TRUSTED COMPUTING
FOR DEFENSE & AEROSPACE

A COLLECTION OF ARTICLES FROM CURTISS-WRIGHT AND MILITARY & AEROSPACE ELECTRONICS

CURTISS-WRIGHT

TRUSTED PROVEN LEADER

CURTISSWRIGHTDS.COM

CONTENTS

INTRODUCTION TO TRUSTED COMPUTING

KEY TRUSTED COMPUTING CONCEPTS

AIRBORNE AND UNMANNED SECURITY

BUILDING A TRUSTED SOLUTION

CERTIFICATION AUTHORITIES

December 18, 2019 | By: David Sheets

TRUSTED COMPUTING: AN OVERVIEW

Attacks are getting sophisticated. Examples include Rowhammer, Meltdown, Spectre, and others. System designers need to consider many attack vectors.

At its core, trusted-computing works to ensure that computing systems operate safely, securely, and correctly every time. Trusted computing matters at every level of operation, whether it be the processor level, software level, or system level. Each layer of a computing system ensures that a system can operate securely. Because malicious attackers are able to poke at all layers of a system, securing only one single layer often is not the most effective use of resources.

Attacks are becoming increasingly sophisticated. Examples include Rowhammer, Meltdown, Spectre, and others. System designers need to consider many attack vectors. The security of hardware components can no longer be assumed. System designers must verify and monitor their hardware for future vulnerabilities. However, secure hardware alone is not enough. For a system to be secure, its software also must be secure. Securing software can include hardening free operating systems like Linux, or software built from the ground up to address security, such as StarLab Crucible.

After securing the software, the security architect's work is still not done. Today, systems must integrate and interoperate to complete a mission. That means that network and physical interfaces that connect individually secure elements of a system also must be analyzed for vulnerabilities and then locked down to mitigate possible attacks.

The good news is that many groups and documents are available to help guide the architect and monitor a trusted computing system. Here are some of the most critical documents that system security architects need to understand.

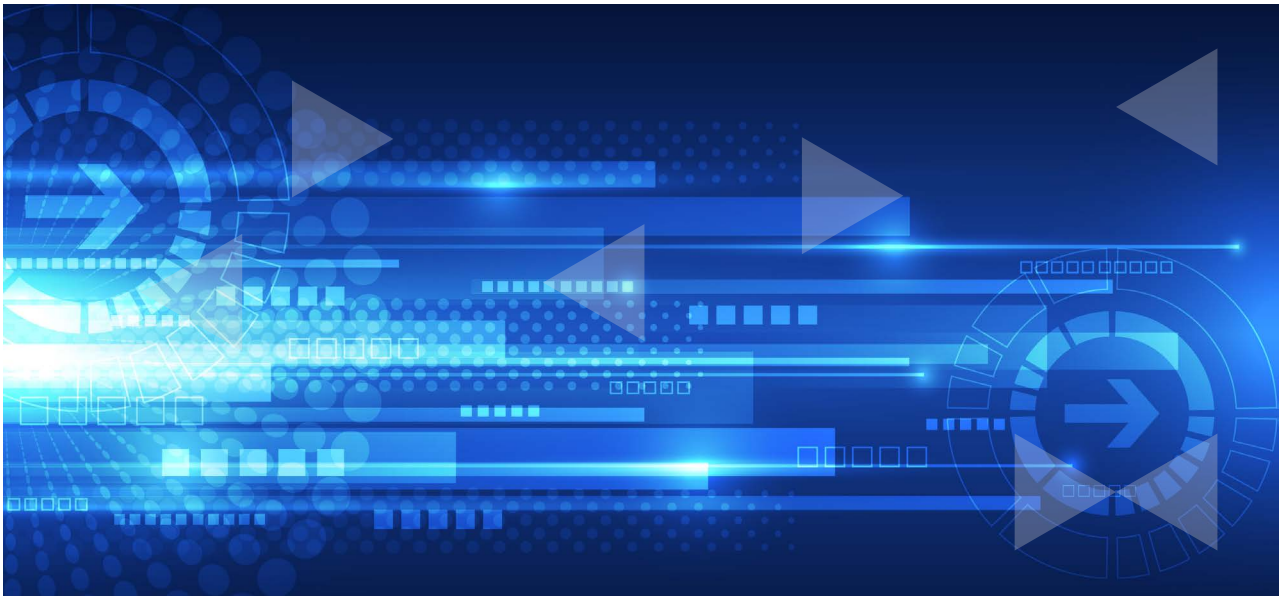
At the hardware level, NIST 140-2 can provide guidance on evaluated cryptographic hardware. Common Criteria, administered by National Information Assurance Partnership (NIAP), can provide trust in the design process for systems and security. One recent example is the evaluated Curtiss-Wright DTS-1, the embedded industry's first commercial off-the-shelf (COTS) data-at-rest (DAR) network attached storage (NAS) solution for secure data storage. For hardware security, the Trust Computing Group (TCG) provides guidance on certification for Trusted Platform Modules (TPM).

Within the U.S. Department of Defense (DOD), the Anti-Tamper Executive Agency (ATEA) provides guidance on physical security for military systems. On the cyber security front, the Risk Management Framework (RMF), presented in a series of National Institute of Standards and Technology (NIST) and FIPS (Federal Information Processing Standards) documents, provides a mechanism to evaluate system security across confidentiality, integrity, and availability, as well as guidance on how to meet required security levels.

Overlays also can be used with RMF to further refine the guidance based on particular system applications, classification level, or other aspects of system operation. Much as DO-178B provides guidance on safety critical software, and DO-254 provides guidance on safety-critical hardware for aviation platforms, DO-326A provides similar types of guidance on cyber security for aviation. For programs that require more concrete and easily implementable guidance, the sets of Security Technical Implementation Guides (STIGs), managed by the Defense Information Systems Agency (DISA), can provide an easy and helpful resource if an applicable STIG is available for the system being protected.

Underpinning the integrity and confidentiality of security for trusted computing is the use of cryptographic algorithms. Cryptography should not be considered as a static discipline. Because processing capabilities are always improving, designers need to understand their security requirements and how those requirements relate to and help drive decisions about which cryptographic algorithms and key sizes need to be used. For example, many systems will have requirements as to how long information confidentiality must be maintained. Those requirements will influence the selection of algorithms and key sizes.

Systems designers also need to understand symmetric cryptographic algorithms, such as AES, and where they are being employed. In addition to symmetric algorithms, security architects also must understand secure hashing algorithms that are used during image and data integrity verification, such as SHA-2 or SHA-3, and asymmetric algorithms that are used to sign and verify images, and are also used in key agreement schemes, such as ECC or RSA.



Apart from existing algorithms and guidance, designers also must be aware of advances in quantum computing power and how those advances might impact the security of asymmetric cryptographic algorithms. Security architects must keep an eye towards understanding how newly developed algorithms, such as those now being competed by NIST, might be integrated into their systems once new implementations of accepted quantum resistant algorithms are available.

Going forward, it's imperative to understand the trusted computing implications for every program.

> ***Trusted computing cannot be an afterthought. Instead, it must be built in from the start of every program to ensure that appropriate risks are understood and appropriate mitigations are put in place.***

That does not mean that every program needs to implement the highest levels of security, but it does mean that every program should do the analysis to make the decision about what level of security is needed based on which risks can be tolerated and which risks are unacceptable.

Trusted computing is hard. Unlike many other disciplines in engineering, it's not just about trying to solve complicated problems. The added complexity and challenge comes from trying to solve complicated problems while facing adversaries who are constantly advancing and evolving.

Even more difficult, unlike most enterprise systems that can accept periodic updates and relatively inexpensive upgrades, deployed embedded systems need to be able to stay relatively static while staying resilient in the face of advancing attack capabilities.

Trusted computing can impact every facet of a computing system, including hardware, software, system integration, maintenance activities, and testability. By ensuring that the program addresses security and trusted computing issues early in the program life cycle, program risks and costs can be managed. It's when security is addressed at the end of the program that most programs run into real problems.

While implementing trusted computing is difficult, it is not an insurmountable problem. It just requires work and starting with the appropriate expectations. By diligently working through potential issues, and working closely with suppliers and vendors, programs can successfully provide secure solutions on time, and on budget.

November 21, 2019 | By: David Sheets

THE DIFFERENT TRUSTED COMPUTING AND CYBER SECURITY APPROACHES FOR EMBEDDED COMPUTING AND ENTERPRISE SYSTEMS

Embedded systems for deployment in the field can have vastly different needs for trusted computing and cyber security than those of enterprise systems.

When discussing trusted-computing, and cyber security, it is important to understand which of two types of computer systems that needs protection -- enterprise or embedded.

Enterprise systems manage the information technology (IT) and infrastructure of large organizations the world over. These systems often are distributed and connected, and typically are upgraded at a comparatively frantic pace.

Embedded systems, on the other hand, typically are deployed in the field. They tend to be much more rugged and much more tightly integrated than enterprise systems, and commonly undergo more rigorous certification and verification processes.

Enterprise systems normally are made up of all-commercial components, networked together potentially over long distances and often connected via firewalls to public networks. These systems often integrate normal desktop and laptop computers, switches, routers, and firewalls, as well as other commercial market products that usually are produced in large volumes.

For each of the products that comprise an enterprise system there are specific security considerations that systems integrators must address. Most enterprise hardware will use standard interfaces, such as Ethernet and USB. They will run a version of a common commercial operating system like Microsoft Windows or RedHat Linux, and depend on support from commercial vendors for security updates.

In contrast to enterprise systems, embedded computing systems exist within a defined environment. They normally are more rugged, which enables the electronics to operate in the harsh physical environments that fielded military hardware must endure. They also are much more closely integrated, perhaps using interfaces that are less common in commercial architectures, such as MIL-STD-1553.

Embedded systems typically also use purpose-built hardware, often in response to the difficult tradeoffs that designers make regarding size, weight, and power consumption (SWaP). Also, because of the lower production volumes, these systems often will have much different vendors and support structures than do enterprise systems.

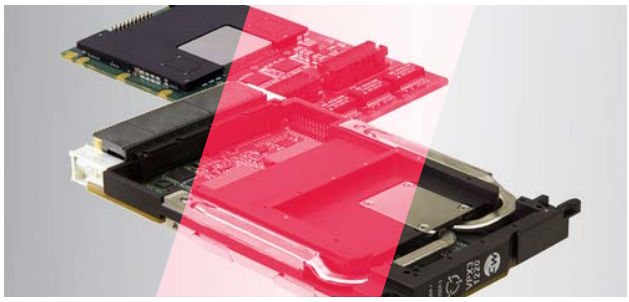
> ***It is important to consider the history of cyber security when choosing trusted computing approaches for enterprise systems and embedded computing.***

The criticality of cyber security emerged as a major concern as systems became more connected, and opened the door to potential attackers via remote access. Because they connect disparate geographically isolated heterogeneous systems together, enterprise architecture designers consider requirements for authenticating endpoints, encrypting communications, locking down interfaces, and controlling the flow of information.

The community of cyber security professionals has developed many products, tools, and frameworks geared specifically to address the needs of enterprise systems. Some common examples of these are virus scanners, firewalls, and intrusion-detection systems.

A virus scanner is a relatively simple tool that can look at code and data to determine if it has nefarious intent. One important facet of using a virus scanner is ensuring that the definition of what constitutes a virus stays up to date. If not, new viruses can circumvent the scanner.

This tool relies on the connectivity of the enterprise system to ensure it stays current and effective. Using such a tool in a disconnected, or even infrequently connected embedded environment can be problematic. In comparison, a firewall that filters network traffic, or an IDS that monitors a system for anomalous behavior, might not require as updates as frequently as a virus scanner does to stay relevant, but may still require the ability to trigger warnings that instigate investigation and response when necessary.



In an enterprise environment, the person handling the resulting investigation and response often will be an IT professional. In comparison, in an embedded environment it can be difficult to say where that needed update information can be reliably stored and processed in a timely and effective manner.

One difficulty of applying cyber security to embedded systems revolves around assumptions and practices concerning with a specific type of environment. An example is the Risk Mitigation Framework (RMF). The U.S. military often relies on RMF to ensure that U.S. Department of Defense (DOD) systems go through sufficient cyber security scrutiny to operate securely on DOD networks.

Because the RMF process is applied to many different types of systems, from enterprise to small embedded systems, however, there are many possible controls that systems designers must analyze and document that may not apply to embedded systems.

For instance, there are RMF controls related to login credentials and password length that do not apply to embedded systems. For a small embedded system without a login, systems designers may mark many of these kinds of controls safely as inapplicable.

Still, it takes time to analyze, document, verity, and monitor the embedded system to ensure that none of the assumptions change. It's also important not only to prevent any backdoors into the system, but also to understand that additional work may be necessary to secure DOD approval for these sorts of systems.

The RMF was developed with the understanding that it applies across a wide variety of systems. As a result, the concept of overlays was designed to help address this concern.

An overlay is a selection of controls specific to a particular type of system. An overlay can either add or remove controls from the required set used to analyze the system security risks. It adds and removes controls when necessary. It also can refine controls, add additional text for clarity.

Although systems designers have discussed developing an overlay that makes it easier to apply the RMF to embedded systems, that hasn't happened yet. Still, there is progress in developing overlays specific to weapon systems or mission computers that may apply to embedded systems. Programs should look to the military services to understand if an overlay that is applicable to their type of system has already been developed before they spend unnecessary time going through all RMF controls and documenting their decisions.

Obviously, programs should look at all relevant overlays -- including any overlays for systems that need to process classified or sensitive information. Designers should look at the Joint Special Access Program (SAP) Implementation Guide (JSIG).

Although the RMF does address systems updates, more discussion may be necessary. Enterprise systems often rely on their ability to update to stay safe, such as weekly operating systems patches and periodic virus scan updates.

On the other hand, embedded systems often must go through a robust verification process before being deployed. That means that any frequent updates may require expensive re-verification. What's more, updating a common shared library potentially could invalidate weapon safety testing, flight safety requirements, or key performance indicators.

To bottom line is that re-running all the required certifications often is simply infeasible for complex embedded systems. The challenge of security updates is an area that DOD systems designers still are dealing with to develop appropriate responses.

Although military embedded systems may have minimal connectivity, it's become apparent that they still have cyber security issues. Additional work is necessary to reduce the overhead on every deployable embedded system while maintaining security metrics.

October 30th, 2019 | By: David Sheets

UNMANNED SYSTEMS VULNERABLE TO THE ENEMY, WHICH MAKES TRUSTED COMPUTING A CRITICAL CYBER DESIGN CHALLENGE

Reliable, secure communications becomes much more critical in unmanned systems. If communications are lost, the unmanned system may need to throttle back.

Trusted-computing is a difficult concept to implement, even in some of the best scenarios. Implementing adequate cyber security and other protections becomes even more challenging when the system being protected will be deployed into the harsh world without a trusted service member nearby to operate the system.

The first challenge when designing a deployable solution for an unmanned platform is the lack of constant supervision from a trained and trusted human operator.

> ***While supervision is typically available on manned systems, unmanned systems may only have periodic contact with an operator.***

What's more, an unsupervised system may need to make autonomous decisions based on minimal and potentially untrustworthy sensors. Another important issue is that, compared to manned systems, unmanned systems are more prone to falling into the hands of potential adversaries. One example is the seizing of a U.S. unmanned underwater drone by China at the end of 2016.

The availability of reliable, secure communications becomes much more critical when dealing with unmanned systems. While a manned system that loses communication still can operate, as long as the onboard operators can work to restore communications or complete the mission, a communications failure is much more critical for an unmanned system. If communications are lost, the unmanned system may need to throttle back, proceeding with much more limited functionality. Loss of communications also means that the unsupervised system won't be able to quickly respond to changing circumstances.

For all of these reasons, systems designers must ensure that their unmanned systems are well protected. These protections must take into account the possibility of an attacker gaining physical access to the system, and must include adequate cryptographic protection of stored data and appropriate cyber security protections to maintain communication integrity and confidentiality.

To help system designers meet the challenge, there are specific documents that provide guidance on how to maintain trusted operation on unmanned systems.

One such document is the Committee on National Security Systems Policy 28 (CNSSP-28), "Cybersecurity of Unmanned National Security Systems." Much of the guidance in this document references other guidance documents from which to pull information (e.g. CNSSP-7). Much of the guidance that CNSSP-28 provides indicates particular technologies that must be employed (e.g. encryption for all command and control data links). In addition, CNSSP-28 indicates guidance on which processes must be adhered to. This includes, for example, describing required aspects of the risk-mitigation framework that must be applied to unmanned systems.

When applying risk-mitigation framework to unmanned systems, designers must take care to ensure that their system has been appropriately analyzed so that all controls have been selected to provide protection across all potentially compromised environments, not just nominal operating environments.

The NIST 800-53 security controls document describes the set of controls that might need to be applied to unmanned systems. Another document, CNSS Instruction 1253, provides guidance on the overlay of controls that are mandated to be applied to national security systems.

This guidance is based on selecting the level of protection required (low, medium, or high) across three protection categories (confidentiality, integrity, and availability). System designers should select the appropriate level of protection for each of these three categories so that their programs don't under-protect or over-protect their system based on its unique system level risks.

