Composable Mission-Critical Embedded System Architecture for High Assurance

Michael Vai[1], Eric Simpson[1], Alice Lee[1], Huy Nguyen[1], Jeffrey Hughes[1], Ben Nahill[1], Jeffery Lim[1], Roger Khazan[1], Sean O'Melia[1], and Fred Schneider[2]
[1]MIT Lincoln Laboratory, [2]Cornell University
PoC: mvai@ll.mit.edu

Abstract—**Mission-critical systems must go through a laborious and lengthy high assurance certification process. Slight modifications of a certified system often trigger a new certification cycle. We have leveraged a Modular Open Systems Approach (MOSA) and developed a composable Embedded-Security-as-a-Service (ESaaS) architecture for mission-critical embedded systems. A zero-trust approach has been applied to incorporate security and resilience technologies and address mission assurance requirements. In this paper, we discuss an ecosystem that supports the acquisition and certification processes of high assurance ESaaS modular embedded systems for critical missions.**

Keywords—MOSA; ESaaS; mission assurance; composable architecture; root-of-trust; zero-trust; certification

I. Introduction

Mission-critical applications, such as aircraft safety and national defense, require high assurance, as the consequences of failure can be loss of human life or compromise of national security. Military missions are multi-domain and multi-modal and increasingly dependent on C6ISR (Command, Control, Communications, Computers, Cyber, Combat, Intelligence, Surveillance, and Reconnaissance) capabilities. Various embedded systems (e.g., an airborne radar signal processor, a secure radio, etc.) have been designed and optimized for these and other dedicated functions.

As embedded systems have been tightly coupled to specific functions, vendors, and uses, military platforms often carry several dozen embedded functions to support their missions. The integration, operation, and management of these systems are complex and also cause additional overheads in size, weight, power consumption, and operating costs (SWaPC).

With today's powerful embedded technologies, multiple applications are regularly integrated into a single unit using a modular approach. With standardized interfaces, such embedded systems are also upgradeable to keep up with new technologies.

Indeed, defense acquisition programs are required to take a Modular Open Systems Approach (MOSA) to enable the incorporation of severable components to promote competition, technology refresh, and reuse [1]. Open standards enable the development of composable embedded systems for enhanced flexibility and agilty. Standardized physical, electrical, and logical interfaces allow plug-in modules conforming with the standard to readily work together within a system. Multiple capabilities could be designed as modules and integrated into a single system.

Mission-critical systems are apparent targets of adversarial attacks, and must be certified for high assurance (i.e., confidence in their security and resilience features) before receiving authorization to operate. In many cases, the arduous and lengthy certification process dominates the cost and latency of developing mission-critical systems and overshadows MOSA benefits.

The development of security and mission assurance for MOSA systems has lagged behind the standardization of electrical and physical interfaces [2]. Without a standard, different vendors, and even products from the same vendor, often implement diverse, proprietary security postures. As a result, the security of a MOSA system becomes a closed design and often breaks by adding, removing, or replacing modules. The implications of composable and upgradeable MOSA systems in their certification (for its original system design) and re-certification (for an upgraded system) processes remain an open question.

There are a few ways to reduce certification timelines: one is to apply more resource (e.g., hire more certifiers), another is through increased efficiency, and lastly through re-use. The heavy workload and the required subject matter expertise and experience render the approach of obtaining more certification resource impractical at the moment. We propose to leverage the modularity emphasized in a MOSA to increase efficiency through standardization and reusability.

We have been developing a composable Embedded-Security-as-a-Service (ESaaS) architecture with a system manager that provides common security and management services to payloads [2]. This paper describes the benefits of an ESaaS architecture in its verification and certification for high mission assurance.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering.

In the rest of this paper, we first briefly describe current strategies in the acquisition and operation of mission systems. After that, we provide an overview of the Lincoln Laboratory ESaaS architecture and provide a context for composable embedded systems. The leverage of modularity and reusability in MOSA to improve the efficiency of certification and re-certification of composable ESaaS modular systems will then be discussed.

