MILS Architectural Approach Supporting Trustworthiness of the IIoT Solutions

An Industrial Internet Consortium Whitepaper

Rance J. DeLong (The Open Group); Ekaterina Rudina (Kaspersky)

This paper describes the details of the MILS architectural approach, which has emerged as a strategy for cost-effective construction of systems requiring dependability with high assurance.

The paper is built like a pyramid. Every chapter explains the MILS approach from a certain aspect up to a certain level of detail, and then sets up a base for the next chapter.

In the first chapter, we provide the context and overview for the need for trustworthy system operation, explain generally what MILS is and how it addresses safety, security, privacy, reliability, and resilience of the systems in the industrial internet of things (IIoT). This provides a basic understanding of MILS for executives and business planners.

Using this knowledge, in the second chapter we dive into MILS concepts like domain separation, policy architecture, MILS platform and learn about MILS foundational components. This chapter is targeted to system architects and engineers.

Building on the understanding of MILS concepts, how the components can be used in a MILS-based system and how a MILS Platform is constructed, we explain how the architectural approach helps in building trustworthy systems. The third chapter describes MILS assurance methods and how they can be used to get trustworthiness in the IIoT environment. This chapter is required for technologists, prospective adopters and certification planners. To system architects and engineers, it provides extended knowledge on how to find a balance between trust by design and trust by assurance.

The last chapter describes MILS evolution over the last 40 years and current variations of the MILS platform, provides MILS case studies for the IIoT, and considers how MILS can be further extended to be used in the contemporary IIoT environment and cover the key safety challenges for IIoT. This chapter is useful to all previously mentioned audiences as it can be understood up to a certain extent with the knowledge obtained from any of the previous chapters.

1 Context and Overview

MILS builds on and extends a long tradition of work on architectural approaches to security to provide methods and tools to create high-assurance architectures for secure information sharing and dependable systems. These approaches aim to leverage system architecture to prohibit an unauthorized subject from accessing or modifying sensitive information while ensuring authorized subjects are able to do so. A thoughtful architecture may also limit the damage that can result from a compromised or failed subject.

The origin of the term ‘MILS’ was an acronym standing for ‘Multiple Independent Levels of Security/Safety’. Today it is used as a proper name for the approach that starts with partitioning the system under design into isolated compartments, or security *domains*. The MILS community today prefers the term ‘domain’ over ‘level’ because the latter implies ordering. Isolation and information flow control are the only required conditions for an architectural design, while ordering restricts the information flow control model.[1] The community has retained the name 'MILS' as a well-established term that is widely understood.

Domain separation is at the heart of MILS today, though the focus has shifted from hierarchical domains to mechanisms of domain separation and domain management that support any kind of security policy, including one that is not layered and not constrained by strict levels.

1.1 Need for Trustworthy System Operation

The foundations for MILS had appeared in the seminal separation kernel work of John Rushby published in 1981, distilling lessons from the security kernel developments of the 1970s, which he had closely studied for the UK Royal Signals and Radar Establishment. His conclusions and proposals were subsequently refined through his own work and that of others in the security community. In 1989 Rushby proposed kernels for safety, and in the 1990s the approach was practically adopted in the aviation safety sector as Integrated Modular Avionics (IMA) and partitioning real-time operating system (RTOS) kernels. By the early 2000s the name "MILS" was coined and a lively consortium of government customer, integrator, vendor, consultant, and academic participants, the MILS Initiative, came to be hosted by the Real Time and Embedded Systems Forum of The Open Group. Numerous sponsored research projects further refined the concepts and investigated the theoretical and practical aspects of MILS.

Over the subsequent two decades new application domains for MILS have emerged, such as industrial automation and critical infrastructures, bringing requirements for distributed secure systems and trustworthy adaptive systems, and with these requirements new scope and challenges for MILS. Accordingly, the concepts and objectives of MILS have been expanded and new MILS research initiatives undertaken to extend and develop MILS theory and technology to meet those challenges. These efforts have always kept sight of MILS' core objectives so as not to sacrifice past accomplishments. We will revisit in more detail the history and evolution of MILS in later sections.

The *raison d'être* of MILS, from Rushby's separation kernel concept onward, has been the need for assuredly secure systems, and later for assured safety and other critical properties. The ability to provide assurance has been a prerequisite at each step taken to advance the scope and

[1] For example, the Bell-LaPadula access control model in computer security theory considers the lattice on the partially ordered set of confidentiality and clearance levels. The information flows between the levels are constrained according to the simple rules "no read up" and "no write down" so that a subject with low clearance can't read highly confidential data, and there is no data leakage from subjects with high clearance to the low-level containers (like throwing a confidential document to the unclassified trash bin). There are some other similar models with isolation, flow control and ordering.

capability of MILS. The addition of a new operational capability to the MILS platform must be accompanied by corresponding theory, modeling, verification, and tool advances so that the distinctive attributes of the approach are preserved. Our understanding of how to achieve assurance and how to construct and present assurance artifacts for increasingly challenging use cases has advanced as a byproduct of pushing MILS forward to new capabilities.

MILS sets forth a bold vision for building and evaluating critical systems from separately constructed and evaluated components. The intuitive security architectural design proceeds by decomposing a system into a “circles and arrows” diagram and to continue splitting big circles into other circles and arrows so that security depends on only a few trusted circles, and that those are trusted to do relatively simple things. The arrows represent the needed communication channels among the circles. The behavior of each trusted circle exhibits the local policy that it is trusted to enforce. This *policy architecture* is then implemented on mechanisms that enable trusted and untrusted circles to share physical resources securely. It is presumed that circles and arrows are cheap so that decomposition may be used liberally to simplify the trusted circles and their associated policies.

1.2 WHAT IS MILS TODAY

The MILS architectural approach is a strategy for the cost-effective construction of systems requiring dependability with high assurance. It is a component-based approach to the construction, assurance and certification of trustworthy systems. In the design and implementation of systems, MILS emphasizes decomposition, policy architecture, separation, component integration, secure sharing of computing resources, and compositional assurance.

MILS is appropriate for systems requiring a high level of assurance for security, safety or other key characteristics in sectors such as automotive, avionics, industrial automation, defense and critical infrastructures. In terms of the IoT Security Maturity Model, the MILS approach applies to systems requiring the highest (formalized) level of comprehensiveness for the implementation of practices of the Enablement domain.[2] [3]

Popularly, MILS is often characterized (simply) as the use of a separation kernel to run applications belonging to diverse security domains, or having different levels of safety requirements (safety criticalities), on the same computer. The MILS Idea[4] is in designing an intuitive logical architecture to achieve a purpose and then creating an implementation structured to reflect that architecture faithfully. At the *operational* upper level, the main objective is to create a bespoke logical *component architecture* intended to achieve a specific

[2] [CEH+20] IoT Security Maturity Model: Description and Intended Use White Paper.

[3] [CHRZ19] IoT Security Maturity Model: Practitioner’s Guide.

[4] [Rus07] John Rushby. Compositional Certification for MILS. HCSS, 2007.

purpose with associated properties. At a *foundational* lower level, focus shifts on a *technology* intended to achieve partitioning, separation and secure resource sharing on which to realize the component architecture.[5]

"Modern MILS"[6] should first be characterized as the two levels mentioned above, and may be viewed through a "management view" and a "technical view". The MILS management view stresses the use of components, a commercial marketplace of high-assurance components, and a cost-effective development and certification strategy. Key concepts of the MILS technical view include aggressive design decomposition, policy architecture with few and simple trusted components, a resource-sharing implementation providing strong separation, and a methodology of component integration to facilitate compositional assurance and certification.

A set of standardized MILS foundational components have been defined to compose with a separation kernel to create the MILS Platform. The contemporary MILS Platform, supporting scalable distributed and heterogeneous environments and dynamically changing configurations, can provide an excellent platform for IIoT, enhancing trustworthiness of the five critical characteristics featured in the Industrial Internet Security Framework (IISF).[7] The allocation of the verification efforts over the various trustworthiness aspects may vary according to the needs of a particular system or part of a system. Thanks to component-based design, compositional verification tools, independent evaluation of commercial MILS components, and flexible automated assurance case support, such variations of need or focus of trustworthiness can be readily accommodated.

A key MILS objective is to encourage a competitive commercial marketplace of off-the-shelf high-assurance components. The technologies underlying the MILS Platform and the tool chain supporting MILS system development must enable reasoning about the interaction of the components, their differential criticalities and the resulting functionality and trustworthiness characteristics of the composition of the components. Many details of the elaboration of MILS and efforts to establish MILS standards have been driven by consideration of this objective.

1.3 How MILS Addresses Safety

In a safety-critical environment, which is characteristic of much of IIoT, a MILS approach has already shown its worth. In the safety partitioning of a system, the domains usually represent applications, perhaps at diverse levels of safety criticality. If the failure of one application, such as flight control on an aircraft, could cause severe damage, viz. the loss of the aircraft, the

[5] [RD07] John Rushby and Rance DeLong. Compositional Security Evaluation: The MILS Approach. ICCC, 2007.

[6] [BDRS08] MILS Component Integration and several John Rushby MILS presentations in 2008.

[7] [IIS16] Industrial Internet Security Framework Technical Report.

application requires high assurance for safety; if the failure of another application would cause a minor nuisance (e.g. loss of the passenger entertainment system), the application does not require high assurance for safety. However, a failure of the entertainment system must never result in the loss of the aircraft. Applications with high safety-assurance levels have more stringent process requirements (planning, development, verification) than applications with low safety assurance levels.

Safety-critical sectors such as avionics or automotive have employed partitioning techniques to improve safety and enable certification of the integrated systems. Resource isolation, fault isolation and enforcement of timing constraints all contribute to safe behavior. An example of safety-rated separation is the approach defined by ARINC Specification 653.[8] It describes Partitioning Operating Systems (POSs) that support applications with shared access to critical resources within integrated systems. They implement the separation in space and in time for independent execution of applications in domains, or partitions in the terminology of ARINC 653. Within each partition, multitasking is allowed. The API decouples the real-time operating system platform from the application software and provides services to manage partitions, processes and timing, as well as partition and process communication and error handling. The partitioning environment can implement this using a hypervisor to run partitions in separated virtual machines, but this is not the only possible implementation approach.

1.4 How MILS Addresses Security

Security architectural design starts with identifying the trusted components, the local policies and the communication channels, then choosing trusted components to minimize their complexity and their associated policies.

Component-based security approaches aim at creating system architectures that deny unauthorized parties’ access to sensitive information while ensuring authorized parties are able to do so. The authorization scheme is implemented by security mechanisms that monitor access attempts. The authorization policy, based on the classification of informational objects and the clearance level of the subjects accessing them, separates the environment into security domains and prevents information leakage or tampering. Fault isolation constrains the compromised or failed subject from inadvertently disclosing the information or causing damage to it.

When technological advances reduced performance concerns, isolation and separation were used as independent mechanisms to support secure and reliable execution. More recent technologies such as hardware support of virtualization facilitate generic domain separation so they are used to implement the security mechanisms or to mitigate their imperfections (such as security flaws or attack exposures).

[8] [Aer19b] ARINC Specification 653: Avionics Application Software Standard Interface.

1.5 How MILS Supports Reliability, Resilience, and Privacy

If domains do not need to interact, they should remain isolated to mitigate the effects of failures and attacks, including propagating and cascading failures, to increase reliability.

Reliability is the ability of a system or component to perform its required functions under stated conditions for a specified period. Applying domain-based security and safety controls enables independent measurement of the system's availability and simplifies support. Identification of components critical for system functionality and their separation from the non-critical components therefore facilitates the assurance of reliability and availability by establishing clear dependencies on influencing factors and determining the likelihood of failure.

If the system design supports backing up the domains or redundant implementation of their functions, the system may be reconfigured after the failure, adapting to the hazardous events and providing resilience.

Resilience requires anticipation of dynamic adversarial conditions, the determination of how to withstand them, how to recover after they have taken place and how to adapt where possible to the predicted changes in technical, operational or threat environments.[9] Trustworthy adaptation requires that a system can be dynamically reconfigured at runtime without compromising the robustness and integrity of the system. While MILS initially is a paradigm for rigorously developed and assured composable systems with a static configuration of domains, adaptive MILS is an extension to this paradigm with adaptation mechanisms for safe and secure reconfiguration of the domains within the constraints of a configuration policy.

MILS also facilitates privacy. Separation and isolation may segregate information between different domains based on 'need to know'. Guaranteed domain isolation ensures non-leakage of the private data to unauthorized agents.

2 MILS Concepts

2.1 Centralized vs Distributed Security Architecture

A centralized approach to security architecture has conceptual simplicity and a clearly identified reference validation mechanism. Such was the *security kernel* approach, in which the kernel was intended to be the sole trusted arbiter of the system wide security policy. In practice, it was necessary to create trusted components in addition to the kernel to handle operationally essential functions that required exemption from the kernel's security policy. Such components were given special privileges, sometimes more broad than necessary, to carry out their function.

[9] [RPG+19] Draft NIST Special Publication 800-160 VOLUME 2.

The attempt to enforce a single security policy over the entire system usually fails because trust must be distributed over components in different system layers. Even minimal use of shared resources, such as printers, backups, networked file systems and authentication services, must violate the central security policy because of the different functions they are required to perform. The security kernel taken together with these trusted components represents the trusted computing base (TCB) that enforces the resulting security policy. Much of the effort to develop and assure a security kernel to enforce a complex policy is wasted because of the quantity of functionality in the TCB that is privileged to violate kernel policy to be reasoned about, particularly if a methodology to reason about combined policies is lacking.

Moreover, some applications must not adhere to the security policy for safety reasons or due to the need for continuous execution, creating exceptions. Kernelized architectures cannot guarantee safety when the application cannot tolerate the delays caused by interposing centralized security policy enforcement into its execution.

A system where trust does not rely upon a central mechanism is a functionally distributed system. Its functions are provided by individual subsystems that are physically or logically separated from each other and provided with only limited channels for communication. A distributed architectural design of a trustworthy system, achieved through domain separation, is the key to overcoming the weaknesses of a centralized approach. Such a design presumes, and is dependent on, reliable establishment and strict enforcement of the architectural design, since critical properties of the design are derived from the architecture. The isolation of necessarily trusted components can both forestall their compromise and limit the damage to the rest of the system that could result from their compromise.

2.1.1 Domain Isolation

A domain is a unit of separation created and maintained in such a way that a system architecture constructed from a collection of interacting domains accomplishes its objectives, both functionally and in support of system trustworthiness. The reliable way to guarantee trustworthiness objectives is to isolate domains and support the minimally required communications among them on the basis of a default-deny information flow control policy. That is, there is no information flow among components that is not explicitly granted in the policy architecture. Strict isolation splits domains, preventing interaction. Effective isolation is a prerequisite for the enforcement of any access control or information flow policy.

Isolation of applications, such as by running them on different standalone computers, is the ultimate way to ensure non-interference. As no data is exchanged among the computers, the state of each application is determined only by its own execution and thus it is secure. Similarly, as the application doesn't produce external data or control signals, it can't affect the environment, or the applications running on other computers, so it is also safe. This is a good starting point. But a system is a collection of cooperating applications, and cooperation is the result of specific beneficial interactions, not unregulated interference, among applications.

Isolating independent parts dramatically decreases system complexity. A proper design can reveal the system functions that do not depend on each other and assign them to separate domains. Resources or functions that are tightly coupled may be assigned to the same domain provided that there is no risk associated with information being freely shared among them. Separation identifies the loosely coupled components or sets of components, reveals their connections and makes these connections explicit. How to achieve the separation depends on many factors.

2.1.2 Isolation and Information Flow Control

Minimally required communications among domains must be supported. This distributes the system as a set of domains communicating through explicitly defined interfaces. Explicit communication links enable control of information flows between domains. This is a primary function of the foundational components of the MILS platform, and first among them the separation kernel.

A *separation kernel* is a specific kind of security kernel that is limited to the enforcement of policies of isolation and information flow control among exported resources by management of shared physical resources to create an environment that is indistinguishable from that provided by a physically distributed system. It must appear as if each domain (regime) is a separate, isolated machine and that information can only flow from one machine to another along known external communications lines.[10] The implementation of a separation kernel is the aggregated hardware, firmware and software that together implements an abstract machine that performs this function. The separation kernel is the key foundational component for a MILS-based system.

One approach to implementing communications among domains is to define communication objects, exported resources associated with components or domains. Communication objects can be shared between components according to the security policy.

The combination of strict isolation with an information flow control policy is referred to as domain separation, or simply, separation, which lies at the heart of MILS today.

2.1.3 Separation as a Basis for the MILS Architectural Approach

Component-based approaches aim at creating system architectures that channel information only to the components that need it to perform their function. The policy represented by the architecture separates the computing resources into security domains and prevents information

[10] A quote from [Rus81].

leakage or tampering by using information flow control rules. Isolation of resources contains the potentially adverse effects of compromised or failed subjects from inadvertently disclosing information or causing damage to other resources.

The implementation of strong separation comes at a price in physical resources (to create physical separation) or at a price in energy for additional computing resources (to maintain logical separation with a support of appropriate technologies). As technological advances in microprocessor capacity reduce the performance concerns, separation can be used to support secure and reliable execution in embedded systems. More recent technologies to facilitate isolation and separation (such as hardware support of virtualization) can also be used to implement the security mechanisms or to increase their performance and robustness, extending the applicability of the MILS platform.

A *MILS Platform* is a composition of foundational components, created from hardware and software, that provide isolation and information flow control over a set of exported resources required for the implementation of a *MILS policy architecture.* The MILS Platform has its own distinct architecture whereby the foundational components are combined to achieve the properties that must be provided by the composition of those components.

The MILS Platform has the dual role of:

- creating abstractions of physical or other lower-level resources and exporting resources corresponding to these abstractions and
- enforcing policies of isolation and information flow control seamlessly over the set of exported resources according to the policy architecture that its foundational components have been configured to realize.

Architecture enforcement is achieved in practice through mechanisms that enforce domain isolation and information flow control.

2.2 MILS Policy Architecture

2.2.1 Policy Architecture

Architecture alone does not guarantee trustworthiness. To reason about trustworthiness, we need to conduct assurance procedures, which require properly defined assurance argument(s).

An assurance argument is a set of claims that assert that the system is acceptably assured relative to a trustworthiness aspect (such as safety or security). This set is usually well-structured and supported by the criteria, assessment scope, assumptions and limitations, list of the undesired events, mitigations for undesired events and the evidence, intended to justify that concern is properly addressed. The process of defining assurance arguments and reasoning about the claims comprising these arguments comprise the core of assurance procedures.

Assurance arguments may be defined for the whole system and for its separate parts. The ideal approach is to decompose a system and consider the parts and assurance arguments related to these parts separately. If the system assurance argument does not decompose on architectural lines, it is difficult to evaluate system behavior because the assurance procedure needs to reveal hidden connections and investigate subtle behavior.

The critical task for the system architect is to construct a system so that assurance decomposes along structural lines. The MILS approach addresses this issue. Policy architecture accumulates the knowledge about the system structure, components, trust given to these components and their possible communications. The strength of the guarantee that the architecture is properly implemented lends strength to the architecturally aligned assurance claims.

The MILS policy architecture is a set of abstract entities (active subjects, passive objects, communication primitives describing permitted interactions) that represents the system decomposition based both on the functional purpose of the system and requirements on the trustworthiness aspects and assurance.

MILS policy architecture is usually represented with a “boxes and arrows” diagram (or “circles and arrows”, as shown on the Figure 1). Circles encapsulate data, information, or control. Arrows are channels for information flow. Some circles are marked as trusted to enforce local policies for the communications control. The trusted ones should be as simple as possible, so the architect must decompose the policy architecture to achieve this.

The MILS approach has two phases: first, the development of a policy architecture that accomplishes a desired goal, and, second the implementation of that policy architecture on a platform that manages shared resources in a way that the fundamental assumptions made by that policy architecture are satisfied.

The MILS approach encourages a vigorous decomposition of a function into an abstract policy architecture that has components that exhibit simplicity and singleness of purpose, and that embodies the principle of least privilege, permitting only the necessary interactions among the components. Figure 1 illustrates a simple policy architecture comprising five components and their permitted interactions. It also depicts a component, C6, that is independent of the other components of the policy architecture, on which it has no effect and vice versa. One of the components, C1, is a trusted subject.[11] The local policy enforced by C1 determines what information will flow to C4, and in what form, of the information C1 receives from C2, C3 and C5.

[11] [CITa] CITADEL Project introduction to MILS.

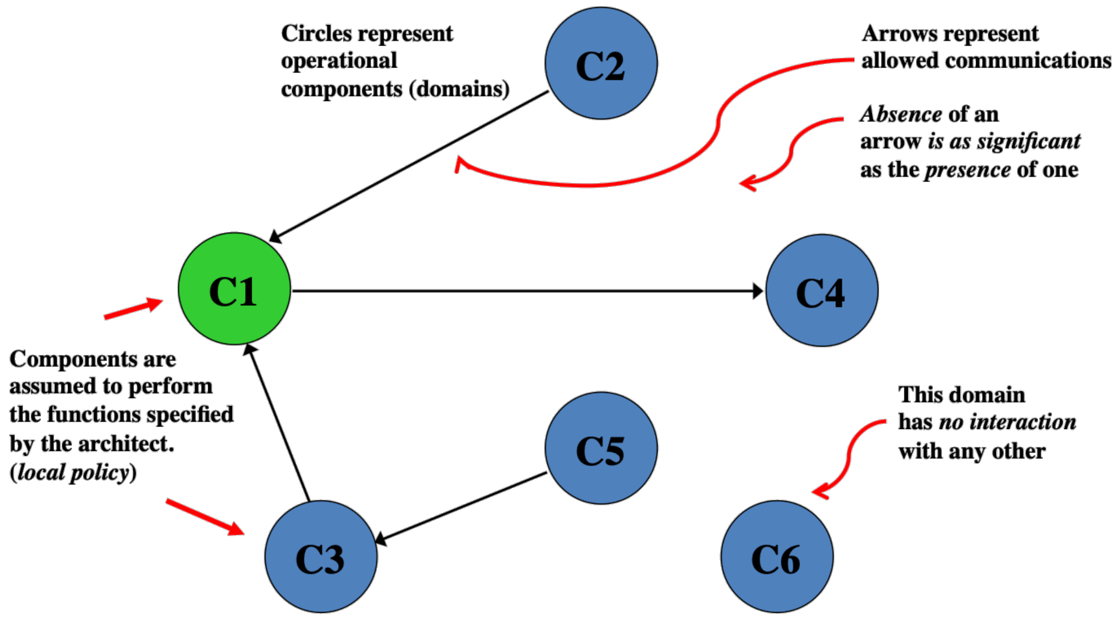

Circles represent operational components (domains)
C2
Arrows represent allowed communications
Absence of an arrow *is as significant* as the *presence* of one
C1
C4
Components are assumed to perform the functions specified by the architect. (*local policy*)
C5
This domain has *no interaction* with any other
C3
C6

Figure 1: Example of MILS Policy Architecture

For example, C1 may anonymize usage information from applications C2, C3 and C5; or, C1 may be a "downgrader" that aggregates potentially classified information received from C2, C3 and C5 and filters it to create an unclassified summary. The non-interacting application C6 may be the backup that stores information independently in a way that does not allow any damage to it in case of cyberattack.

2.2.2 Distributed Policy Architectures

A distinction must now be made between the notion of a MILS policy architecture as a conceptually distributed system that is implemented on a single computer, and that of a MILS policy architecture implemented across a physically distributed system. In *Distributed MILS* the MILS platform comprises a special network of MILS computer nodes that is accompanied by tools to support the development and analysis of policy architectures that will be deployed on such a platform. Here, the elements of the policy architecture may run on different MILS nodes subject to constraints attached to the design and constraints imposed by the physical attributes of the networks and nodes making up the platform. The example policy architecture of Figure 1 is shown deployed over two distributed MILS nodes in Figure 2.

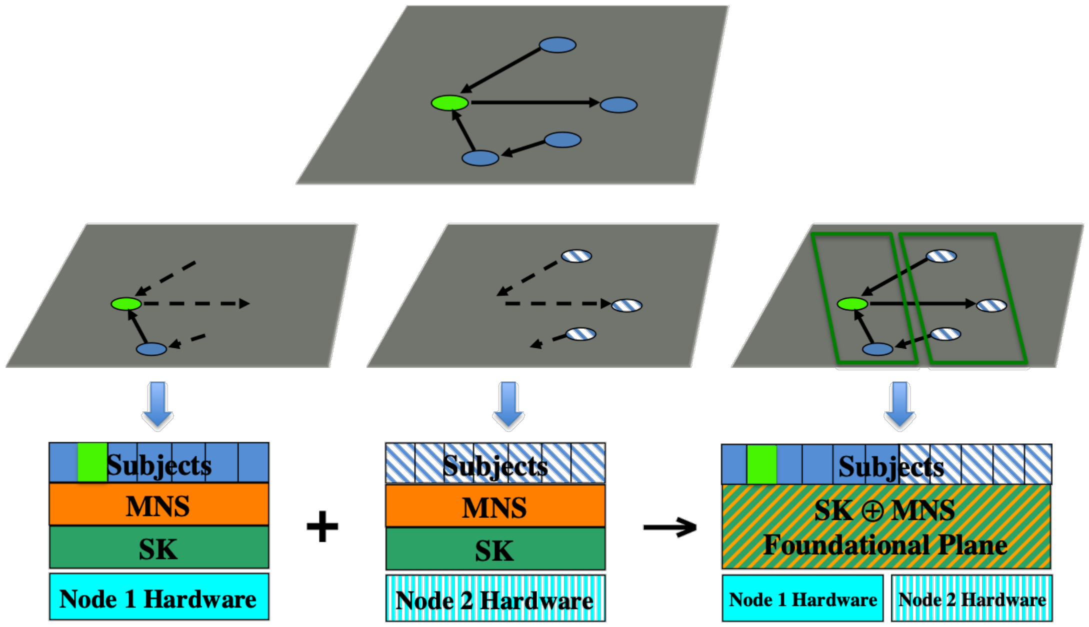


Figure 2: A distributed MILS policy architecture deployment

A new MILS foundational component, the MILS networking system (MNS), is added to the separation kernel (SK) to create a node of the new *foundational plane*. The subjects are distributed over the nodes. Separation kernel and MILS networking system form the distributed MILS foundational plane over which the policy architecture is implemented.