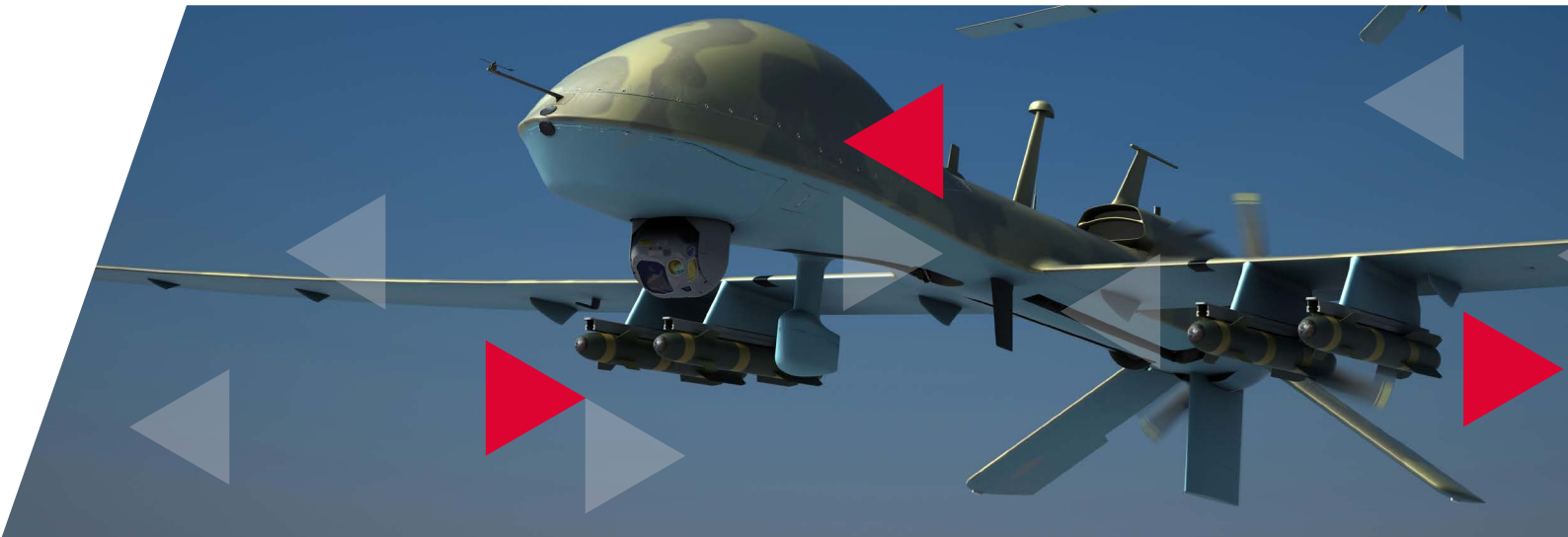
Similarly, there are also policies that provide guidance on cryptographic standards for unmanned systems. One example, the "Policy on the Use of Commercial Solutions to Protect National Security Systems" (CNSSP-7), also available on the CNSSP policy page, provides guidance on using Commercial Solutions for Classified (CSfC) for national security systems.

While CSfC has gone through multiple revisions of the Data at Rest (DAR) Capability Package (CP), the latest version, v4.8, which is currently in review and accepting comments, provides additional guidance on unattended operation.

In this document, the unattended Use Case defines the operational parameters for unattended operation. Requirements at the end of the document allocate from the set of potential CP requirements to the Unattended Operation use case. Since the document is now in the review period, interested programs that may want to use approved CSfC DAR solutions, such as Curtiss-Wright's DTS-1 Rugged Network Attached File Server, on unmanned systems, are advised to review the draft document and provide feedback on applicability of the requirements to their specific platforms and systems prior to the deadline of December 5, 2019.

Beyond the guidance on protecting unmanned systems that is available from publicly available documents, additional guidance for ensuring complete protection can be obtained from leading, experienced COTS vendors.

Programs can work with their vendors to ensure that the products they plan to integrate into an unmanned system have been engineered to meet stringent security requirements and can provide protection in the face of all potential attack vectors. System designers should verify that their COTS vendors are actively involved in appropriate trusted computing communities to ensure that they maintain awareness of and follow appropriate guidance from current documents across their product lines.



September 25, 2019 | By Denis Smetana

FPGA-ENABLED TRUSTED BOOT IS PART OF BUILDING SECURITY INTO EVERY ASPECT OF TRUSTED COMPUTING ARCHITECTURES

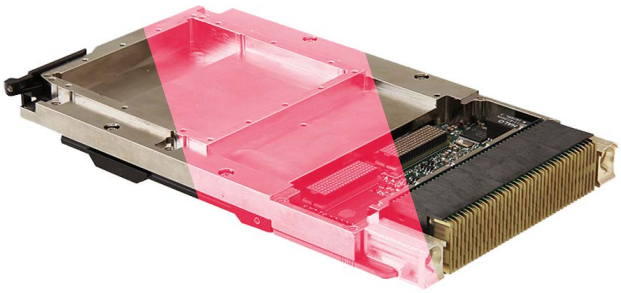
FPGAs can defend against reverse engineering and offers capabilities such as black key storage or side channel resistant cryptography in some devices.

Embedded computing systems designers can establish enhanced trusted boot protection through use of an field-programmable gate array (FPGA)-based root of security (RoS).

An FPGA-based RoS defends against reverse engineering and offers capabilities such as black key storage or side channel resistant cryptography in some devices. It also can enable users to customize the FPGA to add in other protections to secure their system and meet specific program needs.

These enhanced protections provide the necessary hardware infrastructure to enable the RoS to interface with security sensors and processors, while maintaining the security of the system throughout the boot process.

What's more, these enhanced trusted boot techniques provide mechanisms to ensure that any new code is authenticated prior to being stored in non-volatile memory, and that they also deliver additional trusted-computing checks and mitigations during the boot process.



An important concept in trusted computing is a holistic view, beyond just the hardware itself, with an eye to building security technologies and techniques into every aspect of the solution -- from design and testing to supply chain and manufacturing.

In other words, security doesn't stop at the card edge. This comprehensive, end-to-end approach, often referred to as defense in depth, creates a mesh of protection layers that ensure the solution's reliability.

At the module level, tying together as many of the available security techniques as possible, such as integrating the unique trusted boot technology on the processor with the unique trusted boot technology provided with an FPGA, enables security system engineers to realize a multiplier effect.

Here's a real-world example: in September 2019 Curtiss-Wright introduced the CHAMP-XD1S powerful digital signal processor (DSP). It features a 12 core Intel Xeon D processor, a Xilinx Zynq UltraScale+ multi-processor system-on-chip (MPSoC) FPGA, and a Flash-based Microsemi SmartFusion2 IPMC FPGA to provide a secure processor board designed for high-performance embedded computing (HPEC) applications that must operate in harsh environments.

To protect against malicious attacks and reverse engineering, the board incorporates data security, including cyber security and physical protection. Its built-in advanced security features include enhanced trusted boot capabilities like FPGA-based authenticated boot code.

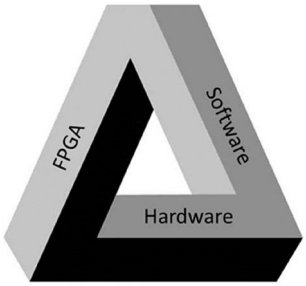
The board supports the Intel TXT secure boot technology, and also can support UEFI secure boot via the operating system. This module features additional protection that uses the physically unclonable function (PUF), which resides in the on-board MPSoC FPGA, to authenticate the boot code.

The PUF, only available in Xilinx Zynq UltraScale+ devices, provides a unique identifier associated only with one piece of silicon. The PUF, which takes advantage of silicon variations unique to Zynq UltraScale+ devices, generates a cryptographically strong encryption key unique to that device.

The resulting key is unreadable by anyone, including the user, and can be used in combination with the FPGA's built-in advanced encryption standard (AES) cryptographic core. The PUF, in addition to generating the encryption key, also generates the helper that enables the PUF to regenerate the encryption key later to provide a heightened level of key security.

Use of the PUF makes it essentially impossible for a malicious actor to spoof, clone, or change the FPGA, since the system would not be able to recognize the device with its altered PUF value. The unique identifier signature can be a seed for encryption and for authenticating the boot image and boot code for the FPGA. Even better, the authentication can be extended to protect other portions of the system, including other boot artifacts apart from the FPGA itself.

Two FPGAs are better than one. In addition to the module's MPSoC FPGA, the board also features a Microchip SmartFusion 2 FPGA which provides not only health and management, but also integrates additional security functions.



The SmartFusion 2 FPGA is a flash-based FPGA, rather than SRAM-based like the Xilinx MPSC. It can be encrypted, and has its own set of internal protections to provide the user with two different types of FPGA technologies.

This new board also features a rev 2.0 Trusted Platform Module (TPM) security chip that uses cryptographic methods to ensure platform integrity throughout the entire boot process until applications are running. The TPM often functions as the basis for supporting Intel TXT but more generically, it manages and generates keys. TPM 2.0, compared to earlier versions, has updated and additional cryptographic algorithms, providing trusted computing designers with more flexibility. It also supports several different cryptographic keys, rather than just one. Users also can use the TPM's security features post-boot as well.

Due to sensitive nature of trusted computing design, it is not possible publicly to discuss all of the security capabilities and features built into the CHAMP-XD1S. It is important to note, though, that the security architecture is designed with flexibility as a core requirement, so that customers can add their own unique capabilities as needed.

To ease that process, the board comes with an FPGA toolkit that enables customers to add in their own custom security capabilities or additional enhanced security capabilities provided by Curtiss-Wright or even other third parties. For example, we develop additional IP that can be integrated into the FPGA to further enhance secure boot and trusted computing operation beyond what is available off-the-shelf from the device vendors.

While it's important for security designers to understand and implement each of the available individual security approaches available with their hardware, it's also important that they consider how implementing several different approaches, and tying them together, can deliver a multiplier effect. Of course, all correct hardware architecture guidance must be followed, but when done correctly, the whole is truly greater than the sum of the parts.

August 21, 2019 | By: Paul Hart

UNDERSTANDING CYBER ATTACKS IN EMBEDDED COMPUTING ENABLES INTEGRATORS AND SUPPLIERS TO CONSIDER OPTIONS

The trusted computing task is to apply practical measures that can mitigate the effects of known cyber threats to aerospace and defense systems.

Cyber attacks on embedded computing systems are an ever-increasing threat, and system integrators need to have an understanding of cybersecurity concepts to design-in sufficient protections. This includes understanding what commercial off-the-shelf (COTS) vendors can bring to the table to mitigate the effects of cyber attacks and ensure that their suppliers have the necessary expertise to meet their system protection requirements.

The trusted-computing task is to apply practical measures that can mitigate the effects of known cyber threats to aerospace and defense systems. Meanwhile, the battlespace is becoming ever-more computerized, from the infantry soldier with his personal radio and situational awareness, to communications with the battle group, and information coming from datalinks and intelligence, surveillance, target acquisition and reconnaissance feeds from unmanned aerial vehicles (UAVs).

Consider the basic computer architecture of a personal computer. It has a processor, memory of different types, a BIOS to initialize the system and a clock to step through the program counter. There also are I/O ports that connect the system to the outside world, and perhaps graphics for displaying data and user interfaces.

Military embedded computing systems generally share the same generic architecture as a personal computer, yet with different levels of processing power. Like the personal computer, the embedded computer also has a processor surrounded by memory. It has a clock, some form of initialization, I/O, and sometimes some graphics and power supply. One main difference is the military system may not have a keyboard, mouse, or electronic display.

Instead, the computer is part of a subsystem such as an aircraft flight-control computer, with applications that launch straight from power-up and have the ability to monitor themselves. Also, the applications might communicate with other nearby embedded computers that essentially are doing the same thing.

The first step is to understand the cyber vulnerabilities of a given hardware system, called attack surfaces, which typically are the interfaces that connect the computer to the outside world. Everyone is familiar with the interfaces on his or her own laptop computer, such as the USB, Ethernet, and WiFi interfaces. Still, an attack surface also can be a switch, a mouse, keyboard.

One well-known method of exploiting an attack surface on a personal computer is via a phishing email with a subject line like “Congratulations, you’ve won a vacation! Just click here.” If the unsuspecting user follows the email’s call to action, he actually will launch an application that gives a hacker access to the computer’s data, and perform all sorts of nefarious activities.

Cyber attacks that threaten the integrity of personal computers and embedded systems include software viruses, worms, and denial of service (DoS) attacks. A virus is malicious software that can replicate itself and modify the code stored in the system’s memory. The infection typically comes via the inputs.

A worm doesn’t require a host application to run, but can execute itself in code. Worms can extract data from memory and send it to the outside world, without anyone realizing it. For example, it a worm can move from an output port, to a switch, to a remote Internet Protocol address. A Trojan Horse is a type of “back-door” software that can capture and log a computer user’s keystrokes to obtain passwords and other valuable data.

In the last year or two, the industry has become aware of a different sort of sophisticated malicious software that manipulate very low-level microprocessor cycling, and enable them to execute machine instructions. Spectre, Meltdown, and Rowhammer are examples of these microprocessor attacks, which are very hard to detect, and can bypass all anti-malware defense mechanisms.

In a DoS attack, a malicious actor overloads the network so that messages can't get through because it claims all of the network bandwidth. These sophisticated software techniques identify large numbers of IP addresses, figures out who's online, and send out enough messages to overload that network. DoS attacks become even more likely as AI and machine learning applications become more common.

Cybersecurity in the military also includes authenticating datalinks that update the equipment and identify if there's a DoS. While embedded systems don't typically connect to the Internet, they often have connectivity to the battlefield to receive new uploads on communications code.

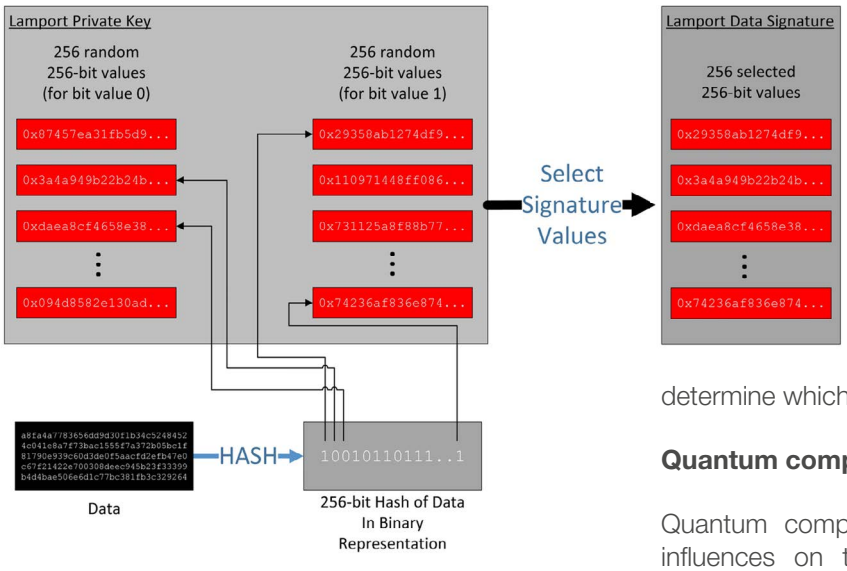
Today, with software-defined radio, systems designers implement new communications waveforms in software that must update regularly. It's critical to ensure that these updates aren't being jammed. The communications must be authenticated so that anyone legitimately sending an update to a computer on the battlefield can establish that connection with absolute assurance that no one is mimicking or spoofing that message or intercepting and understanding the transmitted protocols.

Protecting the data and hardware of deployed embedded military systems is an increasingly critical task facing today's systems integrators. Once designers understand basic concepts and requirements, they can have a true dialogue with their hardware suppliers to ensure the availability of appropriate mitigating technologies and techniques to support the warfighter and help ensure a successful mission.

July 24, 2019 | By: David Sheets

TRUSTED COMPUTING AND THE CHALLENGES OF CRYPTOGRAPHIC ALGORITHMS IN QUANTUM COMPUTING

Quantum computing processes an algorithm by configuring quantum gates and initializing the qubits. Taking a measurement resolves the quantum states.



At its core, quantum computing is very different from a traditional computer. The quantum computer operates on qubits, using quantum gates and measurements, rather than operating on bits per a Von Neuman architecture -- using memory and a processing unit made up of digital gates.

A qubit is not binary. It does not simply encode a 1 or a 0 as a bit does. Instead a set of n qubits encodes a superposition of 2^n possible quantum states. This means that while a traditional digital computer operating on 32-bits only can process one 32-bit value at any given time, a quantum computer with 32-qubits can look simultaneously at the probabilities of all possible 32-bit value combinations.

Quantum computing has several https://www.militaryaerospace.com/trusted-computing challenges. A quantum computer processes a particular algorithm by configuring the set of quantum gates, and initializing the values of the qubits. Taking a measurement causes the quantum states to resolve, and users can measure an actual state of the qubits. For example, once the quantum states resolve in a 32-qubit system, it is no longer in an unknown set of the 2^32 possible combinations, but has resolved to a single possible solution.

Because a quantum computer works with quantum states, each value is often only probabilistic in nature. Normally, to determine a result with sufficient accuracy, a quantum computer runs several times, re-initializing the system, configuring qubits, and then measuring a state. Each subsequent measurement records as a potential result. Users then can analyze these sets of results to determine which of the results most likely are correct.

Quantum computing and cryptography

Quantum computers have a couple of significant influences on the current state of cryptographic algorithms. The first is the ability of quantum computers to implement Grover's algorithm. Using Grover's algorithm a quantum computer can find the input to a black box function that results in a given output, and can do so in half the time of traditional brute-force algorithms. This means that protection from symmetric key algorithms reduces effectively by 50 percent. To mitigate against this threat, it's crucial to ensure that the key-length of any symmetric cryptographic algorithms take this halving effect into account. Doubling the key lengths of symmetric cryptographic algorithms would suffice to protect against Grover's algorithm running on sufficiently robust quantum computers.

The second influence results from the ability of quantum computers to run Shor's algorithm efficiently. Shor's algorithm can solve the underlying integer factorization problems on which RSA and Diffie-Hellman cryptography are based. Unfortunately for trusted computing, Shor's algorithm also can solve the discrete logarithm problem on which the Elliptic Curve Diffie-Hellman algorithm is based. Compared to the threat of Grover's algorithm to the effectiveness of symmetric algorithms, mitigating against Shor's algorithm is much harder and will require substantial reworking of trusted computing systems.

The good news is that quantum computers that can operate on sufficiently large number of qubits don't yet exist. It's important though to understand that these machines are coming, and that their arrival will render the three most used asymmetric algorithms obsolete. Today, cryptographers are working hard to determine the best ways to ensure trusted computing systems can continue to maintain security and trust well into the future.

Hash algorithms for signatures

There are some algorithms that resist known quantum computer-based attacks that already are available and in use. Many of these algorithms can help bolster security in the face of quantum computing systems. For example, hashing algorithms appear just as able to resist attacks from quantum computers as they are from traditional computers. Some hashing algorithms, in fact, can help defend against quantum computer based attacks just as well as asymmetric algorithms can.

In many systems, asymmetric algorithms sign data, with the private key held outside of the system. The public key then can embed in the system and verify authenticity prior to operation. Since these systems keep the private key outside of the system, they resist attackers obtaining the private key and signing their own data. Hashing algorithms can be the basis for other data structures that can provide similarly secure authentication capabilities to systems.