It should be noted that our discussion with respect to certification does not necessarily reflect the view of authorities. We are in the process of engaging with relevant authorities so that we could receive guidance for the further development to streamline the evaluation and certification processes.

II. Acquisition and Operation Strategies

National Defense Authorization Act Section 805 states that defense acquisition programs are required to take the Modular Open Systems Approach [1]. Also, Executive Order 14028 and Memorandum M-22-09 describe a government-wide effort to migrate to a Zero Trust Architecture mandate [5].

A. Modular Open Systems Approach (MOSA)

MOSA is an integrated business and technical strategy for designing and acquiring affordable and adaptable systems [1]. A modular architecture is developed using highly cohesive, loosely coupled, and severable modules with interfaces designed according to standards with which conformance can be verified. Such an architecture supports an open business model to incrementally add, modify, replace, and remove system components across the system life cycle.

On the technical side, a modular design approach has been shown to improve system design and maintenance processes by allowing reusability, workload distribution, and easier debugging. Technology could be rapidly deployed, upgraded, and refreshed. On the business side, MOSA has been used to reduce schedule and save cost in system acquisition, mission integration, and sustainment.

MOSA has enabled the establishment of a variety of industry and government initiated open systems standards. OpenVPX [3] has long been the defacto industry standard for embedded system development. By standardizing electrical and physical properties of the backplane, OpenVPX enables plug-in modules conforming with the standard to readily work together within a system.

System security and assurance requirements must be established with respect to the mission's application domain and are difficult to develop without specific mission concept of operations (CONOPS). The success of OpenVPX is due to that it is generally application domain agnostic. This property, which is a definite advantage in application development, has negatively affected the development of security and assurance in embedded systems [2].

Different payload modules, and even products from the same vendor, often have different security postures. Designers then build mission specific security and assurance upon payload specific, often incompatible schemes. This is an error prone design approach and the result is difficult to reuse and update. As a consequence, any modifications could easily break the system security profile, causing a new cycle of certification and hindering MOSA advantages. Recently, the development of domain specific open systems standards, such as Sensor Open Systems Architecture (SOSA) [4], have begun to address security.

SOSA is a revolutionary standard currently being developed by a consortium for the application domains of electro-optical/infrared sensors, electronic warfare, radar, and signal intelligence. It has leveraged many industry and government standards and initiatives and defined an architecture with physical, electrical, and logical interface standards for sensor development and acquisition. SOSA has significantly improved OpenVPX in terms of security. The recently released technical standard has begun to define the requirements on security services such as authentication, zeroization, verification, key management, encryption, audit, etc., enabling it to be considered as a baseline for the development of high assurance applications. We have adapted SOSA to implement a proof-of-concept, composable modular architecture prototype in order to leverage existing government and industry investment and acceptance.

B. Zero-Trust (ZT)

The government has called for a shift to a ZT architectecture, particularly for enterprise and cloud computing [5]. For the development of embedded systems, ZT offers a new perspective of designing for mission assurance [6].

Zero trust is a set of principles that treats every component, service, and use of a system as continuously exposed to and potentially compromised by a malicious adversary. At a high level, a ZT architecture depends on three attributes: compartmentalized access, continuous monitoring and adjustment, and applying security measures throughout the overall system. A ZT enterprise system can thus support its intended mission by following the ZT security principles [6]:

- Identity verification – strong multi-factor user and device authentication;
- Access control – secure and approved access to resources;
- Resource protection – fine-grained control of approved resource utilization based on identity;
- Policy and orchestration – dynamic management of system use;
- Monitoring and analytics – analysis of system usage and security functions;
- Continuous operations – process to manage risks while supporting usability.

These ZT principles, succinctly captured as the "never trust, always verify" tenet, are well associated with the need to enforce minimization, isolation, least privilege, monitoring, and recovery in the development of high assurance embedded systems. A ZT approach has been applied in a variety of embedded system designs to incorporate security and resilience technologies and enhance mission assurance.

