Intelligence at the Edge Part 4: Edge Node Security

Ian Beavers and Erik MacLean
Analog Devices, Inc.

IoT system attacks are making headlines and continue to showcase the security vulnerabilities of networks, edge nodes, and gateways. A recent Mirai botnet infected over 2.5 million IoT nodes by logging into devices running telnet servers in which the default password had not been changed.[1] Mirai later was able to invoke a denial of service for servers that disrupted internet access for a large portion of the world. The Reaper Botnet attacked over a million IoT devices by exploiting software vulnerabilities and infecting them. An internet connected fish tank provided the entry point into a casino's network, leading to the theft of 10 GB of data. Smart televisions have been exploited and used for espionage and surveillance.

Embedded sensor systems are just starting to be connected and exposed to the internet. As part of the Industrial Internet of Things (IIoT), these sensors lack the past two decades of evolution that web servers have had in this hostile environment. Hence, the industry is witnessing many of the attacks commonly seen in the 1990s and earlier in these systems. The lifecycle of an IIoT system is often much longer than one in traditional computing. Some devices may continue operating for decades after they are deployed, and with unknown maintenance schedules.

While servers and PCs are complex enough to allow for security provisions, IIoT nodes are usually low in power consumption and processing power. This leaves a small power budget for intentional security measures. Security is largely a tradeoff, as there are development costs involved. Although IIoT may have higher costs than consumer IoT, it will still face challenges in cost for scalability. If security is ignored there are hidden impacts that will arise after products are deployed, and these costs will eventually need to be addressed.

Sensors and actuators allow IIoT devices to interact with the physical world. Cyber attacks have been mostly limited to the loss of data, although an IIoT hack allows potential entry into the physical world easier than it has in the past. Attacks now have the potential to cause physical harm. This is even more significant in IIoT, where a failure could potentially shut down or destroy a multimillion dollar industrial process or lead to a life threatening situation.

A Connected World

IIoT devices are generally connected to some network and often the internet. This connectivity is what exposes them the most to an attack. Similar to the realm of epidemiology, infection is spread by contact with other machines. Attack vectors exist where systems interact with the outside world. Attackers are able to interact with systems strictly due to their connected access. The first system design security question to be asked is: "Does the device really need to be connected to a network?" Connecting it to a network dramatically increases the security risk.

The best way to secure a system is to prevent it from connecting to a network or limiting it to a closed network. Many IIoT devices are connected to networks solely because they can be without much reason. Does the benefit of having the device connected to a network outweigh the security risks associated with it? In addition, any other legacy systems that interact with the internet facing system can also be put at risk.

In many cases, an otherwise secure network and secure nodes must also interoperate with a legacy incumbent network that could be far inferior in its own security. This poses a new problem in that the weakest security risk could be outside the influence of the IIoT system. In that case, the IIoT system also needs to protect itself from within the network.

Security considerations at the node:[2]

- Confidentiality—protection from data disclosure to unauthorized people, such as from a spoof attack
- Authentication—use of digital certificates to validate the identity between two machines
- Secure boot—ROM bootloader storage validates authenticity of second-stage bootloader
- Secure firmware updates—only authorized code from the manufacturer is permitted
- Authorization—only authentic nodes should be able to gain network access
- Integrity—protecting data from being altered
- Accounting—proper accounting of data, node counts, and timestamps can help prevent unwanted access to IIoT networks
- Secure communication—encrypted protocols that can reside on a low power node
- Availability—ensuring users have access when they need it
- Nonrepudiation—assurance that authentic communication requests cannot be denied
- Reliability—even in harsh electrical environments, access needs to be reliable

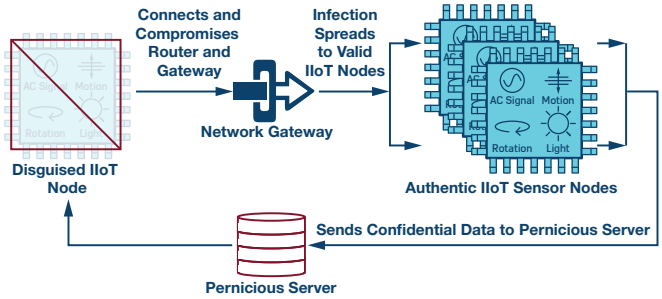

Figure 1. A spoof masquerades as a known node to a gateway.