2.2.3 Static and Dynamic MILS Policy Architectures

Since the early MILS platforms were based on statically configured separation kernels, such MILS implementations provided only for fixed runtime policy architectures on a single computer. The only variation in the policy architecture possible at runtime was limited to a small set of predefined *modes*, the resources for which were statically configured. A change to a different policy architecture required a restart.

Dynamic MILS makes possible more extensive or even unbounded runtime change to a policy architecture, or the addition or deletion of physical resources to the platform in support of policy architecture changes. This is supported by a dynamic MILS platform composed from dynamically reconfigurable foundational components and is governed by advanced techniques and tools for modeling and analysis of dynamic architectures that enables it to be done safely (as explained below in 3.3.2).[12]

[12] [CST18] Cimatti. Formal Specification and Verification of Dynamic Parametrized Architectures.

2.2.4 Realization of a Policy Architecture

The realization of the policy architecture, illustrated in Figure 3 as a configuration of a MILS Platform, must guarantee that components and communication channels do not exchange information or interfere in a way that is not explicitly represented in the architecture even though their implementation may share physical resources such as processors, memory, mass storage and networks.

The implementation of a policy architecture is accomplished by refining the components and connectors of the policy architecture down to MILS platform exported resources. The basic elements of the policy architecture, such as subjects, objects and permitted communication primitives, are resources exported by the hardware and software of the MILS platform foundational components. These components enforce the system architecture by isolating the resources they export and controlling the interactions of the entities they represent. Together they implement a foundational framework for integration of the operational components of a policy architecture.

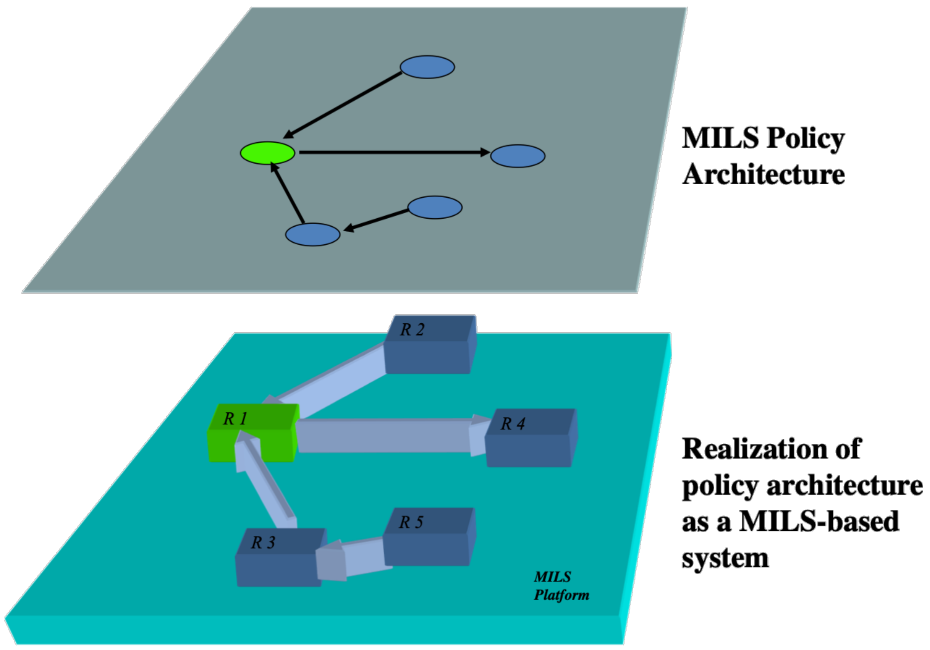


Figure 3: Realization of MILS-based system based on the Policy Architecture

The realization of policy architecture is referred to as an operational plane. A *MILS-based system* may implement one or more MILS policy architectures, each an independent graph component as shown in Figure 4. Distinct policy architectures may be considered to be independent operational planes.

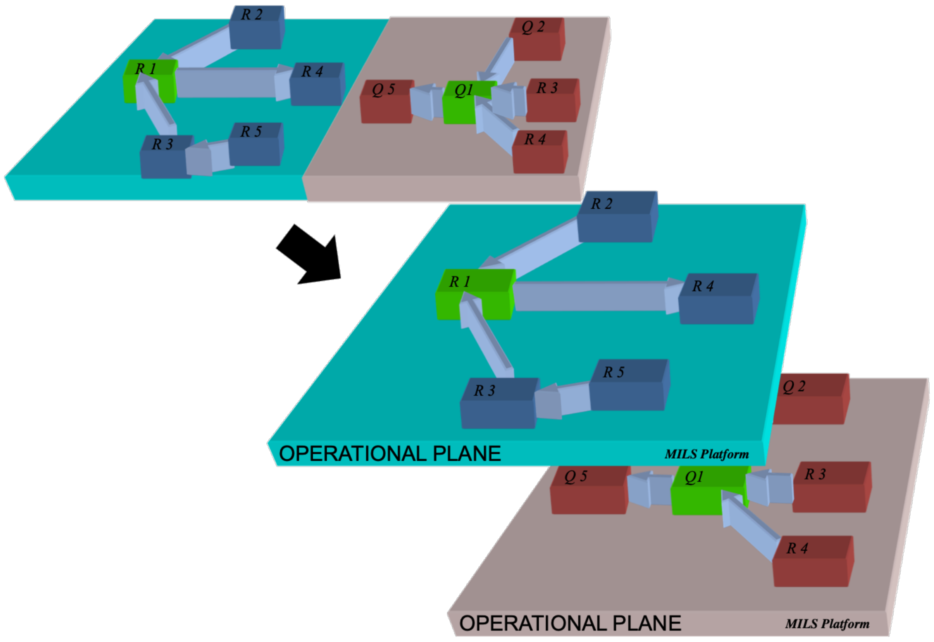


Figure 4: Isolated subsystems as distinct policy architectures

As policy architectures vary from case to case, using a common integration framework for building such systems is needed.

A *MILS platform* is the set of hardware and software components that provides the capacity for entity isolation and controlled information flow necessary for the implementation of a MILS policy architecture. A MILS platform may host multiple policy architectures.

A MILS platform must be capable of supporting the functionality of the system and assuring system trustworthiness. These high-level demands transform into a complex set of interrelated technical requirements for hardware and software components. Incorrect assumptions about the implementation may lead to unexpected behavior of these components and system compromise. Balancing the complexity of policy architecture and the structure of components underlying the platform is the essential task for an architect.

Different sources provide different ways of decomposing the hardware and software varieties comprising the MILS platform.

2.3 MILS Platform

2.3.1 Separation Kernel

A *security kernel* is a combination of hardware and software that implements a reference monitor for a system-wide security policy model.[13] After studying several security kernel projects, including UCLA secure Unix,[14] MITRE kernel,[15] KSOS[16] and ACCAT Guard,[17] in his 1981 paper Rushby criticized the security kernel approach while observing the simplicity of physical construction of high-assurance critical systems of the day. Rushby contends, "the overall security of a [physical] distributed system rests partly on the physical separation of its components and partly on the critical functions performed by some of those components", and proposed the separation kernel as a new kind of security kernel with the role to "re-create within a single shared machine, an environment which supports the various components of the system, and provides the communications channels between them, in such a way that individual components of the system cannot distinguish this shared environment from a physically distributed one."[18]

Rushby proposed a separation kernel as a new kind of security kernel with the role to "re-create within a single shared machine, an environment which supports the various components of the system, and provides the communications channels between them, in such a way that individual components of the system *cannot distinguish* this shared environment from a physically distributed one." The partitioning kernel[19] is a kind of separation kernel for integrated modular avionics and concerns safe separation largely based on an ARINC 653[20] style separation scheme. We will use the term *separation kernel* for both. They act as reference monitors for a separation policy in the system.

Separation kernels provide features to enforce isolation and information flow control. They must be small enough to allow formal verification of their correctness. Functionality may slightly vary depending on the intended use of a system built using the kernel.

Currently, a separation kernel is not the same as an operating system kernel. Some of the low-level functions implemented by an OS kernel may also support separation of domains but it also typically provides many features not included in a separation kernel. On the other hand, a

---

13 [And72] Anderson. Computer Security Technology Planning Study.

14 [PKK+79] Popek. UCLA secure UNIX.

15 [Sch75] Schiller. The design and specification of a security kernel for the PDP-11/45.

16 [MD79] McCauley, Drongowski. KSOS—the design of a secure operating system.

17 [Woo79] Woodward. Applications for multilevel secure operating systems.

18 [Rus81] Rushby. Design and Verification of Secure Systems.

19 [Rus99] Rushby. Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance.

20 [Aer19a] ARINC Specification 653, Part 0 - Overview of ARINC 653.

separation kernel has information flow control policy and enforcement as essential features. Currently, a separation kernel is not a “kernel” in the conventional sense but a combination of hardware technologies and software components that enforce separation in an assured way. The requirement for assurance is the distinguishing characteristic of a MILS separation kernel.

Separation kernels enforce spatial and temporal partitioning of domains, but allow them to communicate in a controlled manner. The extent to which isolation is provided by each of these types of separation depends on the system purpose and particular architectural design.

Spatial partitioning maintains data separation, information flow control and fault isolation. Data separation means that each domain is deployed as an isolated exported resource. Applications in one domain can neither implicitly affect data in other domains nor control applications and devices in them. Information flow control enforces only properly authenticated information flow between domains. Fault isolation restrains failure propagation from one domain to others.

Temporal partitioning maintains scheduling of execution of different domains at distinct times and supports measures to prevent leakage of information between domains over time.

Sanitization ensures that resources allocated to a domain or temporary data areas used by a domain are cleared when the resource is used to process data in other domains, thus enforcing safe processing of information from various domains at different times on the same hardware resources (so-called periods processing). Sanitization also helps mitigate cold-boot attacks and other attacks on confidential data within domains.

Thus, the relevant basic functionality of the separation kernel includes:

- partitioning resources into domains,
- support of interdomain communication,
- separation policy enforcement,
- memory management,
- scheduling,
- periods processing with physical resources,
- minimal interrupt servicing of devices by the kernel, passing interrupts on to be handled by subjects,
- essential synchronization primitives, timers and watchdogs and
- instrumentation (if required).

2.3.1.1 Partitioning Resources into Domains

Partitioning of exported resources into domains and providing seamless connections among resources of diverse types is the cornerstone feature of the MILS Platform. Each foundational component of the MILS Platform manages a different kind of physical resource and exports distinct kinds of exported resources constructed from those physical resources.

A separation kernel must support a consistent partitioning for the resources across the system layers, from hardware to applications. This is why the implementation of separation kernels remained impractical until microprocessors became capable of hosting multiple applications concurrently and separating them robustly and efficiently, using, for example, memory management units and hardware virtualization features.

The Radio Technical Commission for Aeronautics (RTCA) DO-297 guidance for Integrated Modular Avionics (IMA) developers, application developers, integrators and certification applicants defines the aim of "robust partitioning" to provide a level of functional isolation and independence equivalent to that of a federated system implementation. A partitioning analysis demonstrates that "no application or subfunction in a partition could affect the behavior of a sub-function or application in another partition in an adverse manner". The guidance given to achieve partitioning splits validation, verification, configuration management and certification processes into tasks done at the application level, the platform level, and the system level.[21]

2.3.1.2 Support for Interdomain Communication

Separation kernels provide the ability to define authorized channels between domains for interdomain communication. Data isolation can be contravened only through these channels, implementing the default-deny principle.

Different MILS systems and platforms implement interdomain communications using different communication models and transport mechanisms. One variant is a message-passing system conceptually similar to ARINC 653 ports. Another is shared memory buffers with synchronization mechanisms. The only requirement imposed by MILS is support for distinguished effectively unidirectional channels for data communication. However, as Architecture Analysis and Design Language (AADL) has become the de facto specification language for MILS, we recommend using a mechanism providing a semantically compatible model. An abstract convention established in MILS research has been to use ports that are located in the sender, readable and writable by the sender, but only readable by the receiver. Synchronous communication may be preferable in some applications, as asynchronous mechanisms require allocation and management of buffers (by the application) and provision of synchronization primitives (ideally by the kernel) or other flow control.

The complexity of the separation kernel implementation should be limited to enable verification. This limit is determined by the available verification methods and will change as verification technology advances.

[21] [RTC05] RTCA SC-200 / EUROCAE WG-60, DO-297: Integrated Modular Avionics (IMA).

The performance of information exchange is constrained by the need to check and enforce the security policy for the communications. Timing or other non-functional requirements may also restrict the available communication capacity. A balance therefore needs to be found between performance and security and other non-functional requirements.

Lastly, to maintain efficient interaction among applications in the separated domains, the MILS platform should provide appropriate communication primitives, such as "Unix domain" sockets as a protected library or implemented by the separation kernel. Communications among domains, especially on the same computer, may be very intensive as a result of the MILS principle of decomposition, and therefore should have an efficient implementation. Moreover, in a Distributed MILS system communicating subjects may be deployed on different computers, provided that the needed throughput and latency requirements of the applications can be met. The applications should not need be modified for a distributed deployment, so that deployment flexibility is maximized. Inter-subject communication primitives should be transparently backed by appropriate higher-level networking protocols and predictable network implementations. A draft MILS Platform API based on POSIX, including communication and synchronization primitives, has been produced by the MILS Platform API Working Group of The Open Group and is under public review.

The implementation of Distributed MILS and of the inter-subject communication between MILS nodes involves the provision of a "communication stack" and access to specialized networking hardware for deterministic network communication as part of the MILS Networking System foundational component. Its responsibilities include time-sensitive networking to create the MILS backbone for inter-subject communication in a Distributed MILS system and best-effort communication over conventional networks.

2.3.1.3 Security Policy Enforcement

Relative to a conventional operating system kernel, taking the decision engine for a system-wide access control security policy out of the separation kernel decreases its complexity and at the same time enables enforcement of arbitrary access control security policies by decision engines outside of the separation kernel. Following the reference monitor concept, reference validation mechanisms (RVM) for multiple policies are compatible with a common policy architecture pattern, one that provides for the tamperproofness and non-bypassability of the RVM.[22] The isolation and information flow control policies of the separation kernel are used to establish and enforce the policy architecture within which the RVM is applied to the exported resources of the MILS Platform.

[22] According to [And72] an RVM must be tamperproof, always invoked (non-bypassable), and "small enough to be subject to analysis and test the completeness of which can be assured" (i.e. assurable).

Access control enforcement requires the availability of security-relevant metadata or attributes to the security policy decision engine. For this reason, the definition of the MILS Platform includes a foundational component called MILS Extended Attribute System (MEAS).[23] The purpose of this system is to bind arbitrary attributes optionally to the exported resources of other platform components so that these resources may serve as the controlled entities of attribute-based access control policies through their RVMs. The resources exported by the MEAS component are the attribute metadata bound to other exported resources. The MEAS is an integral part of the MILS Platform because it must provide for strong binding and integrity of the attribute metadata with assurance commensurate with that provided for the resources to which they are bound. The separation kernel and the MEAS provide the trustworthy ability to include arbitrary security policy enforcement mechanisms within a policy architecture implemented on the MILS Platform.

In a network context the security-relevant metadata often comprise part of the message content. To retrieve it involves message parsing and supporting the security context for the system. This must be done in a trustworthy way so as not to compromise the purpose of the MEAS or the validity of policy decisions rendered on the basis of the metadata. These considerations should come into play as part of the design of complete MILS Networking System (MNS) and MEAS components. Except for possible minor extensions, no changes to a functionally complete separation kernel should be necessary to implement the MNS and MEAS, and thereby to support RVMs for arbitrary attribute-based access control policies. Support for this claim may be found in the success of implementation of the MNS for Distributed MILS without kernel changes for two separation kernels on two different projects.

2.3.1.4 Memory Management

Data separation requires memory address spaces of each partition to be independent of others. This requirement may be covered by different hardware-assisted features together with strongly assured memory management algorithms implemented by the separation kernel.

Memory management techniques within the kernel should be conservative so that the kernel cannot be the source of an unexpected memory error to a subject. Practices such as dynamic memory reallocation within the kernel increase the complexity of the kernel and create such a possibility. This is inevitable in a dynamically reconfigurable kernel and must be appropriately dealt with in kernel assurance. The kernel may provide the ability to share memory objects among subjects or among partitions but dynamic management of such shared memory should be delegated to the MILS application and the associated risks managed by the application developer.

[23] The MILS Extended Attribute System has not yet been implemented.

2.3.1.5 Scheduling

Processor schedules are determined by requirements of temporal partitioning and timing requirements of components and policy architectures such as end-to-end latency of inter-subject communication paths. Applications running on the MILS platform may require real-time execution or pose other constraints on schedules. For example, partitioning kernels used in avionics typically use a table-driven scheduling algorithm, which is well suited for safety-critical and hard-real-time systems as it enables checking the feasibility of the scheduling in advance. Other approaches such as fixed-priority preemptive scheduling, or dynamic planning-based scheduling, or dynamic best-effort scheduling are not excluded; however, the requirement of keeping the algorithms as simple as possible remains.

There are scheduling considerations that are complicated by MILS. Unlike many previous partitioned systems, which have relatively few partitions on a single processor, MILS encourages aggressive decomposition that could result in hundreds or thousands of subjects on a single processor and far greater numbers in a Distributed MILS system. The possibility of communications among subjects is established by the policy architecture, and the completion of any communication cycle requires both the sender and the receiver to execute. The latency of communication, even on a single processor, is therefore influenced by scheduling. The scheduling algorithm should take this into account so that overall execution of the system can be predictable. There are proposed separation kernel features to ameliorate this complication.[24]

In a Distributed MILS system, the configuration system must consider any latency requirements specified for a policy architecture when that architecture is being deployed on the Distributed MILS Platform, as network latency is combined with scheduling latency. The schedules created for the various nodes of the D-MILS Platform must account for inter-node inter-subject communication, just as the schedule for one node would do, and the runtime schedulers of the nodes of the platform must be synchronized so that clock drift among the nodes will not invalidate the global scheduling decisions. To meet end-to-end latency constraints, time-critical applications may require the use of networking technology, such as Time-Sensitive Networking (TSN) or Time-Triggered Ethernet (TTE), that allows network communications to be scheduled in a manner similar to processor scheduling.

2.3.1.6 Periods Processing

Periods processing is the processing of information from various domains (e.g., classified or unclassified information*)* at different times on the same hardware resources. Under periods processing, the hardware resources must be purged of all information from one processing period before transitioning to the next when different domains are being serviced. A separation

[24][VF10] Velykis. Formal Modelling of Separation Kernel Components.

kernel implementation requires particular attention to the context switching between domains when the resources being used to process a domain are re-allocated to process another domain.

2.3.1.7 Minimal Interrupt Servicing

An interrupt can occur at any time, so an interrupt handler can execute at any time. So as not to affect the validity of schedules, and the timing objectives they were created to achieve, each interrupt handler must run quickly and resume execution of the interrupted code as soon as possible. For some embedded systems interrupt response time must be deterministic.

However, interrupt handlers have a lot of work to perform, and they must acknowledge the interrupt's receipt to the hardware. Because of these competing factors, the implementation of interrupt servicing by the separation kernel is a subject for aggressive optimization. Nodes with multiple processor cores may have a single core designated to handle interrupts that is either dedicated to this task or otherwise runs processes that are not time critical.

2.3.1.8 Minimal Synchronization Primitives, Timers and Watchdogs

A separation kernel must handle hardware traps such as timers and watchdogs and implement synchronization primitives to control interference and preserve correctness of legitimate communication. Requirements for timer management are determined by the scheduling constraints and use cases for the system. The resulting overhead should be contained by the proper implementation of scheduling mechanisms. Some applications, such as TCP protocol implementations, and thus all networked applications based on this protocol, use the timers and synchronization primitives in a predictable way. The information about the intended use of the system may be used for minimization of overheads and optimization of schedules.

2.3.1.9 Instrumentation

Embedded system components commonly record operational health and status data to support, recovery, post-operation analysis and debugging (*instrumentation*). If a single mechanism is used to manage both instrumentation and audit data, then the confidentiality and integrity of collected data must be protected as appropriate.[25]

2.3.2 Separation-Supporting Hardware

Many operating systems and separation kernels use hardware-assisted CPU and Memory Management Unit (MMU) virtualization to efficiently implement proper separation of resources, but they don’t have to. Hardware requirements for a MILS-based system depend on the MILS policy architecture and the external interfaces as determined by the purpose of the system and its functionality and performance requirements. Generally, spatial separation features rely on

[25] [SKP07] Protection Profile for Separation Kernels in Environments Requiring High Robustness (SKPP).

common hardware protection mechanisms such as MMUs and Input/Output MMUs (IOMMUs). Separation kernels use hardware timers to trigger interrupt-driven context switching and periods processing of processor resources for temporal separation.

These hardware elements play an indispensable role for an efficient and robust separation kernel implementation. Other hardware features may be useful for achieving specific additional properties of the separation kernel or MILS platform as a whole. Reliability mechanisms may help to mitigate hardware failures or to meet quality-of-service uptime requirements. In-depth defenses like trusted distribution, authenticated boot, OS code integrity checks, secure storage and similar measures help to thwart low-level attacks.[26]

2.3.2.1 CPU

CPUs execute code, and in MILS-based systems, they support spatial and temporal separation. This applies particularly to the interaction with register files, multi-level caches and system memory to which requirements of periods processing apply.

While MILS features may be implemented in varying ways, the support of separation at the level of CPU(s) or System-on-Chips (SoCs), facilitate easier implementation and more predictable system behavior. One of the technologies boosting the rise of MILS was hardware-assisted virtualization because it enables separation to be more effectively and efficiently achieved.

A key function of MILS hardware support is separation of privileges as reflected in the system policy architecture. A simple executive or an RTOS may provide only one privilege level. Thus, in effect there is only one domain. Traditional operating systems rely on two privilege levels, kernel and user, to separate applications from the kernel. If full processes are supported (rather than just multi-tasking or threads) then applications are separated from one another to the extent provided by the process model. Virtualization, which is an ongoing trend in embedded Commercial-of-the-Shelf (COTS) microprocessors and SOCs conceptually adds a new hardware privilege level, giving a conventional operating system, that would normally have unfettered control of the hardware, indirect or constrained access to the hardware. In this case, the operating system depends on another piece of software, called a *hypervisor* or *virtual machine monitor* (VMM), to manage the hardware and to perform periods processing that provides the operating system with a hardware view that corresponds to the machine without the added privilege level. This makes it possible to run multiple, possibly different, unmodified operating systems on a single processor, and to partition hardware resources on a SoC between operating systems.

[26] [BTL$^+$14] Blasum. EURO-MILS Whitepaper; [EURa] EURO-MILS project.

Processors with multiple cores can be supported and well utilized by a hypervisor, VMM or separation kernel, running each at different clock frequencies to lower overall power consumption or to increase performance and requiring fewer context switches among virtual machines. Multicore systems add complexity to the platform but can enhance domain separation by providing an added dimension to time partitioning and the control of illicit information flow.[27]

For multicore processors, the hypervisor is responsible for creating a virtualized environment for each core, configuring memory protection required for each domain, and loading and running the relevant software. The increasing deployment of the MILS-based systems on single-core processors, and the increasing commercial availability of multicore processors, has led to the convergence of these two technology trends, resulting in the requirement to extend the MILS architecture to exploit multicore performance security-critical systems.[28]

There are some significant challenges to using modern, state-of-the-art, and highly integrated COTS microprocessors as a base for special purpose, highly-assured systems, including:

- may not provide adequate visibility and debug features to reveal internal functionality,
- are less predictable with respect to timing due to the interaction of advanced features,
- may have programmable configuration capabilities available to application software and
- may share resources across multiple cores and devices.[29]

Some MILS applications require robust consideration of system failure, anomaly detection, correction and recovery. To do so, mitigations and protections at a level above the functional elements containing the microprocessor(s) or SoCs can be used to address the lack of design assurance for highly integrated, complex, nondeterministic hardware. The trade-offs between the complexity and attainable level of assurance for the platform and its efficiency and natural support of a multi-domain environment should be evaluated individually for each MILS-based application.

2.3.2.2 Memory Management Unit

A memory management unit (MMU) manages memory references. It translates virtual memory addresses to physical ones; it can control access to certain memory areas too. It is usually implemented as part of the central processing unit, but it can be a separate integrated circuit.

Whenever a process wants to access a virtual memory address, the MMU performs a translation to find the corresponding physical address in main memory. The translation lookaside buffer (TLB) in each CPU core stores most of the recent translations to speed up the memory access.

[27] It is not all good news, as multicore platforms do have low-level shared hardware resources.

[28] [Par16] Parkinson. Applying MILS to multicore avionics systems.

[29] [MLGM11] Mahapatra. Microprocessor Evaluations for Safety-Critical, Real-Time Applications.

However, whenever a TLB miss occurs, the MMU needs to walk the page tables (PTs) of the process (also stored in main memory) to perform the translation. To improve the performance of that MMU walk, the PTs may be cached in fast data caches much like process data is cached.