III. Embedded-Security-as-a-Service (ESaaS)

ESaaS is an ongoing Lincoln Laboratory effort that uses a MOSA to revolutionize how mission-critical, composable embedded systems are developed [2]. Figure 1 shows an example ESaaS modular architecture aligned with SOSA, which consists of several mission-specific, programmable payloads, a switch, and a system manager, housed in a single chassis. Representative payload modules are programmable processors such as single board computers, accelerators such as General-Purpose Graphic Processing Units (GPGPUs), and the popular Multiple Processor System-on-Chips (MPSoCs). In the current Commercial Off-The-Shelf (COTS) market, the security posture of a payload module ranges from having no security features, adapting a crypto-processor such as a Trusted Platform Module (TPM) [7], to incorporating a proprietary security technology [8].

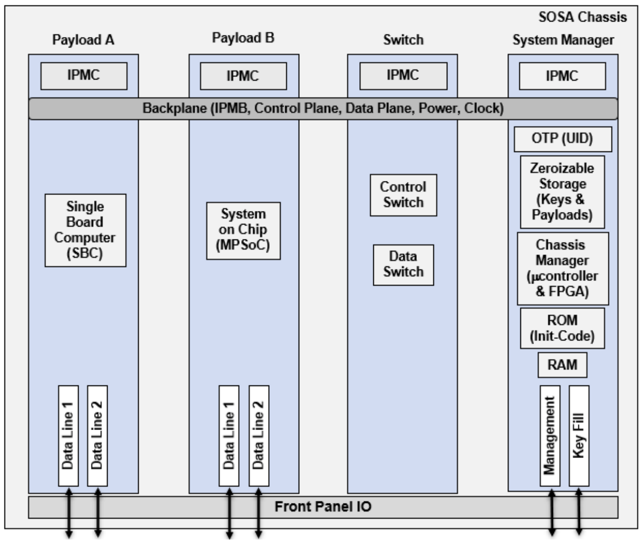

Figure 1: SOSA-aligned ESaaS modular architecture (UID: Unique Device ID, IPMC: Intelligent Platform Management Controller, IPMB: Intelligent Platform Management Bus, Init-Code: power-on bootstrapping code).

A. System Manager

Instead of relying on individual payload security features to develop system-level security, the system manager extends the SOSA chassis manager function and provides common security services (cryptography, key management, etc.) for reusability and updatability. The system manager provides a root of trust that is attestable by the mission operator. System configurations and updates are done through the system manager with government mandated, payload agnostic, updatable processes.

We now explain the process that the system manager establishes a root-of-trust and sets up a chain-of-trust. The ESaaS modular architecture in Figure 2 represents a distributed high assurance processing system containing independent payload modules. As all of them are critical to the mission operation, care must be taken to assure the integrity and authenticity of software/firmware loaded onto each and that the overall state of the system is consistent with expectation. This can be achieved by authenticating the payload modules and measuring each stage of software loaded during the boot process providing assurance that each stage has not been modified and that each was derived from a trusted source. The steps used to establish a chain-of-trust are summarized in Table 1.

Table 1: ESaaS modular architecture chain-of-trust establishment steps (Yellow highlighted steps: implicitly trusted, Green highlighted steps: trusted, @: attested).

		Chain-of-Trust Establishment Steps →								
System Modules	Components	Initialize CM	Configure SM	Provision Mission Keys	Discover Payloads	Load Payload Images	Bootstrap Payloads	Load Payload RTE/APP	Key Payloads	...
System Manager (SM)	Chassis Manager (CM)	Boot	🟩	🟩	🟩	🟩		🟩	🟩	
	IPMC	Boot			🟩					
	Crypto Core (CC)		CM @	🟩		🟩		🟩	🟩	
	Key Storage			CC @		🟩			🟩	
	PL Images					CC @		🟩		
	Management Interface		CM @	🟩		🟩				
	Key Fill Interface		CM @	🟩						
	CIK Interface		CM @	🟩						
Switch	IPMC	Boot			🟩					
	Control Switch				IPMC @					
	Data Switch				IPMC @					
Payload A SBC	IPMC	Boot			🟩					
	HW				IPMC @		🟩	🟩	🟩	
	SW Images						Boot	CC @	CC @	
Payload B MPSoC	IPMC	Boot			🟩					
	HW				IPMC @		🟩	🟩	🟩	
	FW/SW Images						Boot	CC @	CC @	