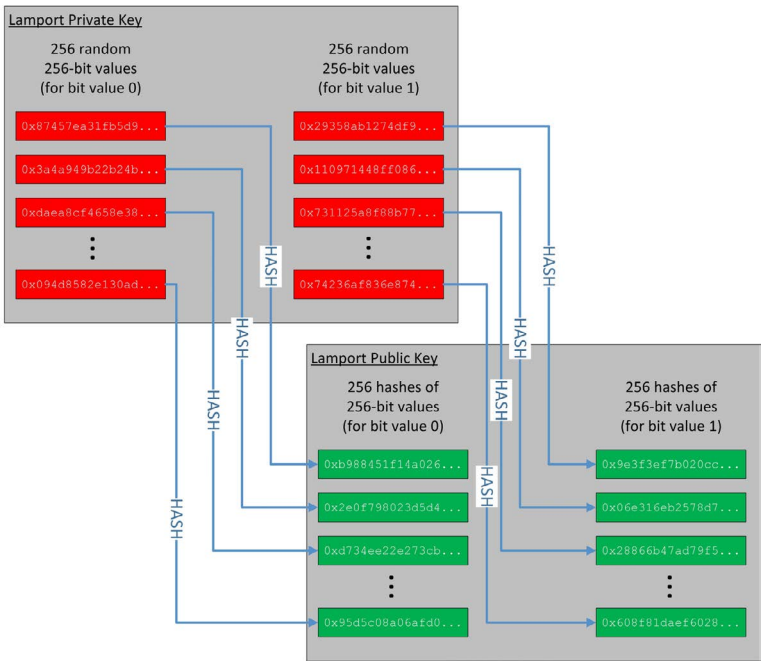
Lamport Signature single-use algorithm

As an example, the Lamport Signature algorithm is based on hashing algorithms and allows a secure one-time-use public-private key pair. In the Lamport Signature algorithm the signer picks a hash algorithm, such as SHA-256. Based on the selected hash algorithm, the signature can support 256-bit signatures. The signer then generates 256 pairs of 256-bit numbers, which generates a total of 512 256-bit numbers. These random numbers serve as the private key.

This approach hashes each of the resulting 512 numbers to generate a set of 512 256-bit hashes, which becomes the public key. The system signs data by hashing, which generates a 256-bit value. The system then chooses one of the two random values from the set of saved private-key values. If a bit is 1, the first value is chosen, if a bit is 0, the second value is chosen. This selection process repeats for all 256-bits of the hashed value of the data, moving through the set of 256 256-bit private key pairs. The final signature passed with the data is the set of selected 256 256-bit values.

To verify the signature, the recipient takes all 256 of the 256-bit values from the signature and hashes each value, resulting in 256 256-bit hash values. The recipient then goes through the pairs of 256 256-bit values in the public key and selects one from each pair, as the signer did, based on the bits of the hash of the data. The data authenticates if the selected values from the public key hashes match the calculated hashes of the signature.

While this signature scheme works, it requires large signatures, and requires new one-use public keys, for every data item it authenticates. Using it in a system requires infrastructure to deliver new one-use public keys securely. This is necessary to maintain security, or the system must come loaded with a very large preinstalled set of public keys.



Merkle Tree Signatures

Another approach for protecting against quantum computer attacks uses the Merkle Signature Scheme, which builds on a one-time signature scheme, such as the Lamport Signature algorithm. The Merkle Signature Scheme generates a set number of one-time-use public-private key pairs. Each of the public keys is then hashed, and the resulting hash becomes the leaf in a Merkle Tree.

This binary Merkle Tree consists of leaf data nodes, where each parent node is a hash of the concatenation of the two children nodes. As with the Lamport Signature algorithm, the signer generates a private key using one of the previously unused public-private key pairs from the Merkle Tree leaf nodes. The signer then also appends the set of sibling nodes going up the Merkle Tree from the selected leaf node. (Continued...)

This enables the recipient to verify that the received signature is actually part of the Merkle Tree. All of this means that instead of preinstalling all 2^n public signatures initially generated as part of the Merkle Tree, the public signatures can be sent out as needed, and verified as part of the Merkle Tree during the signature verification.

However, there are still caveats with using Merkle Trees -- namely all of the public-private key pairs need to be pre-generated. Also, there are a limited number of possible signatures that can be used based on the value of n chosen when the Merkle Tree is initially generated. However, for systems that require long term protection, these algorithms can provide security in the face of increasing quantum computing capabilities.

NIST post-quantum cryptography status

Given the serious concerns of using existing algorithms to sign and authenticate data well into the future, the National Institute of Standards and Technology (NIST) in Gaithersburg, Md., has initiated a process to select possible algorithms to replace currently available asymmetric cryptographic algorithms.

The first round of NIST's evaluation has been completed and a status update was released in NIST IR 8240 on January 2019. The second-round standardization conference will be next month to refine the set of 26 algorithms selected from the 69 first round candidate algorithms. Of the 26 algorithms currently being analyzed, 17 are public-key encryption/key-establishment algorithms and 9 are digital signature algorithms.

There are categories of types of algorithms that are being developed. Most algorithms that made it to the second round use lattice-based cryptography, multivariate cryptography, or code-based cryptography. There was also one algorithm, based on a random walk over elliptic curves, which so far looks to be not susceptible to quantum computers.

While there are currently available implementations for each of these algorithms on the NIST website, since they are still under review, it doesn't make much sense yet for most trusted computing systems to rely on these unproven algorithms. Industry should wait until further analysis has been completed.

It is important for those of us involved in trusted computing to keep an eye on the progress of NIST in their oversight of the analysis of the future of post-quantum cryptographic algorithms at https://csrc.nist.gov/Projects/Post-Quantum-Cryptography.

Based on currently available timelines from NIST, we can hope to have draft standards of algorithms available sometime between 2022 and 2024. At that point, to mitigate against the threat that quantum computing poses to data security, it will be our responsibility to help bring these algorithms into the trusted computing systems we develop and deploy.

Jun 26th, 2019 | By: David Sheets

TRUSTED COMPUTING CAN DEPEND ON ASYMMETRIC CRYPTOGRAPHY ALGORITHMS TO ASSURE THE INTEGRITY OF PROTECTED DATA

Asymmetric algorithms enable systems designers to pair keys to facilitate more novel cryptographic operations than traditional symmetric algorithms.

There are three general categories of cryptographic algorithms commonly employed in trusted computing: secure hashing, symmetric cryptography, and asymmetric cryptography.

> ***Asymmetric algorithms enable systems designers to use a pair of keys to access data. One key signs or encrypts data, while the other verifies or decrypts data.***

This pairing of keys provides the opportunity for novel cryptographic operations when compared to more traditional symmetric algorithms. Normally, in asymmetric cryptography the key pairs are called the "private-key," or one that is securely stored and not-shared, and the "public-key," a matching key that is freely shared with others.

Asymmetric cryptography

Asymmetric cryptographic algorithms enable several use cases. One is personalized encryption, where the sender encrypts data using the public key. Only the matching private-key can decrypt it, which protects the data's confidentiality against external attackers.

Digitally signing of data also uses asymmetric cryptography by processing the data to be signed using the private-key of the sender. Once the other system receives the data, it can use the already shared corresponding public-key to verify the signing.

What's more, the public-key can verify data authenticity after signing This use case fails, however, if the does not maintain the private key's confidentiality securely. Most digital signing algorithms do not work directly with the data at signing, but instead operate on a hash of the data to sign. This approach enhances performance as most algorithms operate with large numbers, making it infeasible to process a large amount of data directly converted into a number.

A third use case for asymmetric cryptography involves key agreement protocols, which use asymmetric algorithms because of the increased speed of symmetric algorithms. The slower asymmetric cryptographic operations enable initial key agreements and then use a secret key for fast symmetric cryptographic operations to protect the confidentiality of communications.

Key agreement protocols are appropriate when both parties know the other's public-key from the start. Each party computes a shared secret value using his own private-key and the other party's public-key. Using secure hashing algorithms to derive additional keys may protect the shared secret value further.

Asymmetric cryptographic algorithm attack

Computer hackers can use several types of attacks on asymmetric algorithms. In most of these attacks, the attacker already knows the public portion of a key pair and searches for the corresponding private-key. A successful attack would enable the attacker to sign non-authentic data, or decrypt data encrypted for that specific recipient.

Brute-force attack

The most straightforward type of attack against asymmetric algorithms is by brute force. Most asymmetric cryptographic algorithms defend themselves from this type of attack by operating on very large numbers, which makes the search space of possible keys sufficiently large to make a successful brute force attack infeasible.

Man-in-the-middle attacks

A man-in-the-middle attack attempts to put the attacker in the middle of some portion of a key exchange operation, enabling them to fool both sides into reaching a key agreement with the attacker instead of with the expected partner. These sorts of attacks rely on insufficient protection during the sharing of public-keys. The use of Public Key Infrastructure (PKI) can help prevent these sorts of attacks in distributed environments. PKI provides a trusted authority that can securely authenticate and distribute the public-keys of individual participants.

Side-channel attacks

Side channel attacks also can be effective against asymmetric cryptography by targeting information leaked due to the implementation details of the algorithm and the specifics of the key used. Normally, this attack analyzes some side channel of information over several cryptographic blocks to determine the secret portion of the key pair.

Examples of the type of information that can be analyzed are timing information; cache hits and misses; and power usage.

The first asymmetric cryptographic algorithms were developed in the early 1970s. The first published work was the Diffie-Helman (DH) Key Exchange algorithm, which was based on the difficulty of reversing exponentiation in a finite field.

The next set of asymmetric algorithms to be published was RSA, named after the three authors Ron Rivest, Adi Shamir, and Leonard Adleman. RSA depends on the mathematical difficulty of factoring large prime numbers.

Over the years, RSA has increased key sizes to maintain security while keeping up with technological progress. In response to this trend, elliptic curve cryptography (ECC) has come into prominence. ECC depends on the mathematical difficulty of finding a discrete logarithm for a pair of points on an elliptic curve. Both RSA and ECC can be used as the foundation for one-way encryption or a signing/verification algorithm.

Diffie-Hellman (DH) key exchange is designed to reach agreement on a common shared secret key when communicating over a non-secure channel. It does not provide any authentication, so it is vulnerable to man-in-the-middle attacks when not paired with additional protection.

The DH protocol is described in detail in the IETC RFC 3526 standard. In general, it works by having each party agree on a non-secret base and a non-secret modulus. When operating mathematical functions with a modulus, all results are constrained between 0 and the modulus value.

If a mathematical operation result is over the modulus, the modulus is subtracted from the result until it is within the acceptable range. For example, 57 mod 10 = 7 (aka 57 – 10 – 10 – 10 – 10 – 10 = 7). With DH, each party raises the non-secret base to his own secret exponent constrained to the modulus, and then sends the result.

Each side then computes the received value raised to his own secret constrained to the modulus value. Because both sides raised the same base to both of the (secret) exponents constrained to the modulus, they now share the same secret value -- one that is hard to compute because reversing exponentiation in the finite field of the modulus is mathematically difficult.

The extent of the mathematical difficulty is dependent on ensuring that large enough values are chosen for the modulus and for both secret exponents.

RSA asymmetric algorithm

RSA refers to a set of protocols, all based on the mathematical difficulty in factoring large prime numbers. RSA algorithms are categorized based on the bit-length of the modulus (also called the “key size”). Early implementations of RSA used the smaller key sizes of 128 or 256 bits, which today can be broken in seconds.

Current implementations typically use 2048 or 4096 bit keys, with most recommendations leaning towards 4096 bit keys lengths where possible.

RSA public/private-keys pairs can be used for encryption, signing and verification, or key agreement. Encrypting a message involves converting the message to a number, which then is raised to the recipient’s public-key exponent modulus n. The recipient can retrieve the plain text by raising the received message to the private-key exponent modulus n.

The sender signs the message by converting it to a value, and raising that value to the sender’s private-key modulus n. The recipient then raises the signature to the sender’s public-key modulus n. If the result matches the sent message, then the signature was genuine.

ECC provides an alternative to RSA by helping address the performance of the mathematical operations that compute the RSA exponentiation for larger key sizes.

ECC relies on a completely different mathematical principle by operating on numbers that exist on a defined elliptic curve -- most often by using the named curves recommended in the NIST FIPS 186-4 Digital Signature Standard (DSS).

Curves in the DSS Appendix D are named according to the size of the prime in bits, with larger values indicating a larger set of possible points. The larger the set, the greater the mathematical difficulty in breaking the math underlying the cryptographic algorithm. Named elliptic curves in DSS include P-192, P-224, P-256, P-384, and P-521.

In ECC, public/private-key pairs represent two values. The private-key is an integer in the range of the prime value. ECC calculates the corresponding public-key mathematically by using a generator point, the private-key integer, and the concept of addition within the finite field of the elliptic curve.

Once it chooses an appropriate curve and generates public/private-key pairs, the key pairs can be used in a key exchange protocol or Elliptic Curve Diffie-Hellman Key Exchange (ECDH), or to digitally sign data using the elliptic curve digital signature algorithm (ECDSA).

ECDH is similar to the DH protocol, but operates on points on the curve instead of with exponents. In ECDH, each side keeps his private-key secret and shares their public-key, but then generates a shared secret point on the elliptic curve using their own private-key and their counterpart's public-key.

ECDSA is a variant of DSA that uses ECC instead of modular exponentiation. ECDSA requires a random value for each signature. One potential weakness is the reuse of the same random value with the same signature keys.

Implementations that fail to provide sufficiently random values can potentially be compromised because of the underlying math of the signature algorithm. For example, this vulnerability calculated the private-key to authorize games on the PlayStation-3. Correct implementations of ECDSA are still secure, but should use algorithms properly to avoid unnecessary vulnerabilities.

Current recommended key lengths

The Committee on National Security Systems (CNSS) Policy 15 Appendix B specifies a set of recommended key sizes for protecting national security systems. For asymmetric cryptographic algorithms, the publication recommends the following minimum sizes: curve P-384 for ECC; and a minimum of 3072 bit for RSA and DH key exchange.

Quantum computers, based on quantum bits (qubits), operate very differently from traditional Von Neumann architecture computers, and have significant implications for cryptography.

While traditional machines require 2^n bits to represent all possible values of a key of n-bits, a quantum computer, with n qubits, has a probability for those qubits to enter each of the possible key values.

This means that quantum computers can attack many mathematical problems that are infeasible with traditional computers. The bad news is that quantum computers can potentially solve integer factorization, exponentiation, and discrete logarithm problems, which are the mathematical underpinnings of RSA, ECC, DSA, and DH.

Fortunately, no quantum computer has been built with sufficiently large numbers of qubits to solve today's keys sizes reasonably. Still, the days of using existing asymmetric cryptographic algorithms may be numbered because of the fast rate of progress.

May 29th, 2019 | By: David Sheets

CRYPTOGRAPHY IS CRUCIAL FOR EFFECTIVE SECURITY IN TRUSTED COMPUTING: INTRODUCTION TO SYMMETRIC ALGORITHMS

Systems designers must employ the correct cryptography algorithms in their proper modes to secure secret or sensitive data in military trusted computing.

There are three general categories of cryptographic algorithms that are common in trusted computing: secure hashing, symmetric cryptography, and asymmetric cryptography. This column focuses on symmetric cryptographic algorithms, which are designed to take a block of data and encrypt it so that it is mathematically infeasible to extract the original information from the cipher-text without the key, which is used to both encrypt and decrypt the data.

In this approach, the same key encrypts and decrypts data to ensure the secrecy of that key, which is critical for ensuring the secrecy of the protected data.

> ***Most symmetric algorithms work on fixed sizes of data called blocks. The plaintext input is divided into blocks and each block is encrypted on its own to create the entire cipher-text stream.***

There are several types of attacks against symmetric algorithms. In some, the goal is to determine the key to enable the attacker either to decrypt the protected data or encrypt different data. This helps the attacker insert his own data without otherwise modifying the system. Other types of attacks on symmetric algorithms are just designed to obtain the unencrypted plaintext, which might be protected data or protected algorithms.

Brute-force attack

The most straightforward type of attack is a brute-force attack, which tries to defeat cryptography by decrypting data with all possible keys until it finds the right key. The difficulty in this type of attack results from the vast number of possible keys, which increases exponentially as the number of bits in the key are increased.

The birthday attack is a corollary to the brute-force attack. It attempts to try different likely inputs to find a collision between the result of encrypting the likely plaintext inputs and the known cipher-text. A birthday attack is most effective against symmetric algorithms with small block sizes and less variability in the input so that the search space is small. A birthday attack also requires a system that enables the attacker to enter his own input stream to be processed by the symmetric algorithm.

Most modern symmetric algorithms can resist known plaintext attacks, which attempt to determine the secret key based on the input text and the produced cipher-text. Once determined, the key can decrypt other messages that were not previously known.

Side-channel attacks

While all the previous attacks discussed work on the mathematical algorithm itself, side-channel attacks target information leaked due to the implementation details of the algorithm. Normally, some side channel of information is analyzed over several cryptographic blocks to determine the secret key.

Examples of the type of information that can be analyzed are timing information; cache hits and misses; and power usage. To obtain the needed level of information requires access to the system, either virtually or physically, depending on the type of side-channel information and the level of fidelity required.

As computing power has increased, several generations of symmetric algorithms have been developed and put into use. One of the first standardized symmetric cryptographic algorithms was Data Encryption Standard (DES), developed in the late 1970s. It set the stage for symmetric algorithms operating over blocks of data.

Later symmetric algorithms included Blowfish and Twofish. Today, the most popular symmetric algorithm is Advanced Encryption Standard (AES).

These algorithms operate on blocks, so they must run in a particular mode to be effective. The mode dictates how the keys are (or are not) changed in between each block.

DES/3DES symmetric algorithms

DES, one of the first publicly available block symmetric cryptographic algorithms, was developed by IBM Corp. with input from the U.S. National Security Agency (NSA) in the 1970s. It had a fixed block size of 64-bits, and an effective key length of 56-bits. After the security of DES began to wane, a new standard, called triple-DES (3DES), was formalized in FIPS 64-3. 3DES effectively runs DES three times in succession, with independent keys. While Triple-DES can still be used securely, AES is recommended for new applications.

The Blowfish algorithm was developed by Bruce Schneier to provide an efficient secure symmetric encryption algorithm unencumbered by patents. It operates on 64-bit blocks and supports effective key lengths of 32 to 448 bits. Because Blowfish suffers from slow initialization during key rolling, it is less efficient because many applications use frequent key rolling to combat side-channel attacks.

Twofish, designed as a follow-on to Blowfish, was entered in the National Institute of Standards and Technology (NIST) challenge to determine the AES standard algorithm, but was not selected. Twofish operates on 128-bit blocks and supports key sizes of 128, 192, or 256 bits. While neither Blowfish nor Twofish have any verified cryptographic vulnerabilities, they have received less attention than AES, so are not generally as trusted as AES.

The AES algorithm, formalized in the FIPS 197 Advanced Encryption standard, is the world's most widely used symmetric cryptographic algorithm. It resulted from a competition managed by NIST starting in 2001. A slightly modified subset of the Rijndael algorithm, AES operates on block of 128-bits and supports key lengths of 128, 192, and 256 bits. AES is widely accepted as the standard to use for most symmetric cryptography, and since its adoption, has received the most attention of any algorithm in research activity. Despite repeated attempts to defeat it, AES still remains effective when used properly.