Virtualization implies an additional level of memory addressing, requiring independent memory mapping onto physical address space for every domain. The translation may be organized in two stages such as in the ARMv8 platform for which the translation regime translates a virtual address to an intermediate physical address (IPA), and then from that IPA to the physical address. Or it may employ hardware-assisted MMU virtualization, called rapid virtualization indexing (RVI) or nested page tables (NPT) in AMD processors and extended page tables (EPT) in Intel processors, provide hardware support to virtualize the MMU. Hardware-assisted MMU virtualization has an additional level of page tables that map guest physical memory to host physical memory addresses, eliminating the need for the hypervisor to maintain shadow page tables.

A similar solution needs to be implemented for MILS-based systems, as they require separation of resources including system memory into domains. The implementation may be supported by virtualization technology or developed from scratch. Memory mapping may be static or dynamic.

When implementing static memory-management, a separation kernel supports static entries in the MMU’s translation table, which do not change at runtime. If identical virtual addresses are referenced by different domains, a runtime mechanism indicates which partition is currently active and adjusts MMU translation entries. Static MMU configuration entails a static spatial separation of the memory resource without support of dynamic memory allocation and reallocation for domains. This arrangement is suitable for relatively small, static MILS systems, affording a reduction in complexity and thus assurance effort.

Dynamic memory-management allows reconfiguring the translation tables for domains. This does not necessarily require hardware support but might require additional processing cycles during switching between the domains. It is more difficult to ensure that domains are properly separated and resources are sanitized when using dynamic memory management. However, such an arrangement is necessary for larger and for dynamic MILS systems, and the additional assurance burden must be addressed.

There is a requirement to address threats that may exploit caching mechanisms. For example, in cache attacks, attacker-controlled code sharing the cache with a designated victim can leak confidential data by timing the execution of cache-accessing operations. Another kind of attack lures an external, trusted component into indirectly accessing the cache partition of the victim and mounts a so-called confused deputy side-channel attack. Recent research shows that the isolation enforced by existing defense techniques is imperfect and that generalizing such techniques to mitigate arbitrary cache attacks is more challenging than previously assumed.[30] Effective and efficient solutions to such issues necessarily entail hardware support and likely some sacrifices. Such solutions are unlikely to appear in commercial processors until the requirement to mitigate cache attacks is more widely shared.

2.3.2.3 IOMMU

System software may use hardware I/O memory management units (IOMMUs) to implement transparent, isolated access to virtual instances of I/O devices to one or more partitions. An IOMMU is intended to prevent arbitrary use of direct memory access (DMA) by certain hardware subsystems to access main system memory independently of the CPU. This may potentially violate spatial partitioning.

The IOMMU treats the address in a DMA as an I/O virtual address (IOVA) and maps it to a physical address using OS- or hypervisor-provided mappings, blocking the DMA if no mapping exists. Transient IOMMU mappings restrict device DMAs to only valid DMA targets. Systems that isolate drivers as untrusted out-of-kernel components apply this technique to protect code and data in the trusted kernel and in other components.

If a system's functionality demands using external DMA-capable devices in addition to those for which trusted subsystems are provided, IOMMUs protect the system memory from invalid direct memory access triggered by the device and so achieve spatial separation. The task of an IOMMU is similar to that of an MMU as both perform address translation, but the purpose is different: while an MMU is intended to increase the performance for address translations between virtual and physical addresses, an IOMMU is primarily deployed for memory-protection reasons.

IOMMU-based protection is provided in page granularity, and if I/O buffers reside in the same page with sensitive data, devices that are allowed to access a buffer gain access to this data too. IOMMU identifiers must be provided securely to prevent such attacks using DMA. One class of attacks abuses message signaled interrupts (MSIs) to trigger interrupts that do not belong to the device. Another class of attacks uses a vulnerability of peripheral component interconnect (PCI) to PCI express bridges, where the identifier is added by the bridge but not by the devices connected to the bus "behind" the bridge. IOMMUs may also be compromised making it possible to access one domain's memory from another. Some attacks are intended to bypass IOMMU protection. In addition, IOMMUs are not capable of withstanding timing attacks, like exhausting bandwidth, interrupt bombing or long uninterruptible bus transactions.[31]

[30] [vSGBR18] van Schaik. Malicious Management Unit: Why Stopping Cache Attacks in Software is Harder Than You Think.

[31] [BTL$^+$14] EURO-MILS Whitepaper.

2.3.3 MILS Platform Foundational Components

Early on in MILS the separation kernel was the star of the show, and nearly everyone was focused on the *separation kernel protection profile*[32] and the development and use of separation kernel products. When a MILS-based system needed to include devices such as keyboards, displays, network interfaces or mass storage, they were added to the application architecture *ad hoc*. Since MILS is supposed to be component-based, it was recognized that these should be standardized and an initiative was undertaken to specify protection profiles for the subsystems managing these devices, so independent efforts could develop and evaluate them separately.

At the same time there was a need to implement systems having higher level functions and policies, such as Multi-Level Security (MLS) by enabling the various components to be composed to create more complex systems. Unfortunately, the rather uncoordinated approach for the added device components did not provide a practical yet principled way forward beyond conceptual presentations. With the existing approach, new components did not compose in the ideal way envisioned by Rushby.[33] The success of MILS depended on the creation of a COTS marketplace for MILS components that can successfully interoperate securely with high assurance. This prompted a closer study of MILS component integration and modular platform specification that led to the current concept of the MILS Platform as a composition of foundational components, and spawned an effort to develop a MILS Integration Protection Profile[34] (later renamed MILS Platform Protection Profile (MPPP)[35]) as a framework for functional and assurance composition.

To transform the practice of development of trustworthy systems, it is necessary to offer techniques, tools, and technology that is widely applicable, based on standards and a compositional certification approach. The required standards need to include but go well beyond what is offered by the Common Criteria (CC), though it is attractive to leverage the CC as it is perhaps the most well-developed world-wide evaluation and certification infrastructure. The CC focuses on security functional and assurance requirements. To succeed MILS needs to specify

[32] [SKP07] SKPP.

[33] [Rus08] Rushby. Separation and Integration in MILS (The MILS Constitution).

[34] The MILS Integration Protection Profile (MIPP) concept and commentary was developed concurrently with a MILS Network System Protection profile as part of the Secure Interoperability for Real-time Embedded Systems (SIRES) and High Assurance Middleware for Embedded Systems (HAMES) projects, performed by Raytheon and SRI International for the Air Force Research Laboratory (AFRL) and the AF Cryptographic Modernization Program Office (CMPO). Further development of the MIPP was done as part of the Research Enabling MILS Development and Deployment (REMDaD) project performed by SRI International for AFRL and CMPO.

[35] Development of the MILS Platform Protection Profile and accompanying commentary was subsequently further developed by DeLong, MPPP [DeL13b] and Commentary [DeL13a].

functional and assurance requirements for the full range of key system characteristics. To achieve MILS' vision of a commercial marketplace of high-assurance components and associated products and services, it is essential that MILS standards extend to architectures, components, interfaces, protocols, interoperability considerations. For this reason, nowadays we tend to talk about *specifications* or *models* rather than protection profiles, which are identified with the CC. Specifications include security requirements. Accordingly, as a continuing evolution of the MPPP, the Mils™[36] Working Group of The Open Group has planned a set of Mils™ standards to include a future Mils™ Platform Specification, and specifications for each of the foundational components and the Mils™ Platform API.

The role of the MILS Platform Specification is to create a framework for the specifications of the individual components and how they compose. The MILS foundational components are identified and their roles circumscribed by the Commentary on the MILS Platform Protection Profile[37]. Properties that the MILS Platform must exhibit, both functional and non-functional, are to be derived from the functions and properties of the foundational components and the manner of their composition. As the Commentary prescribes, the specifications of the foundational components must conform to the MILS Platform Specification. Foundational component implementations may be developed and evaluated separately and later composed to create MILS Platform instances with confidence.

The components of the MILS Platform may be thought of as a modular operating system. A high-assurance general-purpose operating system could be constructed from a separation kernel composed with a collection of suitably defined foundational components and a stack of other trusted software components running on top. This may be the way that someday the holy grail of a high-assurance GPOS will be achieved with MILS.

Each MILS foundational component creates exported resources of a certain type from other more primitive resources. These exported resource types combine to form the vocabulary from which policy architectures are constructed. Most of the foundational components own a portion of the physical computing resources and create exported resources while managing the physical resources so that they are safely and securely shared. They permit information to flow to or from a resource only when configured to do so, otherwise the resources are isolated.

[36] The Open Group has trademarked Mils™ to differentiate its family of consistent platform, component, and API standards for Mils™ from the diverse "reservoir" of MILS concepts and terms found in the broader commercial, academic and research MILS community.

[37] Preliminary MPPP [DeL13b] and Commentary [DeL13a].

2.3.3.1 MILS Separation Kernel (MSK)

The separation kernel is the distinguished foundational component of the composed MILS Platform since it is necessary to execute the software portions of the other foundational components and to provide the separation kernel exported resources they comprise. The exported resources of the separation kernel are presented as a raw, distinct and possibly proprietary interface that may be then normalized to a standard API (for example, by the Mils™ Platform API considered below). Other foundational components are expected to also be high-assurance and are likely to use the separation kernel through a high-assurance implementation of the platform API, allowing them to share assurance burden with the API implementation and to gain portability among proprietary separation kernels.

For a separation kernel to be suitable for use in conjunction with the other MILS foundational components to create the MILS Platform it must conform to the separation kernel role and requirements in the MILS Platform Specification.

2.3.3.2 MILS Network System (MNS)

Initially viewed only as an operating system kernel, the separation kernel concept has been extended to complex networking and heterogeneous environments. The Distributed MILS (D-MILS) platform extends the MILS platform to include distributed system configuration features and a network subsystem. The D-MILS deployment platform enables an application architecture to span multiple computer systems seamlessly, with scalable deterministic operation.

A D-MILS node relies on its separation kernel and a MILS networking subsystem to support its integration into the whole platform. Networking and connectivity in D-MILS are not handled at the application-level but by the platform. The security, safety and other characteristics of the MILS networking subsystem are as important as those of the separation kernel. The requirements for the networking components of the D-MILS platform include taking the security and safety concerns into consideration from the beginning of the platform and component design.

Today’s IoT systems evolve to integrate ever more applications, sometimes with largely varying requirements on the underlying communication subsystem. Some applications may require short latencies, and lossless transmission guarantees, while others do not demand any transmission guarantees at all. These differing requirements may lead to the implementation of several dedicated communication networks in a single system serving the respective application classes.

A better approach is a communication platform that uses a single, physical network for all applications in a system. This can be achieved by traffic shaping and policing algorithms that ensure partitioning of the network and minimizing the effect of messages from applications on each other. For messages of highest time and safety criticality, the time-triggered communication paradigm[38] can eliminate the effect of other messages in the network completely achieving strict time partitioning. The realization developed for the D-MILS Project employs a hardware-based Time-Triggered Ethernet (TTE) “backplane” in its MILS Networking Subsystem (MNS).[39] Ethernet packets in this subsystem are sent over the network at scheduled times and take precedence over all other traffic types. The occurrence, temporal delay and precision of time-triggered messages are predefined and guaranteed. The realization of the Distributed MILS paradigm for the CITADEL Project employs Time Sensitive Networking (TSN) in its MNS. This is a boon for future adoption of Distributed MILS since the TSN technology has been made an industry standard, in part through the efforts of the CITADEL Project.

2.3.3.3 MILS Console System (MCS)

The primitive resources managed by the MILS Console System are human interface devices of all kinds, and it uses resources provided by the MSK to export abstractions such as audio input/output, display output, graphics, windows, trusted display regions, cursors, positions, keystrokes, gestures, trusted path, etc. Examples of the MCS have been implemented for specific use cases as discussed in the Frequentis Voice Service use case in Section 4.2.3 and the Trusted Smart Phone use case in Section 4.2.4.

2.3.3.4 MILS File System (MFS)

The primitive resources managed by the MILS File System are mass storage devices (or virtual mass storage devices), and it uses resources provided by the MSK to export abstractions such as volumes, files, directories and file systems. An example of the MFS is found in the Trusted Smart Phone use case.

2.3.3.5 MILS Extended Attributes System (MEAS)

The MILS Extended Attributes System uses resources provided by the MSK and the MFS to bind an arbitrary set of attributes with the exported resources of any MILS Platform foundational component. The binding has high integrity and may be persistent for persistent resources. The attributes may be used for system or application purposes, but is typically used for adjudication of access control and other security policies that pertain to the associated resources. A general purpose MEAS has not yet been implemented but the demand for flexible high-level access control policies and enforcement mechanisms in IIoT and other complex distributed systems is spurring research on unified information flow control and access control policies within the MILS system context.

[38] [KG93] Kopetz. TTP - A time-triggered protocol for fault-tolerant real-time systems.

[39] [D-M13] Requirements for Distributed MILS technology.

2.3.3.6 MILS Audit System (MAS)

The MILS Audit System uses resources provided by the MSK and the MFS to create an audit system with audit generation, audit configuration, audit management and high-integrity and persistent audit storage.

2.3.3.7 MILS Platform Interface

A higher layer of the MILS Platform, above the resource exporting foundational components, may include other services, but even these services should be standardized in a modular fashion. Such services include file interaction, network interaction, memory management, synchronization, cryptographic operations, threading and mathematical functions. These should be specified, implemented and assured once-and-for-all over all implementations of the MILS platform components. A set of such functions is currently defined by the draft Mils™ Platform API Specification[40] developed by a collaborative MILS working group within the Real Time and Embedded Systems Forum of The Open Group.

The MILS Platform API as a whole provides a unified and complete interface to the MILS platform and the modular packages of the API correspond to and build upon the exported resources of the various MILS platform foundational components, while others provide the additional functions itemized above. The modularity of the API follows the modularity of the MILS Platform itself. Just as the MILS Platform requires, at a minimum, a separation kernel foundational component, so too the Mils™ Platform API requires the base package. As other optional foundational components are added to the MILS Platform the corresponding packages of the API are added. Figure 5 illustrates the correspondence of the Mils™ Platform API packages to the MILS Platform foundational components. For this illustration the MILS Platform is taken to be composed of a MILS Separation Kernel (SK), a MILS Console System (MCS), a MILS File System (MFS), and a MILS Network System (MNS).

[40] The Open Group has trademarked the name "Mils" as a variant of the term MILS in order to identify its particular family of Mils™ standards.

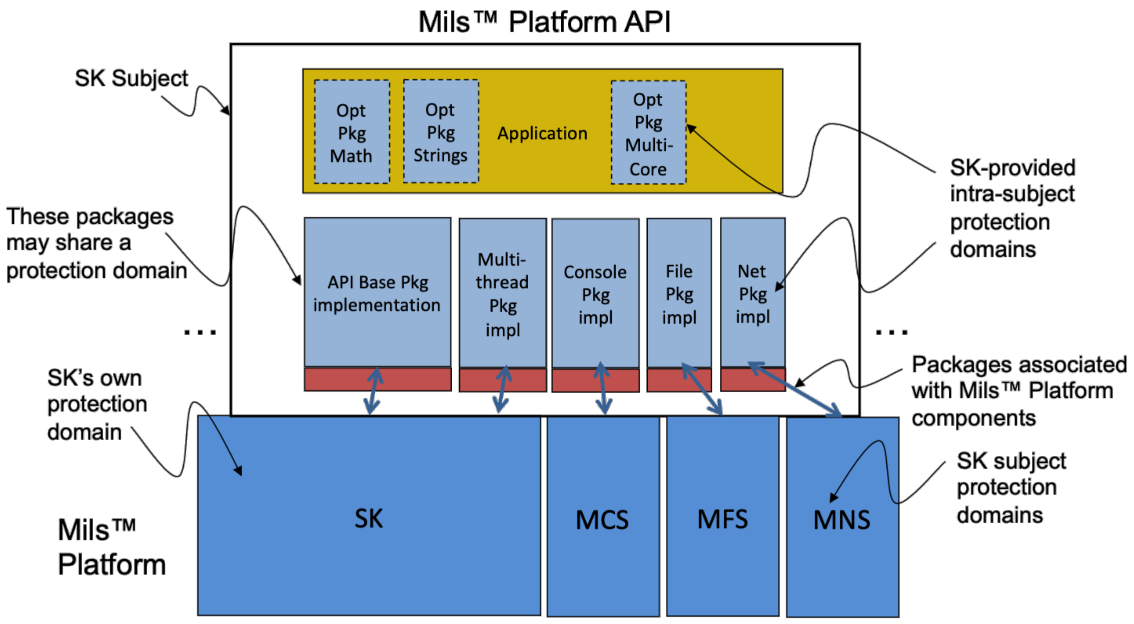

Mils™ Platform API
SK Subject
Opt Pkg Math
Opt Pkg Strings
Application
Opt Pkg Multi-Core
SK-provided intra-subject protection domains
These packages may share a protection domain
API Base Pkg implementation
Multi-thread Pkg impl
Console Pkg impl
File Pkg impl
Net Pkg impl
SK's own protection domain
Packages associated with Mils™ Platform components
SK subject protection domains
Mils™ Platform
SK
MCS
MFS
MNS

Figure 5: MILS Platform components and corresponding API packages

As it is intended primarily as a vehicle for the implementation of high-assurance applications, a MILS API itself should be implemented with high assurance. Needless to say, it may also be used for applications that do not require high assurance, or that are not yet implemented with high assurance, but may evolve into high-assurance applications. It may also be used in any situation where a small footprint and lightweight runtime environment is preferable to a heavyweight OS kernel. As a MILS API raises the level of abstraction, effort invested in its assurance reduces the assurance burden for software implemented on it while also bestowing enhanced portability to that software.

The Mils™ Platform API is currently defined as a POSIX profile that attempts to strike a balance between simplicity and the rich APIs expected by application developers. In defining a modular API, the goal is to adhere to the MILS principle of modularity of the platform, to include only the portions of the API that correspond to foundational components present in a particular instance of the MILS platform, to provide services suitable for building trusted subjects and to simplify the verification of the API implementation and independent compositional verification the applications written using the API.

3 MILS Assurance and Trustworthiness for IIoT

3.1 Assurance

As stated in the Industrial Internet Security Framework, trust flows down from users to all parts of a system in response to being earned through operational experience and a presentation of sufficient evidence to instill confidence that a system will perform as expected with respect to all the key system characteristics.

3.1.1 Assurance for the MILS Platform and a MILS Policy Architecture

In the MILS approach, the assurance of a deployed policy architecture depends upon the assurance of a number of things, including the MILS Platform on which it is deployed. Assurance of the MILS Platform is dependent in turn upon the assurance of the individual foundational components and upon the assurance of their composition. Compositional assurance allows these component assurances to be confirmed at different times, and the assurances to be later combined into a system assurance, perhaps in a parametrized fashion, within a complete assurance case. Aspects of the MILS Platform assurance case can be constructed once-and-for-all, such as the foundational component composition argument justified by the MILS Platform Specification. The specific assurance case for each foundational component instance demonstrates that it meets the requirements levied on it by the MILS Platform Specification. The composition of foundational component instances into a MILS Platform instance is a once-and-for-all construct valid for all application policy architecture instances deployed on the platform. A schema for a compositional MLS Platform assurance case is illustrated in Figure 6. Claims for the MILS Platform (MP Claims) are justified by an assurance argument having the MP Claims as its conclusion derived from a combination of inference rules and supporting evidence. An item of evidence employed in such an assurance case may, in turn, be the top-level claim from another assurance case. In the figure such is the case for MILS Separation Kernel Claims, MILS Network System Claims, and MILS Console System Claims. The top-level assurance case is devised to be provisionally valid provided that the assumed evidence items are presented. Claims from subordinate assurance cases are substituted for certain evidential items in the platform assurance argument. When the claims are independently demonstrated as guarantees the MILS Platform assurance case is valid and guarantees the MP Claims.

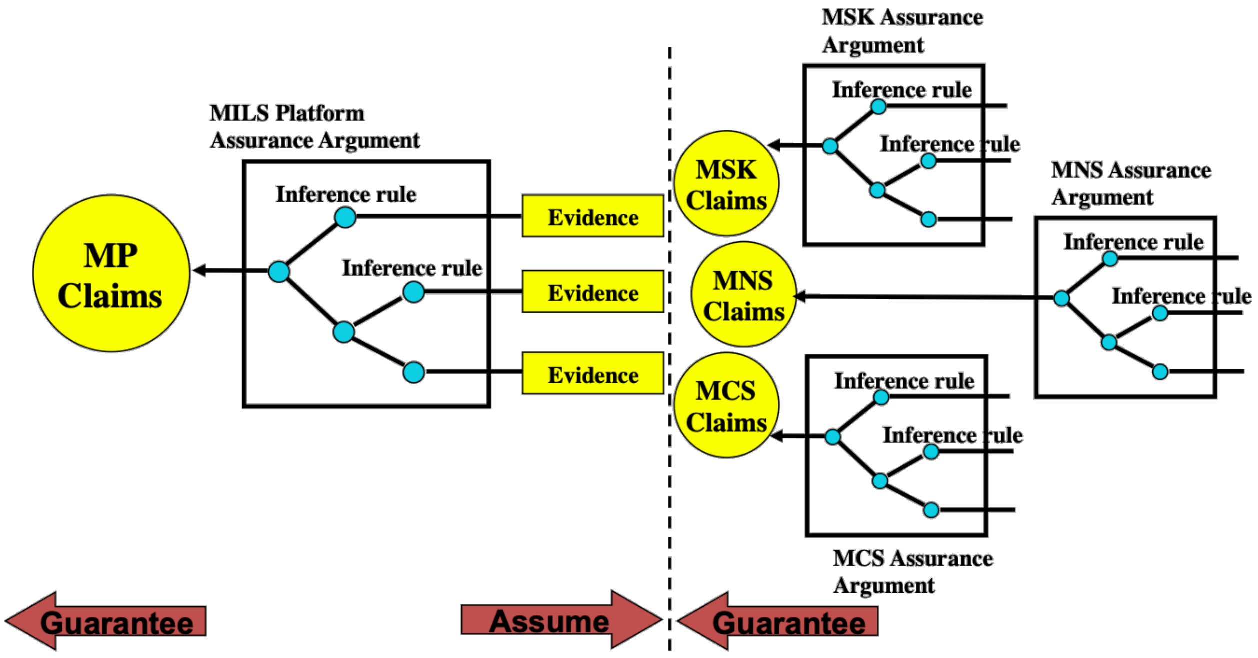


Figure 6: Composition of assurance cases to establish MILS Platform Claims

3.1.2 Compositional Assurance

We have seen that there are many aspects to the assurance of a system constructed and analyzed according to the MILS architectural approach. Automation of the computation of compositional results and the construction and maintenance of the assurance cases for components and systems is a vital prerequisite for the practical and economic viability of the approach. Fortunately, the theory and tools[41] for compositional assurance of static, distributed and dynamic MILS systems are now in place for the use cases considered until now. It is to be expected that these may need further refinements and extensions as MILS takes on new application domains with new requirements and the supporting theory evolves.

The key system characteristics in which we are interested, including security, safety, privacy, resilience and reliability, are *emergent properties* of a system. These properties arise through composition of the system's components and depend on their properties and the manner of their composition.

In MILS we distinguish between three composition concepts:

- Composability—A condition whereby components may be composed without affecting their individual properties. This is a crucial property provided by the isolation service of a

[41] [KAN$^+$18] Koelemeijer. A Model-based Approach to Certification of Adaptive MILS.

separation kernel (or the MILS Platform). Components A and B can both be run in isolation with neither one impacting the operation of the other.

- Compositionality—A condition whereby components may be composed in a way to create new properties of the composition from their individual properties. Compositionality represents the ability to connect two components in such a way that their behaviors combine constructively to create a new behavior.
- Additivity (or Additive Compositionality)—A specific kind of compositionality whereby the “addition” of components’ individual but distinct properties are preserved in the composition along with their common properties. This concept is required to ensure composition of the MILS Platform from the MILS foundational components.

Reasoning about composition may seem complicated, and it has long been so regarded. In the early years of MILS reasoning about composition in MILS policy architectures was only an aspiration, but today it is a reality. MILS research projects have adopted and developed tools that make such reasoning possible for MILS systems in the standalone, distributed, static and dynamic cases. [42]

When reasoning about the composition of components in a MILS policy architecture, to prove that the composition provides the needed system properties, a crucial assumption is being made: that the refinement of the policy architecture as a concrete system will also exhibit the properties proven of the model. Though it is intuitively appealing, this assumption (the *refinement assumption*) cannot be taken for granted. It is necessary to analyze this assumption and confirm its validity.

3.1.3 Assurance Case

An assurance case structures the reasoning and supporting evidence to justify claims made for a system. There are multiple legs to the top-level assurance case for a MILS system. These include assurance cases for the assumptions as well as an assurance case for the claim that the policy architecture delivers its claimed properties that are needed to meet the system requirements. Essential legs of a MILS assurance case are the composition, implementation, deployment and platform arguments.

Because the purpose of MILS is to create assured systems, assurance cases have been an integral part of the MILS discussion since the mid-2000s. In the 2010s both the Distributed MILS (D-MILS) and CITADEL Adaptive MILS projects pursued policy architecture-driven automated assurance

[42] [D-M14, D-M15, CIT18a, CIT18b] Compositional reasoning tools developed for the D-MILS and CITADEL projects, for example, allow reasoning over properties expressed as temporal logic formulae for modal and parametrized policy architectures expressed in a specification language based on an extended subset of AADL.

case support as a major activity of the project. The refinement assumption was regarded to be valid because van der Meyden's result[43] was already known at the time of the start of D-MILS. In both of these projects the composition, implementation, deployment and platform arguments are included as sub-arguments of the assurance cases. However, the implementation argument was generally not elaborated and the platform argument was only shallowly elaborated because these projects treated component verification as a solved problem, and focused instead on compositional verification, what had heretofore been unsolved.