The system boot process is designed to minimize the elements that must be implicitly trusted. Most COTS programmable devices require a small segment of implicitly trusted bootstrapping code to set up interfaces, verify software signatures, etc. before further attestation could happen, which are highlighted yellow in Table 1. It is highly desirable to leverage advanced COTS products as they offer instant technology insertion and availability opportunities, thus helping to accelerate the development and reduce the cost of government-sponsored R&D activities. We have used best practice to reduce associated risks and minimized posting requirements on COTS products. Powering up details are explained as follows.

B. Powering Up

The system manager module is powered up before other modules to set up a root-of-trust regardless of what mission specific functionality is currently deployed. Referring to the architecture in Figure 2, the system manager contains a Unique Device ID (UID) and UID derived mission keys. Zeroizable storage is used to store keys that validate stored certificates to prove root-of-trust authenticity and other essential data. An assembly of a microcontroller and an FPGA provides chassis management functions and common security services.

The microcontroller powers up and runs its ROM based init-code and first-stage bootloader, which is a small segment of bootstrapping code that has to be trusted. The first-stage bootloader sets up an interface with the operator, encrypts its UID and certificate with the operator public key to authenticate itself to the operator. The operator validates the UID from a list of valid UIDs and sends a symmetric key encrypted with the system manager's public key (from its certificate) and signed. The first-stage bootloader decrypts the symmetric key and saves it in the zeroizable storage. The operator sends the system manager FPGA firmware encrypted with the symmetric key and signed by its software trust anchor/authority. The system manager module decrypts the firmware and sets up a crypto core in the FPGA. The chassis manager reboots with code verified and authenticated by the crypto core and the system manager is now in operation.

C. *Mission Provisioning*

After authentication, the operator provisions the system manager for a mission. All keys and certificates required for the mission are securely loaded into zeroizable storage using a key loader through the key fill interface. A mission specific payload profile (device list, run-time environments, applications, configuration parameters, etc.) is also loaded into the zeroizable storage after its integrity and authenticity have been verified by the crypto core. The system manager will acknowledge the acceptance or rejection of payload profiles to the operator.

D. *Mission Updating*

Payload profiles, firmware, or software may need to be updated due to newly discovered bugs, threats, or malfunctioning behavior. Similar to mission provisioning, the operator could choose to reconfigure the system for a different mission. An updated payload profile would be accepted after its integrity and authenticity have been verified.

E. *Payload Configuration*

The Intelligent Platform Management Controllers (IPMCs) on the payload modules are set to wait for commands from the system manager before payload activation. Using the VITA 46.11 standard [9], the system manager performs a discovery on the payloads and confirms that the discovered payloads match a device list included in the mission payload profile.

Following the payload boot order specified in the mission profile, the payload IPMC is instructed by the system manager to begin its boot process starting with its embedded boot ROM. The payload continues to boot with verified payload boot code and sets up the run-time environment through the backplane from the system manager storage. The payload application, verified by the system manager, is now loaded and execution begins.

F. *Payload Monitoring*

There are two levels of payload monitoring. Using the VITA 46.11 standard, the payload hosted IPMC could provide a variety of out-of-band payload information such as temperature, voltage, etc. As embedded in the SOSA requirements, security services will periodically check a payload's health and security state. Action may be required when its health or security state has degraded. Additional measures for runtime application monitoring, such as watchdogs running in a software container could provide information on payload behavior and abnormality detection (e.g., unexpected inter-process accesses, latencies, and crashes) to the system manager. Disrupted payloads could be restarted as an attempt to recovery.

IV. Verification and Certification

In this section, we discuss the leverage of the modularity and reusability in an ESaaS modular system with ZT operations to facilitate the certification process. For a mission-critical system, regardless of its architecture, security and assurance requirements must be established for the intended mission and certified that such requirements have been met by the design. Generally, this has been a lengthy process as the establishment of effective security and assurance requirements for an embedded system is notoriously difficult. Providing evidence for meeting such requirements is harder yet, as it demands that we prove a negative.

