thus, porting the entire software to another hardware platform is a matter of the availability of the platform support package. Currently, the software is available for x86, Leon4, PPC and ARM platforms.

Especially, multicore platforms offer many advantages when performance is an issue. For example, the encryption component (SPCC-E) could be assigned to one (or more) CPU allowing software en-/decryption in real-time. Thus, expensive hardware in terms of power consumption and mass could be eliminated.

To summarize, using Time and Space Partitioning, message routing according to well defined rules and software en-/decryption allows to combine safety and security features on one platform. Thus, the additional costs when introducing security on-board a satellite in terms of processing ressources, mass, power consumption and development effort is limited.

The presented architecture will also allow to combine OBC and IOM functionality on one computer board as shown in Figure 14.

Acknowledgments

The authors thank the ESA for supporting the study *Software Elements for Security Partition Communication Controller* through a research fund.

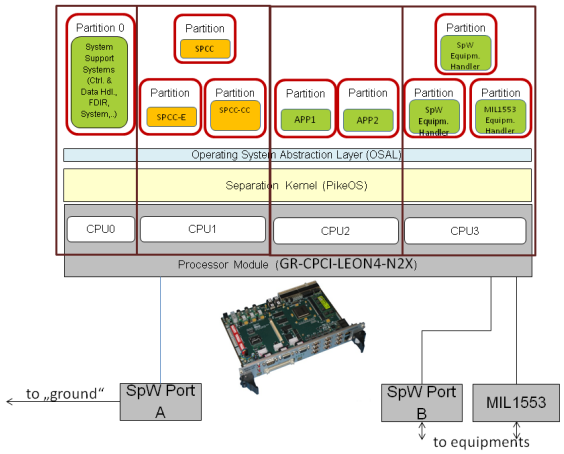


Figure 14. Single Board Solution for SPCC

REFERENCES

[1] *EBIOS v2 Section 4, Tools for Assessing ISS Risks*. FIPS PUB 197, 2001.

[2] *Common Criteria for Information Technology Security Evaluation Version 3.1, Revision 1*, number CCMB-2006-09-001, 2006.

[3] David Elliott Bell. Looking Back at the Bell-LaPadula Model. In *Proceedings of the 21st Annual Computer Security Applications Conference. Tucson, Arizona, USA. pp. 337351.*, 2005.

[4] K. J. Biba. Integrity Considerations for Secure Computer Systems, MTR-3153. Technical report, The Mitre Corporation, 1975.

[5] Clarke. Mission Requirements Use Cases. Technical Report SWES-ASU-TN-03, Airbus DS, 2014.

[6] Clarke. Security Threat Assessment (External) - Security Target. Technical Report SWES-ASU-TN-01, Airbus DS, 2014.

[7] Clarke. Security Threat Assessment (Internal). Technical Report SWES-ASU-TN-02, Airbus DS, 2014.

[8] ECSS. Interface and communication protocol for MIL-STD-1553B data bus onboard spacecraft. Technical Report ECSS-E-ST-50-13C, ESA, 2008.

[9] ECSS. Telemetry and telecommand packet utilization. Technical Report ECSS-E-ST-70C, ESA, 2008.

[10] ECSS. Software Engineering. Technical Report ECSS-E-ST-40c, ESA, 2009.

[11] ECSS. SpaceWire protocol identification. Technical Report ECSS-E-ST-50-51C, ESA, 2010.

[12] ESTEC. IMA-SP D08-11 - Process and role definition. Issue: 1.3. Technical report, ESA, 2011.

[13] ESTEC. Software Elements for Security - Partition Communication Controller, Statement of Work. Technical Report TEC-SWS/12-543/SOW, ESA, 2012.

[14] HJ Herpel. KARS: Entwicklung der Systemsteuerungssoftware: Kontroller fuer autonome Raumfahrtzeuge. Technical Report KAR-SYS-ASI-RP-21034, Airbus DS, 2014.

[15] HJ Herpel. Validation Test Report. Technical Report SPCC-SYS-ASI-RP-022034, Airbus DS, 2015.

[16] A. Krutak HJ Herpel, M. Kerep. SPCC Software Requirements Specification. Technical Report SPCC-SYS-ASI-RS-022021, Airbus DS, 2015.

[17] M. Kerep HJ. Herpel. Demonstrator Specification and Manual. Technical Report SPCC-ASI-SYS-SUM-024004, Airbus DS, 2015.

[18] W. Scott-Harrison Jim Alves-foss, Paul Oman and Carol Taylor. The MILS architecture for high-assurance embedded systems. In *International Journal of Embedded Systems, Volume: 2, Issue: 3/4, Inderscience, pp. 239*, 2006.

[19] A. Krutak. Toolset User Manual. Technical Report SPCC-SYS-SGO-SUM-022024-3, Airbus DS, 2015.

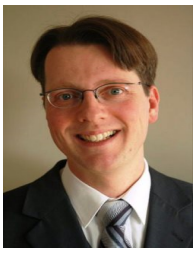
[20] NN. Technology Readiness Level Definitions. Technical report, NASA.

[21] NN. Quad Core LEON4 SPARC V8 Processor LEON4-N2X Data Sheet and Users Manual. Technical report, Aeroflex Gaisler, 2014.

[22] Devarajan Park, Joon and Ganesh. Fine-Grained and Scalable Message Protection in Sensitive Organizations. In *Journal of Software, Vol 2, No. 6*, 2007.

BIOGRAPHY

Hans Juergen Herpel *received a PhD degree from Darmstadt University in Electrical Engineering in 1995. He has now more than 20 years of experience in the field of embedded systems. This includes the implementation and design of hardware (boards, FPGAs, ASICs) and software for embedded system as well as the definition and implementation of a design methodology for these systems. The field of application ranges from automotive/train, airborne and spaceborne systems. In the past 10 years at Airbus he was also involved in various projects as system engineer and project team manager. In 2014 Mr. Herpel was nominated as Expert for Advanced Avionics Software. In his current position at Airbus DS he is coordinating the R&D related projects within the department.*



Knut Eckstein *is a computer and network security specialist working in the communications section of TEC-ET, the RF payloads division of the technical directorate at ESTEC. His main area of work are security aspects of the Galileo Navigation Programme. Other work he does or has done at ESTEC includes space, ground segment and system security aspects of Inmarsat Programmes and the European Data Relay Satellite (EDRS), as well as technology development. His R&D Interest include advances in distributed secure computing, in particular multi-level security, computer forensics and radionavigation security aspects. Knut holds an aerospace engineering degree from the University of Stuttgart (1993) and a PhD in computational mechanics (University of Stuttgart, 1997). He is a member of the IEEE and the ACM. Before joining ESTEC, he held network security engineering and team lead positions in private industry and an intergovernmental organisation.*