3.2 Trust by design

The prospect of "trustworthiness by design" has great intuitive appeal though it does not have a universally accepted definition or approach.[44] MILS is concerned with methodology, techniques and tools that can be applied to create an architecture-centric technical design that will assure that its implementation will be trustworthy for application-dependent functionality and essential system characteristics. As such MILS qualifies as an approach to, and provides a framework within which to pursue, trustworthiness by design. It allows a variety of techniques and tools to be applied to produce evidence in support of claims for the needed properties and provides a unifying framework within which such evidence may contribute.

The effort and expense needed to attain trustworthiness were first justified only in application domains where security or safety characteristics were of paramount importance because of the severity of the consequences of a failure to exhibit essential characteristics. Each such domain strived to develop a design approach by which the requisite trustworthiness could be achieved. "Security by design" and "safety by design" emerged in such domains to provide and more systematic and cost-effective way of attaining trustworthiness according to the requirements of their corresponding certification authorities.

"By design" approaches can be applied to most trustworthiness characteristics. For example:

- Security-by-design is the concept of applying architecture, design and software development methods to minimize the security risks connected to information leakage, information integrity violation and disruption of authorized access to data early in the design process and implementation.[45]

[43] [CvdM12] Chong. Using Architecture to Reason About Information Security; and van der Meyden author version, Architectural Refinement and Notions of Intransitive Noninterference.

[44] In some circles "trust by design" or "trustworthiness by design" is used in a way that is quite different to that which we consider, instead referring to organizational reputation and trust.

[45] [OWA] Security by Design Principles.

- Prevention-by-design is a shift in approach for on-the-job safety. It involves evaluating potential risks associated with processes, structures, equipment and tools and accounts for the construction, maintenance, decommissioning and disposal of waste material.[46]
- Privacy-by-design was initially developed and formalized in a joint report on *privacy-enhancing technologies*.[47] Privacy must be accounted for throughout the whole engineering lifecycle process. It is an example of value-sensitive design, i.e., to take human values into account in a well-defined manner throughout the entire process.

All approaches to making a system trustworthy "by design" have one thing in common. They recommend best practices at the architecture and design phase to increase the cost-effectiveness of enhancements to trustworthiness during the entire lifecycle, in particular integration activities and system usage. In some cases, properly implemented design may even eliminate the necessity of such enhancements. By identifying an issue, it is possible to make a design-time decision about how to eliminate or withstand it.

With time it has been shown that common methods can replace some of the bespoke techniques developed for specific domains. MILS has taken advantage of this potential commonality by pursuing a methodology, modeling technique, and verification tools that can support a broad range of properties, and at the same time by relying on an evidential tool bus that permits other new tools and techniques to be used for the generation of supporting evidence.

A "trust-by-design" approach does not replace "trust-by-assurance". Systems that are "trustworthy-by-design" still need assurance activities but they are simpler to implement and give more guarantees when a "by design" approach was employed.

The claim that a system is secure- or safe-by-design means that it has properties such as:

- Separation of functions among the security domains to create simpler, local security policies and strengthen assurance guarantees for their implementation.
- Separation of responsibilities among the security domains to address arising issues with a minimal involvement of a human factor.
- Isolation of resources and creation of domain partitions to enforce access control on a default-deny basis and prevent data leakage and data integrity violation.
- Control of communications between domains allowing for mixing applications and components with different trustworthiness requirements.
- Clear loosely-coupled architecture of trusted computing base allowing for strengthening of the security, safety and other assurance guarantees and scaling from local systems to networked applications.

[46] [SRO+08] Schulte. *National Prevention through Design (PtD) Initiative*.

[47] [Cav] Cavoukian. Privacy by design. The 7 Foundational Principles.

- Fault isolation to increase system reliability and support its resilient execution.
- A design that provides diagnosability of the source of faults and failures, analysis of failure modes, and the ability to monitor for those.
- Architectural support for recovery from failures and attacks.
- Platform-based approach to support system designers with templates, guidelines and good practices at all stages of system development criticalities.

MILS significantly reduces the possibility of undermining a system's policy architecture during detailed design, implementation or maintenance by specifying the architecture separately from the design and implementation of the architectural components, and by strongly enforcing the architecture, both during development and at runtime. This approach has the beneficial side effect of protecting the integrity of the architectural components, so that the architect's assumptions on the behavior and properties of each component cannot be undermined by another component, which is essential to the support of trustworthiness aspects by design.

3.3 TRUST BY ASSURANCE

Assurance is gained by the collection and analysis of evidence that supports the design, construction, deployment and test of the system and its operation. The evidence must support the claim that the correct mixture of innate system capabilities and compensating controls has been put in place to mitigate risks.[48]

MILS takes a compositional approach to assurance, evaluation, and certification. The evaluation of a system composed of previously evaluated components can be substantially based on those previous evaluations, without requiring the details of all of the components to be reconsidered.

A MILS policy architecture is a collection of domains interacting with each other. The absence of an interaction is as important to the architecture as the presence of one. Some domains, such as security services may have strict behavioral requirements specified by the architect because these behaviors are necessary for the policy architecture to fulfill its purpose. Such behavioral requirements are known as *local policies*, and the implementations of such components are trusted to enforce the specified local policies. Local policies may relate to any kind of local action that the architect deems important. Examples include, that a storage component must transparently incorporate the backup feature, that the cleartext received at the input of a cryptographic component is never passed unencrypted to the output, and that an access control decision mechanism renders the correct decision for given parameter values and a given access control policy. Some trusted components, such as the access control adjudicator, may be

[48] [IIS16] Industrial Internet of Things, Volume G4: Security Framework.

recognizable as conventional security enforcement mechanisms, but all trusted components enforce some policy, and are treated in MILS as policy-enforcing components. An untrusted component is not required to enforce a local policy; thus, it does not require assurance. [49]

Assurance for a MILS system involves four steps:

- assurance for individual resource-sharing components, such as separation kernels, partitioned file systems, and partitioning communications systems that deliver the required guarantee of separation (reasoning about noninterference) among exported resources created from managed resources,
- assurance that individual trusted components enforce their specified local policies,
- assurance that the individual resource-sharing components compose additively to enforce the policy architecture (reasoning about policy architecture) and
- assurance that the individual trusted components, in the context of the policy architecture, compose to enforce the required overall system policy (compositional verification).

Systems assurance is the process of building clear, comprehensive, well-defined and defensible arguments to justify claims for the safety and security properties of systems. Certain claims are supported through reasoning expressed by explicit annotated links between claims, where one or more claims (called sub-claims) combine to provide inferential support to a larger claim. Certain associations (recorded as assertions) between claims and sub-claims can require supporting arguments of their own (e.g., justification of an asserted inference). Claims are propositions that are expressed by statements that may be informal, such natural language, or formal, such as a semantics-based logic. The degree of precision in formulation of the claims may contribute to the comprehensiveness of an assurance case.[50]

3.3.1 Reasoning about Noninterference

Two domains may communicate according to an information flow policy determined by the policy architecture. Interference implies any communication with or influence on a domain that is not explicitly authorized by the policy architecture for this domain. An example of such interference is improper influence on the execution of a component, or invalid modification of a component's state (e.g. using a bypass via a directly mapped device).

Informally, noninterference between domains means that the execution of one domain does not affect the execution of another domain. It demands that each domain's complete internal state is well-defined and determined at all times independent of the processing status and condition

[49] [BDRS08] MILS Component Integration.

[50] [Obj18] OMG's Structured Assurance Case Metamodel 2.0 (SACM 2.0).

of the other domain. Noninterference does not presume the total absence of interactions between domains but the absence of hidden channels and unspecified information flows.

To demonstrate noninterference, evidence connected to the lifecycle is needed.

- The separation kernel and other resource-sharing components provide domain isolation and information flow control, supported at the hardware level, thereby increasing assurance of the absence of hidden channels and unwanted information flow.
- All interfaces between the domains are clearly defined and completely and accurately described.
- End-to-end testing of the components and the integrated platform relies on the analysis and assessment of the appropriately defined interfaces between the domains and guarantees that the existing communication mechanisms cannot be misused.
- In contrast, vulnerability assessment and penetration testing techniques may consider defined interfaces and any other ways that information may flow among domains.

The MILS approach specifies the architecture separately from the design and implementation of the architectural components. This makes the *a priori* evaluation of the noninterference property necessary. The *a priori* method transfers evaluation efforts from the last steps of system development (integration testing, vulnerability analysis) to efforts on separate testing and vulnerability assessment of the components with additional focus on their noninterference properties. For an initial certification of a composed system the *a priori* non-interfering evaluation methodology will not reduce the evaluation efforts in total. However, it enables reusability of certified noninterfering operational components for subsequent non-interfering composed evaluations. These components may be composed in newer versions of the composed system or in a new composed system having a different policy architecture. [51]

3.3.2 Reasoning about Static and Dynamic MILS Policy Architectures

Building on the noninterference property proven for the MILS platform, the evaluator considers the policy architecture as a base for reasoning about system behavior.

In a static MILS-based system the configuration for the exported resources of the separation kernel and other MILS resource-sharing foundational components is finalized before initialization of the MILS application. After initialization there is no creation or destruction of exported resources and no changes in the information flow policy. The situations in which static configuration of a MILS-based system is adequate are typically simple applications of a small number of MILS components provided through a short supply chain. This approach has been

[51] [FSW+15] Furgel. Non-Interfering Composed Evaluation (EURO-MILS).

applied to safety-critical real-time operating systems and to security-critical systems needing the highest levels of assurance.

The need for adaptability of IoT systems in safety and security critical environments creates an additional requirement for development and certification procedures for systems that may dynamically change configuration during execution. Dynamic MILS provides low-level mechanisms to reconfigure the MILS platform dynamically, including creation and destruction of exported resources and changes to permissible information flows, and by interposing above these mechanisms a configuration change monitor to enforce a set of configuration change constraints that comprise the configuration change policy. The monitor mediates configuration change requests performed by a configuration change agent, deciding whether to permit them according to the configuration change policy.[52]

The constraints embodying the configuration change policy may be examined and approved during the system certification process, making configuration change a part of the one-time certification, or may be determined while the system is in operation as part of a just-in-time certification approach. The configuration change monitor is able to invoke pluggable decision procedures to determine whether to permit pending configuration changes.

Reasoning about Dynamic MILS policy architecture systems is supported by a system adaptation framework that includes a certification assurance subsystem that maintains an up-to-date, consistent, and always-available assurance case as an embodiment of the presentation of claims, arguments and the supporting evidence. The main challenge for the certification assurance subsystem relates to the proper creation and maintenance of the assurance case. Techniques and mechanisms for generating and managing the evidence needed for certification are also needed. These include model checkers, predicate abstractors, decision procedures, constraint solvers and other tools. They may introduce assumptions and constraints into algorithms and their implementation to maintain the practical solvability of the verification tasks. The task of supporting these mechanisms is to make the constraints and assumptions consistent to each other and with a goal of verification.

The evidential tool bus (ETB) is an engine enabling the verification tools to work in tight collaboration. The ETB, introduced by Rushby,[53] provides a framework for propagating system-level and inter-component constraints, invoking tools appropriate to the need, enforcing standards, checking consistency and providing automated review of evidence as a proxy for certification experts.

[52] The configuration change monitor is effectively a reference monitor for the configuration state. It has been modeled and implemented in CITADEL as a configuration transition system.

[53] [Rus05] Rushby. An Evidential Tool Bus.

The essential difference for reasoning about static and dynamic policy architectures is the presence of the ETB or similar component in the MILS platform architecture that provides automated support for continuous assurance case generation and evidence checking. Two major usages for this component are:

- *a priori* construction, when the assurance case and evidence is constructed for a future system configuration, usually before an adaptation or reconfiguration step is applied on the system and
- *just-in-time* construction, when the assurance case and evidence is re-constructed for the new configurations of the system.

In both cases, the evidence covers two orthogonal aspects. The first is the correct operation of the system in some stable configuration. The second is the correct reconfiguration, for example, ensuring that the system transitions only to configurations that are valid. Of course, we also ensure that the system behaves correctly during reconfiguration and does not violate security and safety constraints.

An example of the ETB implementation for MILS Policy Architecture assurance is the Adaptive MILS ETB (AM-ETB) prototype developed in the context of the CITADEL project.[54] From an operational perspective, AM-ETB includes a workflow sequencing system that manages assurance case and evidence generation as part of a process that results in an up-to-date assurance case. Steps of this process are carried out by agents for tools that perform reasoning or verification tasks. Results of the process establish higher-level goals through a chain of reasoning expressed by assurance case patterns. The AM-ETB gathers the evidence from the tools into a database and records the logic of the combination of the evidence to support the assurance case claims.

Assurance case patterns allow the general structure of frequently used argumentation and evidence to be expressed independently from the specific details of any particular assurance case through abstraction and parameterization. The patterns can then be instantiated and composed for the target system by using relevant information.

Figure 7 shows the overall architecture of AM-ETB comprising a core workflow engine, a database including assurance case patterns and evidence, and interface agents to run external analysis and verification tools.

[54] [KAN+18] A Model-based Approach to Certification of Adaptive MILS.

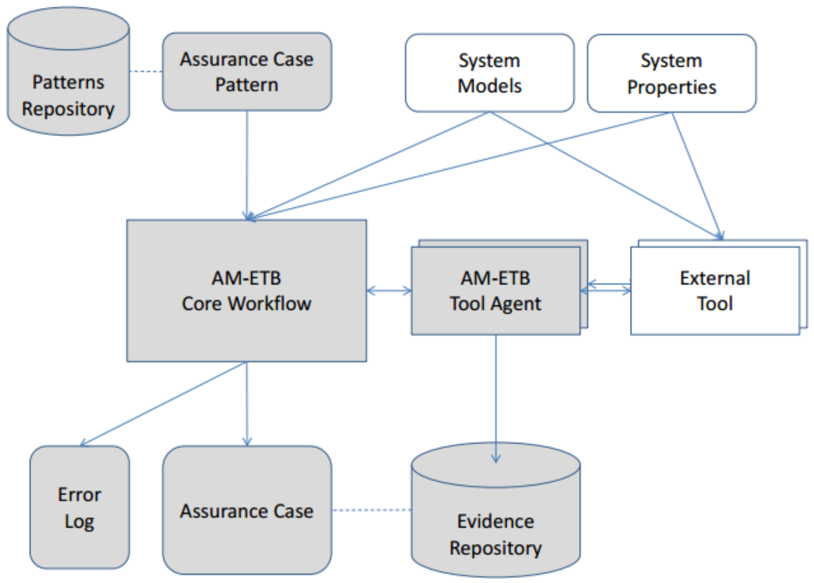


Figure 7: The architecture of AM-ETB, an ETB implementation for the Adaptive MILS Framework

3.3.3 Compositional Verification for MILS-based systems

Development of the assurance case both for static and dynamic MILS Policy Architecture seeks to minimize the cost and effort associated with assurance, while ensuring that when required, the highest levels of assurance can be demonstrated. Another objective is the support of compositionality of independently developed components, because there are many features that will be common to MILS-based systems, in particular those achievable by generic components at the level of the MILS platform (e.g., services and middleware). It is expected that many common application-level components will be created within a MILS ecosystem to leverage development efforts and create business opportunities.

Compositional methods in verification have been developed to cope with the state-space explosion problem. These methods attempt to break monolithic verification problems into smaller sub-problems by exploiting either the structure of the system or of the property. Compositional reasoning can be used in different ways, for example for deductive verification, assume-guarantee reasoning, contract-based verification, compositional generation.[55]

While an assurance case is the embodiment of the claims and supporting evidence for them, assurance case patterns enable reuse and the effective composition of assurance cases along

[55] [ARB+14] Compositional Invariant Generation for Timed Systems.

with the underlying argumentation supporting goals.[56] This can be successfully applied to support compositionality and independently developed components in MILS-based systems.[57] A detailed and confirmed assurance case for a component can be provided by the component's developer, or if the details are to be retained as intellectual property, the assurance case can be evaluated and its claims validated by an independent third party evaluator for use in other assurance cases and assurance case patterns. This presents new business opportunities for existing evaluation facilities such as Common Criteria Testing Laboratories.

An assurance case uses a structured set of arguments and a corresponding body of evidence to demonstrate that a system satisfies specific claims in respect to its security and safety properties. The construction of assurance cases also may rely on a set of *assurance-case argument patterns* to enhance the flexibility of the validation and certification of MILS-based systems. The argument patterns specify the requirements to instantiate the claims, and the evidence to support these claims for safety- or security-critical components. When constructing an assurance case for a system, the argument patterns are instantiated with information concerning the design, development, analysis and verification of the system.

The MILS Platform is designed as a set of components, grouped according to their functional purpose (separation-supporting components, services, middleware, operational components, monitoring components). These groupings usually contain similar components even for different MILS-based systems. Architectural patterns may be established for these groups of components and compositions thereof. A composition argument refers to a group of components and an argument about their composition. The composition argument for a MILS-based system can be created by instantiating appropriate argument patterns.

To verify that the components in a MILS-based system satisfy their local policies, evidence is required to be gathered and assessed. Evidence leaf nodes are therefore included in an argument pattern. Additional evidence nodes can be incorporated in the assurance case by adding them to the appropriate arguments when instantiating the respective argument patterns. Moreover, components may have additional properties such as person, organization, artifact, technique and trusted component arguments associated with them, which are also capable of providing evidence to support the claims made in the composed assurance case.[58]

[56] [HCAK11] Using a Software Safety Argument Pattern Catalogue: Two Case Studies.

[57] [HKH15] Richard Hawkins, Tim Kelly, Ibrahim Habli. Developing Assurance Cases for D-MILS. Systems. MILS Workshop 2015, Co-located with the HiPEAC Conference, Amsterdam, 2015.

[58] [KAN$^+$18] A Model-based Approach to Certification of Adaptive MILS.

3.4 Assurable Kernel and MILS Platform Components

A security kernel as a reference validation mechanism must be "assurable". To be able to verify a reference validation mechanism that implements the reference monitor[59] it must be "small enough to be subject to analysis and test, the completeness of which can be assured." Some interpret this as "as small as possible" or by some arbitrary size limit in lines of software code, but a well-performing full-featured security kernel may be fairly complex, which makes an arbitrarily small size requirement unattainable in practice. What is "small enough" depends on the current state of technology available for "analysis and test" that is sufficiently rigorous and systematic that its completeness can be convincingly argued. Such capabilities are not constant but can change with time, technology and available resources.

The *security kernel* is all the hardware and low-level software components for interception and control of operations that realize the reference monitor abstraction.[60] It implements the mechanisms for security policy definition and configuration and has interfaces sufficient for any usage scenario. It may have links to other system components and dependencies with external mechanisms. The reference monitor may be a part of the system kernel, be identical with it or encompass it.

A security kernel along with other trusted components are parts of the Trusted Computing Base (TCB), the totality of protection mechanisms within a computer system—including hardware, firmware, and software—the combination of which is responsible for enforcing a unified security policy over a product or system.[61] As such the entire TCB is a reference validation mechanism for the unified security policy and should meet the assurability requirement.

4 MILS Evolution, Examples and Case Studies

4.1 MILS Evolution and Key Directions for the Future

4.1.1 A Whirlwind Tour of MILS 1980-Present

Inspired by the separation kernel papers of SRI International's John Rushby in the early 1980s, the MILS initiative of the early 2000s developed from the integrated modular avionics (IMA) design approach. IMA proposes an integrated architecture with application software that is portable across an assembly of common hardware modules. An IMA architecture imposes

[59] [And72]

[60] [AGS83] S. A. Ames, M. Gasser, and R. R. Schell. Security kernel design and implementation: An introduction. IEEE Computer, 16(7):14–22, 1983.

[61] [TCS85] Department of Defense Trusted Computer System Evaluation Criteria. In the glossary under entry Trusted Computing Base (TCB).

multiple requirements on the underlying operating system. The resulting approach was adopted by NSA and the Air Force Research Laboratory as advances in microprocessor technology made it a more realistic venture. MILS continues to evolve and extend over technology areas requiring highly assured security, safety and resilience.

Advances in MILS methodology and technology, as well as the interests of the involved communities, enable us to divide the history of MILS up to the present into four eras.

4.1.1.1 Pre-MILS Era 1980 ~ 1999

John Rushby conducted a study of numerous secure operating system research projects underway circa 1980, including government funded R&D of formally specified and verified security kernel-based operating systems. During his research it became obvious that elaborate kernels were being constructed to support relatively simple secure systems. Much effort had been put into proof of correctness of kernel enforcement of multilevel security. Paradoxically, every practical implementation included software that was related to security but was not a part of the kernel, and needed to violate the kernel's meticulously verified policy enforcement. As demonstrated by the Kernelized Secure Operating System in the late 70s, this software may "aid the day-to-day operation of the system (e.g., secure spoolers for line printer output, dump/restore programs, portions of the interface to a packet switched communications network, etc)."[62]

Rushby questioned this approach to security kernel development in his seminal 1981 paper.[63] His alternative approach mimicked the construction of high-assurance hardware devices based on physical separation, where functional units were connected explicitly by wires. It is thus easy to see what units are connected and which are not connected. The physical architecture reflected the logical architecture and the desired policy. Rushby proposed a way to do this in software, by isolating functional units and providing explicit connections among them using a mechanism having assurance approaching that achieved by physical separation.

Later, Rushby showed that the verification of enforcement mechanisms for specific security policies was simplified by this approach. The Distributed Secure System (DSS)[64] separated the security concerns of policy enforcement from those of resource sharing and used a variety of mechanisms (dedicated components, cryptography, periods processing, separation kernels) to manage resource sharing more simply.

[62] [BB79] Berson. KSOS-development methodology for a secure operating system.

[63] [Rus81]

[64] [RR83] Rushby and Randell. A Distributed Secure System.

Beyond security, Rushby proposed separation kernels for safety in 1986,[65] and published separation, channel control, partitioning, safety and assurance research results continuously through the 1990s.[66]

4.1.1.2 Classic MILS Era 2000 ~ 2007

"Classic" MILS emerged circa 2000 on the recognition that commercial partitioning kernels for avionic safety could also be applied to security concerns. Strong partitioning (separation or isolation) provided a basis for controlled information flow and damage limitation. Leaps in processor performance, inclusion of memory management units with smaller processors, and emerging support of hardware-assisted virtualization by COTS hardware platforms was an additional reason for the propagation of ideas of secure systems based on partitioning. That led to the popular rediscovery of Rushby's separation kernel and then to the development of Common Criteria Protection Profiles for Partitioning Kernels by The Open Group[67] and Separation Kernels by NSA.[68] A MILS protection profile issued in 2016 as a whitepaper by the EURO-MILS project addresses primarily the operating system as part of a MILS integrated system.[69] Funded research on MILS by Rushby and others continued at SRI International and elsewhere from 2004 to 2012.[70]

4.1.1.3 Modern MILS Era 2008 ~ 2012

The term "Modern MILS" was coined by Rushby around 2008 to refer to refinements emerging from research ongoing since 2005, including a "two-level view" (the policy architecture level and the resource sharing level), MILS foundational components, operational components, the MILS platform, and compositional assurance and certification.[71] In the Modern MILS paradigm, properties of the system are assured according to a given high-level policy in two steps. First, a policy architecture shaped by the high-level policy requirements is established and implemented by an appropriately configured MILS platform, and second, assuming that the policy architecture is properly established by the configuration tools and enforced by the platform, the composed properties of the components are checked according to the high-level policy.

[65] [Rus89] Rushby. Kernels for Safety?

[66] The complete list of Rushby's papers is supported by the author at *http://www.csl.sri.com/users/rushby/biblio.html*

[67] [LMBC$^+$03] Protection Profile for Partitioning Kernels in Environments Requiring Augmented High Robustness (PKPP).

[68] [SKP07] SKPP

[69] [FS16] Common Criteria Protection Profile "Multiple Independent Levels of Security: Operating System" [V2.03].

[70] [Rus08] Rushby. Separation and integration in MILS (The MILS Constitution).

[71] [BDRS08] The MILS Component Integration Approach to Secure Information Sharing.

The Modern MILS era saw the development of Common Criteria protection profiles for MILS Network and Console System components, work towards a MILS Integration Protection Profile (aka MILS Platform PP), and investigations of MILS applications[72] and the issues of MILS trusted delivery, configuration and initialization.[73] It also gave birth to the concepts of heterogeneous, distributed and dynamic MILS, which are prerequisites for MILS in IIoT, spurring the "Progressive MILS" era.

4.1.1.4 Progressive MILS Era 2012 ~ Present

Further refinements and additions to the MILS conceptual landscape of "Progressive MILS" began in 2012 to include integration planes, distributed MILS, dynamic MILS, MILS delivery, configuration and initialization, the benefits of least-privilege separation kernels, and the use of assurance cases that are structured along architectural lines. This period covers:

- Distributed MILS: assured scalable distributed deterministic systems
- Dynamic MILS: assured reconfigurable systems, cloud computing, IoT systems
- Adaptive MILS: assured critical infrastructures, adaptive & resilient systems
- MILS platform components: separation kernel, network system, and console system development
- Heterogeneous MILS: separation kernels based on heterogeneous processors (CPU, GPU, FPGA) and
- Mixed-Critical MILS: assured mixed-critical cyber-physical systems.

These MILS developments were advanced through several research and technology development projects funded by the European Commission, including D-MILS, EURO-MILS, CITADEL, CertMILS and PHANTOM[74] projects. These advancements position MILS to address the broad and stringent requirements for functionality and trustworthiness levied by IIoT.

4.1.2 Variations on the MILS Platform

4.1.2.1 Distributed MILS Platform

Distributed MILS relies on extensions to a MILS separation kernel and the addition of a MILS network subsystem using a hardware-based, time-triggered Ethernet "backplane". It is possible, for the first time, for an application architecture to span multiple computer systems seamlessly, with scalable deterministic operation over a set of nodes, opening many new practical application

[72] [DHR09] Rance DeLong, David Hanz, and John Rushby. An Application of the MILS Approach to Secure Information Sharing. November 2009.

[73] [DeL12] Delivery, Configuration and Initialization of MILS Components and Integrations.