An embedded system that is designed to operate according to the zero trust tenet of "never trust, always verify" has numerous benefits for mission assurance [9]. With a security analysis, mission essential functions and their dependency on system components could be identified. Security requirements for the development of mission critical embedded systems could be created by considering the origins of essential functions and their resources. The designer would then incorporate technologies to establish a chain-of-trust to proper system functionality, place the system into an attestable state, and maintain that trust over its operation. The ZT analysis and reasoning will lay a foundation for system verification and certification processes.

MOSA emphasizes modularity and reusability. Modularity enables the use of a divide-and-conquer approach for certification and reusability enables the use of an ECP (Engineering Change Proposal) approach. Both contributes to streamline the certification process.

A. *Composable Mission-Critical System Architecture*

With open systens standards, a proper ecosystem could be developed to support a MOSA for system acquisition and certification processes. The ESaaS goal is to enable the development of composable systems in compliance with assurance certification, and support the acquisition, growth, agility, and maintenance of mission needs. Figure 2 shows an ESaaS mission-critical system architecture with its key components defined as a chassis, an ESaaS system manager, one or more mission application payloads, and optional crypto applications. The system manager, equipped with standardized system management and security services, communicates with an operator or an administrator, and manages the multi-purpose, reprogrammable payloads.

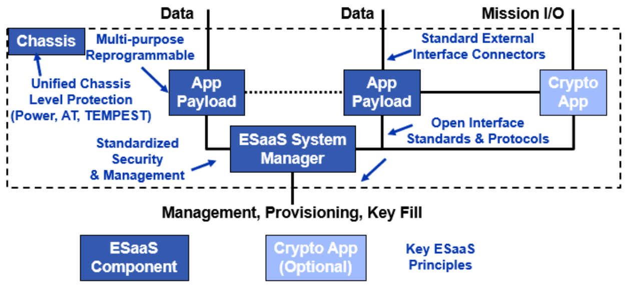
Data
Data
Mission I/O
Chassis
Multi-purpose Reprogrammable
Standard External Interface Connectors
Unified Chassis Level Protection (Power, AT, TEMPEST)
App Payload
App Payload
Crypto App
Open Interface Standards & Protocols
Standardized Security & Management
ESaaS System Manager
Management, Provisioning, Key Fill
ESaaS Component
Crypto App (Optional)
Key ESaaS Principles

Figure 2: The ESaaS composable mission-critical architecture concept, components, and principles (App: applications, AT: anti-tamper, TEMPEST: side-channel protection, I/O: input/output).

The system manager and payload modules could be implemented in various form factors. For example, the components could be developed into plug-in boards as Line Replaceable Units (LRUs) or Intellectual Property (IP) cores for embedded purposes.

Figure 2 also captures a set of design principles of a mission-critical composable embedded system. The ESaaS architecture defines component functions, their interfaces (physical, electrical, and logical), and communicating protocols. For mission assurance purposes, component design requirements, e.g., red-black separation and software security, are established to enable their compliance evaluations and facilitate system certifications. For example, for high assurance, the system must be designed to tolerate at least a certain number of functionally or physically independent failures. Specific mission CONOPS and application domains would play a key role in determining requirements. For example, red (plain data) and black (encrypted data) signals must be carefully separated in cryptographic equipment.

The deficiency of a *generic* standard such as OpenVPX in security and assurance implementation has been mentioned above. Even in domain-specific standards, such as SOSA, which include many properties desirable for mission-critical architecture, gaps do exist. In some cases, the standard could be configured for mission critical operations. In other situations, the standard needs to be enhanced. We use SOSA as an example to explain the challenges below.

SOSA is built for heavy inter-payload communication through a central data plane switch (switched vs. direct IO). In an ESaaS architecture, additional requirements need to be posted on switches and ensure the proper signal separation required by the operation.

In the development of an ESaaS system targeting a SOSA, we have relied on the extensive optical IO (Input Output) capabilities which may pass directly from a payload card, through the backplane, to the exterior of the chassis. Optical connectivity provides tremendous throughput on the order of hundreds of Gbit/s and minimizes crosstalk and emissions relating to signaling at the interfaces. This can be configured to allow many low-rate channels or fewer high-rate channels using the same hardware and cabling.