Block symmetric algorithm modes of operation

Because all of the symmetric algorithms discussed above operate on blocks of data, there is a need to determine the relationship of the cryptographic operations performed on each subsequent block of data. The different methods of cryptographic operation of each algorithm are referred to as modes.

Many of these modes were described originally in relation to DES in the FIPS 81 publication "DES Modes of Operation." Additional modes were later added in NIST special publication 800-38A and NIST special publication 800-38E.

Electronic Cookbook Mode (ECB) is the most simplistic mode of operation and performs no operations on the key in between blocks. This can have disastrous effects on the confidentiality of the resultant cipher-text. As an example, see the original image of Tux [Image 1], and the same image after being encrypted using ECB mode [Image 2].

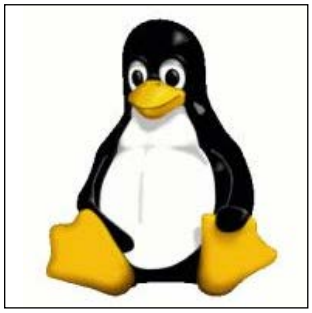


IMAGE 1

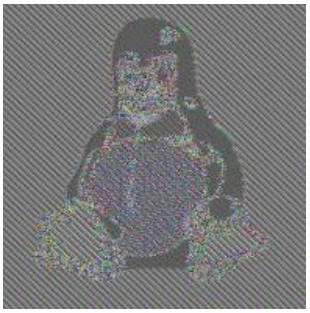


IMAGE 2

Because of the lack of key modification, blocks that are the same in the plaintext also come out the same in the cipher-text. That allows large patterns to emerge. Because of these issues, ECB should almost never be used in operational environments where confidentiality is desired.

Cipher block chaining (CBC)

Cipher block chaining (CBC) takes each plaintext block and performs the logical exclusive OR operation (XORs) with the previous cipher-text block. For the initial plaintext block, since there is no previous cipher-text, an initial value called initialization vector (IV) seeds the algorithm.

Although the IV does not need to remain secret or require protection like the secret key, in CBC mode, it is important to ensure that IVs are not reused with the

same key. Reusing the same IV with the same keys can leak information about the contents of the first block.

Cipher Feedback Mode (CFB) encrypts the IV for the first block, and then XORs the resultant value with the plaintext, instead of encrypting the plaintext. Each cipher-text block then re-encrypts to provide the next value to XOR with each subsequent plaintext block. As with CBC, IV should not be reused with the same key because it can leak information.

> ***Counter mode (CTR) uses the algorithm to encrypt an IV concatenated with a non-repeating pattern -- normally a simple incrementing counter. Then it XORs the plaintext with the resultant value to generate the cipher-text.***

Using a counter maintains an easy way to view an overall pattern, and ensures that decrypting an individual block does not depend on decrypting previous blocks. This can enable decrypting several blocks in parallel. In CTR mode, the IV must not be reused to ensure continued confidentiality.

Galois Counter Mode (GCM) with authentication works with AES to provide confidentiality and integrity verification. It uses the same CTR mode of operation for encryption and decryption, yet adds an authentication tag by taking the cipher-text through a polynomial function, and then encrypting the result. This approach verifies the block's integrity by reversing the process. If the decrypted tag doesn't match, then the block verification fails and should be discarded.

While authentication in AES-GCM can verify the integrity of blocks, this process has different security implications from asymmetric cryptographic algorithms that use public-private key pairs. Since public-private keys pairs can separate out the signing key from the verification key, ensuring the confidentiality of the key becomes much more important for maintaining integrity with symmetric cryptography.

Symmetric cryptography in trusted computing

Symmetric cryptography can provide confidentiality for data-at-rest and data-in-transit. Systems designers choose different algorithms based on their performance requirements, their desired level of security, and necessary certifications.

Systems designers should choose the correct mode of each algorithm to maintain confidentiality across the life of the product. Mode selection can have profound effects on system performance to take advantage of potential parallelization across several data blocks.

Some modes can decrypt, but not encrypt, in parallel. This may fit well with a secure boot model that prepares an image once, and then processes it many times. It's not a good match, however, for a message streaming model that must encrypt messages dynamically.

For symmetric algorithms, the key to security is protecting the key itself. Keys used during system startup have physical and process controls that can protect the secrecy of symmetric keys. For dynamic systems, key agreement protocols using asymmetric cryptographic algorithms can generate secret keys dynamically during operation.

April 24th, 2019 | By: David Sheets

CRYPTOGRAPHY IN TRUSTED COMPUTING: AN INTRODUCTION TO SECURE HASHING

Cryptography forms the foundation of many aspects of trusted computing. This article considers recent algorithms and cryptographic attacks, as well as some future directions for cryptography in deployed embedded systems.

Cryptography forms the foundation of many aspects of trusted computing. This article considers recent algorithms and cryptographic attacks, as well as some future directions for cryptography in deployed embedded systems.

This article provides a foundational understanding of the context of cryptographic algorithms, and delves into the details of secure hashing, common secure hashing algorithms, and some trusted computing applications of secure hashing.

> ***Today there are three general categories of cryptographic algorithms that are common in trusted computing: secure hashing, symmetric cryptography, and asymmetric cryptography.***

Hashing takes a relatively large piece of data and generates a relatively small unique value from that data. Given a large amount of data, if even one bit of the input data is changed, the resulting hash will change by a large amount and in a deterministic but unpredictable way. Cryptographic algorithms that use symmetric cryptography employ the same key to encrypt and decrypt the data.

Symmetric algorithms take data and encrypt them to be mathematically infeasible to extract the original information from the cipher text without the key. One key encrypts and decrypts to ensure the secrecy of that key, and so ensures the secrecy of the protected data.

In contrast, the third type of algorithm, asymmetric cryptography, uses a pair of keys. One is private and always should remain secret, while the other can be shared freely.

Asymmetric cryptography has uses like signing data, since it can prove who generated the data. Asymmetric cryptography also is for key agreement protocols to enable secure agreement on a shared secret key over an open network.

Hashing passes a large set of data through a mathematical one-way algorithm, and then generates a more compact, deterministic, and unique value. Hashing algorithms are fast, require minimal computational resources, and need to be “one-way” to ensure that it is computationally infeasible to find another input that matches the output value of the hash function. Some commonly used hash functions include MD5, SHA-1, SHA-2, and SHA-3.

Secure hash algorithms need the ability to resist collision attacks, pre-image attacks, second pre-image attacks, and length extension attacks. A collision attack attempts to find any two inputs A and B that, when hashed with the hash algorithm H(), generate the same output.

Find any two A and B such that H(A) = H(B)

During a pre image attack, the attacker knows output H(A) but does not know the input A. The attacker next attempts to find any input A, B, C… that will generate the same hash output H(A).

Given H(A) find any value B which may include A such that H(A) = H(B)

A second pre-image attack, while similar to a pre-image attack, stipulates that the attacker is trying to find some input other than the original input A that will generate the same hash as H(A).

Given H(A) find a B such that A != B and H(A) = H(B)

A length extension attack is when an attacker, given the output hash H(A), and the length of the original message, tries to design a new input A || B without needing to know A, that can generate a known output.

Given H(A) design H(A || B) with known B such that H(A || B) is a given output

MD5 hash algorithm

The MD5 hash algorithm first was published in 1992, and uses a fixed digest size of 128 bits. While designed initially for security, several vulnerabilities have been found in MD5 for collision and preimage attacks. While still often used as a checksum for files to check for unintended corruption, MD5 should not be used for security in trusted computing systems.

SHA-1 hash algorithm

SHA-1 was released in 1995, with a fixed digest size of 160 bits. As its secure hash algorithm name implies, it was designed for security. Unfortunately several papers have shown that SHA-1 is vulnerable to collision attacks. Since 2010, SHA-1 has not been recommended for use in the security of U.S. trusted computing systems, although exceptions are made for interfacing with legacy systems.

SHA-2 hash algorithm

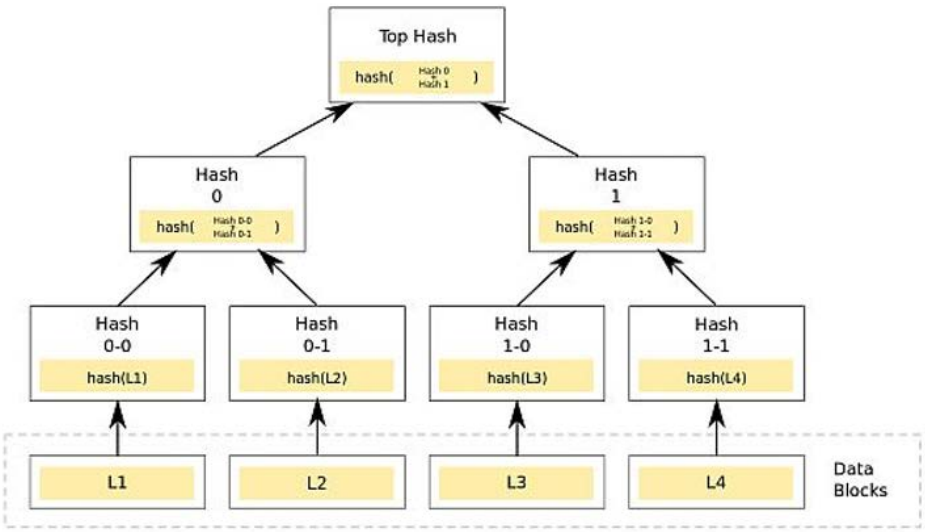
SHA-2 refers to a family of hash algorithms that were released in 2001. They are SHA-224, SHA-256, SHA-384, and SHA-512: the value denotes the digest size in bits. All members of the SHA-2 family use the same fundamental mathematical algorithm, with only slight variations to change the digest size.

SHA-224 and SHA-384 essentially are truncated versions of the SHA-256 and SHA-512 algorithms. While there have been several papers published that reduce the computational complexity required to break SHA-2, it is still infeasible for larger digest sizes. The CNSS Policy 15, published 20 Oct. 2016, recommends SHA-384 as the minimum for protection of classified information on U.S. National Security Systems.

SHA-3 hash algorithm

The SHA-3 family of algorithms was released by NIST in 2015. These algorithms fundamentally are different from the MD5, SHA-1, and SHA-2 algorithms, which all share a similar structure. While SHA-3 is not considered "better" than SHA-2, these hash algorithms provide additional tools to ensure there are robust solutions readily available to fill the same role if fundamental vulnerabilities are found in SHA-2.

There are no current plans to require a move to SHA-3 from SHA-2 algorithms for U.S. National Security Systems. The SHA-3 algorithms don't have a fixed digest size unlike the other common hash functions. That's because they employ a different mathematical structure. SHA-3 relies on sponge construction that provides for the concept of absorbing data and then "squeezing" out data after applying permutations.



NIST Publication 202, though, does specify a set of defined digest sizes and algorithms that are specified as SHA3-224, SHA3-256, SHA3-384, and SHA3-512. These would allow SHA-3 to be used as a drop-in replacement for SHA-2. SHA-3 also offers two additional algorithms, called extensible output functions (XOF), that can generate any given output length at a specific security strength. These XOFs are called SHAKE128 and SHAKE256; SHAKE is a combination of SHA and Keccak, the fundamental mathematical algorithm that SHA-3 depends on.

Secure hashing algorithms generate digests from data, an approach that many higher-level protocols use in trusted computing systems. For example, while not required for generating digital signatures over arbitrary data, secure hash algorithms can increase the processing speed of signing and verification operations for asymmetric cryptographic operations.

Secure hashing can help process large quantities of data quickly, and generate a relatively small hash. Slower asymmetric cryptographic operations then can sign or verify the smaller digest. This approach can help reduce overall processing time over a wide range of data sizes.

Secure hashing algorithms also is for hashing message authentication code (HMAC). HMAC algorithms, defined in FIPS publication 198-1, concatenate a secret key (K) with the message (M) and then hash the resulting combined message H(K || M). The specific implementation includes additional padding and key length details that are important to security, but the overall concept is as above. The recipient of the message M uses its own copy of the shared secret key K, concatenates it with the received message, and then performs the same hash operation to verify that the message is authentic.

Secure hash algorithms also can help in key derivation functions (KDFs), which are for higher-order key management techniques like key stretching or key rolling. Defined KDFs that depend on hash algorithms acting as a pseudo-random function are specified in NIST special publication 800-108.

Defined KDFs include a KDF operating in counter mode and feedback mode. In counter mode, keys are computed using a base key and a defined counter, which allows for missed blocks not impacting an individual key. In feedback mode, keys are computed using some set of previous keys that require some number of previous blocks to correctly calculate the next key.

One increasingly popular use of secure hashing is for developing hash chains, or more generally Merkle Trees. Hash chains and Merkle Trees are based on the repeated application of a hash function to data. Hash chains apply a hash recursively. This approach can support applications like one-time-use passwords.

A Merkle Tree applies a hash data blocks, which then generate the leaves of the tree. It computes each parent node of the leaf nodes as hashes of the set of the children nodes. This data structure enables application like verification of data integrity across distributed systems (git or mercurial content management system) or the verification of data integrity in file systems like ZFS.

Blockchain cryptocurrency, such as bitcoin, also relies on hash chains for recording the ledger of bitcoin transactions, with each transaction containing a hash linking it to a previous transaction.

Since secure hashing does not depend on mathematical operations that are solvable on quantum computers, secure hashing structures such as hash chains and Merkle Trees help to mitigate concerns over increasing capabilities in quantum computers to break other cryptographic algorithms.

March 28, 2019 | By Andrew McCoubrey and David Sheets

TRENDS IN TRUSTED COMPUTING AND NETWORK SECURITY IN AEROSPACE AND DEFENSE EMBEDDED COMPUTING APPLICATIONS

Andrew McCoubrey and David Sheets discuss network security in embedded computing for airborne networks.

Network security in embedded computing is getting more scrutiny these days. In a constantly evolving threat environment, where new attacks arrive virtually every day, system architects must design networks to be as secure as possible. That requires a constant review process to enable the necessary adaptation, modification, and updates to keep systems safe.

Network security involves providing protections against all devices that are connected or could have access to the network. In this area, embedded architectures are catching up to enterprise networks. In the enterprise environment, where there has always been the risk of an unauthorized person connecting on a port in an office or conference room, the need to lock down the network is well understood.

In comparison, airborne networks typically have been very controlled, with no network ports exposed. Physical access to ports in the past was easy to control. Today, however, we are seeing embedded networks connecting more devices and making more connection ports available, which makes trusted computing approaches imperative. Aboard commercial jetliners, for example, Ethernet might be available at every seat, and Wi-Fi might be provided for entertainment.

As more devices connect to the embedded network, the more of the network needs protecting. Adding to the security challenge is the growing use of converged networks. Instead of one purpose network, today's fast links can transport data from disparate systems over the same network. More systems sharing the network increases not only the potential for contention, but also the security challenge; more end points means more potential threats. We are seeing increased use of converged networking in military embedded systems.

The good news is there's growing awareness of what's necessary for effective network security; many of the important tools are familiar and readily available. One tool for securing the network is white-listing, or limiting access to trusted devices. This could be as simple as enabling each port only to allow traffic from a known MAC address. While simple to implement, MAC addresses can be changed and spoofed. Trusting a device just because it has the right address turns out not to be a very robust security solution.

A more advanced technique to keep out unknown users involves IEEE 802.1x for port-based network access control (PNAC). 802.1x enables the network to authenticate a network endpoint using a cryptographic exchange. Instead of trusting a MAC address, trust is based on a certificate or other credentials. It implements port security via a feature on the network switch. 802.1x is a hybrid feature that needs support on the switch; that's what controls turning the ports on and off). Still, it also requires clients, called "supplicants," on the end points. That means that implementing a protection like 802.1x requires a whole system solution in which both the switches and the connected computers provide support.

Another challenge for providing network security on embedded systems involves upgrade cycles. Adding a security layer on which only one device is secured can introduce a weak link -- unless all other devices on the network also have that layer of security.

While hard-coding and 802.1x enable control over what devices can access the network, MACsec and IPsec tools use encryption to protect data on the move and prevent someone from snooping into that data. IPsec is an end-to-end protocol used originally for VPNs that connect from one office to another office over an untrusted network. In comparison, MACsec secures only a point-to-point connection.

IPsec and MACsec help encrypt network data, and validate keys when establishing connections, but differ in how much data they encrypt. IPsec, for example, supports tunneling and transport modes that offer tradeoffs between overhead and the amount of encrypted data.

Apart from IPsec and MACsec, there are encryption standards like transport layer security that work at the application level. These require less support from the network infrastructure, but consume more processor overhead and encrypt even less, because they exist at the highest layers of the network stack.

Today, we typically see IPsec in local networks like airborne networks that are contained entirely within an aircraft. This protects against data being intercepted by other devices on the network. It also provides protection if the network switches are compromised.

It's important to select network equipment and end points that provide good performance because they encrypt network traffic at high rates. MACsec encryption is typical for hardware, and is built into the PHY devices that provide the link-layer connections. IPsec encryption typically happens in software, but can require hardware acceleration to keep up with the network.

Standards also can change over time, so it is important to stick with the latest versions. Early implementations of MACsec supported only AES 128 bit encryption keys, while AES 256 bit encryption was added later.

A major challenge can be finding people with the necessary expertise. Most are familiar with enterprise IT, yet those who know rugged embedded systems may not be up to date on the latest networking technologies. The intersection of people who understand both networking and embedded systems is small, but growing. What's helping that intersection grow is the Internet of Things (IoT) phenomenon, which is taking all manner of embedded devices that were traditionally stand-alone appliances, and connecting them to networks.

As systems designers ask for network security solutions, their specific requirements still can be vague. Security, as yet, isn't something you can just buy from a vendor. Instead, systems designers need to implement security across all products in the system, and either perform the architecture work themselves, or hire experts to do it for them. To design-in effective network security, it's wise first to reach out to vendor network security experts at the very beginning of the project.

Keep in mind, too, that network security doesn't end with the architecture or initial implementation; it needs to be revisited on a regular basis. Patches likely will be necessary to address vulnerabilities at each iteration, at every software upgrade, and every time a new device gets added.

Even some fundamental tools to secure networks may need to change. Most underlying network security today is based on public key cryptography. Future generations of quantum computers likely will require a reexamination of some cryptographic primitives on which network security designers depend.

Network security is a complex and evolving challenge, and solutions are being developed continually to address emerging threats. Leveraging the latest commercial technology is key to ensuring that connected embedded systems are secure, today and in the future.