[74] [PHA], though not primarily a MILS project, PHANTOM developed a heterogenous and parallel computing platform for executing "component networks" based on MILS platform principles.

areas for MILS. Automated design and verification assistance, as has been developed and applied in the Distributed MILS project, provides development process support and is indispensable for the verification of dependable distributed systems.[75] System architects, developers, integrators, installers, operators, and the organizations and populations that depend on critical systems, benefit from the resulting assurances that errors that lead to added cost and dangerous failures can be eliminated.

The Distributed MILS platform comprises MILS nodes that communicate over a deterministic network. The requirements to provable domain isolation and information flow controls across the platform increase assurance of the absence of hidden channels and unwanted information flow.

A MILS node implements a minimum separation kernel that controls the information exchange between the applications and virtualizes hardware resources. The networking system with proven fault-tolerant synchronization strategy, e.g. time-triggered Ethernet,[76] guarantees message delivery and separation. The separation kernel of each node as well as each switch of the network is statically configured to realize the distributed MILS policy architecture. In any case, the configuration must guarantee the absence of unintended information exchange in the system deployed on the distributed MILS platform.[77]

The Distributed MILS platform extends to distributed embedded systems by creating in effect a distributed separation kernel using deterministic and predictable network communication. In this way the Distributed MILS platform provides the capability to use one policy architecture that seamlessly spans across multiple MILS nodes. Moreover, time and space separation of the Distributed MILS platform might be used to minimize the effect of faults or attacks to certain parts of the distributed system, thereby avoiding propagation of harm and increasing resilience.

One of the running case studies for the D-MILS project was based on demonstrating various privacy and security requirements for a prosumer-based smart grid.[78] Successful decentralized and prosumer-based smart grids need to be at least as dependable and secure as the prevailing one-way, generation-transmission-distribution-consumer power grids. A two-phase model-based design methodology for secure architectural design and secure deployment of such a

[75] [D-M] The D-MILS project (2012-2015).

[76] [SAE] AS6802: Time-Triggered Ethernet.

[77] [BQIR14] Distributed MILS Architectural Approach for Secure Smart Grids. fortiss GmbH, An-Institut Technische Universität at München, Germany.

[78] A prosumer, a term from the dot-com era, refers to a person who consumes and produces a product. In smart grid prosumer a new type of energy user who consumes, produces, stores and shares energy with other grid users whose collaboration is critical for the sustainability and long-term efficiency of the energy-sharing process.

security architecture on a distributed separation kernel helps to gain assurance on the representative privacy properties. These properties are considered for the micro grid case study and they can be encoded in terms of information flow properties.[79] These encodings help in detecting a case where privacy is broken. In such a case an alternative model should be proposed. This case study successfully demonstrated that the D-MILS approach is suitable to reason about security requirements.

The D-MILS project established a common framework for critical system construction and certification, encouraged innovation with component and service suppliers, and led to improved dependability and reduced cost to develop, certify and deploy trustworthy critical systems.

4.1.2.2 Dynamic MILS Platform

MILS implementations, including distributed MILS, initially provided only for fixed runtime architectures, as they are based on statically configured MILS platforms. That is, the configuration information used to configure the exported resources of the separation kernel and other MILS resource-sharing foundational components was finalized before initialization of the MILS platform. After initialization there was no creation or destruction of exported resources, and no changes in the information-flow policy. This is a characteristic shared with safety-critical real-time operating systems (RTOSes). The rationale, inherited from the safety domain, was that only static systems can be adequately understood and analyzed to achieve the required level of confidence that they will behave as expected. The approach had also been applied to security-critical systems needing the highest levels of assurance.

Two capabilities are required of the platform to enable dynamic configuration change: configuration introspection and dynamic reconfiguration primitives. Configuration introspection enables a running subject to query the configuration data of the platform. Dynamic reconfiguration primitives enable change at runtime to the current configuration of the MILS foundational components. To reason about reconfiguration, the use of reconfiguration primitives must be constrained by a configuration change policy. Enforcement of the configuration change policy is implemented through monitoring the evolution of a dynamic MILS configuration and mediating runtime configuration change requests.

One of the necessary properties of dynamic MILS is the ability to prove that the configuration of a MILS system is maintained until it is changed by authorized configuration-change operations, and that portions of the configuration that are not the object of a configuration change operation are guaranteed to be unaffected by the configuration change provided certain conditions on the configuration are met. Fortunately, this is a familiar paradigm for the developers of high-

[79] [CS09] Clarkson. Hyperproperties.

assurance secure systems, and the problem is amenable to known verification strategies for it, such as property validation by model checking.

Dynamic reconfiguration entails additional assurance activities. Certainly, for the runtime system, system security and safety objectives must be continuously met during operation of dynamic MILS systems. Fortunately, the activities for dynamic MILS are not substantially different from those for static MILS, provided that a thorough and diligent effort is done for the static case, and that certain extensions are provided to the specification language and analysis tools.

4.1.2.3 Adaptive MILS Framework

To be resilient, a system must first be adaptable. Trustworthy adaptation requires that a system can be dynamically reconfigured without compromising its robustness and integrity. Traditional certification practices have conservatively required critical systems to be static and have required assessment of the complete integrated system for certification. Adaptability is at odds with certification because it changes the system.

The Adaptive MILS Platform is dynamic, implementing a full and flexible ability to change its configuration during runtime. It is adaptive, including mechanisms to monitor its operation and its interaction with the environment. It must also include mechanisms that use its dynamic reconfiguration capabilities to maintain safe and secure operation, and to fulfill the system's mission in the face of environment change or internal failures.

The reasons for a reconfiguration vary (e.g., new requirements, component failures or attacks) and require input from different internal components of the platform, as well as from external agents and the environment. The constraints on reconfiguration may be defined in entirely different ways. The global demand for adaptive physical systems with enhanced requirements pertaining to their safety and security is increasing due to the rise of the internet of things and the sustainable growth of smart industry. All these factors lead to the conclusion that building adaptive MILS-based solutions must include the platform, as a combination of hardware, firmware and software for resource sharing, and the means to describe system architecture, component failures and their propagation, mechanisms for runtime monitoring and adaption, and an automated framework in which to state the claims for the adaptive system, expose the overall assurance strategy, and track the evidence resulting from analysis of system components, using diverse assurance measures and tools. Adaptive MILS refers both to the platform and all the necessary artifacts and methodology for building adaptive and resilient solutions.

The ordinary MILS Platform is a composition of a foundational plane, an operational plane, a monitoring plane, and a configuration plane. However, a monitoring plane had not previously been implemented as part of a MILS platform and the configuration plane previously consisted of off-line configuration tools provided with the separation kernel.

The Adaptive MILS Framework developed in the CITADEL project extends the MILS platform to dynamic and distributed operation by augmenting the prior MILS Platform's three online planes (foundational, operational, and monitoring), with the online configuration plane, and by adding an adaptation plane and a certification-assurance plane. The monitoring plane provides the feedback for reconfiguration and becomes an essential part of the adaptation loop (Figure 8).

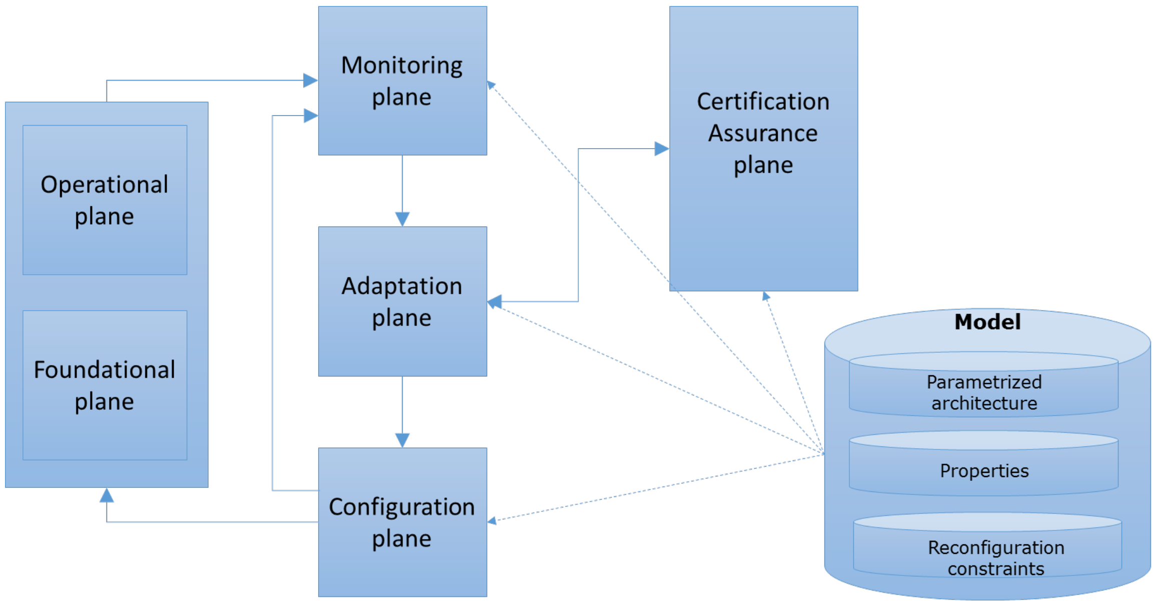


Figure 8: The generic architecture of Adaptive MILS framework

Configuration mechanisms (which are provided not only in the configuration plane) implement configuration introspection and dynamic reconfiguration capabilities that are essential for the support of Dynamic MILS. The supportive mechanisms and overall architecture of the Adaptive MILS Framework support safe and secure use of these mechanisms for reconfiguration within the constraints of a configuration change policy.

The implementation of the framework includes a network subsystem that employs a hardware-based time-sensitive networking (TSN) backplane, dynamic re-configuration primitives, a reconfiguration policy enforcement mechanism, platform and environment monitoring.

The Adaptive MILS Framework covers the following technology areas for the development of systems of high criticality that function in complex challenging environment:

- modeling language for dynamic systems,
- analysis & verification methods and tools,
- detection & monitor synthesis,
- recovery & adaptation methods,
- Dynamic MILS platform foundation plane, including separation kernel, MILS network system,

- Dynamic platform configuration plane,
- Dynamic platform adaptation plane,
- Dynamic platform monitoring plane and
- Dynamic platform certification assurance plane & Adaptive MILS Evidential Tool Bus.

4.1.2.4 Heterogeneous MILS Platform

As IoT and big data trends go hand in hand, IoT applications start to require more and more resources–mainly due to the increase of the amount of data that must be handled and to the increase in the complexity of the algorithms used to process it, such as machine learning techniques. Data gathered from connected devices can be enriched with social media, video analytics, weather and other third-party data and turned into real-time actionable insights. Patterns and anomalies detected in real time help to build, manage, and maintain better products and services. Managing IoT information with traditional data and analytics often mean operating with data that is organized in an outdated way and dealing with complex and specialized systems, usually leading to high latency and exorbitant costs.

This situation has led to an increase of interest in techniques that take advantage of non-CPU based computations and efficient parallelization of applications and processes. For example, complex analytical queries on large and streaming datasets may be optimized by applying computations based on a graphical processing unit (GPU) database approach. GPU databases are more flexible in processing different types and larger amounts of data.[80] [81] Similarly, the field-programmable gate array (FPGA) technology plays an important role for the future of IoT. Low-power optimized FPGAs are able to enhance speed and power consumption of several types of algorithms compared to microcontrollers of commercial sensor nodes. The architectures based on the combination of systems-on-chip and FPGA can play a key role in the future of sensor networks and in fields where processing capabilities such as strong cryptography, self-testing and data compression are paramount.[82] [83]

Time and resource optimization is where system or service quality meets trustworthiness. Many safety, reliability and resilience aspects are defined through predictable time-based characteristics such as response time. Thus, using non-CPU based technologies makes sense for trustworthiness aspects. At the same time, the integration of diverse processing technologies into a complex mechanism carries a risk of undermining trustworthiness if it cannot be convincingly achieved.

[80] *https://developer.nvidia.com/taxonomy/term/785*

[81] *https://www.kinetica.com/solutions/iot/*

[82] [dlPBT12] Sensor Systems Based on FPGAs and Their Applications: A Survey.

[83] [SAM15] FPGA for Internet of Things.

To make such a mechanism feasible, we consider heterogeneous processors as a MILS platform that generalizes the separation kernel concept. It is based on a strict observance of the fact that a separation kernel currently is *not* considered a piece of software that runs on a piece of hardware. The separation kernel may be realized as an arbitrary combination of hardware, firmware, and software that comes together with modeling and specification languages, tools for validation and verification and features for enforcing the bespoke security policies on the components interaction.

There is presently no public information about MILS platform implementations for such heterogeneous IoT environments. The demand remains relevant and will only grow. The PHANTOM Project[84] aimed to enable the development of next generation heterogeneous, parallel and low-power computing systems through innovative tools that hide the complexity of computing hardware from the programmer, provides a low-level solution to ensure integrity of component network execution on the heterogeneous platform based on isolation and information flow control inspired by the MILS approach.

4.2 MILS Case Studies for IIoT

Security, safety, resilience, reliability and privacy may be assured in several ways and with varying levels of confidence. The goal of these case studies is to demonstrate how conflicting and non-trivial concerns may be addressed under growing demands and complexity of IIoT infrastructure. These case studies are not intended to cover all industries, nor do they cover all five trustworthiness aspects in every case.

4.2.1 MILS-based security platform for railway command and control systems

Control, command and signaling are at the core of railway operations. They determine safety and performance of the rail network. Railway command and control systems must allow safe train operations and meet capacity demands. As these demands and the complexity of railway infrastructure grow, the signaling system, its methods for ensuring safety, and factors on which it relies, change. In the area of control, command and signaling specifically, we have seen time separation ("generation 1.0"); then space separation, by electro-mechanical signals (the absolute block principle, "generation 2.0"); and finally track-to-train communication-based signaling, still based on block sections ("generation 3.0"). With new capabilities, a fourth generation of railway traffic management system enabled by vehicle-to-vehicle communication ('command and control 4.0') will become possible.[85]

[84] [PHA]

[85] [Dop18] Doppelbauer. Command and Control 4.0.

To support these changes, the railway infrastructure, proprietary systems and protocols are being replaced with IP-based networks and COTS technologies. But there are safety concerns because these technologies were not initially designed to be safe and resilient.

Railway command and control systems implementing the safety logic of digital interlocking require assurance of safe and resilient behavior. This assurance may be obtained using a MILS design approach as demonstrated, for example, by the HASELNUSS project.[86]

The aim of the HASELNUSS project is the development of a customized hardware-based security platform for railway command and control systems that provides required security and safety. The platform features provisions to ensure system integrity and constitutes a foundation for secure infrastructure networking.

Current signaling systems can be divided into three layers: operation layer, interlocking layer and field element layer. These three layers are connected via a wide-area network.

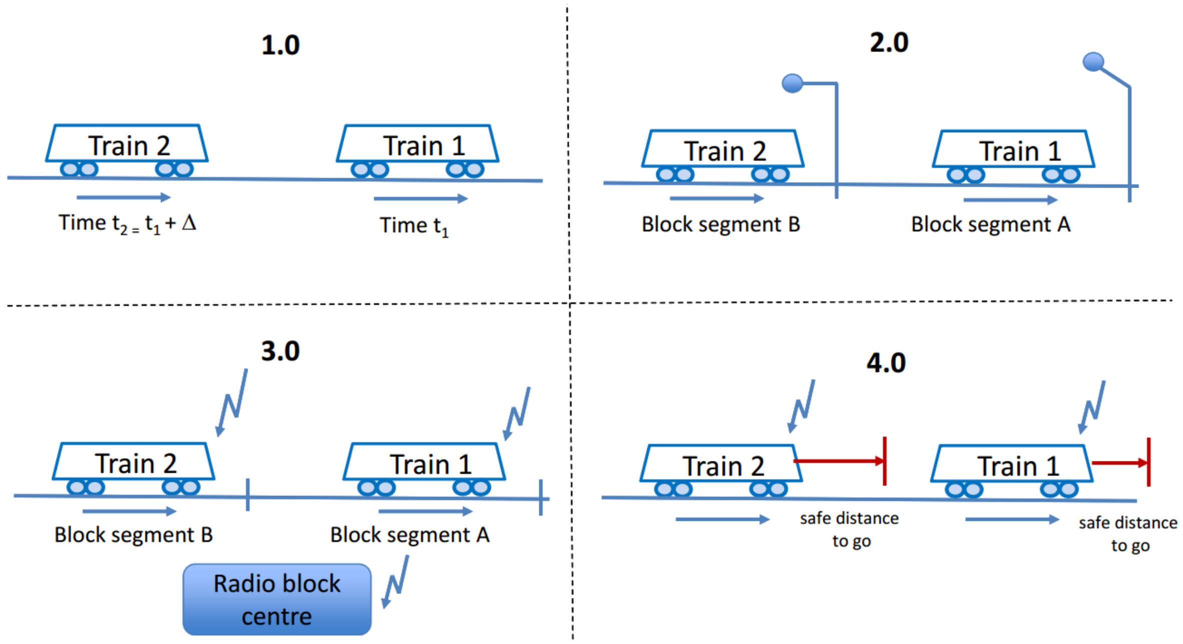


Figure 9: Four generations of control, command and signaling and their basic principles

Source:
https://www.era.europa.eu/sites/default/files/library/docs/command_and_control_en.pdf

Most of the safety systems are located on the interlocking layer, which checks the commands from the operation layer for validity and whether they respect the safe operation rules. It also

[86] [HAS]

monitors the components on the field-element layer for correct operation and in case of anomalies, falls into an error state. Systems like the interlocking and the European Train Control System (ETCS)[87] are located on this layer.

Components in this layer are developed according to several safety standards like EN 50126[88] and only the required functionality is available. Additionally, these components are built redundantly, which means that in case of a defect one of the standby systems comes in place and the maintenance personnel is notified to replace the failing component. The data networks and the power supplies are redundant too.

The Haselnuss Reference Architecture (HRA) integrates in railway systems at the field-element level as object controllers for field elements.

Information channels to the safety application are realized using the communication objects provided by the separation kernel that allows precise control over the information flows in the system. This partitioned architecture based on the certifiable separation kernel that provides evidence of non-interference between the high-assurance safety applications (i.e., Safety Integrity Level (SIL) 4) and the security applications that do not contribute to the safety of the system (and so has a lower SIL). This freedom-from-interference evidence is needed to keep the existing certification of safety application when integrated with the security applications. The separation kernel is also certifiable at the same assurance levels (i.e., SIL 4) as the safety application, e.g., a railway object controller.

The MILS approach is used to run the critical infrastructure's safety application(s) on the same hardware as the security applications that protect the safety functionality against attacks. Strict partitioning of underlying platform resources, particularly in static separation kernels, guarantees that one domain cannot demand or use more than its allocation, and separation allows defining the exact contact points of information flow between the safety application and the security application such as an intrusion detection system. This structures the safety case, where the influence of security has to be investigated and freedom from interference with the safety has to be proven.

The Haselnuss reference architecture provides several security functions. This includes mutual authentication of Haselnuss nodes with the interlocking system, protecting the software integrity of Haselnuss nodes at boot- and run-time, integrity reporting and remote attestation of Haselnuss nodes, remote software update of Haselnuss nodes and an intrusion detection system

[87] [EURb] European Rail Traffic Management System (ERTMS).

[88] [Eur17] Railway applications Reliability, Availability, Maintainability and Safety (RAMS).

(IDS).[89] The part of this functionality such as firewall and IDS is implemented in a separate network security enhancer installed between interlocking and operational layers.

The rest is supported at the level of Haselnuss nodes, which can implement the required MILS policy architecture due to the use of hardware root of trust and the appropriate software stack, secure boot to prevent unauthorized modification or tampering of software and configuration, and high-assurance separation kernel allowing running the applications of mixed criticality on the same platform.

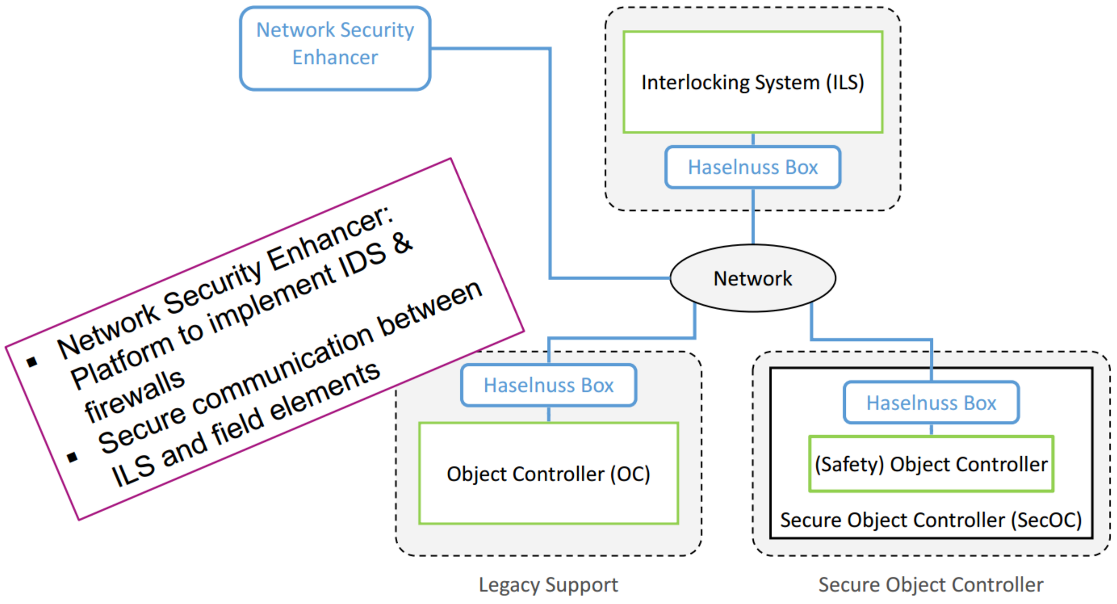

Network Security Enhancer
Interlocking System (ILS)
Haselnuss Box
Network
▪ Network Security Enhancer: Platform to implement IDS & firewalls
▪ Secure communication between ILS and field elements
Haselnuss Box
Object Controller (OC)
Haselnuss Box
(Safety) Object Controller
Secure Object Controller (SecOC)
Legacy Support
Secure Object Controller

Figure 10: The place of Network Security Enhancer in the HRA.

Source:
http://ifev.rz.tu-bs.de/SiT_SafetyinTransportation/SiT2018/Freigegeben_Katzenbeisser.pdf

[89] [BKZ+18] A Reference Architecture for Integrating Safety and Security Applications on Railway Command and Control Systems.

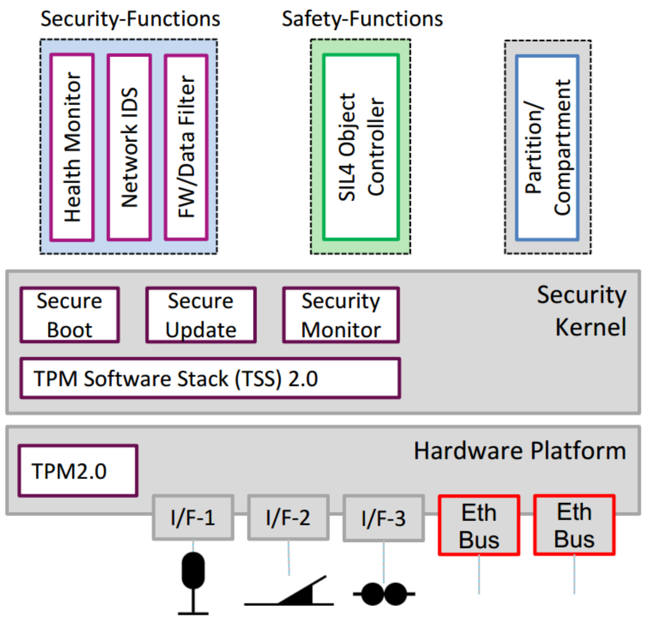


Figure 11: Haselnuss node architecture.

Source:
http://ifev.rz.tu-bs.de/SiT_SafetyinTransportation/SiT2018/Freigegeben_Katzenbeisser.pdf

4.2.2 Distributed MILS Platform for Secure Smart Grids

Smart power grids hope to provide sustainable energy services using bi-directional flow of data and power enabled by advanced information, communication and control infrastructure. Prosumers are important stakeholders in future smart grids and have a vital role in peak demand management.[90]

The control of smart grids requires change towards decentralized energy management systems with a tight coupling of energy control with new monitoring, processing, optimizing, and controlling devices based on real-time information and communication technology.

To maintain privacy, no prosumer should know the consumption of another prosumer. Assurance on this requirement must be demonstrated.[91] [92] This requirement can be detailed as:

[90] [ZMR⁺18] Prosumer based energy management and sharing in smart grid.

[91] [BQIR14] Distributed MILS Architectural Approach for Secure Smart Grids. In: Cuellar J. (eds) Smart Grid Security.

[92] The D-MILS project [D-M] is funded by the European Commission under the 7th Framework Programme for Information and Communications Technology. The smart grid case study by Fortiss has been supported by Siemens, the EIT ICT Labs, and the Bavarian Ministry of Economics.

RQ1: No prosumer is able to bypass the defined communication channels to find out the consumption plan of any other prosumer.

RQ2: No prosumer is able to deduce the consumption plan of any other prosumer with any received information.

The first requirement refers to the low-level implementation of the system. It is enforced through separation capabilities of the platform and its configuration by a configuration compiler. The configuration files are built from a formal model of the system. A configured platform guarantees the absence of unintended communication channels that are not in the formal model. The second requirement is ensured by checking that the formal model satisfies a security property.