When an application requires inter-module connections, such as in the case of a radio consisting of a modem module and a crypto module, the designer could adapt the direct inter-module connections through an expansion plane, which are defined in SOSA and OpenVPX. Individual data lines would be needed when a crypto module serves an external application, which could be implemented as connectors to a payload module through the front panel IO. These and other specific properties are defined in SOSA and OpenVPX, and could be configured for an ESaaS architecture.

The system manager component should be powered up before other modules and set up a root-of-trust. Mutual authentication needs to be established between the system manager and the operator. After authentication, the system manager is provisioned for the mission with payload configurations. Keys and certificates need to be loaded securely into zeroizable storage (anti-tamper requirement). During operation, the system manager has to monitor payloads and performs secure logging. Our study and prototyping have indicated that these requirements are implementable in SOSA with a few caveats.

As mentioned, ZT operation lays a foundation for certification. The development of a high assurance system needs to rigorously establish a chain-of-trust and correctly place the system into an attestable state [8]. SOSA has specified a secure boot process for a system, which needs to be further fortified for high assurance crypto operations.

The ESaaS powering up sequence needs to minimize the elements that must be implicitly trusted. As discussed, most COTS programmable devices require a small segment of implicitly trusted bootstrapping code. Also, SOSA/OpenVPX relies on implicitly trusted bootstrapping devices such as the IPMC (Intelligent Platform Management Controllers) for payload control and monitoring [9]. These and other properties must be further investigated in the development of ESaaS systems. Best practice needs to be explored and incorporated into requirements to reduce associated risks. For example, the bootstrapping code, typically minimal and concise, could be formally verified to rule out design errors and stored in a Read Only Memory.

B. *Component Certification*

First of all, in the ESaaS architecture, every component in a system is in a specified application domain with well-defined functions and interfaces. For example, a payload could be a signal processor for target recognition, and another payload could be a secure radio. Design requirements could thus be created for individual components. Against the requirements, each component could be individually certified to be in compliance. Component level certification is particularly logical for the chassis and system manager components, as their hardware and software are designed to work with various payloads and usually unchanged when the system is being upgraded.

Figure 3 illustrates the design and certification process of components for specific mission CONOPS and application domains. For example, the component is a secure radio embedded in an air-borne platform operating in Contiguous United States (CONUS). We envision that a handful of application scenarios could be established.

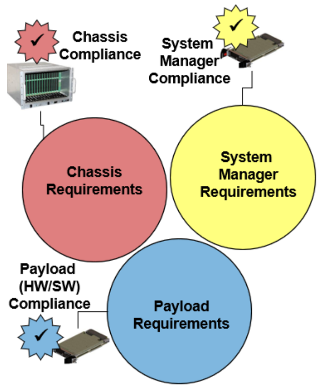
Chassis Compliance
System Manager Compliance
Chassis Requirements
System Manager Requirements
Payload (HW/SW) Compliance
Payload Requirements

Figure 3: Component compliance certification (function, interface, protocol, and security) with respect to mission CONOPS and application domains.

Vendors would design components, including their hardware, software, and firmware, to meet mission-specific requirements (function, interface, protocol, and security). Such components are then submitted for certification against their requirements. This first step of the divide-and-conquer methodology is similar to the current practice of module level certification.

C. *Mission System Certification*

A mission system is then composed of certified components, potentially from multiple vendors. Figure 4 illustrates the system certification process. Even though the components themselves have been individually certified, inter-component relations need to be further examined and determined if the system is in compliance. The mission system, having been verified to be in compliance with design requirements, can now go through its certification process.

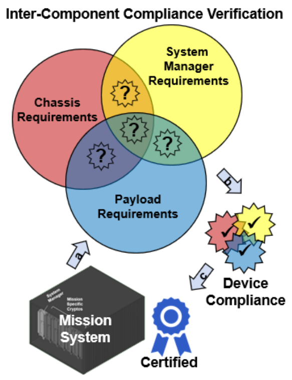

Figure 4: Mission system compliance verification and certification; (a) Compose mission system, (b) Verify the assembly of certified components do not violate system requirements, (c) Certify mission system.