Feb 27th, 2019 | By Paul Hart

OPTIMIZING CYBER SECURITY AND TRUSTED COMPUTING ON TODAY'S CONNECTED MILITARY AND COMMERCIAL AIRCRAFT

If not properly protected by taking trusted computing measures, every system on an aircraft can create a potential vulnerability. Paul Hart talks about optimizing cyber security on today's connected military and commercial aircraft.

The number of data communication links and interactions between an aircraft and the ground systems supporting it are ever increasing. If not properly protected by taking trusted computing measures, every system, sensor, and module on the aircraft can create a potential vulnerability that unauthorized users can exploit to obtain confidential sensitive data or, worse, disrupt the safe operation of an aircraft.

Such threats are of paramount concern for all areas of aviation, including military aircraft and the increasingly top-of-mind unmanned aerial vehicle (UAV) market. Systems developers must safeguard the exchange of tactical information and the integrity of command and control links between ground stations and airborne platforms.

> ***Designers also must incorporate additional cyber security features into avionics systems to minimize the number of different points at which an unauthorized user can input or extract data -- also known as the "attack surface."***

The electronic connectivity onboard commercial or military aircraft includes wireless data links for downloading flight recorders or uploading terrain databases and navigation databases. On commercial aircraft, it also includes the networks within the aircraft that support passenger in-flight entertainment and passenger satellite communications (SATCOM) connectivity that enables passengers to surf the Internet while in-flight.

Today, it's not uncommon for aircraft pilots to use tablet computers in the cockpit. These tablets, known as electronic flight bags, connect through a Wi-Fi receiver integrated into an avionics interface device in the cockpit. This connects to various avionics databuses to transfer data from the avionics systems to the tablet, which the pilot then uses to run applications like calculating takeoff V-speeds and load sheets.

Connectivity means potential targets of opportunity for malicious actors. There may be a computer hacker flying as a passenger, who has the duration of the flight in which to attempt access to the in-flight entertainment system, or more seriously, to the avionics interface device. There also may be a disgruntled airline employee with a valid pass and access to the aircraft and its equipment, who can operate inside the wire with no questions asked. The challenge is how to protect avionics against these sorts of vulnerabilities?

Protecting data-at-rest

Nearly every aircraft operating in controlled airspace today is equipped with a flight management system (FMS). This enables flight crews fly pre-programmed routes from an onboard database containing important information such as airspace structures, ground-based navigational beacons, and airport information like runways and taxiways. The database also contains the flight trajectories for standard instrument departures (SIDs) and standard arrival routings (STARS) that can fly the airplane automatically after takeoff and during approach.

The FMS typically updates every 28 days under the aeronautical information regulation and control (AIRAC) cycle. The database content comes from official state sources by service providers, but the ultimate responsibility of data integrity rests with the end-user.

FMS database updates typically are uploaded by USB memory stick as a line-maintenance function; the USB content having been downloaded from a secure website or FTP server. This crucial FMS data is stored in non-volatile memory and, if compromised, could prevent an aircraft from operating or landing safely. However, using authentication and encryption techniques, it is possible to guarantee to the end user that the data flow from the FMS service provider to the on-board aeronautical database has not been violated.

Protecting data-in-motion

Protecting data in motion on the aircraft is equally important. For the networks, there are security layers that provide authentication. Two key examples of these security layers are the security protocol suites Internet Protocol Security (IPsec) and MAC Security standard (MACsec: IEEE 802.1AE). They can be built into the network layers to ensure that end-to-end communication cannot be disrupted or hacked or tapped into.

The MACsec standard strengthens network security by identifying unauthorized local area network (LAN) connections and excluding them from communication within the network. The protocol authenticates nodes through a secure exchange of randomly generated keys, ensuring data can be transmitted and received only by MACsec-configured nodes, and provides optional point-to-point, Layer 2 encryption between devices on a virtual or physical LAN.

IPSec provides similar protection for a wide area network (WAN). It works on IP packets at Layer 3 (as opposed to Ethernet frames at Layer 2, like MACsec). For an FMS, which traditionally requires data to be uploaded manually yet can now receive such uploads via wireless technology, such protocols are important for protecting the integrity of the data during transfer.

In addition to the connected aircraft itself, with the Next-Generation Air Transportation System (NextGen) in the US and Single European Sky ATM Research (SESAR) in Europe, advanced air traffic control systems are coming online quickly to enable denser air traffic with reduced tolerance for error.

Security and next-generation air traffic control

A surveillance technology that uses GPS satellite navigation data to determine an aircraft's position, called Automatic Dependent Surveillance-Broadcast (ADS-B), will be mandated in many controlled airspace regions from 2020. ADS-B will send out automatic position report pulses, called extended squitters, to broadcast the aircraft's position. This information can be received by air traffic control as a replacement for secondary surveillance radar, since no interrogation signal is needed from the ground.

ADS-B also can be received by other aircraft to provide situational awareness and allow self-separation. On more advanced aircraft, it will be used to report not just the aircraft's current position but also its flight path to destination so that the ATC system and other aircraft also can predict where the aircraft will be in the future.

This predictive data enables other aircraft in the same area to compute their own route to ensure that they don't converge with the first aircraft's flight path.

The search acquisition radars used at airports for years are rapidly going away because they are only good for providing line-of-sight data.

> ***With today's sophisticated onboard navigation systems, it's now better to let separate aircraft work out amongst themselves how closely they can approach each other safely in the sky.***

To provide guidance for handling the threat of malicious interference with aircraft systems, the Radio Technical Commission for Aeronautics (RTCA) released DO-326A, titled "Airworthiness Security Process Specification." This document complements other advisory material, such as the hardware and software safety certification guidance documents DO-254 and DO-178C. DO-326A outlines compliance objectives and data requirements for aircraft and airborne equipment manufacturers.

The DO-326A Airworthiness Security Process Specification

DO-326A provides guidance on the interactions between security and safety. As the DO-254 safety certifiability standard for hardware requires a plan for hardware aspects of certification (PHAC), and DO-178C requires a plan for software aspects of certification (PSAC), the DO-326A standard calls for a plan for security aspects of certification (PSecAC). Today, any new aircraft system that is connected to the outside world must address the DO-326A requirements that are flowing down from the regulators.

DO-326A covers such things as navigation databases and terrain awareness warning databases, though it doesn't provide specific direction on how to implement required safeguards. Instead, it mandates a process under which all threat scenarios and use cases are identified and adequate measures are put in place to mitigate them.

Under DO-326A, a systems integrator deploying any new avionics, such as a navigation system, onto an aircraft must demonstrate that they have protection measures in place and that they've identified the necessary aircraft and security perimeters to mitigate against a malicious actor.

In recent years, as FAA and EASA regulators are being asked more about cyber security, there has been a growing emphasis and insistence that DO-326A processes are put into action. That makes it incumbent on avionics systems integrators to educate themselves so that they are aware of this standard, as it will only become more important in coming years.

November 29, 2018 | By: David Sheets

INTRODUCTION TO CERTIFICATION AUTHORITIES FOR TRUSTED COMPUTING IN MILITARY AND AVIONICS PRODUCTS

In the world of security and trusted computing, there are many different disciplines involved. With a constantly evolving set of standards, learn what certifications might apply.

In the world of security and trusted computing, there are many different disciplines involved, from cyber security to safety certification. With a constantly evolving set of standards and possible certifications, it can be confusing to understand what certifications might apply to your system, which of those are worthwhile, and what the important aspects are when considering certification or certified products.

> ***There are certification authorities involved in trusted computing, which oversee different disciplines they oversee. It can be a challenge on when to get these certification authorities involved, and make judgments on which these bodies are relevant in the U.S. and some international markets.***

The National Institute of Standards and Technology (NIST) manages myriad standards across many industries. Some of these standards include areas of trusted computing, including cryptographic algorithms and documents used to define the Risk Management Framework (RMF).

Here are some of the certification programs related to trusted computing in military and avionics applications that are administered by NIST.

The Cryptographic Module Validation Program (CMVP) is administered together by NIST and the Canadian Centre for Cyber Security (CCCS). This program performs independent testing of cryptographic modules at independent labs for conformance to FIPS 140-2 Security Requirements for Cryptographic Modules. For stand-alone cryptographic modules, this certification can show thorough testing to provide confidence to customers on the security and implementation of cryptographic algorithms. FIPS 140-2 provides for multiple security levels (1 to 4) mainly related to physical security capabilities, so vendors need to ensure that they apply for the appropriate level of certification, and customers need to verify that products are certified to meet their required level of security.

The Cryptographic Algorithm Validation Program (CAVP) ensures that cryptographic algorithms have been faithfully implemented, either in hardware or software. CAVP is a prerequisite to CMVP. Systems designers can select subsets of algorithms for validation, and NIST maintains the list of certified testing laboratories and validated algorithm implementations.

The Risk Management Framework (RMF) is for assessing risk and is designed for federal information systems. Apart from assessing risks, the RMF also provides guidance on selecting controls to mitigate risk, and then authorizes and monitors those systems. While the framework itself is maintained by NIST in special publications SP 800-53, SP 800-34, SP 800-61, SP 800-53A, SP 800-37, SP-800-137, SP 800-60, and others, NIST does not perform assessment or certification of systems under RMF. Programs should be assessed under guidance of the SP 800-53A Guide for Assessing the Security Controls in Federal Information Systems and Organizations. Different departments and agencies will determine who provides the oversight needed to sign off on after appropriate assessments are completed in order to allow systems to operate.

The Common Criteria (CC), administered by the International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), is a framework to help evaluate products against a defined security target (ST) and security functional requirements (SFR). Normally, products will pull much of their ST from an already defined protection profile (PP) for a given set of products. Products are then evaluated against their defined ST and SFR at an independent lab. Furthermore, when being evaluated, the Evaluated Assurance Level (EAL) can also be selected from level 0 to level 7. The EAL dictates the strictness of the evaluation, with higher levels looking at the entire development process of the product,

and the highest level requiring formal verification of security claims. It is important to remember that higher EAL levels do not imply higher security; they simply show that there is a higher level of confidence in the verification of the security claims. Because CC certification by itself only states that the evaluated product meets its defined ST and SFR, vendors must select and define, and customers must evaluate the ST and SFR of any product to ensure that the defined capabilities meet the security needs for their system and have been evaluated to the appropriate confidence.

The National Information Assurance Partnership (NIAP) manages the certification of commercial off-the-shelf (COTS) components to Common Criteria (CC) certification. NIAP works with certified testing laboratories to perform CC certification and maintains the list of validated products.

The Defense Information Systems Agency (DISA) within the U.S. Department of Defense (DOD) helps ensure continued operation and security of the DOD Global Information Network. DISA also manages a repository of Security Technical Implementation Guides (STIG) that can help secure computing systems. STIGs can range from general to product-specific. While DISA does not perform certification, they do maintain the set of STIGs used to secure systems, and they approve submitted STIGs prior to including them in the list. Vendors who want to generate and provide specific STIGs for their own products can submit them to DISA for approval and inclusion.

The Commercial Solutions for Classified (CSfC) is a program of the U.S. National Security Agency (NSA) that takes CC-certified security solutions, layers those solutions to produce a product, and certifies that the product can securely protect National Security Systems (NSS) that operate on classified data. NSA may put additional requirements on a product, or require that CC protection profile selections for products are included on the CSfC list. The designer should start discussions on CSfC with NSA prior to going through CC certification for individual portions of that CSfC product. CSfC provides an alternative to using Type-1 NSA certified cryptography. Its use does present tradeoffs that can affect product life cycle, key management requirements, and product classification.

The RTCA DO-178C/EUROCAE ED-12C Software Considerations in Airborne Systems and Equipment Certification is a U.S. Federal Aviation Administration (FAA) design assurance guideline to approve the airworthiness of aviation software. It details the requirements for software development, testing, test coverage, and reliability. There are multiple design assurance levels (DALs) based on the level of criticality of the system failing, with “A” indicating catastrophic danger and “E” indicating no impact on safety. Since DO-178C can influence the entire software development process, ensuring that the requirements are well understood prior to starting development is essential.

The RTCA DO-254/EUROCAE ED-80 Design Assurance Guidance for Airborne Electronic Hardware standard is the FAA hardware counterpart to DO-178C, and provides guidance for certification of complex avionics components that can influence flight safety. As with DO-178C, levels of criticality exist (A to E), and the DO-254 guidelines can influence the entire hardware development process. Requirements to meet DO-254 certification should be understood before beginning development of a new complex hardware avionics component.

December 19, 2018 | By David Sheets & Paul Hart

A GUIDE TO INTERNATIONAL AUTHORITIES FOR GLOBAL TRUSTED COMPUTING STANDARDS CERTIFICATION

David Sheets and Paul Hart provide a guide to the international government agencies that perform security accreditation for equipment used in military applications.

Trusted computing standards are evolving constantly within the United States, which can be confusing enough. Now consider international trusted computing standards that may apply to a growing number of embedded computing projects and you can encounter a real dilemma.

This overview could help. It's a guide to the international government agencies that perform security accreditation for equipment, such as network switches, storage devices, and ruggedized computers used in military applications.

International agreements exist like the National Information Assurance Partnership (NIAP), that recognize Common Criteria (CC) schemes and Protection Profiles (PP) to reduce the level of re-assessment on international military program standards.

Nevertheless, applicants generally must submit a Security Target document that describes the Target Of Evaluation (TOE) and the relevant protection features built around the critical security areas, such as hard drive encryption, key management, and secure boot that cryptographically verifies executable code on power-up.

The extent of these features depends on the type of program. For example, secure boot can range from validating checksums prior to loading code, to verifying cryptographic signatures and decrypting all boot artifacts. Government agencies and international authorities will issue high-level requirements that specify the protection levels. There also are specific agencies and protection schemes for individual countries.

The United Kingdom Ministry of Defence (MOD) in London issues a Security Aspects Letter (SAL) on a new military program. In turn, the National Cyber Security Centre (NCSC) in London determines the assessment process level -- the highest grade being the CESG Assisted Product Service (CAPS), with official-level programs following the Commercial Product Assurance (CPA) certification route.

CAPS assessments are performed directly by NCSC, whereas CPA approvals are outsourced to licensed evaluation facilities.

Companies performing development work in this field also need to meet facility-level IT and access security requirements which flow down from Defence Condition DEFCON 658 (Cyber) and Defence Standard DEFSTAN 05-138 (Cyber Security for Defence Suppliers). The MOD sponsors a scheme known as Cyber Essentials Plus, whereby potential suppliers can seek accreditation against these requirements.

The French government agency Agence nationale de la sécurité des systèmes d'information (ANSSI) is equivalent to the United Kingdom NCSC. It two certification schemes that depend on the security level, which are assessed by facilities approved by the Centres d'évaluation de la sécurité des technologies de l'information (CESTI).

Among the CESTI schemes is the Critères communs (CC), which is equal to the Common Criteria. It is applicable to products already accredited in another country, like the United States, United Kingdom, and Canada, that is a signatory to the ITSEC accord. It also is based on ISO15408 (IT Security) and focused on network, enterprise-level computing.

The Certification sécuritaire de premier niveau (CPN) represents first level security certification. Accreditation to this scheme is not normally recognized outside of France.

Equally, the German federal office for information security, known as the Bundesamt fur Sicherheit in der Informationstechnik (BSI) offers similar accreditation schemes.

The Italian Ministry of Economic Development is that country's regulatory agency for security & integrity for electronic communication systems.

The Spanish government agency for certifying cryptographic equipment is the Organismo de Certificación - Centro Criptológico Nacionalis (OC-CCN).

In Turkey, the Ulusal Elektronik Ve Kriptoloji Arastirma Enstitüsü (UEKAE) is the Turkish National Research Institute of Electronics and Cryptology.

In South Africa, COMSEC Electronics Communications Security is a company owned by the South African government. It secures government communications against unauthorized access, and provides verification services for electronics communications security systems, products, and services used by the government.

KISA is the South Korean government agency responsible for international cyber security. It is managed by KrCERT/CC within KISA. South Korea has an NCSC within the National Intelligence Service that coordinates with KrCERT/CC.

The Australian Department of Defence State Security Agency provides cryptographic evaluations under the Australian Signals Directorate (ASD).

For the European Union, the European Union Agency for Network and Information Security (ENISA) is the certification organization for trusted computing.

Cryptographic products in a NATO member nation are subject to approval by the developing nations National Communications Security Authority. These include those that are evaluated and approved in accordance with the INFOSEC Technical and Implementation Directive on Cryptographic Security and Cryptographic Mechanisms.

These products are eligible for inclusion to the NATO Information Assurance Product Catalogue (NIAPC). The list of cryptographic products and cryptographic mechanisms is updated and maintained by the NATO Communications and Information (NCI) Agency Cyber Security on behalf of input provided by the National Communications Security Authority of the NATO member nation.

The NCI agency develops interoperable C4ISR capabilities, operates the NATO Information Assurance Product Catalog (NIAPC), and recognizes Collaborative Protection Profiles under the NATO Common Criteria. The NATO Cooperative Cyber Defence Centre of Excellence (CCD COE) is based in Tallinn, Estonia.

To sell trusted computing and security products effectively into North American and international markets, companies must ensure that the correct certification authorities and regimes are identified and understood. This will help pave the way for the acceptance of their offerings.

Also, suppliers can deliver significant cost reductions to their customers, as well as greater security testing and resiliency, by capitalizing on certifications across different markets.

October 24, 2018 | By: Kira Reid

ESTABLISHING A TRUSTED SUPPLY CHAIN FOR EMBEDDED COMPUTING DESIGN

Read how the Curtiss-Wright trusted computing services are one example of the processes and procedures essential for protecting the supply chain of embedded computing components.

Best practices for establishing a trusted computing supply chain involves establishing state-of-the-art processes for rugged industry-standard open-architecture embedded computing like VME, VPX, PMC, and XMC. These processes are in place to reduce risk and mitigate malicious threats against hardware or data.

> ***The best way to lead is by example. The Curtiss-Wright TrustedCOTS services are one example of the processes and procedures essential for protecting the supply chain of embedded computing components.***

These services include data protection for data-at-rest and data-in-transit and technology protection for anti-tamper. they also set the bar for trusted supply chain processes, including physical security; manufacturing security; component supply chain integrity; secure handling and chain of custody protection; product reliability and VITA 47 testing; counterfeit parts mitigation; and parts inspection.

Secure supply chain

The most important starting point in a secure supply chain is only to buy components directly from franchise sources, from the component OEM, or through authorized distribution channels. Control over the components chain of custody can be ensured only by purchasing through controlled, authorized channels.