Isolation

Isolating systems from each other can reduce the attack surface and limit the spread of malware. Isolate systems that do not require network connectivity from systems that are exposed to networks. Consider setting up a separate air-gapped or tightly monitored network that is separated from other networks for high risk systems. Ideally, critical systems should be completely isolated from the outside world.[3]

The infotainment system of a connected car can expose the vehicle to many new attack vectors not previously seen before. The main engine control unit (ECU) has nothing to do with the infotainment system and there should be no way to interact with it through the infotainment system. Though there are typically two separate CAN busses in vehicles separating the most critical systems from the rest, they are still connected together in some way. It is still possible to compromise one and gain control of the other. If there was total isolation between these networks, the risk of compromise would be reduced from potentially life threatening to something far less serious.

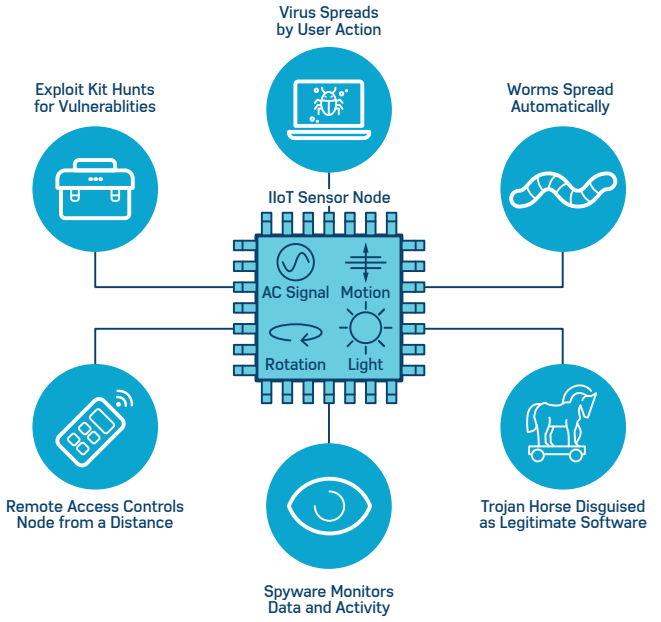

Figure 2. Various types of malware that can potentially infect an IIoT system.

Moving to the Edge

Many IIoT systems connect to a cloud server that collects and processes information sent to it by the device and also manages the device. As the number of devices scales to large numbers, the cloud can have difficulty keeping up with all of them. Many systems are moving processing out to the edge on the IIoT devices to reduce the amount of traffic to the cloud.

We often think of data as an asset. Data is mined and sold to find hidden patterns in large data sets. However, the bulk of collected data is usually not very useful, though it may be useful to an attacker. Sensitive data creates a target for attackers and creates a liability. Collected data should be filtered down to only what is needed, and the rest should be deleted as soon as possible. This not only improves security, but also the utility of the collected data. It is important to identify potentially sensitive information and eliminate or limit its collection.

Processing data at the edge can reduce the amount of data sent and exposed to the cloud. The more locations data is sent, the more difficult it is to keep it confidential. Each new node is another potential compromise where data can be leaked. The attack surface can grow exponentially.

Keeping sensitive data contained at the edge can limit the attack surface specifically on confidential data. If it is confined to one edge node, it is less likely to be stolen. A parking occupancy sensor that detects and only reports the presence of a vehicle through a binary signal after image processing will not stream video. It eliminates the large amount of unnecessary data contained in an image. This reduces the burden on the receiving server so that it cannot be reused maliciously for surveillance.

Similar to consumer IoT systems, industrial IoT systems also have proprietary and confidential information that must be maintained:

- Proprietary algorithms
- Embedded firmware
- Customer information
- Financial information
- Asset location
- Equipment usage patterns
- Competitive intelligence
- Access to a larger network

Through the Fog

Some IIoT devices still lack the power and performance to be edge-based. Another topology emerging, the fog model, is a hybrid between cloud- and edge-based systems. In the fog model, the edge nodes first connect to a gateway that receives data and does some processing before sending it to the cloud. There may be one gateway for many IIoT devices. The gateway does not need to operate on battery power, can afford a much higher budget in processing power, and costs more than constrained IIoT devices.

The fog has risen more from scalability issues, but could also come to play a role in security. The gateway device could help protect vulnerable edge nodes that may be too constrained to provide security on their own, but it may be better to provide some level of protection instead of none. The gateway can be used to help manage all the nodes underneath it instead of managing each individual node directly. The fog model can also allow for incident response in IIoT while avoiding disruption of service. For example, security may respond by interacting with the gateway instead of shutting down a mission critical manufacturing line.

Provisioning and Deployment

Among the greatest challenges in IIoT is the deployment and management of large numbers of devices. Wide reaching IIoT systems are notoriously difficult to set up and configure. With the long lifecycle of IIoT, systems may be deployed by one team and still be operational years later when yet a different team supports it.