The proposed mission system certification process has two advantages. First, it is identical to the current practice of certification. Second, the component and system compliance verification and certification steps could facilitate and accelerate the certification process. Rather than a completely unique mission system certification, MOSA enables re-using known good approaches and certification artifacts. Tools could be created to automate or facilitate the creation of inter-module test suites and system certification plans, the review of architectures (e.g., with model-based systems engineering), and the generation of documents.

D. *Upgrade and Re-certification*

The biggest benefit is that it could potentially streamline the re-certification of a mission system after it has been upgraded. Upgrade could be performed by loading new software/firmware into an existing payload in a system, replacing an existing payload, removing a payload, or adding a new payload.

The re-certification process of an updated mission crypto system with a new payload is similar to the original system certification step in Figure 4. Since the new payload should have been verified to be in compliant with mission requirements, the updated mission system only needs an engineering change proposal (ECP) level certification. The ecosystem discussed above could be developed to facilitate and accelerate the re-certification process.

V. SUMMARY

The benefits of using a MOSA strategy in system acquisition and development have been demonstrated. However, the development of security and mission assurance for MOSA systems has lagged behind the standardization of electrical and physical interfaces. The result is that it has been slow to adapt MOSA in crypto development.

ESaaS has been developed into a composable, reprogrammable modular architecture so that mission specific modules can be upgraded or swapped out to address evolving mission needs. Standardized common management and security services reduce the burden for users and maintainers of each platform. ESaaS has thus extended MOSA benefits into the development of high assurance mission systems. The current ESaaS architecture targets SOSA, an open systems standard highly invested by both the government and defense industry.

It is critical to accelerate the initial certification of a mission system, and even more so for the following re-certification when it has been updated. We have proposed to rely on modularity and reusability to accelerate the certification processes. We will continue to identify and document appropriate ESaaS modularity, interfaces, and protocols. We will develop the current SOSA aligned prototype into a testbed to identify and investigate necessary actions to close gaps in SOSA, from all technical, business, and compliance verification aspects, for ESaaS development.

VI. ACKNOWLEDGEMENT

Fred Schneider is supported in part by Air Force Office of Scientific Research under award number FA9550-23-1-0435, as well as by funding from MIT Lincoln Laboratory, Amazon, and Google. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of these organizations.

VII. REFERENCES

[1] https://www.dsp.dla.mil/Portals/26/Documents/PolicyAndGuidance/Memo-Modular_Open_Systems_Approach.pdf, accessed June 3, 2023.

[2] M. Vai, E. Simpson, D. Kava, A. Lee, H. Nguyen, J. Hughes, G. Torres, J. Lim, B. Nahill, R. Khazan, and F. Schneider, "Security-as-a-Service for Embedded Systems," IEEE MILCOM, 2023.

[3] ANSI/VITA 65.0-2021, OpenVPX System Standard, October 4, 2021.

[4] Technical Standard for SOSA Reference Architecture, Edition 2.0, The Open Group SOSA Consortium, August 2022.

[5] Department of Defense (DoD) Zero Trust Reference Architecture, Version 2.0, July, 2022, Defense Information Systems Agency (DISA) and National Security Agency (NSA) Engineering Team, https://dodcio.defense.gov/Portals/0/Documents/Library/(U)ZT_RA_v2.0(U)_Sep22.pdf, accessed June 27, 2023

[6] M. Vai, D. Whelihan, E. Simpson, D. Kava, A. Lee, H. Nguyen, J. Hughes, G. Torres, J. Lim, B. Nahill, R. Khazan, and F. Schneider, "Zero Trust Architecture Approach for Developing Mission Critical Embedded Systems," IEEE HPEC, 2023.

[7] https://trustedcomputinggroup.org/work-groups/trusted-platform-module/, accessed May 26, 2023.

[8] https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html, accessed May 26, 2023.

[9] ANSI/VITA 46.11-2022, System Management on VPX, April 5, 2022.