Embedded computing subsystem vendors must follow the same stringent controls applied to semiconductor device suppliers when they purchase third-party mezzanine modules. The vendor of commercial off-the-shelf (COTS) embedded computing must control not only the source of supply for components, but also for the source of supply for mezzanine modules or subsystems that could have the potential to harbor counterfeit parts or tampering.

The COTS vendor must flow-down all of his requirements for using franchise suppliers to the module vendor. If the module vendor must go outside of standard channels, he also must ask the COTS vendor for approval. That way the COTS vendor can control the component's source and purchase, as well as required authentication, testing, and other requirements necessary before proceeding.

Obsolescence and the supply chain

The embedded computing industry faces a contradiction; companies build builds products made with commercial parts with short life spans, yet that must function in integrated systems with very long lifetimes. In this environment, component obsolescence is a fact of life.

When a part goes through end-of-life (EOL), the first point of mitigation is to identify if the part manufacturer or his authorized distributor has a drop-in replacement available. If not, the vendor should pursue a replacement via a last-time buy (LTB). An LTB can help minimize the risk of obsolescence by extending product life and avoiding the need later to procure obsolete parts of questionable lineage from brokers.

Sometimes COTS vendors can find obsolete components only through parts brokers that are secondary sources unauthorized by the component OEM. The COTS vendor should use components from these suppliers only when authorized sources are no longer available, and then only with the approval of the customer. A brokered part should NEVER be used on a board without explicit customer approval.

All parts from these suppliers should be tested at authorized third-party test facilities to ensure that they are authentic components meeting the original design specifications, and have not been subject to prior use or tampering.

Broker sources must be subject to intense audits, and any device provided by a broker must be tested by internal or accredited labs in compliance with customer-, supplier-, and industry-mandated validation methods. All test reports should be reviewed by the

COTS vendor's component engineering and quality teams prior to accepting the material into inventory. When using an authenticated broker part, the COTS supplier must perform disciplined configuration control to ensure that the brokered part receives a unique part number. Any brokered material must be controlled and segregated from franchised components so that its source of origin is always clear.

COTS vendors should be active participants on relevant standards committees, such as the Society of Automotive Engineers (SAE) International AS6081, which defines best practices and requirements for working with trusted broker partners. For example, under AS6081, manufacturers must establish a quality management system and retain appropriate records for supply chain traceability. Any broker that Curtiss-Wright works with must be authenticated to stringent industry requirements, which we review internally.

Also, all material must be authenticated before it comes into the manufacturing site. Even then, the customer must approve any part acquired through a broker before it can go on a board. The customer has the option of whether or not to approve the authentication reports, ensuring that they are comfortable with the material.

Counterfeit parts mitigation

Counterfeit parts are a major concern for the U.S. Department of Defense (DOD). COTS vendors and electronics suppliers routinely discuss with their customers and suppliers how to prevent use of counterfeit parts.

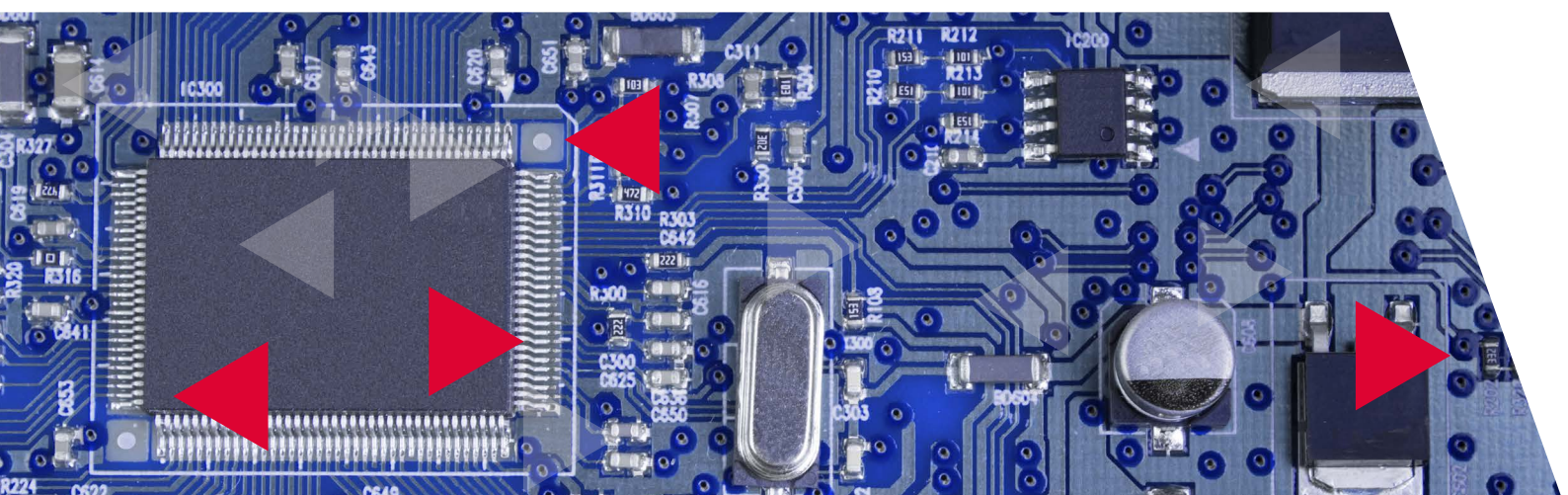
Curtiss-Wright has implemented controlled processes designed to prevent counterfeit parts in the supply chain at any point in the product life cycle. This includes stipulating an SAE AS5553 (Counterfeit Electronic Parts; Avoidance, Detection, Mitigation, and Disposition) compliant process that uses approved vendors who have been audited and monitored by the COTS vendor to provide an audit trail for all components used, and to ensure their authenticity.

Many of the best practices for counterfeit mitigation -- including methods, systems, and trend monitoring -- result from participation in industry committees. Leading the way in efforts to prevent counterfeiting are the SAE G19 committee and various other industry consortia, such as the Electronic Resellers Association International (ERAI) and the Independent Distributors of Electronics Association (IDEA).

The continued development and evolution of standards results from partnerships with suppliers and customers, and through counterfeit-mitigation systems. New relevant regulatory standards are continually released.

For example, Curtiss-Wright also complies with the SAE AS6174 standard ("Counterfeit Materiel; Assuring Acquisition of Authentic and Conforming Materiel"), which includes nonelectrical component counterfeit mitigation. Defense and aerospace suppliers -- all of whom are routinely audited by customers to verify their compliance for electrical, electronic, and electromechanical (EEE) parts -- are required to evolve their methods in recognition of the non-EEE component counterfeit threat.

To ensure that their supplier is up to date with the latest trends in underworld counterfeiting and the most up-to-date techniques for mitigating them,



systems integrators should ask their suppliers about the industry organizations they belong to, and which organizations they monitor. Maintaining membership in and monitoring the latest proposals from committees such as SAE G19 AS5553 is critical for staying ahead of the counterfeit game.

Leading COTS suppliers should monitor organizations that track trends in counterfeit materials, and share their experiences with groups like the Government Industry Data Exchange Program (GIDEP), the United Kingdom's ESCO Council, and counterfeit-mitigation suppliers such as ERAI. Active involvement and communication with these bodies is critical for reducing the risk posed by counterfeits.

A company's certifications, like as AS5553, AS6081, or ISO/IEC 17025, is one way to verify how well a COTS supplier can mitigate counterfeits. SAE has released a technical certification in their National Aerospace and Defense Contractors Accreditation program (NADCAP) suite that provides technical acknowledgement, via auditing, that a supplier adheres to AS5553 requirements.

It is imperative that COTS suppliers protect their own manufacturing facilities and mitigate any insider threats. Curtiss-Wright manufactures its computer module products in the United States, Canada, and Europe. Employees working on the manufacturing floor are subject to background checks, and ITAR cleared before they are allowed to work on the manufacturing line. All employees are trained to ensure their competency. All sites are accredited to handle ITAR information. In addition, the company's U.S. manufacturing site is a secure facility and can handle classified data and components.

COTS vendors should use only trusted contract manufacturers. It is ideal to work with a contract manufacturer that is IPC Validation Certified as a Class 3 Trusted Source, and has a long legacy of high-performance in the military C4ISR sector.

When working with a U.S. Eyes-Only contract manufacturer it's critical to ensure that all materials are U.S. Eyes Only-based, including circuit boards and metal work. It's also critical to ensure the contract manufacturer adheres to all of the processes and standards that are used in the COTS vendor's own facilities.

Design Integrity

To ensure design integrity, the COTS vendor should comply with the DOD's Risk Management Framework (RMF). This framework provides controls for the information technology infrastructure to mitigate against unauthorized access, modification, loss, or theft of critical design information.

Communications is key in the relationship between COTS vendors and system integrators. One approach for improving communications is to establish a 24/7 web-based information portal that provides the system integrator with updates and visibility into what parts are approaching EOL, informing about LTB opportunities, or assuring them that a drop-in replacement part is available.

Curtiss-Wright has established its Total LifeCycle Management (TLCM) services portal, an optional communications channel that enables customers to manage the trusted supply chain. It provides insight into what's coming down the pipeline from industry and disseminates information such as reports, and material inventory status.

In today's world, threats are unending and constantly evolving. That's why COTS vendors should remain constantly vigilant, to protect and verify the integrity of the supply chain. Trust in your supplier's certification but verify their performance to ensure that they, and any brokers and distributors they use, have demonstrated their abilities in counterfeit mitigation.

September 26, 2018 | By David Sheets

DECOMPOSING SYSTEM SECURITY TO PREVENT CYBER ATTACKS IN TRUSTED COMPUTING ARCHITECTURES

Trusted computing systems designers should consider system security early in the design process to prevent cyber attacks.

High-level security requirements in trusted computing systems can flown down to the system from several sources, such as a request for proposal (RFP) or from reference documents that provide system design guidance. The best approach for dealing with security requirements is to follow a framework that determines the design process, based on the necessary security levels.

This framework not only will specify how to implement necessary security, but also will lead the designer through the thought process. For example, the Committee on National Security Systems (CNSS) at Fort Meade, Md., provides direction on the controls that designers should consider for system confidentiality, integrity, and availability.

Similarly, the CNSS provides a path for designing-in authorization capabilities or message integrity into system security. Although some security guidance documents for U.S. military systems are classified, there are similar sorts of documents that tell the system security designer what he needs to think about as he goes through the architecture and design process.

After establishing frameworks for high-level requirements, the customer and integrator must agree on how best to meet those requirements. This considers the context of the system and the use cases in which it will operate, as well as a risk analysis that looks at where and how the system will operate.

Will there be armed guards always present, for example, or will the system operate unattended for weeks or years? The risk analysis will evaluate how these different use cases influence the risks that different types of cyber attacks pose.

High-level architecture development

The first step in risk analysis is creating a high-level logical system description to model data and flows; this should happen when the system architecture is still being defined to best mold tradeoffs between security requirements and the implementation.

It's important to have the system security engineers active and involved early to optimize design trade-offs. There will fewer opportunities to mitigate security vulnerabilities once the physical implementation is set, which makes any needed modifications more difficult and costly. Even worse, systems designers may be forced to accept some vulnerability in the system if they don't take care of security concerns early.

Identify what to protect

Next, the system security architect must identify two keys aspects of system security: determining what he needs to protect, and identifying the threats he must protect against. What needs protecting can involve specific algorithms, data, interfaces, and capabilities. Anticipated threats, meanwhile, can involve preventing potential attackers from viewing or modifying certain algorithms -- especially if systems or software developers need to share those algorithms.

Some systems call for protecting the actual data -- especially in implementations like signals intelligence or reconnaissance imagery from an unmanned aerial vehicle (UAV) to protect against remote hacking. The security team also must consider other applications with which their system will communication, and their classification levels.

Systems designers also must take planned system availability into consideration when they identify potential threats. It's important to mitigate against a denial of service (DoS) attack that could bring the system down.

Next, the designer should identify expected types of system attacks, like remote attacks, local attacks, or insider threats. Will potential attackers be typical hackers, resource-rich organized criminals, or nation states?

To protect against an insider threat, it's important to ensure that no backdoor routes were developed in the system software. Similarly, systems designers must protect hardware from nefarious components that could fail if someone sends a key phrase. The entire hardware supply chain, moreover, must be tested and verified.

Remote attacks can include attempts to access a UAV video feed. An attacker may gain access to the system data to modify its operation or deny availability -- especially if it's connected to the Internet. Local attacks can be as simple, yet dangerous, as a malicious actor plugging in a USB key to capture data.

Map logical attacks to physical components

System security engineers can map a nominal physical system architecture to the logical system architecture to identify potential physical avenues of attack. The system logical diagram defines what the system does and what functions it needs to perform.

The next step is to decompose that information to an actual physical architecture that identifies the circuit boards on which those functions reside, and how those boards communicate with each other. The designer needs to define the backplane, network interfaces, and how data will flow throughout the system. This will enable system security engineers to map system operation to potential attack vectors, and propose remediations for each anticipated attack vector.

Defining the final system security plan

When systems designers finish all this, they can start working with the customer to make changes to the system's physical architecture. It may be necessary to remove some connections from a single-board computer, boost security by adding an encryptor into a data path, or add an encrypted hard drive to store data. The designer may want to apply additional logical constraints to provide remediation. In this way, they can change the security aspects of the way the system operates, instead of making changes to the physical architecture. They could implement software to verify the current functionality of the system.

Designers should weigh the cost of remediation for each type of anticipated attack as they identify which techniques to apply. Equally important, however, is determining new remediations have created new system vulnerabilities. Are there new remediations that need to be applied?

Designers must weigh the risks of different attack vectors with the costs of remediation; this will influence how best to proceed. One cost-effective remediation might protect against many attacks, while it may be difficult to justify using a very expensive remediation to protect against one attack vector.

Risk analysis might reveal a potential attack type that isn't very probable, or a potential attack with little chance of system access. In this case, system security engineers may decide not to protect against this kind of attack, or to mitigate it with an inexpensive less-effective approach. A low-risk potential attack vector may be acceptable. Designers might apply additional remediation to processes and procedures, like checking the system regularly after deployment to ensure that it's still in good shape.

Real-world considerations

In an ideal world, all the groups involved in developing the system would work together; unfortunately, that's not always possible. Different groups are available at different times; contracts may start and stop; engineers may not address security considerations before defining the physical architecture, forcing them to layer security on top of existing architecture.

Government customers may require systems designers to accept all identified risks. There may be clashes between requirements for cyber security, anti-tamper, and system certification. For these reasons, it's crucial to get system security engineers involved as early as possible before the architecture gets finalized. Just as early, the security team should engage their suppliers so its members can understand the solutions that are available, and the steps they can take to meet their security and performance goals.

August 22, 2018 | By Steve Edwards and David Sheets

THE TRUSTED COMPUTING IMPLICATIONS OF INTERFACES, AND HOW THEY CAN INFLUENCE SYSTEM PERFORMANCE

Steve Edwards and David Sheets explore the implications of how interfaces influence system design in trusted computing.

A trusted computing system can ensure security at the potentially vulnerable entry points of vulnerable system interfaces, yet this may compromise performance through design trade-offs that systems designers must recognize, understand, mitigate, and compensate for.

Systems development often involves several different engineering groups, and the team dealing with security isn't necessarily the one responsible for project performance.

The group concerned with security will identify which parts of the system need protecting to meet the program's security plan. An entirely different team can dictate performance issues involving processor speed, compute power, memory, and I/O bandwidth -- all of which affect system hardware. It's not unusual for these disparate teams to be out of sync with each other. This affects trusted computing systems because decisions made about system security inevitably affect system performance.

Design teams must have conversations internally at the highest level to understand trade-offs that implementing security can create. Authentication and encryption can influence processor use and available data bandwidth, which can force designers into augmenting the system's processing power to make up for security's affects on overall performance. It also could force system designers to consider relaxing security requirements to maintain performance.

Designers must consider how to define security at the I/O boundary at the board and subsystem level. At the subsystem level are interfaces like Ethernet, MIL-STD-1553 and others that communicate outside of the box. As these interfaces communicate and receive information from other equipment, designers need authentication to ensure that communications happen only among authorized entities, and that only trusted data flows in both directions.

When design teams discuss security and performance tradeoffs, they must make choices about how to implement authentication. Will it occur only once, at power-up, or every very time a message is sent? What kind of authentication should be performed? Should some sort of key exchange be used to pass keys back and forth? Such decisions have associated overhead costs. Depending on the system architecture, those overhead costs may increase startup timeline, reduce overall system throughput, or introduce additional system latency.

Authentication is just one issue involved with security and performance tradeoffs. Calculating the implications of complex processes like cryptography, and acknowledgements as data passes through the system also can be difficult. Designers may opt to build and test a system mockup to gauge the overhead that security will impose, yet the most important thing is to learn the different costs of implementing authentication.

Encryption is a two-way street when it comes to processing costs. Encrypted data must be decrypted, adding additional processor overhead. Key exchanges can introduce overhead when creating ephemeral session keys. throughput hiccups caused from key renegotiation can impose additional overhead.

Another often-overlooked consideration is the availability or lack of existing industry standards that define how to ensure security over data interfaces. System integrators need to understand which interfaces their system will use, and determine if standard security protocols exists for them. Ethernet, for example, has standard security protocols like Internet Protocol Security (IPsec) and Transport Layer Security (TLS). Designers might implement such standard protocols at different levels of the IP stack; their impact on system performance will differ depending on where in the IP stack the security

standard is located. What's being authenticated, and what is being encrypted also influences system performance.

Using standard interface security protocols has clear benefits. For one, the designer can implement and interoperate them on hardware modules from several vendors. When no pre-packaged security protocol exists for a particular interface, the designer, the program, or the standards body must define a secure approach for using that interface.

Sometimes the system designer must use an older interface that wasn't designed with security in mind. Doing this brings related concerns, like deciding whether to layer additional security code on top of the interface, or designing a unique solution.

The target platform itself may drive many of these interface and performance issues. If an Ethernet network is available, the designer can use its built-in security. If a sensor must communicate over MIL-STD-1553, there won't be as rich an ecosystem to support security. MIL-STD-1553 has been around for many decades, and is common in military systems. A lot of deployed equipment can't support modern trusted computing authentication techniques.

If a legacy sensor cannot perform authentication over MIL-STD-1553, the designer must decide how, or if, to implement authentication; he has to determine risks and vulnerabilities, like how critical it will be not to authenticate that link.

The designer should identify and understand not only all interfaces in a system, but also any associated potential security concerns. This includes the module interfaces that already are enabled, as well as those interfaces that possibly could be enabled. Common interfaces that can have serious security implications, so design teams should not overlook them during security reviews. Designers also should consider debugging interfaces and maintenance interfaces.