The policy architecture indicates how information is allowed to flow between the different components of the system. A MILS platform establishes and enforces a security policy defining communication channels between components. Consequently, the construction of security assurance cases is separated into establishing security of the high-level model and enforcing the policy architecture defined by the high-level model through configuration of the distributed platform. The MILS platform must have provable domain isolation and information flow controls, thereby increasing assurance of the absence of hidden channels and unwanted information flow.

The distributed MILS architectural approach enables using a single policy architecture that seamlessly spans across multiple MILS nodes, as required for demonstrating various security and privacy requirements for a prosumer-based smart grid.

A configured D-MILS platform ensures that the only possible communications are the ones defined in the security policy. The configuration compiler is a tool that produces a configuration for each node and each switch of the platform. To this purpose, the configuration compiler is fed the policy architecture, a model of the platform, and deployment information. Then the configuration compiler generates configuration files for running separate partitions and establishing the communication channels between partitions according to the policy. Each node, except the partitions corresponding the policy architecture domains, contains a dedicated partition for hosting a MILS Network Subsystem in charge of the communications over the time-triggered network (Figure 12).

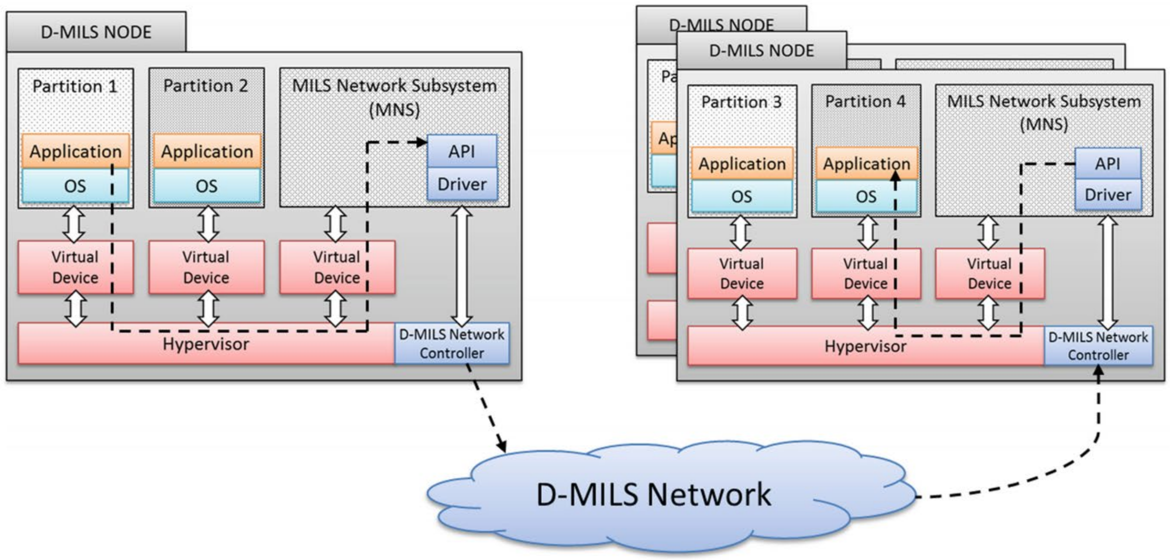


Figure 12: D-MILS platform

This level of policy architecture abstraction, to be assured as consistent to the implementation, requires an additional level of modeling of the low-level hardware components and topology. A hardware topology of Smart Grid demonstrator that comprises computation units (e.g. ECUs, cores, etc.), communication units and sensors/actuators, in D-MILS project was represented as a logical architecture using a model-based tool AutoFOCUS3. This tool allows modeling and validating concurrent, reactive, distributed, timed systems based on formal semantics.[93] It offers several levels of abstraction whereby the logical and the technical architecture views were used for the case under consideration. These models form the starting point for the design and implementation according to the D-MILS architectural design and implementation approach.

The logical architectural view of a system is defined by components communicating via message passing through typed channels, using a clearly defined model of computation. Message exchange is synchronized with respect to a global, discrete time base. Components can directly implement behavior or consist of other components that do so.

D-MILS platform consists of abstract subjects instead of ECUs and other hardware. Those subjects are connected by a communication medium, namely, the time-triggered Ethernet switch. Each subject has sensors that represent the interfaces to the data that is received from outside the Smart Microgrid system (i.e., a Micro Grid component receives the current energy price and the deviation event). The subjects can be on the same machine but do not necessarily have to.

[93] *http://af3.fortiss.org*

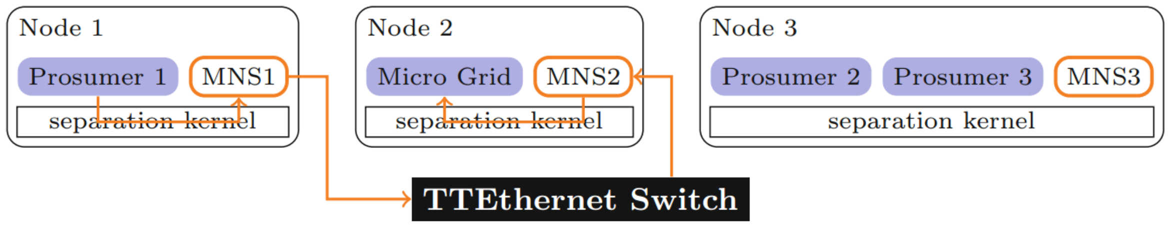


Figure 13: A communication channel in a deployed D-MILS smart micro grid

The platform guarantees that the information exchanged between components follows the channels defined in the high-level model. Each component is only aware of the values received on its input ports. For instance, each prosumer can only see the information sent by the Micro Grid component, and cannot directly communicate with another prosumer. This (but represented in a more formal way) provides ground for certainty about completion of RQ1.

However, a prosumer may get some information about other prosumers through the microgrid component. RQ2 demands that no prosumer can deduce the consumption plan of any other prosumer. The implementation is secure if there is no observation that allows a given prosumer to deduce the consumption of another one.

In the theory of knowledge, an agent observes only a part of the system and knows the set of all possible traces allowed by the system. If a particular property holds in all possible traces and are consistent with the current observation, then the agent knows that this property holds in the current execution, thus formalizing this requirement.

Focusing on the knowledge gained during one exchange of plans and acknowledgements and assuming that each prosumer knows the set of possible traces (or, equivalently, the bounds for the prosumer plans and the line capacity) further constraints are set on prosumers plans. As researchers demonstrate for the case under consideration, if the global contribution of the prosumer[94] remains within the bounds corresponding to the maximal consumption or production, it is not possible to make any conclusions regarding the level of energy consumption for this particular prosumer. In this case, the extreme values for each consumption plan in a group of three or more prosumers can be reached in several ways, hiding the real consumption value.

Thus, when representative privacy properties of the microgrid case study are encoded in terms of information flow properties, the privacy property is defined through a preliminary encoding of facts deduced from the execution traces, which are visible at the input ports. These encodings

[94] Contribution is computed by adding non-negative consumption + non-positive production + battery usage plan indicating whether the battery will store energy as negative value or provide energy as positive one.

allowed detecting a case where privacy is broken and propose an alternative model that is more secure. The role of MILS approach for this privacy assurance is in the clear identification of the reference points for the platform architecture allowing describing the traces and ensuring that they fit the necessary constraints. The analysis can be further extended by accounting for the history of actions. Ultimately, the privacy property should be stated in term of the quality of approximation that a prosumer can obtain from a given observation (if this level of attitude to the security and privacy is considered as appropriate for the case).

4.2.3 Adaptive MILS for Resilient ATC Remote Tower Communications

Frequentis AG, Austria, has developed MILS use cases for safety-critical communications. The first of these, Frequentis Voice Service, developed in the D-MILS project, achieved separation of data and voice domains as a demonstration of static distributed MILS. The second, Frequentis Communications Service, developed in the CITADEL project, is a demonstration of adaptive MILS for resilient air traffic control communication. The demonstration made use of the MILS specification and analysis tools to design the demonstration, and of the dynamic MILS platform with the CITADEL adaptation framework for its deployment.[95]

The concept involves an Air Traffic Control (ATC) control room with remote video surveillance instead of *out-of-the-window* view from a real tower. Radar and cameras provide a high-quality real-time image of the airfield and nearby airspace. Sensor data from the airfield and voice communication systems complete the capabilities needed to operate a tower remotely, as illustrated in Figure 14. The communication flows among these domains are separated by the architecture for safety and security. The features necessary for the required communication services are provided by the functional elements: Operator Position, Radio and Radio Gateway, Radar, CCTV and Sensors.

Figure 15 a) depicts the components of the Operator Position and Figure 15 b) depicts the components of the Airfield Services.

[95] [KE19] An adaptive MILS Architecture for Resilient Remote Tower Communication Services.

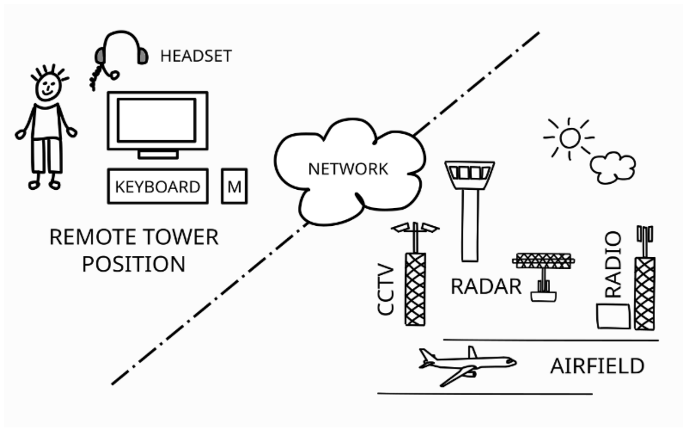
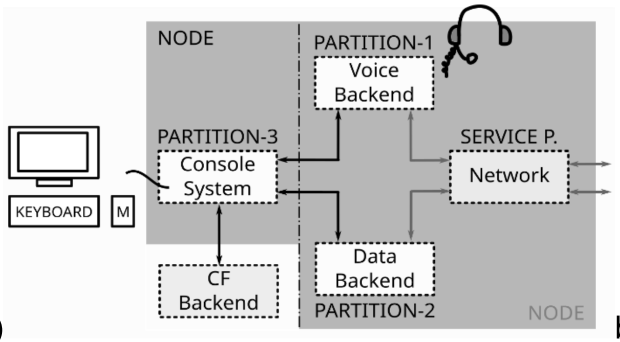


Figure 14: Remote ATC Tower Demonstration

The Radio Gateway application manages access by the Voice Service to a physical radio for several different controller positions. The Operator Position integrates both voice and data applications into a single user interface implemented using the MILS Console System (CS). For demonstration purposes, basic radio emulation (pre-recorded ATC audio) is used to demonstrate radio communication. To test specific inter-component communication flows and evaluate basic performance, the deployment of the demonstration includes two operator positions and multiple remote sites.

a)


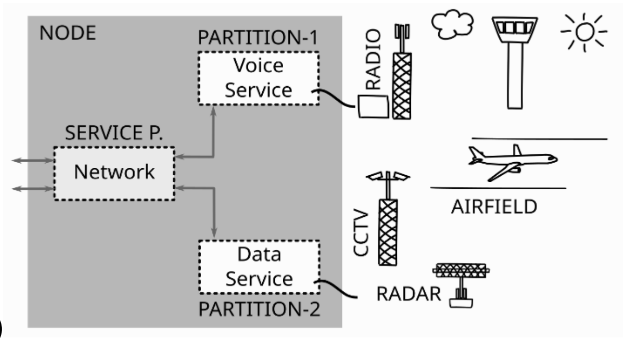
b)


Figure 15: a) Operator Position; b) Airfield Services

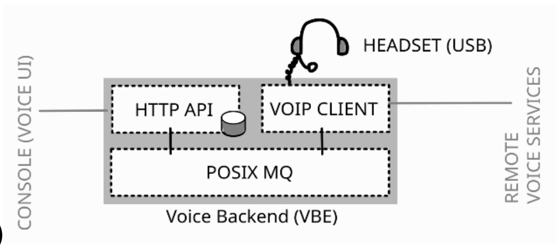
The following briefly describes the main components that integrate with the CITADEL framework (e.g. to initiate a reconfiguration or to receive monitoring data).

Operator Position: The Operator Position (Figure 15a) provides a graphical user interface using the CS and at least one audio device (headset or handset) that connects via USB or an analogue interface to the MILS partition that hosts the voice backend (VBE). The CS runs in a distinct partition that has access to user interface devices (keyboard, mouse and monitor). Further, the CS maintains connections (node internal, black arrows in the figure, to the CITADEL framework, via the Console Framework Backend (CFB), to the Voice Backend, and to the Data Backend (DBE). Backend business logic manages individual user interface elements (rendered to communication screens) and controls the media engine (VBE only). In addition, backend services maintain connections to remote services via the MILS network system (node internal and external, gray arrows in Figure 15. The VBE media processing engine or VoIP client requires access to the connected audio devices and remote voice services (Radio or Radio Gateway). The DBE requires access to remote data services (Radar, CCTV and Sensors) and implements a business logic that manages the user interface (rendered to a surveillance screen) and, if necessary, forwards messages to remote sites. The CS may also run on a separate node or PC workstation (dot-dashed line in Figure 15a).

Radio and Radio Gateway: shown in Figure 15b, are applications that run on standard Linux. Asterisk[96] is used to answer radio calls and to start replaying a pre-recorded wav file in the demonstration. A Radio Gateway may act as border gateway to a voice communication system. Exploiting the MILS Platform's separation capability, distinct Voice Service and Data Service partitions are configured.

Radar, CCTV and Sensors: Radar, CCTV and Sensors, depicted in Figure 15b, are applications that run on standard Linux, e.g. motion[97], or dump1090[98]. Motion monitors the video signal from one or more cameras and dump1090 is used to capture and display aircraft positions.

The MILS *operational plane* of the Operator Position node hosts four applications: The Console System, the Voice Backend, the Data Backend, and the CITADEL Framework Backend. The Voice Backend consists of two main parts, an HTTP API and a VoIP client for air-to-ground and ground-to-ground communication. The HTTP API provides GUI components displayed at the console and connects to the VoIP client via a POSIX message queue as shown in Figure 16a. This separation allows the deployment of individual parts of the VBE for testing purposes.

a)


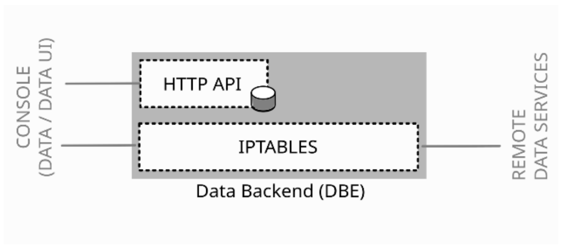
b)


Figure 16: a) Voice Backend (VBE); b) Data Backend (DBE)

The VOIP Client is a headless native Linux application. User Interface (UI) commands to select/key a radio or dial/answer a call are sent to the HTTP API service and forwarded to the VoIP client

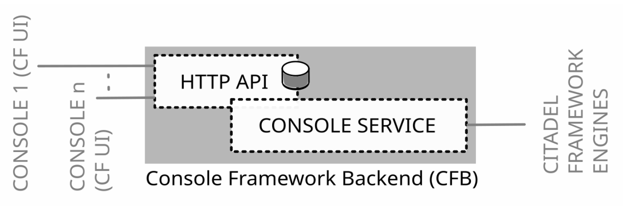
[96] Asterisk: a framework for building communications applications, *https://www.asterisk.org*.

[97] Motion: *https://motion-project.github.io*

[98] dump1090: *https://github.com/antirez/dump1090*, June 2014.

part of the VBE. The VoIP client utilizes plain SIP[99] and RTP[100] as well as domain specific extensions to these protocols. The Data Backend, Figure 16b, consists of two main parts, an HTTP API and *iptables* to route/forward data traffic to the configured remote media proxy. The HTTP API provides UI components displayed at the console (Figure 17a) and runs as native Linux application, *iptables* is part of the Linux distribution.

a)



b)

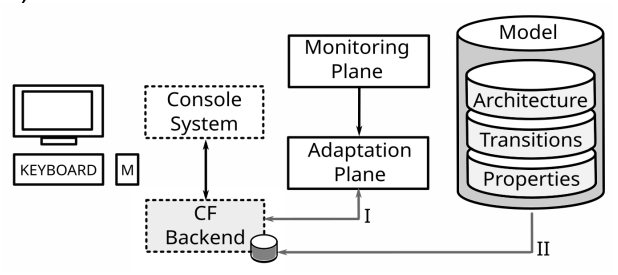
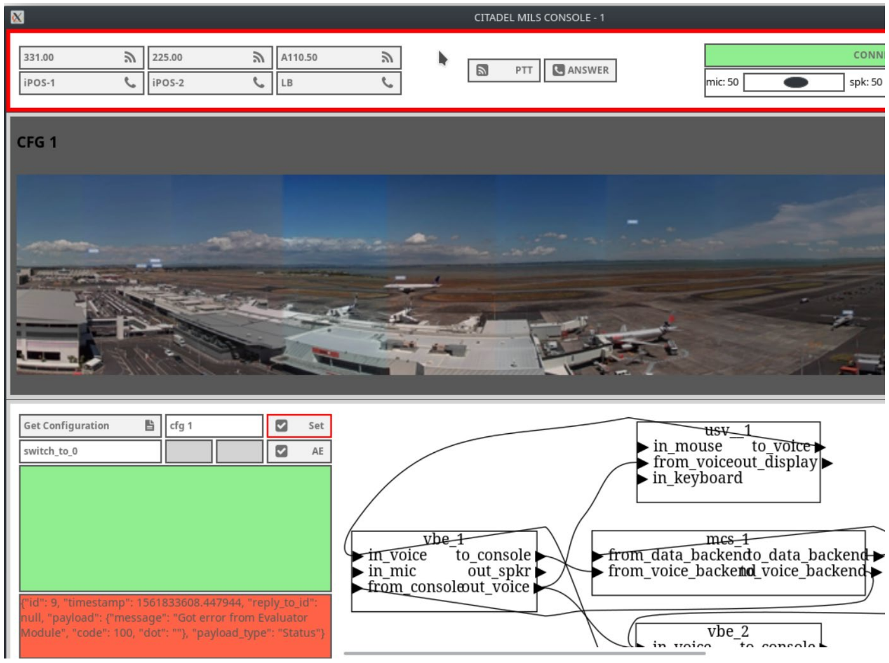


Figure 17: a) Console Framework Backend (CFB); b) CITADEL Framework Backend

The UI (being a video feed, radar data, or sensor data) is loaded via the HTTP API and UDP/TCP traffic is forwarded based on iptables rules configured for the DBE. The CITADEL Framework Backend (Figure 17b) implements the interface to the CITADEL adaptation plane (AP), a service to receive monitoring and status messages from the AP for display on the connected Consoles, and a service to permit multiple instances of the CS to be connected to the AP. An example of the Console Operations Screen is shown in Figure 18.

[99] Session Initiation Protocol, RFC 3261, June 2002. [SIP02].

[100] Transport protocol for real-time applications, RFC3550, July 2003. [RTP03].



```
{"id": 9, "timestamp": 1561833608.447944, "reply_to_id":
null, "payload": {"message": "Got error from Evaluator
Module", "code": 100, "dot": ""}, "payload_type": "Status"}
```

Figure 18: Console Operations Screen

Increasing MILS capabilities increases complexity necessitating new methodologies. For adaptive MILS, a new model based on configuration transitions of a parametrized architecture has been combined with adapted components of the Operational Plane (Console System and Backend Services). In a remote tower reference system, the use case demonstrates that monitoring for failures, in particular, communication failures caused by faulty network components or by malicious acts can be detected to trigger automatic or operator-initiated reconfiguration. The research results of the CITADEL project can be combined with existing commercial safety-critical communications products to yield technology that will increase the capability of future remote tower applications.

4.2.4 Trusted Smart Phone for Enterprise and Personal Communications

As the Bring-Your-Own-Device approach grows in popularity among mobile workers, so do the security concerns of enterprises and individual users. Enterprises are concerned with the confidentiality, integrity, and non-repudiation of enterprise data access from mobile devices. Individuals are concerned with privacy of use, mobile access to media, the security of mobile transactions, and the confidentiality of their personally identifiable information. Commercial-off-the-shelf mobile device offerings did not provide adequate security for commercial enterprises and government sensitive information. The development of bespoke mobile devices for such limited markets is costly and impractical. Thus, commercial and government markets joined together to develop architectures for COTS devices that could provide the necessary, security and flexibility at a sufficiently low cost that would justify providing the necessary foundational capabilities in *all* COTS devices.

The purpose of the Trusted Smart Phone use case[101] is to develop a smart phone having features and security properties that make it suitable for use both as a personal device and as a trustworthy enterprise device, and to influence the adoption of key components by the broader commercial mobile device market. An imperative of the approach is that the device can be marketed as a COTS product that is price, performance, and feature competitive with other commodity smart phones, while providing refined ease of use of its dual-use capabilities that make it an attractive alternative to having two (or more) single-use phones. Further, users must be able to work with familiar operating environments, services and applications without having to adapt to new and different interfaces.

The use of an ARM A9 and A15 processors provides the needed processing power and virtualization extensions to host protected domains and enable an architecture that provides isolation, resource sharing, and device control with adequate safeguards for network applications handling sensitive commercial and secret-and-below government data. The COTS device and software offered must provide a rich execution environment to simplify application development and controlled resource sharing.

As illustrated in Figure 19 the Trusted Smart Phone supports (at least) two domains, or *personalities*: one for use in conjunction with an enterprise (commercial or government) and another for personal use. Applications of the Trusted Smart Phone include access by government personnel to sensitive networks and data with the same device that is used for personal communications and applications. Another configuration of a dual-personality smart phone is to service two enterprise personalities, such as two distinct government domains. The Trusted Smart Phone may be configured with a variety of combinations of personalities.

[101] The Trusted Smart Phone described in this use case was developed and demonstrated by SRI International, Galois, and Open Kernel Labs using LG smartphones based on ARM Cortex A9 and A15 processor architectures as part of the USMC Trusted Handheld Program.

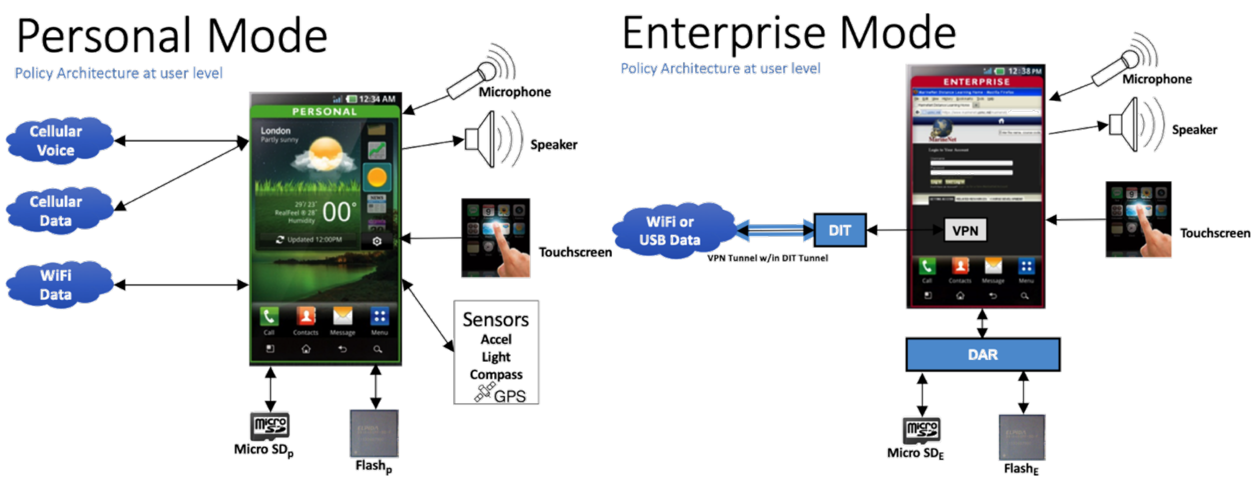


Figure 19: Personal and Enterprise Modes of Trusted Smart Phone

For each such combination of personalities, the device user/owner and the enterprise are both entitled to reasonable expectations concerning the security functions of the device. The Trusted Handheld must provide explicit and unambiguous claims that the user and the enterprise may compare to their expectations. Among the most important of these claims is that none of the personalities have privileges that supersede the expectation of independence on the part of other personalities. Confidence in the validity of these claims is provided to all parties by independent, expert third-party validation. The Trusted Smart Phone differentiates itself from other commercial devices by providing both the functionality and assurance needed to deploy BYOD with confidence and minimal risk.

The security objectives for the Trusted Smart Phone are:

- support multiple personalities, some allowed to connect to sensitive networks and others connecting over-the-air to the internet,
- isolate the domains containing each personality to contain the propagation of failures and the effects of malware,
- implement trustworthy security-critical functions, such as confidentiality and integrity of data-in-transit and data-at-rest, in a protected way,
- implement secure boot and initialization and
- implement non-bypassable platform-wide information flow security policy enforcement.

Despite the obvious benefits of having to carry only one device, users are understandably reluctant to trust a smart phone owned, administered and subscribed by an employer for personal communications and applications. This concern is ameliorated if the user can be assured that personal use is strongly separated from and impervious to enterprise surveillance. Likewise, the enterprise, which can manage its data and communications assets, including the enterprise personality on the smart phone will not accept the risk of unauthorized access or tampering from the untrusted and potentially subverted personal mode of a shared user-owned device without assurance that its control over the enterprise domain cannot be usurped. Accordingly, to forge this multi-party trust relationship, the assurance requirements for the Trusted Smart Phone go beyond the ordinary assurance measures for consumer devices. The Trusted Smart Phone:

- should be amenable to Common Criteria security evaluation according to protection profiles appropriate to the needs of commercial/government enterprises,
- provide trustworthy certification that the concerns of both the device user/owner and the enterprise(s) are credibly served,
- achieve assurance levels commensurate with the value of the assets protected and
- be commercially competitive.