IIoT systems are often insecure with weak authentication mechanisms by default. As seen with the Mirai botnet, most users never log into IIoT devices to configure them. They may even be unaware that they are supposed to be configured. Most IIoT users assume things just work out of the box. Systems must be made secure by default. A system expectation should be set that the user may never configure the device other than the default. Weak default passwords are a common mistake.

Network Security

While the edge receives most of the focus in IIoT, it is important to not neglect the cloud or the server side of a system. Test for common server side vulnerabilities like cross-site scripting, SQL injection, and cross-site request forgeries, and review APIs for vulnerabilities ensure that software running on the server is patched promptly.

Data in transit across the network needs to be secured, or it could be intercepted and modified maliciously. Secure cryptographic protocols such as TLS or SSH are used to protect data in transit. Data should ideally be protected end-to-end.

The perimeter boundary of an IIoT network can often be blurry. IIoT sensor nodes often spatially reside on the periphery of their network. However, they also provide an easy portal into a larger industrial network through a fixed gateway.[4] Proper authentication of these devices to the network can help prevent traffic from being tampered with by a malicious third party.

Securing network data traffic involves the use of a secure communications protocol. The best practices should be to use standard protocols that are known to be secure. Security on an Ethernet LAN can be provided using IEEE 802.1AE MACsec. Wireless LANs tend to be a higher risk since they are more accessible and ubiquitous. WPA2 provides security for IEEE 802.11 wireless networks. The low power IEEE 802.15.4 standard, often used within wireless IIoT solutions, offers its own suite of security protocols. However, these are Layer 2 protocols and only secure traffic on the LAN.

Securing traffic that needs to be routed outside the LAN, such as over the Internet, requires higher layer protocols that provide end-to-end security. TLS is commonly used to secure Internet traffic and provides end-to-end security. While TLS uses TCP and many IoT devices communicate using UDP, there is DTLS (datagram transport layer security), which works over UDP. While IoT devices are constrained in power and memory, it is possible to implement TLS for most constrained applications with minimal effort. For even more tightly constrained devices, there is currently a new protocol, constrained application protocol (CoAP) in development by the IETF.

Endpoint Security

While securing data in transit is important and necessary, attacks are more often targeted at the endpoints. Network facing interfaces need to be hardened against vulnerabilities. One approach to IIoT security is to build protection directly into the sensor node device. This provides a first critical security layer, as the devices are no longer dependent on the corporate firewall for their sole protection. This can be especially critical for mobile corporate devices and IIoT sensors that are deployed in remote locations.

A security solution for IIoT devices must provide protection against a wide range of cyber attacks. It must ensure that the device firmware has not been tampered with, be able to secure the data stored within the device, be able to secure inbound and outbound communications, and it must be able to detect and report any attempted cyber attacks.[5] This can only be achieved by including security in the early stages of design.

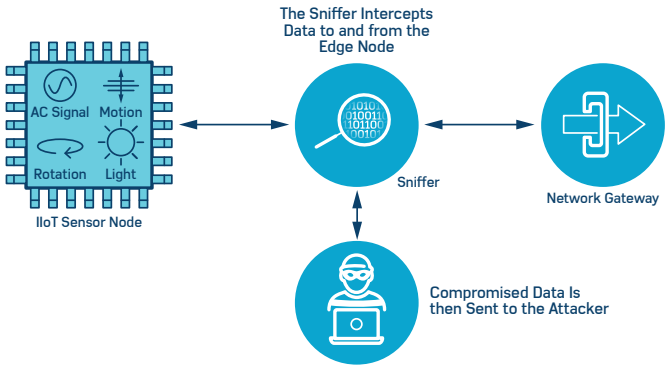

Figure 3. A Man-in-the-middle attack inserts a malicious access point between a node and a gateway.

There can never be a one-size-fits-all security solution for embedded devices. Solutions are available that provide a general framework for OEMs. However, a complete security framework must consider the core capabilities required to protect specific devices, networks, and entire systems. There must also be flexibility to customize the solution to any specific requirements, while also ensuring that critical security capabilities are included.

Autoclaves for Automata

In medicine, sterilization of surgical tools is essential to allow their reuse while preventing the spread of disease. The autoclave is the gold standard for sterilization. It quickly sterilizes instruments with superheated steam at high pressure. It obliterates all bacteria and returns the instruments to a known good state. This allows a surgeon to use a scalpel for surgery and safely reuse the scalpel after sterilizing it.