July 25 , 2018 | By Steve Edwards and David Sheets

DEVELOPING A SECURE COTS-BASED TRUSTED COMPUTING SYSTEM: AN INTRODUCTION

In this article, Steve Edwards and David Sheets look at how system security should start in system design as security touches every element of the system.

Security and trusted computing, at the end of the day, really are all about the system. While the pieces and parts, such as the individual modules, operating system, and boot software, all are important, system security is not an additive process; it can't simply be bolted-on to make the system secure.

The designer must look at each system element and understand how they work together. If a single discrete element of the system is insecure, it is not possible for the designer to claim confidently that the entire system is secure. You need to know how each system element integrates with the rest, what interfaces are available to those elements, and how each element communicates with the other parts of the system.

The systems designer must make all efforts possible to eliminate any risk of inadvertently providing an insecure port of entry into the system that makes it vulnerable to malicious attack.

Defining security requirements

Frequently, designers can meet system-level security requirements in several different ways. For example, to resist a particular malicious attack targeting one weak hardware component, the designer could use either a more resistant component or implement a distributed system approach, presenting the attacker with the far more difficult task of targeting multiple components at the same time.

The systems designer must understand how he can implement different system-level architectures to address the same security requirements. Designers also must understand the tradeoffs of each architectural option. Different program managers will want to make different choices based on their unique priorities and key performance parameters.

Designers can achieve some security protections either at the system or the module level. While that decision already may be established, sometimes cost, schedule, or the limitations of a particular component may drive the designer's decision of what level to implement a protection.

Interfaces and data communications

Another important topic for system level security is how to set up interfaces, within and between systems. The designer must consider the best way to design communication pathways to ensure that the system handles data in transit appropriately.

The designer must make decisions about intra-system communications between the individual modules and how to protect that data. Where interfaces send data also is important, because intra-system communications often have very different security considerations from inter-system communication.

What's more, the ways in which the system uses those interfaces, whether during operation or in maintenance activities, may raise additional considerations. The designer must perform an appropriate level of validation or authentication of data prior to its acceptance for processing at each different levels of integration. That means the designer must make choices about what sort of access control or authentication capability is necessary at the I/O boundary.

Working with COTS vendors

Security requirements flow down either from the customer or the program office to the systems designer to provide specifics for anti-tamper or cyber security protections. The designer must decompose those requirements to define what each one means for the system, and for each of the components within it.

Sometimes designers can meet for anti-tamper and cyber security with one solution. In other cases the designer must resolve the competing requirements of different domains. Security requirements next flow down to the COTS vendors to implement any necessary mitigating technologies.

In an ideal world, one trusted and capable vendor would handle all components to optimize security. Yet security challenges often stem from using modules from different suppliers. Frequently, the system integrator must deal with multiple COTS vendors, but

not every COTS vendor can support the same levels of trusted computing capabilities, so the designer also must weigh supply chain management issues and their COTS vendor's ability to meet the system's needs.

Performance and testing

The designer also must address and understand the performance aspects of security implementations, because secure networking, encryption, and authentication can affect nominal throughput for booting, processing data, networking data, and system latency; the designer must consider all of a system's key performance parameters.

The impact of trusted computing on performance will vary from system to system, and may require tradeoffs between security and performance. Size, weight, and power (SWAP) limitations also can play a part in weighing tradeoffs. For new "bleeding edge" system designs the tradeoff might be between implementing security (and/or other overhead) and meeting the mission requirements.

Testing for system-level security brings its own set of challenges. For some security implementations, the designer might need to classify pieces of a system, and test them in a classified laboratory. Afterward, when security is enabled, the system can be tested in an unclassified environment where the sensitive component can be integrated with additional elements of the system. The challenge is how to undertake the integration of classified and unclassified components in a way that's cost-effective and ensures the ability to perform debugging in each set of environments effectively.

System security in the field

Maintaining and upgrading fielded systems is another area with potential implications for security. The designer has to make decisions about whether to allow software updates in the field, and if so, how to validate the software.

If one system module fails in the field, there must be a process to authenticate its replacement properly. When the system comes in to a maintenance depot for repair, it's important to understand what parts of the system its maintenance ports can access.

Prior to a system's deployment, the designer must decide how to manage security certificates and keys. The key management plan must be fully vetted, and include the ability to revoke keys to reduce the program's long-term maintenance costs. These decisions should be made early in the program, since the choices will affect how they are designed into the system.

When to think about system security

It's always easier to make decisions about trusted computing protections at the very beginning of system development; there is great value in discussing anti-tamper and cyber security at the earliest stages of system design. Security touches every element of the system. To add in needed "hooks" after a module or system is already designed often requires undoing a lot of previously undertaken and costly design work.

While requirements for security might not exist at the beginning of a program, the systems designer often can anticipate them for the future. It's important for the systems integrator and his suppliers to discuss long-term expectations for security early on in the design stage.

Implementing trusted computing protections later in the design cycle can be expensive and wasteful, since all other system development must wait for the security approach to be chosen, and all the related implications that those decisions have on the rest of the system fully understood.

June 27, 2018 | By David Sheets

TRUSTED COMPUTING: APPLICATION DEVELOPMENT, TESTING, AND ANALYSIS FOR OPTIMAL SECURITY

David Sheets looks at Trusted Computing in application development, testing and analysis for optimal security.

Application software in military systems is what actually gets work done, and enables warfighters to carry out their missions, so it's essential that this code is trusted and secure.

All other hardware and software are designed only to start the application software securely; once it starts, application software relies on the system's fundamental security building blocks, but requires special attention to ensure it functions as intended.

This can be easier said than done, however, because only a miniscule number of system developers typically has a chance to look at the application code. Most application software is custom-built to execute a specific mission or run a particular algorithm, so far fewer software engineers will see it than would see open-source or even most commercial software.

This can result in undiscovered vulnerabilities, which can be made worse because opportunities to review and update application code typically are few and far-between. Application code in military systems is geared to a particular specification; once tested, the system often is deployed and has much less opportunity for re-testing than a general-purpose system would.

Complicating matters is the narrow technology refresh window of deployed systems. Limitations on time, budgets, and mission requirements can make it nearly impossible to update application software once it's in the field. Even if users discover code issues or security vulnerabilities, the costs to bring a deployed system back for update is excessive.

On the other hand, it takes far less time and cost to find, fix, and test software problems prior to deployment; it's imperative for system developers to make the right decisions about application code from the very beginning.

Application software can fall into several categories: libraries, middleware, and custom-built code to perform specific functions. Custom-built middleware requires special scrutiny because it often is widely reused. Middleware provides the glue that holds libraries and applications together, and any middleware vulnerability could make the entire system susceptible to malicious cyber attack.

It's preferable for software engineers to organize systems so that system functions and inputs naturally fall into related buckets. It's important to allocate applications correctly to the user and kernel space. Designers should keep the amount of code running in kernel space to a minimum to reduce the impact of security vulnerabilities, since kernel space executes at a high privilege level, with broad access to system resources. It behooves the developer to consolidate related functions closely to each other. That way they don't have to reach out to other parts of the application.

Moving large amounts of data also can create security concerns. Large data sets tend to move in the clear because it's inefficient to encrypt a large amount of data moving at high speeds. Some systems, moreover, must share large pieces of data, especially when separate executable portions of an algorithm are located in multiple places; it's hard to verify and define this data flow in a secure manner.

A better solution is to divide the software such that application using the algorithm treats it like a black box, with no knowledge of its internal workings. Applications can instruct these black boxes to run specific portions of the algorithm, and receive a concise response in return. This provides a much better-defined information flow between applications. This data flow is easy to verify because it validates the information the system sends and receives it. This approach also helps encapsulate information better to enable encryption and authentication.

Systems designers should define the information flow logically in terms of security boundaries. It's best to minimize the number and complexity of logical interconnections because each connection is a

possible point of infiltration. Designers must examine, lock down, test, and verify each connection, as well as consider a secure messaging mechanism that uses authentication to make sure the data comes from a trusted source.

Secure coding practices are an important part of designing-in application security from the beginning. Using standard secure coding practices can minimize security vulnerabilities from programmer errors. The best-known example of such an industry standard is CERT. Others include MISRA, DO-178C, IEC61508, and ISO 26262.

Using secure coding practices can eliminate the likelihood of unintentional errors. A smart approach is to train all application programmers to know and follow the rules and submit to peer reviews. Automation tools also are available to scan code and verify adherence to the rules. Being consistent with coding rules also can enable programmers to move between projects without introducing unnecessary security issues.

Funding constraints can lead to program cutbacks, and when this happens, program managers might be tempted to eliminate security testing. Doing so, however, can introduce critical gaps and vulnerabilities. It's important to budget for testing at the program's front end.

One approach for analyzing how application components work together is static code analysis, which uses tools to check the code for any potential issues. Software engineers often can use the same plug-in tools they use to verify secure coding standards compliance to perform static code analysis. Examples of these tools include Coverity, cppcheck, Klocwork, lint, Parasoft, and Understand. Some of these tools work on several languages, some are language-specific, some are commercial, and some are available as free open-source software.

Using dynamic code analysis tools can help software developers analyze how the application will run under test conditions. Dynamic code analysis hooks into the running software to analyze what the system is doing, and works in the background to enable systems developers to validate their applications with normal inputs and outputs to make sure that everything works properly.

Static and dynamic code analysis each look at different things, and produce different results. Static code analysis typically will report more false positive errors because it flags every possible error no matter how unlikely. Dynamic code analysis, on the other hand, can verify the same error never does occur, under test conditions. Tools that provide dynamic code analysis include BoundsChecker, dmalloc, Parasoft Insure++, Purify, and Valgrind. Dynamic code analysis also can provide additional benefits such as thread coherence validation and code coverage analysis.

Another important decision for application code developers is whether to use regression testing or continuous testing. Regression testing might not uncover failures until the application goes through acceptance testing or somewhere else further down the line.

The continuous method runs tests at on a continuous basis to help find problems as early as possible. It should include a security-specific testing apparatus to identify software changes quickly that break some of the application's security constraints. While continuous testing may add costs at the front-end, it can reduce costs at the back-end.

Some processors, such as the Intel SGX, have built-in security features. Intel SGX allows software developers to create enclaves in the application to partition data securely. If the programmer needs a key to encrypt some data, he can place the variable into a separate enclave. Using enclaves to separate the encryption key from other system functions prevents a cyber attacker from gaining access to system security.

Another processor-specific security feature is Arm TrustZone, which enables an Arm processor to separate memory, operating system, and application into trusted and non-trusted areas. TrustZone ensures that only the trusted area can access trusted data.

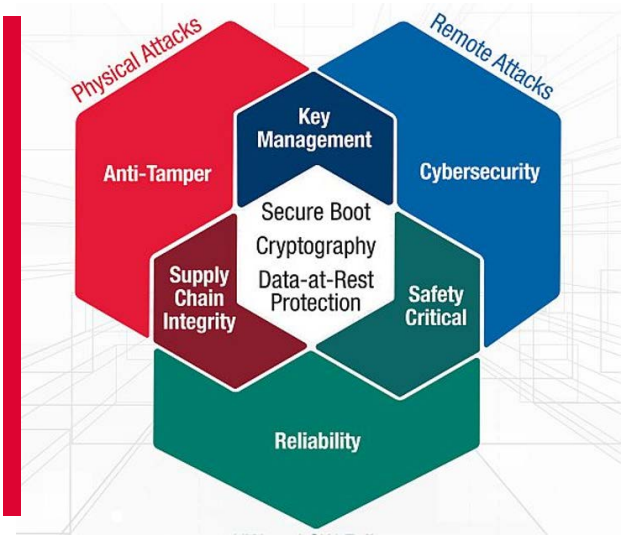
One of the major tenets of cryptography is don't try to build it on your own; it's complex and easy to get wrong. It's much safer and wiser to use something that already exists, is readily available, and is kept up to date continually from the feedback of many users. Take advantage of the security libraries already available within your operating system to implement common security concepts, such as authentication, secure communication, encryption, and key derivation. Examples of robust security libraries include OpenSSL and IPsec.

Mandatory Access Controls (MACs), which most software operating systems support, enable developers to create configuration files that define how different people can use the application and its resources. Operating systems that support MACs include SE Linux and Windows Integrity Levels.

May 30, 2018 | By David Sheets

COMPUTER HARDWARE'S ROLE IN SECURING OPERATING SYSTEMS AND HYPERVISORS IN TRUSTED COMPUTING APPLICATIONS

David Sheets looks at the role of computer hardware in securing operating systems and hypervisors in trusted computing applications.



Many software applications run on "least privilege," meaning that the software only receives minimal access to the hardware, other applications, and other system resources. A security context separation between an application and other resources like operating systems and hypervisors ensures that less-secure applications and software can't access critical data from more-secure and critical trusted computing applications.

Highly sensitive data must be protected to ensure that only the code that needs to operate on that data has access to it.

The responsibility of maintaining this type of secure application separation belongs to the operating system and the hypervisor, if one exists on the system. Think of an application that sits on top of a software stack; each lower layer of that stack must do its part to maintain the application layer's security.

At the bottom of the stack is the hardware, which must be able to enforce the access controls. On top of the hardware is an operating system or a Type-1 hypervisor that manages access controls. Type-2 hypervisors, instead of running on top of the hardware, run on top of an operating system.

For optimal protection, systems designers should configure the system to ensure that the operating system or Type-1 hypervisor exploits all available hardware security to manage scheduling, resources, processes, and security from the next layer. It is very difficult to build a secure application if the foundation is missing those essential security building blocks.

First let's consider the operating system, and how it plays an integral role in ensuring system security on any moderately powerful processor.

For many years now, operating systems have maintained separation between kernel processes and user space applications. In fact, the entire function of an operating system is to ensure the consistent operation of several applications running on one piece of hardware. As part of their responsibility, operating systems have evolved to prevent or limit a malicious actor in one application to influence other concurrently running applications.

> ***As processors have become faster and more efficient, they also have become more complicated, which in turn has made the responsibilities of operating systems even more complex.***

It's complicated, for example, for an operating system to handle cache management between tasks as processes move in and out of memory; the operating system must flush or invalidate any cached memory, which can be difficult and error-prone.

Add to that challenge factors such as direct memory access (DMA), side effects, and security issues like the row hammer, meltdown, and spectre attacks, and the security responsibilities of the operating system become immense.

Hardware security capabilities

Most processing hardware today includes security capabilities for operating system or hypervisor. That's because many of these hardware capabilities perform operations in supervisor mode. In processor-based trusted boot resources like Intel SGX or ARM TrustZone, the operating system must create and manage process and resource access to security domains.

Software engineers must design the operating systems and hypervisor to use the hardware's built-in security features. The operating system and hypervisor typically run in privileged mode; they are the only entities able to use all the features of the processing architecture.

Using an old operating system on new hardware can negate the hardware's security capabilities, and updating operating systems in military and aerospace systems can be costly. Systems designers must consider he potential tradeoffs between risk and program costs to determine when to insert new versions of operating systems during system refresh.

The operating system also must manage resource access securely -- in addition to maintaining security boundaries between processes and using hardware security to its full potential; it's not enough just to maintain process separation. It compromises trusted computing if a process can read another's data from a peripheral device as the data flows in and out.

Software engineers must design operating system software drivers that access peripherals with security in mind. They must understand potential tradeoffs between increasing the access and availability of I/O resources, as well as maintaining and controlling access separation.

Some processors provide enhanced capabilities for managing I/O, such as an input–output memory management unit (IOMMU), which enables the operating system and software drivers to enhance security while making the most of I/O resources.

Security aspects of hypervisors

A hypervisor manages virtualizing resources to enable several operating systems to operate on the same hardware at the same time. The operating systems and the hypervisor must work together when running in a virtualized environment to maintain a secure and trusted environment.

A set of guest operating systems work in the system stack above the hardware and the Type-1 hypervisor. Each of these guest operating systems works similarly to a single operating system when a hypervisor is not present. The Type-1 hypervisor virtualizes all hardware resources and manages access to all the operating systems running above it in the stack. VmWare ESX and Xen are examples of Type 1 hypervisors.

In contrast, Type-2 hypervisors run on top of another operating system, using that parent operating system to access the hardware resources. The Type-2 hypervisor virtualizes those resources to the guest operating systems that reside above. VmWare Workstation and Oracle VirtualBox are examples of Type 2 hypervisors.

Linux KVM is a hybrid hypervisor, and executes in Linux kernel mode directly on the hardware, yet uses the Linux operating system architecture to manage virtualized resources.

The type of hypervisor used dictates where the base security responsibilities for the guest operating systems reside. The Type-1 hypervisor must use all security features available in the hardware to prevent a compromise in which one guest operating system leaks information or access across another guest operating system.

It's imperative for software engineers to write host or parent operating systems to use available hardware security capabilities where Type-2 hypervisors are concerned; the Type-2 hypervisor uses the host operating system to manage and control access to the virtualized resources.

The operating system -- or set of guest operating systems -- separates user space processes from supervisory processes. Each process that helps operate the system has a defined interface that helps it communicate and

interoperate with other processes. The operating system also ensures that processes operate within their defined roles and use interfaces to communicate. For example, it catches and prevents invalid operations and interface access, which prevents one failed process from bringing the entire system to a halt.

Operating system security

Systems designers should give the most stringent security requirements to the operating system or hypervisor when considering the layers of software running on the stack.