To achieve the functionality and assurance objectives the Trusted Smart Phone project embraced the MILS architectural approach. The design, implementation, and assurance strategy for the Trusted Smart Phone are founded upon MILS principles and practices that had been developed and applied to high-assurance systems over the preceding decade. A MILS platform provides the foundational layer for the device, which supports the operating systems, system software and applications of the operational layer. Figure 20 depicts the operational subsystems of the Trusted Smart Phone.

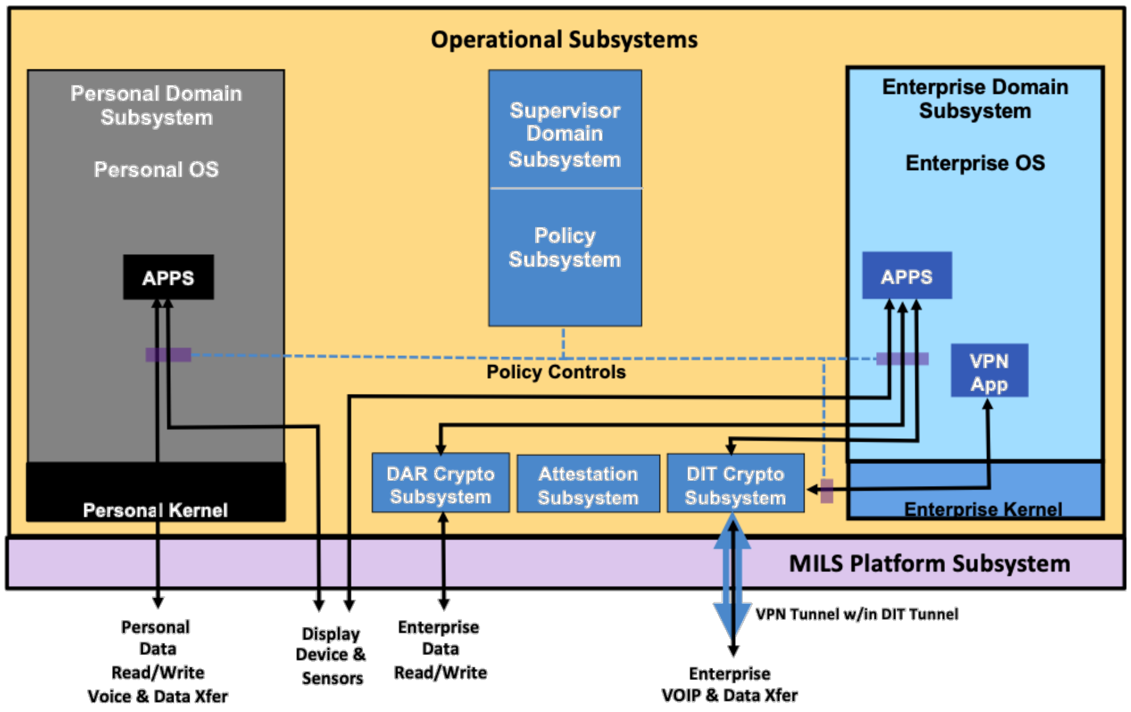
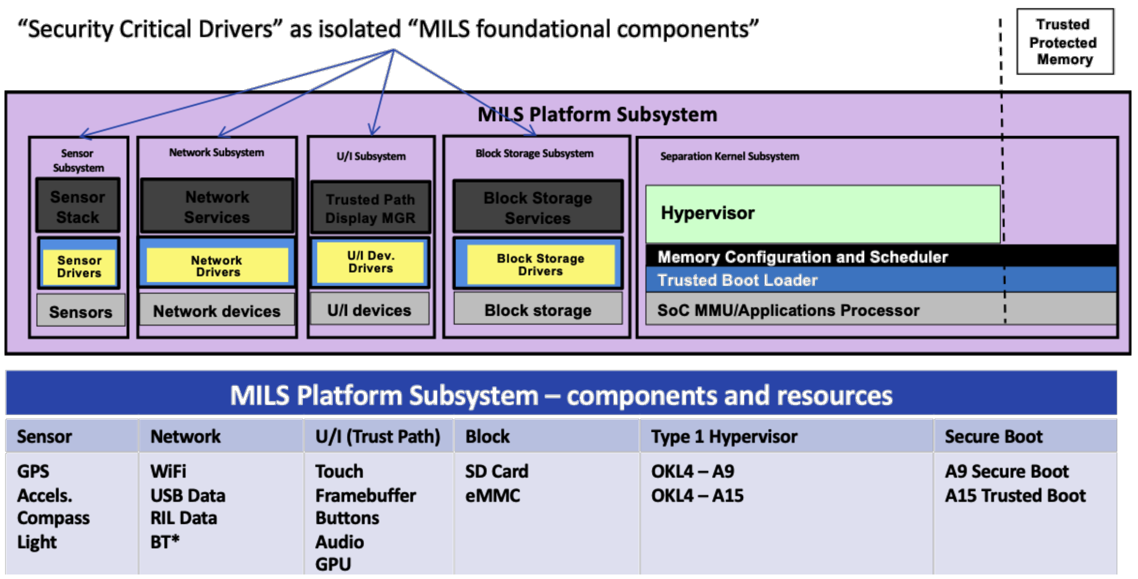


Figure 20: Operational Subsystems of Trusted Smart Phone

The approach retains the operating environment, services and applications with which the user is familiar on existing smart phones, while adding robust resource and device management, enhanced security services, policy enforcement and a trusted path for the user to invoke security services and to switch among personalities.

Figure 21 depicts the MILS platform subsystem of the Trusted Smart Phone. One area of great concern to those who seek robust security is the device control aspect of the operating system. The operating system needs to support various device classes such as sensor devices, block storage devices, network interface devices and user interface devices. They also need the corresponding device drivers and the complex software stacks that create useful resource abstractions from the raw physical resources provided by the device. Each of these software stacks, not to mention third-party device drivers, represents complexity that can harbor vulnerabilities that can threaten device resources and every other part of the system if it is not executed in an isolated domain. Consequently, certifiers demand that security-critical device subsystems be in isolated components. This is what the Trusted Smart Phone did with its sensor, network, UI and block storage sub-subsystems, following the MILS platform paradigm in which such components compose with the separation kernel/hypervisor to form the mobile MILS platform.



Sensor	Network	U/I (Trust Path)	Block	Type 1 Hypervisor	Secure Boot
GPS	WiFi	Touch	SD Card	OKL4 – A9	A9 Secure Boot
Accels.	USB Data	Framebuffer	eMMC	OKL4 – A15	A15 Trusted Boot
Compass	RIL Data	Buttons			
Light	BT*	Audio			
		GPU			

Figure 21: MILS Platform for Trusted Smart Phone

The trusted device sub-subsystems are:

- Sensor sub-subsystem provides access to onboard sensors such as GPS, accelerometer, light sensor and magnetometer.

- Network sub-subsystem includes and provides access to network interface devices, maintaining separation and implementing the policies for sharing of network devices among personalities and other subsystems.
- User interface sub-subsystem includes and provides controlled sharing of the user interface devices, provides switching of the UI devices among personalities, provides trusted display management, implements a UI device management policy, allows invocation of trusted administration and other trusted utilities, and implements the *trusted path*[102] function.
- Block storage sub-subsystem includes and provides access to and sharing of block storage devices among personalities, assuring isolation among allocated regions of such storage devices.

The mobile MILS platform supports the overall architecture of the Trusted Smart phone shown in Figure 22. The foundational separation kernel is a type 1 hypervisor, the OKL4 Microvisor, a commercial version of the L4 verified kernel. It is an important feature of this architecture that the OEM version of the normal operating environment is supported for the personal domain. This permits the smart phone to keep pace with all of the latest technological capabilities essential for competitive differentiation and commercial viability. The OEM's Android OS is supported by its own device driver cell and access to virtual services through client drivers. The virtualization capabilities permit the personal OEM OS kernel to run in the personal domain with little or no modification, and with exclusive read and write access to personal data storage, and transfer of voice and data over-the-air and to virtualized services such the user interface and sensors through the platform's trusted device components.

The enterprise domain runs a para-virtualized version of Android and Linux adapted to run with the OKL4 Microvisor. It provides a user-level end-to-end enterprise VPN service that tunnels through a lower level mandatory Data-in-Transit (DIT) encryption system for communication of enterprise data. Also serving the enterprise domain are Data-at-Rest (DAR) encryption systems for local storage of enterprise data.

Finally, there is a supervisor domain that establishes and manages the runtime architecture of the Trusted Smart Phone. It contains a manager for the partitions or "cells" used to house the other domains and trusted sub-subsystems, a global server for the policy that establishes and manages the dynamic virtual communication that provides the needed communication channels among the components of the architecture, and it provides the secure boot and trusted initialization of the system.

[102] Trusted path is a mutually authenticated communication path between a user and the trusted software. It should be un-spoofable provided the user understands how it operates and how to interact with it securely.

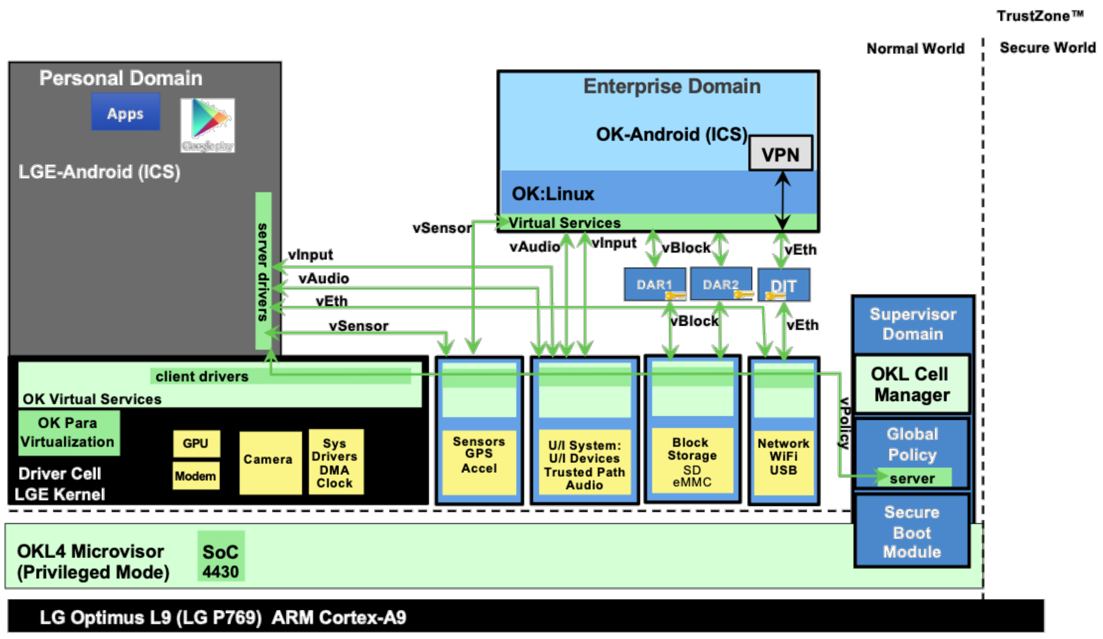


Figure 22: Design Architecture of Trusted Smart Phone

The Trusted Smart Phone has demonstrated the feasibility of a trustworthy commodity device and the utility of the MILS architectural approach.

4.3 Next Steps for MILS in IIoT

4.3.1 MILS Platform Extensions for IIoT

The MILS Platform for IIoT is an extension of the Distributed Dynamic MILS Platform, accompanied by the MILS system configuration tools, an integrated policy framework (discussed below) and the Mils™ Platform API. It should also provide the modeling and analysis tools for distributed dynamic MILS IIoT systems.

The approach to IIoT being pursued by MILS could be characterized as *Dynamic Distributed MILS to the Edge*. The realization of Dynamic Distributed MILS to the Edge depends on several extensions.

A micro-separation kernel (micro-SK) and a micro-network system (micro-NS) running on MILS micro-nodes near to or at the edge of an IIoT edge computing architecture can participate as a full peer in a distributed MILS configuration. The micro-SK/NS in combination with a minimal implementation of the Mils™ Platform API is a lightweight version of a conventional MSK+MNS MILS node providing an execution and communication environment that can run a single (or small number of) MILS subject as though it was running on a larger MILS node. Time Sensitive Networking may be used as the network medium when design constraints require deterministic communication. Smaller edge devices such as sensors and actuators can be managed by such

MILS micro-nodes. Larger MILS nodes can act as edge gateways that offer to MILS micro-nodes local cloud services and distributed MILS system connectivity and gateway services.

The *unified policy* of an IIoT MILS system may be formulated as a combination of:

- a global information flow policy that is the policy architecture of the inter-component flows,
- the local policies (programmed behaviors) of trusted components that are critical to enable the policy architecture to achieve the required system properties and
- the configurable local access control policies of distinguished policy enforcement point components placed in the architecture to enforce fine-grained access control policies.

In addition to information flow policies an integration into the MILS Platform of mechanisms to enforce application-level access control policies is planned. It is based on the very flexible Next Generation Access Control (NGAC) standard[103] which provides a flexible framework for specifying and enforcing multiple fine-grained attribute-based access control policies. It is suitable to be applied in a distributed and heterogeneous setting.[104] The NGAC framework has been extended with multi-domain policies to enable accesses by subjects in a network domain governed by one local access control policy to objects in a network domain governed by another local access control policy through a pair of edge gateways. A prototype of multi-domain distributed access control based on NGAC was developed for the FAR-EDGE Project.[105] An integrated policy modeling framework combining the information flow control of MILS policy architectures with access control is planned, to be supported by unified policy analysis methods and tools.

4.3.2 Covering Key Safety Challenges for IIoT

'*Key Safety Challenges for the IIoT'* published by Industrial Internet Consortium in December of 2017[106] articulates four key challenges unique to IIoT that affect safety characteristics:

- increased security risks due to an increased attack surface,
- convergence of IT and OT,
- pervasive autonomy and
- inadequate regulatory framework and evolving standards.

The recommendation for each of the aspects is to enforce the noninterference of IT and OT elements that share computing and communications platforms. It refers to the IEC 61508 functional safety standard, which uses the term "noninterference" or "independence" to include

[103] [Int20] Next Generation Access Control.

[104] [HBM$^+$16] Next Generation Access Control for Distributed Control Systems.

[105] [FAR] FAR-EDGE Project (2016-2019).

[106] [ZKHK17] Key Safety Challenges for the IIoT.

logical separation while considering both spatial and temporal aspects. Two components are logically separated if it is impossible for one component to affect the operation of another, even if they share a resource. This whitepaper suggests using separation kernel-based operating systems for IIoT applications requiring safety.

A MILS-based approach is capable of addressing all four key challenges.

4.3.2.1 Increased Security Risks due to an Increased Attack Surface

Security risks related to an increased attack surface expand the safety challenge in IIoT systems since there is a larger attack surface that adversaries could remotely exploit to cause unsafe system behavior. A MILS architectural approach addresses this in two ways: by controlling the communication paths and by isolating the domains. MILS reduces the increased attack surface to several communication paths and then filters signals from external agents that may cause harm. The modular design allows deactivation of domains that fail thus preventing the propagation of the faulty behavior. This may be done by a Dynamic MILS implementation that reconfigures the system in a way that supports process continuity while avoiding cascading effects, or by simply deactivating a component.

4.3.2.2 Convergence of IT and OT

MILS mitigates risks associated with convergence of IT and OT through separation of domains. Physical convergence involves hosting both OT and IT functions on the same platform. By assigning IT- and OT-related functions to different domains with controlled communication, IT and OT functions can be split properly while sustaining their joint cooperation. This can be done in a networked environment defined by a MILS Policy Architecture.

Integration between IT and OT implies physical convergence and convergence of expectations and mentalities. Many attributes we typically associate with IT systems become associated with OT systems and vice versa. The MILS concept does not break down the apparent wholeness of the system while keeping its internals properly and safely compartmentalized. MILS technology alone cannot change management or operator mindsets but the process of articulating a MILS policy architecture should bring IT and OT people together if done collaboratively.

Safety-critical systems should be developed top-down using rigorous processes (for example, the V-model development lifecycle[107]) for design and implementation through verification and validation. Moreover, systems designers should explicitly assign safety responsibilities to each system component and consider the control actions a system must implement to avoid unsafe

[107] [INC03] Vee Model of Systems Engineering Design and Integration.

situations.[108] This is simpler for systems with well-defined separation implemented within the MILS paradigm.

4.3.2.3 Pervasive Autonomy

Autonomy is the ability of the system to make its own decisions with regards to external inputs and its changing environment and to be able to continue to operate even if disconnected from the network and remote analytics. Autonomous systems in IoT are often practical applications of machine learning (ML) and artificial intelligence (AI) techniques. Safety challenges posed by autonomous systems are associated with shifting responsibilities of human operators having the need to respond to dynamically changing circumstances as well as the inherent unpredictability of ML and AI algorithms. The risks of using these algorithms cannot be easily quantified with known safety-analysis methods.[109]

Human operators maintain safety by detecting an impending unsafe situation and sometimes breaking rules by driving the system outside of its envelope or violating some normative rule. While the intentional violation of the safety constraint to prevent the more serious damage is not an option for current systems, the greater degree of responsibility requires more behavioral options for the autonomous mechanisms compared to human-assisted ones. These options should be regulated within the whole system behavioral policy that should be validated against the invariant-based safety objectives. The approach based on the policy architecture and assurance techniques developed for MILS-based systems may be applied for this validation. In general, the IoT applications with shifted, transferred and reassigned responsibilities require help with responsibility decomposition and assignment to separated components, such as provided by MILS. In case of concern or safety failure it is then less onerous to recognize the invalid assumptions about safety made by some system component and deliver the appropriate additional constraints.

It is possible to decompose safety considerations according to a “role-based model” of the system and mutually constrain the components assigned to different roles. For example, one of the domains is responsible for monitoring the whole system for security issues and blocking the attacks, and the other domain is supervising the behavior of the first one to avoid blocking actions that may affect safety. In other words, we have the roles of security agent and supervisor, and their interaction model enforces the separation of duties. The system safety mechanisms usually constrain the behavior of “intelligent” ones. At the same time the role of “intelligent” components may be to inform the safety mechanisms about the changing circumstances. Based

[108] [Lev04] Leveson. A New Accident Model for Engineering Safer Systems.

[109] Verification of ML and AI algorithms is a field of growing interest and activity as witnessed by the program of Computer Aided Verification (CAV) conference, 2020, *http://i-cav.org/2020/.*

on the resulting context, the arbitrating component(s) make decisions about changing the configuration of assumptions, rules and constraints to minimize the current operational risks for the whole system.

4.3.2.4 Inadequate Regulatory Framework and Evolving Standards

Many safety-related standards and regulatory frameworks face the challenge of an increasing number of components and interfaces in an IIoT system. Various standards and regulatory bodies address important security-related considerations and how they affect safety regulation and compliance in IIoT. Both safety and security must be considered.

Distinguishing policy architecture at the design level validates the implemented policy against safety and security objectives including those prescribed by the regulatory requirements and standards. MILS facilitates safety and security assurance with the relevant design principles.

One important desired capability of IIoT system components is "plug & play" interoperability to enable systems operators to assemble and integrate a new system quickly. For example, a medical provider could combine a set of medical sensors, actuators and control algorithms on the cloud to automate the delivery of certain therapies. The compositional approach to safety and security assurance for MILS-based systems is currently evolving into a methodology that addresses the challenge of certification for such systems.[110] This methodology animates the contract-based approach, where a component manufacturer specifies constraints on the interface(s) of the component(s) that their component is designed to work with. These components would actively check that they are being composed with other components whose contracts satisfy those constraints. Regulatory submissions would provide evidence and arguments that the component behaves safely when composed with other components that have been designed for connection using those contracts[111] (i.e., that the compositions of behaviors described in the contracts are not unsafe).

[110] [KAN+18].

[111] [KFP+15] Towards Assurance for Plug & Play Medical Systems.

Annex A ACRONYMS AND ABBREVIATIONS

AADL	Architecture Analysis and Design Language
AC	Access Control
AI	Artificial Intelligence
API	Application Programming Interface
ARINC	Aeronautical Radio, Incorporated
BLP	Bell-LaPadula
CAE	Claims, Argument, Evidence
CC	Common Criteria
CITADEL	Critical Infrastructure using Adaptive MILS
CNSS	Committee on National Security Systems
COMPASS	Correctness, Modeling, and Performance of AeroSpace Systems (ESA)
COTS	Commercial off-the-shelf
CPU	Central Processing Unit
D-MILS	Distributed MILS
DMA	Direct Memory Access
ESA	European Space Agency
ETB	Evidential Tool Bus (AM-ETB Adaptive MILS ETB)
ETCS	European Train Control System
EU	European Union
FBK	Fondazione Bruno Kessler
FPGA	Field Programmable Gate Array
GPOS	General-Purpose Operating System
GPU	Graphics Processing Unit
GSN	Goal Structuring Notation
HRA	Haselnuss Reference Architecture
IEEE	Institute of Electrical and Electronics Engineers

IFC	Information Flow Control
IIoT	Industrial Internet of Things
IISF	Industrial Internet Security Framework
IIV	Industrial Internet Vocabulary
IMA	Integrated Modular Avionics
IOMMU	Input/Output Memory Management Unit
ISO	International Standards Organization
IT	Information Technology
KSOS	Kernelized Secure Operating System
LTL	Linear-time Temporal Logic
MAS	MILS Audit System
MCS	MILS Console System
MEAS	MILS Extended Attributes System
MFS	MILS File System
MILS	Multiple Independent Levels of Security (deprecated expansion)
ML	Machine Learning
MMU	Memory Management Unit
MNS	MILS Networking System
NASA	National Aeronautics and Space Administration
NIST	National Institute of Standards and Technology
NKSR	Non-Kernel Security Related [software]
NSA	National Security Agency
OCRA	Othello Contract Refinement Analysis (FBK)
OS	Operating System
OT	Operational Technology
PCI	Peripheral Component Interconnect
PKPP	Partitioning Kernel Protection Profile

POS	Partitioning Operating System
POSIX	Portable Operating System standard
PT	Page Table
RTCA	Radio Technical Commission for Aeronautics
RTOS	Real Time Operating System
SACM	Structured Assurance Case Metamodel
SIL	Safety Integrity Level
SK	Separation Kernel
SKPP	Separation Kernel Protection Profile
SLIM	System-Level Integrated Modeling (FBK)
SoC	System on a Chip
SRI	SRI International
TCP	Transmission Control Protocol
TLB	Translation Lookaside Buffer
TSN	Time Sensitive Networking
TTE	Time Triggered Ethernet
VM	Virtual Machine
VMM	Virtual Machine Monitor
xSAP	Safety Analysis Platform (FBK)

Annex B GLOSSARY

Adaptive MILS Framework (CITADEL Framework)	The Adaptive MILS Framework, or CITADEL Framework, augments a Dynamic MILS Platform with monitoring, adaptation, reconfiguration, and certification assurance planes.
Adaptive MILS system	An MILS-based system possessing two main abilities: First, a full and flexible ability to change its configuration during runtime, which ability is provided by the Dynamic MILS Platform; and second, the ability to address environmental changes or internal failures by monitoring its operation and its interaction with the environment and by adapting to maintain safe and secure operation, which ability is provided by the monitoring, adaptation and reconfiguration planes of the Adaptive MILS Framework.
Communication policy	In a context of *MILS-based system*, the *security policy* describing the allowed pairwise communications between *domains.*
Distributed MILS	The ability to transparently deploy a MILS policy architecture over a distributed MILS platform while preserving its deterministic execution and other specified properties.
Distributed MILS Platform	A set of MILS nodes communicating over a networking technology capable of providing deterministic communication, such as hardware-based Time-Triggered Ethernet or Time Sensitive Networking. Each node consists at minimum of a distributed MILS-enabled Separation Kernel and a MILS Networking System foundational component.
Domain (or: MILS domain, security domain, safety domain)	A unit of isolation created and maintained in such a way that other domains may not interfere with it except in controlled ways as permitted by information flow policy.
Domain separation	See *separation*
Dynamic MILS	A realization of MILS providing dynamic reconfiguration of the resources exported by the separation kernel and other foundational components of the MILS platform, and their permitted information flow relationships.
Dynamic MILS Platform	A realization of the MILS Platform providing the ability to dynamically reconfigure the resources exported by the

	separation kernel and other foundational components of the MILS platform and their permitted information flow relationships.
Dynamic MILS policy architecture	The policy architecture for a MILS-based system that may change during system operation, usually under the strict control of a defined reconfiguration policy and/or a configuration transition system.
Evidential tool bus (ETB)	An automated framework that justifies a claim, made for a system model in support of certification, by building and managing an argument structure and a supporting body of evidence generated by a collection of tools that employ model checking, constraint solving and other techniques.
Foundational component	A component of the MILS Platform that exports resources created from primitive resources and composes with the other foundational components to seamlessly enforce isolation and information flow control policy on all exported resources.
Heterogeneous MILS	A MILS Platform based on an implementation of a separation kernel that runs in a heterogeneous hardware, firmware and software environment combining various technologies to take an advantage of non-CPU based computations (e.g. GPUs, FPGAs or ASICs) and parallelization of applications and processes.
Information flow control (IFC)	Procedure to ensure that information transfers within a system are not made in violation of the security policy.[112]
Isolation	The ability to keep multiple instances of software separated so that each instance only sees and can affect itself.[113]
Kernelized architecture	A system architecture based on a single central trusted component, a security kernel, that can intercept and control the system operations in support of a security policy.
MILS	A component-based approach for the construction, assurance, and certification of trustworthy systems, which emphasizes decomposition, isolation, constrained

[112] [NISTSP800-171] NIST SP 800-171 Rev. 1, NIST IR 7298r3.