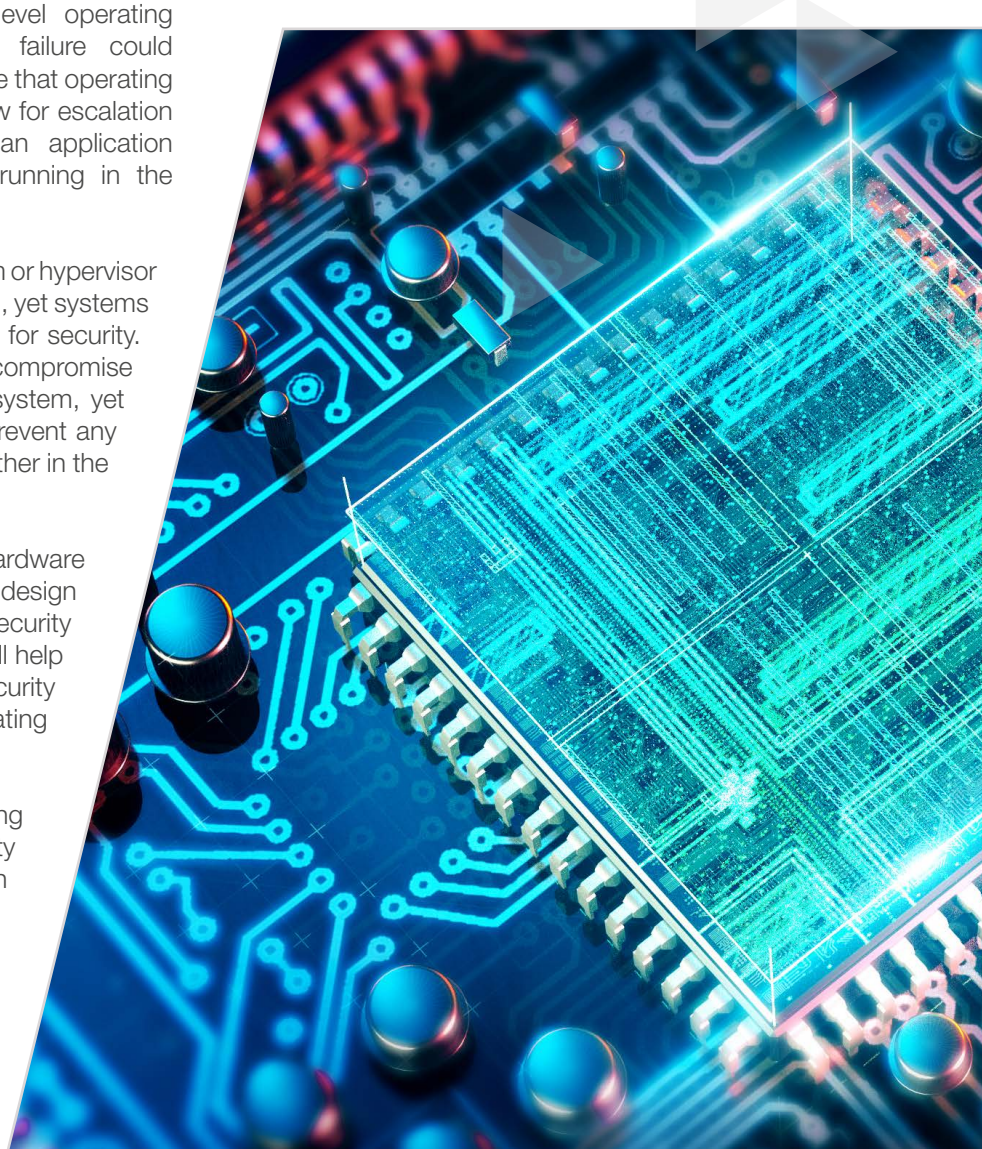
The most stringent security must reside at the lowest layer because of the risk of a security failure at that level. Designers should analyze the operating system and hypervisor to prevent bugs that could allow for inappropriate access. If the lowest-level operating system or hypervisor has a bug, failure could compromise all trusted systems that use that operating system or hypervisor. It also could allow for escalation of privilege, which could leverage an application to compromise all other processes running in the operating system or hypervisor.

The risk of failure at the operating system or hypervisor level normally is much more constrained, yet systems designers still must review applications for security. A failure at the application level can compromise one application or a guest operating system, yet the next level below normally should prevent any such compromise from propagating further in the system.

It's of great value to talk with the hardware vendor as early as possible in the design cycle to best understand system security requirements. A clear understanding will help designers give adequate weight to security capabilities when evaluating operating systems or hypervisors.

Selecting the most recent operating system or hypervisor with built-in security usually will be the best choice. Even better is keeping potential exploits and security vulnerabilities to a minimum when paring down the chosen operating system or hypervisor's feature set.

Embedded systems often offer the opportunity to configure the operating system to remove unnecessary features, processes, and libraries. Tailoring the operating system to minimize any potential exploits also is a good security practice.



April 25, 2018 | David Sheets

TRUSTED COMPUTING HARDWARE FEATURES FOR MAINTAINING CYBER SECURITY DURING OPERATION

David Sheets looks at the hardware features built-in to the most popular defense and aerospace processor architectures to ensure the continued cyber security of a trusted computing system.

It's important to understand hardware features that are built-in to the most popular defense and aerospace processor architectures to ensure the continued cyber security of a trusted computing system during operation.

Understanding these features -- what they protect against, and how to use them effectively -- will enable embedded computing systems to operate securely even in the face of cyber-attacks. Sometimes some software must be modified, as well, to take advantage of these hardware features.

First, it's important to review one's own platform architecture to determine which features are available and relevant to the system's requirements. Generally, the system integrator should use all security features that are available. Variables, such as cost, complexity, security requirements, and threat assessment, can influence the features that the designer will implement.

Each program should review requirements and make tradeoff decisions on security, cost, schedule, and complexity. Discussions with suppliers of commercial off-the-shelf (COTS) embedded computing components and subsystems at the earliest stages of system development can be of great help in making the right choices.

The QorIQ Trust Architecture from NXP Semiconductors, which is part of the Power Architecture and Arm-based devices, provides many hardware security features for secure processing during runtime. Because features of the Trust Architecture vary depending on the processor, it's important to look at the platform to determine which features are available.

To protect sensitive memory from malicious data reading or writing, the QorIQ Trust Architecture implements I/O memory management units (MMUs) to prevent memory access without permission. These I/O MMUs prevent I/O devices from accessing protected memory regions my making sure that input cannot overwrite restricted memory regions. It also helps ensure the integrity of applications and data by protecting against unauthorized modification.

Security monitor

The QorIQ Trust Architecture provides a security monitor (SM) to sense and control the security state of the QorIQ. The SM monitors and responds to potential physical changes to the underlying security features in the hardware. After the QorIQ passes secure boot, SM enters a trusted/secure state, where the SM monitors the QorIQ's security.

If it detects a security violation, the SM can respond with actions ranging from master key lock-out to full system-on-chip (SoC) reset. Monitoring includes checking hardware fuses to verify that error correction is still in line, and that no bits have flipped unexpectedly. Hardware self-checks can detect failures.

The QorIQ Trust Architecture Run Time Integrity Checker (RTIC) can monitor for unexpected changes in system memory by periodically verifying that contents have not changed from what is expected. This sort of protection ensures that even if a cyber-attack modifies application contents, the system can respond to the attack. RTIC security is one example of a layered approach to system security, because it provides a second layer of defense able to notice and respond to attacks.

Secure debug

System development always involves a tension between securing the system and opening it up for debugging. Secure debug enables the hardware to manage the ability to debug by progressively shutting down the ability to debug through the system development cycle. Using QorIQ's secure debug controller enables a range of accessibility, from a completely open system, to a system that requires authentication to debug, to a system that completely disables the ability to debug. This hardware capability ensures that system security can respond to the changing requirements during system development life cycle.

The secure debug controller in the QorIQ architecture controls fuse settings, which helps systems designers control the level of access available to external

debuggers. Secure debug supports four levels of access: open, conditionally closed without notification, conditionally closed with notification, and closed.

When set up, the system is completely open, yet the user can restrict access by burning fuses via software instructions. Then the debug feature is available only after authentication. To lock down access to the debug feature completely requires the burning of additional fuses.

Intel security features

Intel builds security technology into their products, including their embedded IoT lines, with capabilities to ensure secure operation. Available for Skylake and newer processors, Intel's Software Guard Extensions (SGX) act as a firewall to protect portions of memory from unauthorized access by using a security fence. This fence allows software code to interact -- but only via defined instruction sets that the processor provides.

The SGX instructions help systems designers define protected memory regions via software. Hardware enforces access to software-defined regions to provide an additional layer of defense. If an attacker injects code to gain unauthorized access, SGX blocks this malicious code from accessing another enclave's data and instructions. SGX requires more than just modifications to the OS; it also requires modifications to application code.

A system typically runs either in supervisor mode or application code. When the OS finished its kernel operations and is ready to switch to running the application code again, it needs to un-set a bit in a jump register that enables it to move from privileged mode to un-privileged mode. Intel's OS Guard provides a hardware mechanism that prevents a user from executing non-privileged application code when the system is executing in privileged mode. This helps the hardware protect a secure system from attacks that try to escalate privilege.

Arm and TrustZone

Arm IP-based processors not only support the secure operation hardware features of the NXP QorIQ Trust Architecture, but also support TrustZone to add another layer of defense on top of user mode and privilege mode. TrustZone helps designers set a bit to place user or privilege modes into trusted mode. This limits access for such operations as accessing memory regions, or to preventing code from being modified. TrustZone positions security at the heart of the hardware and enables fine-grain management of partitioning hardware resources. It also ensures that non-trusted code cannot gain unauthorized access to resources used for trusted operations.

Power Architecture, Arm, and Intel processors also support virtualization, which enables several different operating systems to run together on one piece of hardware. Virtualization enforces security at a level above the OS, and prevents unauthorized access from leaking to other operating systems. However, this level of security can introduce a performance penalty because of the overhead of virtualizing system resources.

Virtualization happens in two ways. First Type-1 hypervisors operate directly on the hardware. Type-2 hypervisors, on the other hand, operate on top of an OS. Type-2 hypervisors virtualize the host OS's resources so several guest operating systems can use them. Virtualization works similarly to TrustZone in that it provides an additional level of execution privilege distinct from supervisor mode. This enables the hypervisor to manage resource requests from several guest operating systems.

Military-level security

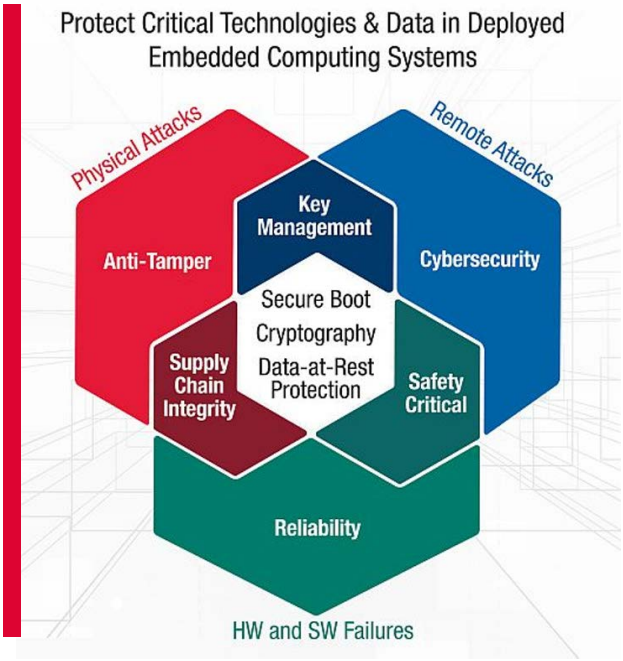
Many military and aerospace systems, however, need even more security. This is where a COTS vendor with security experience can come in. A COTS vendor can design-in and optimize security at the component and module level.

It's important to think of system security as more than just a collection of hardware features. Security results from understanding the system's fundamental building blocks, how the system operates, and the potential attack vectors.

March 28, 2018 | By Steve Edwards

TRUSTED BOOT: A KEY STRATEGY FOR ENSURING THE TRUSTWORTHINESS OF AN EMBEDDED COMPUTING SYSTEM

Steve Edwards looks at how Trusted Boot is a key strategy for ensuring the trustworthiness of an embedded computing system against cyber attacks.



Trusted boot -- a key strategy for ensuring that the trustworthiness of an embedded computing system -- begins with the very first software instruction at system startup to protect against cyber attacks.

Trust in an embedded module or system means ensuring that the system operates exactly as intended. In the context of the boot process, trust means that an embedded module executes only the boot code, operating system, and application code. The only way to guarantee trust in this chain is to ensure that all code -- from the very first instruction that a processor executes -- is authentic and specifically intended by the system integrator to execute on that processor.

Establishing initial trust in the boot process involves various means to do that, although many of these same techniques are also useful for extending trust to the operating system and application code.

Cryptography in the form of encryption and digital signatures is an essential component for establishing trust and preventing a malicious actor from modifying, adding, or replacing authentic code. While encryption ensures confidentiality to prevent prying eyes from understanding the code, it does not guarantee that the code comes from an authorized source and has not been tampered with in some way.

Protections to ensure software authentication typically require the use of public-key cryptography to create a digital signature for the software. This involves the use of a protected private key to generate the signature, and a public key stored on the module. It does an attacker no good to have the public key since he also needs the private key to generate a valid digital signature.

Trusted boot by itself does not make a secure system. There are attacks that can bypass authentication, particularly if the attacker has physical access to the module. There are other ways to gain access to the information an attacker wants. For example, the attacker may simply wait until the module is powered up and then observe the application code while it is running.

These techniques for bypassing authentication do not diminish the usefulness of trusted boot as a component in a defense-in-depth approach in which different security technologies work together to provide a far more secure system than any one technology could deliver on its own.

In the embedded computing world, Intel, Power Architecture, and Arm are the dominant processor architectures, and each one has technologies that support a trusted-boot process. In addition, trusted boot can be designed using external hardware cryptographic engines and key storage devices. The remainder of this article will survey available commercial options for providing trusted boot.

Trusted eXecution Technology (TXT) and Boot Guard are two Intel technologies that create a trusted boot process. Both make use of an external device called a trusted platform module (TPM) -- a dedicated security coprocessor that provides cryptographic services.

TXT makes use of authenticated code modules (ACM) from Intel. An ACM matches to a specific chipset and

authenticates with a public signing key that is hard-coded in the chipset. During boot, the BIOS ACM measures (performs a cryptographic hash) various BIOS components and stores them in the TPM. A separate secure initialization (SINIT) ACM does the same for the operating system. Each time a module boots, TXT measures the boot code and determines if any changes have been made. A user-defined launch control policy (LCP) enables the user to fail the boot process or continue to boot as a non-trusted system.

Boot Guard is a newer Intel technology that works in a complementary fashion to TXT. It introduces an initial boot block (IBB) that executes prior to the BIOS and ensures that the BIOS is trusted before allowing a boot to occur. The OEM signs the IBB and programs the public signing key into one-time programmable fuses that are stored in the processor. On power-up the processor verifies the authenticity of the IBB, and then the IBB verifies the BIOS prior to allowing it to execute. Intel describes Boot Guard as:

> ***"hardware based boot integrity protection that prevents unauthorized software and malware takeover of boot blocks critical to a system's function."***

Many of the Power Architecture and Arm processors from NXP support the Trust Architecture, and the two different NXP processors use variations of the same Trust Architecture. They implement trusted boot differently than Intel. For Power Architecture and Arm processors, the trusted boot process requires the application, including the boot code, to be signed using an RSA private signature key. The digital signature and public key is appended to the image and written to nonvolatile memory. A hash of the public key is programmed into one-time programmable fuses in the processor.

When the processor boots, it begins executing from internal secure boot code (ISBC), stored in non-volatile memory inside the processor. The ISBC authenticates the public key using the stored hash value. The public key then authenticates the signature for the externally stored boot and application code. The external code may contain its own validation data similar to the ISBC. This process can break the external code into many smaller chunks that are validated separately by the previous code chunk. In this way the Trust Architecture can extend to subsequent software modules to establish a chain of trust.

NXP QorIQ processors, meanwhile, enable encryption of portions of the image to prevent attackers from stealing the image from flash memory, and allow for alternate images to add resiliency to the secure boot sequence.

Intel and NXP provide mechanisms to implement trusted boot and ensure the authenticity of the boot process starting from the very first line of code. Still, these important steps do not complete the trusted boot effort.

Designers of secure systems must follow up these with security mechanisms to ensure that trusted boot is maintained throughout the rest of the boot process. Designers also must identify and mitigate potential attack vectors, especially in embedded systems that might be open to issues of supply chain integrity, physical tampering, and remote cyber attack.

February 21, 2018 | By Steve Edwards

COTS-BASED TRUSTED COMPUTING: GETTING STARTED IN NEXT-GENERATION MISSION-CRITICAL ELECTRONICS

Trusted computing involves technologies protect mission-critical embedded electronics from physical and remote attacks and from hardware and software failures.

Trusted computing involves technologies and techniques that protect mission-criticalembedded electronics from physical and remote attacks and from hardware and software failures. Through anti-tamper methodologies and avionics safety certifiability processes, trusted computing also ensures that a system will only execute what is intended and nothing else.

> ***In the aerospace and defense market, solutions based on embedded hardware frequently play in critical applications involving sensitive and classified information; the goal being to operate these systems with complete confidence that they are secure and uncompromised.***

Trusted computing also delivers confidence that any critical data or intellectual property (IP) will not benefit U.S. adversaries if the hardware falls into enemy hands. Anti-tamper defines the set of solutions to protect against physical attacks on the system.

Cyber security defines those protections fielded against remote attacks. System reliability results from activities designed to mitigate hardware and software failures. Approaches for providing trusted computing for anti-tamper, cyber security, and reliability can intersect in various ways.

For example, reliability, includes supply chain integrity, which ensures that supplied parts and software don't introduce vulnerabilities into the system.

Another important element of supply chain integrity is to activate a counterfeit electronics parts control plan in accordance with the AS5553B and AS6174 anti-counterfeit part processes.

The effectiveness of these efforts can have a direct effect on the anti-tamper portion of trusted computing because maliciously altered or counterfeit devices can be weak links that an adversary might exploit to intrude into a system.

When it comes to manned and unmanned aircraft operating in domestic airspace, safety certifiability processes can play a part in trusted computing. These processes can help designers develop commercial off-the-shelf (COTS) avionics hardware demonstrated to be safe for use in domestic airspace.

Safety certifiable processes help speed the design and COTS avionics hardware, as well as the eventual safety certification of the avionics hardware, by providing supporting artifact packages for these products, which otherwise can take years and millions of dollars to develop. Certifiable products can meet the required DO-254 Design Assurance Level (DAL) for hardware, and DO-178C DAL level for software.

Safety certifiable processes can help COTS vendors reduce design risk greatly, and deploy avionics solutions quickly and cost-effectively that are certified safe for operating in domestic controlled airspace.

This series of columns will consider elements that make up an effective trusted computing strategy, including anti-tamper technology protection, cyber security data protection for software and algorithms for data at rest and data in transit, and reliability processes for protecting the COTS embedded computing and electronics supply chain.

This is the first in a series of columns that will address the use of open-standards COTS technologies that address trusted computing in deployable embedded systems for aerospace and defense applications.

To ensure that these approaches are as strong and effective as possible, COTS vendors and systems integrators should begin a conversation about what steps are available to meet anti-tamper, cyber security and reliability requirements as early as possible in the design stage. It's always more difficult to add security by backfilling than it is to build it in from the ground up.

In future columns, we will discuss such topics as data encryption, trusted and secure booting, protection of data-at-rest, protection of data-in-transit, and safety certifiability for airborne avionics systems.

Trusted computing doesn't have an end point; it's an ongoing process that evolves with new threats and new technologies. This column will provide a platform for shining light on known and emerging options for keeping embedded systems secure.

CURTISSWRIGHTDS.COM