[113] [NISTIR7298r3] NIST SP 800-190, NIST IR 7298r3.

	communication according to a policy, and compositional verification of system properties.
MILS Platform	A composition of hardware and software foundational components that provide isolation and information flow control over a set of resources required for the implementation of a *MILS policy architecture.* *The MILS Platform has its own distinct architecture.*
MILS Policy Architecture	A system model comprising a set of abstract entities (active subjects, passive objects, communication primitives describing permitted interactions) that represents a decomposition of the system to achieve both its functional purpose and its trustworthiness requirements.
MILS-based system	The realization of one or more MILS policy architectures on a MILS platform.
(Domain) Noninterference	The property of the system with *domain separation* guaranteeing that the execution of every domain does not undermine the *communication policy* for the *MILS-based system.* An abstract formal security model for information flow. Of particular importance to MILS is intransitive noninterference.
Operational component	A component in a MILS policy architecture.
Partition	A logical unit of separation comprising a set of resources created and maintained by a separation kernel. See also *Domain*
Partitioning	The process and the result of splitting the system into isolated compartments (domains, partitions)
Periods processing	A technique supporting partitioning by purging a physical processing resource of all information from one processing period before transitioning to the next processing period to avoid leakage of information from the domain served by the first processing period to the domain served by the next.
Policy architecture (*MILS policy architecture)*	A graph structure, the nodes of which represent subjects (active) and objects (passive) exported by a MILS platform, and the directed arcs of which represent a relation among these resources, designating permitted information flows among the related resources implemented by platform-

	supported protected operations. The policy architecture represents the system decomposition intended to achieve both the functional purpose of the system and requirements on its trustworthiness aspects and assurance.
Reference monitor	The concept of a system component that mediates all references by subjects to objects in a system and ensures that these references satisfy a specific security policy.
Security kernel	Hardware, firmware, and software elements of a trusted computing base implementing the reference monitor concept. Security kernel must mediate all accesses, be protected from modification, and be verifiable as correct.[114]
Security policy (security model)	A set of criteria for the provision of security services.[115] A set of rules that governs all aspects of security-relevant system and system element behavior.[116] The set of rules and constraints formally defining what controlled operations are permitted (and/or prohibited) or what security properties are to be maintained for a particular system. A formalized security policy is sometimes referred to as a security model.
Separation	a) A synonym for *isolation* (strict isolation), particularly at the inception of its use by Rushby to describe the operation of a *separation kernel*. b) The combination of *isolation* and *information flow control* policies as enforced by a *separation kernel*.
Separation kernel	a) A specific kind of security kernel that is limited to the enforcement of policies of isolation and information flow control among exported resources by management of shared physical resources to create a noninterference relation that is indistinguishable from that provided by a physically distributed system.

[114] [NISTSP800-53r4] CNSSI 4009-2015, NIST SP 800-53 Rev. 4.

[115] [CNSSI4009] CNSSI 4009.

[116] [NISTSP800-160] NIST SP 800-160v1.

	b) Hardware and/or firmware and/or software mechanisms whose primary function is to establish, isolate and separate multiple partitions and control information flow between subjects and exported resources allocated to those partitions.[117] A MILS separation kernel is the essential base foundational component of a MILS platform.
Spatial partitioning	A type of partitioning that compartmentalizes system resources in space and maintains data separation, information flow control and fault isolation.
Static MILS	A realization of MILS providing only static (not changing over time) configuration of the resources exported by the separation kernel and other foundational components of the MILS platform, and their permitted information flow relationships.
Static MILS policy architecture	The policy architecture for a MILS-based system that does not change during system operation, by contrast with a *dynamic MILS policy architecture.*
Temporal partitioning	A type of partitioning that compartmentalizes system resources in time by scheduling and processing of various types of information at distinct times.
Trusted computing base, TCB	The totality of protection mechanisms within a computer system—including hardware, firmware, and software--the combination of which is responsible for enforcing a security policy.

[117] [SKPP] U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness. Version 1.03.

Annex C REFERENCES

[Aer19a] Aeronautical Radio, Inc. (ARINC). *Avionics Application Software Standard Interface, Part 0, Overview of ARINC 653*, ARINC 653P0-2, August 2019.

[Aer19b] Aeronautical Radio, Inc. (ARINC). *Avionics Application Software Standard Interface, Part 1, Required Services*, ARINC 653P1-5, December 2019.

[And72] J. P. Anderson. Computer security technology planning study. Technical Report ESD-TR-73-51, US Air Force, October 1972. (Two volumes, available at *http://seclab.cs.ucdavis.edu/projects/history/seminal.html*).

[ARB$^+$14] Lacramioara As¸tefa˘noaei, Souha Ben Rayana, Saddek Bensalem, Marius Bozga, and Jacques Combaz. Compositional Invariant Generation for Timed Systems. In Ábrahám E. and Havelund K., editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2014)*, volume 8413 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2014.

[BB79] T. A. Berson and G. L. Barksdale. KSOS-development methodology for a secure operating system. In *Conference Proceedings: 1979 National Computer Conference*, volume 48, New York, NY, USA, June 1979. American Federation of Information Processing Societies, AFIPS Press. (Available at *https://www.computer.org/csdl/pds/api/csdl/proceedings/download-article/12OmNy6HQSf/pdf*).

[BDRS08] Carolyn Boettcher, Rance DeLong, John Rushby, and Wilmar Sifre. The MILS Component Integration Approach to Secure Information Sharing. In *27th AIAA/IEEE Digital Avionics Systems Conference*, 2008. (Available at *http://www.csl.sri.com/users/rushby/papers/dasc08.pdf*).

[BKZ+18] Henk Birkholz, Christoph Krauß, Maria Zhdanova, Don Kuzhiyelil, Tolga Arul, Markus Heinrich, Stefan Katzenbeisser, Neeraj Suri, Tsvetoslava Vateva-Gurova, and Christian Schlehuber. A Reference Architecture for Integrating Safety and Security Applications on Railway Command and Control Systems. In *International Workshop on MILS: Architecture and Assurance for Secure Systems, MILS@DSN 2018, Luxembourg.* Zenodo, June 2018. (Available at *https://zenodo.org/record/1314095#.YAaOVS1h2XE* .

[BQIR14] Denis Bytschkow, Jean Quilbeuf, Georgeta Igna, and Harald Ruess. Distributed MILS Architectural Approach for Secure Smart Grids. In J. Cuellar, editor, *SmartGridSec 2014*, LNCS 8448, pages 16–29. Springer International Publishing, Switzerland, 2014. (Available at *https://st1.ning.com/topology/rest/1.0/file/get/1154165430?profile=original*).

[BTL+14] Holger Blasum, Sergey Tverdyshev, Bruno Langenstein, Jonas Maebe, Bjorn De Sutter, Bertrand Leconte, Benoît Triquet, Kevin Müller, Michael Paulitsch, Axel Söding Freiherr von Blomberg, and Axel Tillequin. MILS Architecture (EURO-MILS whitepaper), 2014. (Available at *https://euromils-project.technikon.com/downloads/2014-EURO-MILS-MILS-Architecture-white-paper.pdf*).

[Cav] Ann Cavoukian. Privacy by design, The 7 Foundational Principles. (Available at *https://www.ipc.on.ca/wp-content/uploads/Resources/7foundationalprinciples.pdf*).

[CEH+20] Sandy Carielli, Matt Eble, Frederick Hirsch, Ekaterina Rudina, and Ron Zahavi. *IoT Security Maturity Model: Description and Intended Use*. Industrial Internet Consortium, version 1.2, May 2020. (Available at *https://iiconsortium.org/pdf/SMM_Description_and_Intended_Use_V1.2.pdf*).

[CHRZ19] Sandy Carielli, Frederick Hirsch, Ekaterina Rudina, and Ron Zahavi. *IoT Security Maturity Model: Practitioner's Guide*. Industrial Internet Consortium, version 1.0, February 2019. (Available at *https://iiconsortium.org/pdf/IoT_SMM_Practitioner_Guide_2020-05-05.pdf*).

[CITa] The CITADEL Introduction to MILS. *http://www.citadel-project.org/introduction-to-mils* .

[CITb] The CITADEL Project Web site. *http://www.citadel-project.org/* .

[CIT18a] CITADEL Modeling and Specification Languages. Technical Report D3.1, Version 2.3, CITADEL Project, August 2018. (Available at *https://st2.ning.com/topology/rest/1.0/file/get/14688988?profile=original*).

[CIT18b] CITADEL Verification Techniques and Tools. Technical Report D3.2, Version 1.0, CITADEL Project, September 2018. (Available at *https://st1.ning.com/topology/rest/1.0/file/get/153640922?profile=original*).

[CNS15] *Committee on National Security Systems (CNSS) Glossary*. Committee on National Security Systems, April 2015. (Available at *https://www.serdp-estcp.org/content/download/47576/453617/file/CNSSI%204009%20Glossary%202015.pdf*).

[CS09] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. Technical re-port, Department of Computer Science, Cornell University, October 2009. (Available at *https://www.cs.cornell.edu/~clarkson/papers/clarkson_hyperproperties_journal.pdf*).

[CST18] Alessandro Cimatti, Ivan Stojic, and Stefano Tonetta. Formal Specification and Verification of Dynamic Parametrized Architectures. In *Proceedings of the 22nd International Symposium on Formal Methods*, pages 625–644. Springer, Cham, July 2018. (Available at *https://link.springer.com/content/pdf/10.1007%2F978-3-319-95582-7_37.pdf*).

[CvdM12] Stephen Chong and Ron van der Meyden. Using Architecture to Reason About Information Security. In *Proceedings 6th Layered Assurance Workshop*, pages 1–11, 2012. (Available at *http://www.acsac.org/2012/workshops/law/2012-law-proceedings.pdf*).

[D-M] The D-MILS Project Web Site. Distributed MILS, project funded by the European Union's Seventh Framework Programme, *http://www.d-mils.org/* .

[D-M13] Requirements for Distributed MILS technology. Technical Report D1.3, Version 1.2, D-MILS Project, August 2013. (Available at *https://st2.ning.com/topology/rest/1.0/file/get/1154161314?profile=original*).

[D-M14] Compositional verification techniques and tools for distributed MILS–Part 1. Technical Report D4.4, Version 1.0, D-MILS Project, July 2014. (Available at *https://st2.ning.com/topology/rest/1.0/file/get/1154164405?profile=original*).

[D-M15] Compositional verification techniques and tools for distributed MILS–Part 2. Technical Report D4.5, Version 1.0, D-MILS Project, September 2015. (Available at *https://st1.ning.com/topology/rest/1.0/file/get/1154166001?profile=original*).

[DeL12] Rance DeLong. Delivery, Configuration and Initialization of MILS Components and Integrations. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, 2012.

[DeL13a] Rance DeLong. Commentary on the MILS Platform Protection Profile. (Available at *https://www.researchgate.net/publication/348559519_Commentary_on_the_MILS_Platform_Protection_Profile_Version_003_draft#fullTextFileContent*), April 2013.

[DeL13b] Rance DeLong. MILS Platform Protection Profile. (Available at *https://www.researchgate.net/publication/348559515_MILS_Platform_Protection_Profile*), January 2013.

[DHR09] Rance DeLong, David Hanz, and John Rushby. An Application of the MILS Approach to Secure Information Sharing. (Available at *http://www.csl.sri.com/users/rushby/papers/mils-example.pdf*), November 2009.

[dlPBT12] Antonio de la Piedra, An Braeken, and Abdellah Touhafi. Sensor Systems Based on FPGAs and Their Applications: A Survey. In *Sensors 2012*, volume 12, pages 12235–12264, September 2012. (Available at *https://www.mdpi.com/1424-8220/12/9/12235/htm*).

[Dop18] Josef Doppelbauer. Command and Control 4.0. IRSE News, Issue 246, July/August 2018. (Available at *https://www.era.europa.eu/sites/default/files/library/docs/command_and_control_en.pdf*).

[EURa] The EURO-MILS Project Web site. EURO-MILS: Secure European Virtualisation for Trustworthy Applications in Critical Domains, project funded by the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 318353 *http://www.euromils.eu/* .

[Eurb] European Union Agency for Railways. European Rail Traffic Management System (ERTMS). www.era.europa.eu. *https://www.era.europa.eu/sites/default/files/activities/docs/ertms_making_railway_system_work_better_for_society_general_en.pdf* .

[Eur17] European Committee for Electrotechnical Standardization (CENELEC, CENELEC Central Secretariat, rue de Stassart, 36, B-1050 Brussels. *Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS) Part 1: Basic requirements and generic process*, EN50126:1999 E, October 2017.

[FAR] The FAR-EDGE Project Web site. *http://www.far-edge.eu/* , see also *https://www.edge4industry.eu/* .

[FS16] Igor Furgel and Viola Saftig. *Common Criteria Protection Profile "Multiple Independent Levels of Security: Operating System"*, version 2.3, 2016.

[FSW+15] Igor Furgel, Viola Saftig, Tobias Wagner, Kevin Müller, Reinhard Schwarz, and Axel Söding-Freiherr von Blomberg. Non-Interfering Composed Evaluation. Technical report, EURO-MILS: Secure European Virtualisation for Trustworthy Applications in Critical Domains, September 2015.

[HAS] The HASELNUSS Project Web site. *https://haselnuss-projekt.de/* .

[HBM+ 16] J. H. Huh, R. B. Bobba, T. Markham, D. M. Nicol, J. Hull, A. Chernoguzov, H. Khurana, K. Staggs, and J. Huang. Next-generation access control for distributed control systems. *IEEE Internet Computing*, 20(5):28–37, 2016.

[HCAK11] R. Hawkins, K. Clegg, R. Alexander, and T. Kelly. Using a Software Safety Argument Pattern Catalogue: Two Case Studies. In *30th International Conference on Computer Safety, Reliability and Security (SAFECOMP '11)*, 2011.

[HKH15] Richard Hawkins, Tim Kelly, and Ibrahim Habli. Developing Assurance Cases for D- MILS Systems. In *MILS Workshop*, 2015.

[IIS16] Industrial Internet Consortium. *Industrial Internet of Things, Volume G4: Security Framework*, September 2016. (Available at *https://www.iiconsortium.org/pdf/IIC_PUB_G4_V1.00_PB.pdf* .

[INC03] INCOSE. Vee Model of Systems Engineering Design and Integration. In *INCOSE G2SEBOK 3.30*. International Council on Systems Engineering, September 2003. (Available at *https://web.archive.org/web/20070927005332/http://g2sebok.incose.org/app/qualsys/view_by_id.cfm?ID=INCOSE%20G2SEBOK%203.30&ST=F*).

[Int20] InterNational Committee for Information Technology Standards, Cyber security technical committee 1. *American National Standard for Information Technology— Next Generation Access Control (NGAC)*. ANSI, INCITS 565-2020, April 2020. (Available at *https://standards.incits.org/apps/group_public/project/details.php?project_id=2328*).

[Joi20] Joint Task Force. *NIST SP800-53 Rev 5 Security and Privacy Controls for Federal Information Systems and Organizations*. National Institute for Standards and Technology, September 2020. (Available at *https://doi.org/10.6028/NIST.SP.800-53r5*).

[KAN+ 18] Dorien Koelemeijer, Rasma Araby, Ayoub Nouri, Marius Bozga, and Rance DeLong. A model-based approach to certification of adaptive MILS. In *International Workshop on MILS: Architecture and Assurance for Secure Systems, MILS@DSN 2018, Luxembourg.* Zenodo, June 2018. (Available at *https://zenodo.org/record/1306089/files/Koelemeijer2018a-model-based-approach.pdf* <10.5281/zenodo.1306089>. <hal-01889050>).

[KE19] Wolfgang Kampichler and Dieter Eier. An adaptive MILS Architecture for Resilient Remote Tower Communication Services. In *2019 IEEE/AIAA 38th*

Digital Avionics Systems Conference (DASC), pages 1–7, September 2019. (Available at *https://www.researchgate.net/publication/341078657_An_adaptive_MILS_Architecture_for_Resilient_Remote_Tower_Communication_Services*).

[KFP+ 15] Andrew King, Lu Feng, Sam Procter, Sanjian Chen, Oleg Sokolsky, John Hatcliff, and Insup Lee. Towards Assurance for Plug & Play Medical Systems. September 2015. (Available at *http://repository.upenn.edu/cgi/viewcontent.cgi?article=1839&context=cis_papers*).

[KG93] H. Kopetz and G. Grunsteidl. Ttp - a time-triggered protocol for fault- tolerant real-time systems. In *The Twenty-Third International Symposium on Fault-Tolerant Computing*, volume FTCS-23, 1993. (Available at *https://www.computer.org/csdl/pds/api/csdl/proceedings/download-article/12OmNzuZUsJ/pdf*).

[Lev04] Nancy Leveson. A New Accident Model for Engineering Safer Systems. *Safety Science*, 42(4):237–270, April 2004. (Available at *http://sunnyday.mit.edu/accidents/safetyscience-single.pdf*).

[LMBC+03] Lockheed-Martin, Boeing, Rockwell Collins, Green Hills Software, LynuxWorks, Objective Interface, and University of Idaho. *Protection Profile for Partitioning Kernels in Environments Requiring Augmented High Robustness*. Version 1.3, June 2003. Prepared for The Open Group and NSA IAD.

[MD79] E. J. McCauley and P. J. Drongowski. KSOS–The design of a secure operating system. In *Conference Proceedings: 1979 National Computer Conference*, volume 48, pages 345–353, New York, NY, USA, June 1979. American Federation of Information Processing Societies, AFIPS Press. (Available at *https://www.computer.org/csdl/pds/api/csdl/proceedings/download-article/12OmNB8TUer/pdf*).

[MLGM11] R. Mahapatra, J. Lee, N. Gupta, and R. Manners. Microprocessor Evaluations for Safety-Critical, Real-Time Applications: Authority for Expenditure No. 43 Phase 5 Report. Technical Report DOT/FAA/AR-11/5, U.S. Department of Transportation, Federal Aviation Administration, Air Traffic Organization NextGen & Operations Planning Office of Research and Technology Development, Washington, DC, USA, May 2011. (Available at *https://www.faa.gov/aircraft/air_cert/design_approvals/air_software/media/11-5.pdf*).

[Obj18] Object Management Group (OMG). Structured Assurance Case Metamodel (SACM), Version 2.0, March 2018. (Available at *http://www.omg.org/spec/SACM/2.0/PDF*).

[OWA] OWASP—Open Web Application Security Project. Security by Design Principles. In *OWASP Foundation Wiki*. (Available at *https://wiki.owasp.org/index.php/Security_by_Design_Principles*).

[Par16] Paul J. Parkinson. Applying MILS to multicore avionics systems. In *2nd International Workshop on MILS: Architecture and Assurance for Secure Systems*, volume HiPEAC, Prague, Czech Republic, 2016. (Available at *https://zenodo.org/record/47978/files/09_EuroMILS2016_PaulParkinson_WindRiver_MILS_Multicore_paper_FINAL.pdf*).

[PHA] The PHANTOM Project Web site. *https://www.phantom-project.org* .

[PKK+79] Gerald J. Popek, Mark Kampe, Charles S. Kline, Allen Stoughton, Michael Urban, and Evelyn J. Walton. UCLA Secure UNIX. In *Conference Proceedings: 1979 National Computer Conference*, volume 48, pages 355–364, New York, NY, USA, June 1979. American Federation of Information Processing Societies, AFIPS Press. (Available at *https://www.computer.org/csdl/pds/api/csdl/proceedings/download-article/12OmNzcPAEy/pdf*).

[RD07] John Rushby and Rance DeLong. Compositional Security Evaluation: The MILS Approach. In *Proceedings of the International Common Criteria Conference*, September 2007. (Available at *http://www.csl.sri.com/users/rushby/slides/iccc07.pdf*).

[RPG+19] Ron Ross, Victoria Pillitteri, Richard Graubart, Deborah Bodeau, and Rosalie McQuaid. *NIST SP800-160 Systems Security Engineering: Cyber Resiliency Considerations for the Engineering of Trustworthy Secure Systems*. National Institute for Standards and Technology, November 2019. (In two volumes available at *https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160v1.pdf* and *https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160v2.pdf*).

[RR83] John Rushby and Brian Randell. A Distributed Secure System. Technical report, University of Newcastle upon Tyne, 1983.

[RTC05] RTCA SC-200 / EUROCAE WG-60. *Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations*. Radio Technical

Commission for Aeronautics, Inc. (RTCA) / European Organisation for Civil Aviation Equipment (EUROCAE), DO-297 edition, November 2005.

[RTP03] *Transport protocol for real-time applications, RFC 3550*, July 2003.

[Rus81] John Rushby. The Design and Verification of Secure Systems. In *Eighth ACM Symposium on Operating System Principles*, pages 12–21, Asilomar, CA, December 1981. (ACM *Operating Systems Review*, Vol. 15, No. 5, available at *http://www.csl.sri.com/users/rushby/papers/sosp81.pdf*).

[Rus81] John Rushby. Kernels for Safety? In *Safe and Secure Computing Systems*, pages 210– 220. Blackwell Scientific Publications, 1989. (proceedings of a symposium held in Glasgow, UK, October 1986).

[Rus99] John Rushby. Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance. NASA Contractor Report CR-1999-209347, NASA Langley Research Center, June 1999. Available at *http://www.csl.sri.com/users/rushby/papers/faaversion.pdf* , and *https://ntrs.nasa.gov/api/citations/19990052867/downloads/19990052867.pdf* ; also issued by the FAA.

[Rus07] John Rushby. Compositional Certification for MILS. In *Proceedings of the High Confidence Software and Systems Conference*, May 2007. (Available at *http://www.csl.sri.com/users/rushby/slides/iccc07.pdf*).

[Rus08] John Rushby. Separation and Integration in MILS: The MILS Constitution. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, February 2008. (Available at *http://www.csl.sri.com/users/rushby/papers/mils-constitution.pdf*).

[SAE] SAE International. *Time-Triggered Ethernet (SAE AS6802)*. (Available at *https://www.sae.org/standards/content/as6802/*).

[Sam15] V. P. Sampath. FPGA for Internet of Things. electronicsforu.com, November 2015. (Available at *https://iot.electronicsforu.com/expert-opinion/fpga-internet-things/*).

[Sch75] W. L. Schiller. The design and specification of a security kernel for the PDP-11/45. Technical Report MTR-2934, ESD-TR-69, The MITRE Corporation, Bedford, MA, March 1975. (Available at *http://seclab.cs.ucdavis.edu/projects/history/papers/schi75.pdf*).

[SIP02] *Session Initiation Protocol, RFC 3261*, June 2002.

[SKP07] Information Assurance Directorate, National Security Agency, Fort George G. Meade, MD 20755-6000. *U.S. Government Protection Profile for Separation*

Kernels in Environments Requiring High Robustness, Version 1.03 edition, June 2007. (Available at *https://www.niap-ccevs.org/MMO/PP/pp_skpp_hr_v1.03.pdf*).

[SRO^{+}08] Paul A. Schulte, Richard Rinehart, Andrea Okun, Charles L. Geraci, and Donna S. Heidel. National Prevention through Design (PtD) Initiative. *Journal of Safety Research*, 39(2):115 – 121, 2008. (Available for purchase at *https://www.sciencedirect.com/science/article/abs/pii/S0022437508000054*).

[TCS85] *Department of Defense Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD, 1985.

[VF10] Andrius Velykis and Leo Freitas. Formal Modelling of Separation Kernel Components. In *ICTAC 2010: Theoretical Aspects of Computing – ICTAC 2010*, pp 230-244, 2010. (Available at http://andrius.velykis.lt/publications/VelykisF10.pdf)

[vSGBR18] Stephan van Schaik, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. Malicious Management Unit: Why Stopping Cache Attacks in Software is Harder than You Think. In *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, pages 937–954, USA, 2018. USENIX Association. (Available at *https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-van_schaik.pdf*).

[Woo79] J. P. L. Woodward. Applications for multilevel secure operating systems. In *Conference Proceedings: 1979 National Computer Conference*, volume 48, pages 319–328, New York, NY, USA, June 1979. American Federation of Information Processing Societies, AFIPS Press. (Available at *https://www.computer.org/csdl/pds/api/csdl/proceedings/download-article/12OmNyS6RD4/pdf*).

[ZKHK17] Qinqing Zhang, Andrew King, Frederick Hirsch, and Semen Kort. *Key Safety Challenges for the IIoT*, December 2017. (Available at *https://iiconsortium.org/pdf/Key_Safety_Challenges_for_the_IIoT.pdf*).

[ZMR^{+} 18] Rehman Zafar, Anzar Mahmood, Sohail Razzaq, Wamiq Ali, Usman Naeem, and Khurram Shehzad. Prosumer based energy management and sharing in smart grid. In *Renewable and Sustainable Energy Reviews*, volume 82, pages 1675–1684. Elsevier, February 2018. (Available at *https://www.sciencedirect.com/science/article/abs/pii/S1364032117310894*).

5 AUTHORS AND LEGAL NOTICE

This document is a work product of the Trustworthiness Task Group in the IIC, co-chaired by Marcellus Buchheit (WIBU-Systems AG), Frederick Hirsch (Upham Security), and Robert Martin (MITRE). The Trustworthiness Task Group is the essential part of the IIC Security Working Group, co-chaired by Keao Caindec (Farallon Technology Group) and Vyacheslav Zolotnikov (Kaspersky).

Authors: The following persons contributed substantial written content to this document: Rance J. DeLong (The Open Group) and Ekaterina Rudina (Kaspersky).

Contributor: The following person contributed valuable ideas and feedback that significantly improved the content and quality of this document: Frederick Hirsch (Upham Security).

Technical Editor: Stephen Mellor (IIC staff) oversaw the process of organizing the contributions of the above Authors and Contributors into an integrated document.