The ability to return the system to a known good state after compromise is more important than making it bulletproof to all attacks. A resilient system can quickly recover and resume operation with confidence.

Once a system is infected, how can it be disinfected? When a system is infected, it alters the state of the system in some unknown way. Remote exploits take control of the processor and inject new malicious code into the system. Typically, the firmware is modified or replaced in some way with malware so the system now behaves in a different way. Once this occurs, the processor can no longer be trusted.

Embedded systems are often designed in a way that make it too difficult to reliably recover from a compromise. Often, the only way to sanitize a system and verify that a system is clean is to physically dump all nonvolatile memory directly to an external reader. Then it can be verified against the original firmware and replaced with the original if it is not intact. Most systems are not designed in a way to make this possible.

One method to protect the integrity of a system is to physically write-protect nonvolatile memory with a mechanical switch. When the switch is set to write-protect, the memory is physically protected in hardware. Moving the control over memory outside the domain of the processor makes it physically impossible to remotely install permanent malware into this memory without physical access to the device. This reduces the list of potential attackers from anyone in the world with an internet connection to only those that have physical access to the device for an extended period of time. Firmware updates are usually a very rare event. When a firmware update is required, the user can set the switch to write enable the memory to authorize the update and then write-protect the device once the update is complete.

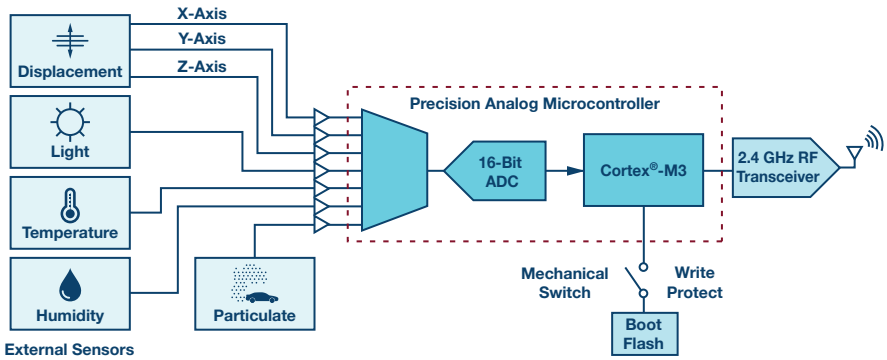

Figure 4. Physically write protecting firmware, except when performing an update, is an effective way to protect the integrity of a device.

Many devices also use their nonvolatile memory to store data needed for write access. In a high security system, a separate nonvolatile memory chip may be used to store data but not software. An attacker may still compromise some systems by writing malicious data to this memory and exploiting software bugs, so the system should be thoroughly analyzed and tested, so no matter what data is stored in this memory, the system will not be compromised. The addition of an extra memory chip increases cost—however, some flash memory allows certain sectors to be write protected, while allowing others to be writable.

Secure Boot

A secure boot prevents unauthorized software from being loaded onto the device during the boot process. It is the beginning of the chain of trust. A secure boot starts with a first-stage bootloader programmed into a read-only, nonvolatile memory location on the node. This bootloader only validates the authenticity of the second-stage bootloader. The second-stage bootloader, which very often is more complex and can be stored in a reprogrammable flash memory, repeats the process.[6] It verifies that the operating system and loaded applications are indeed valid from a trusted source.

An IIoT node with secure boot and secure firmware update capabilities ensures that the device is running authorized code and not altered or malicious code, as this prevents the permanent installation of malware or code. The device will either only run unmodified code or will fail to boot.

The secure boot process usually relies on digital signatures to protect the authenticity of the code. The code images are signed by the device's OEM using the OEM's private key at the time of manufacturing assembly. The OEM's corresponding public key is then used by the node to validate the signature for the firmware image.

The code can also be protected with a message authentication code (MAC) using symmetric cryptography, but this requires the private key to be stored on the device, which puts it at risk. However, it is computationally easier to use a MAC.

While a secure boot can enhance security, it can sometimes be too restrictive to end users since it can prevent them from changing the software running on their devices or running their own software. Depending on the application, users may need more flexibility and the ability to configure secure a boot, which allows it to trust their own code.

Secure firmware updates, similar to a secure boot, validate that new code images have been signed by the OEM during the upgrade process. If the downloaded images are not valid, then they are discarded and the upgrade is halted. Only valid images are acceptable and subsequently saved to the device memory.

Assume that a vulnerability will be discovered sometime. There should be a plan in place for how vulnerabilities will be addressed when they are found or exploited. There usually needs to be a way to allow software updates and patches to be installed on the device to fix vulnerabilities. The update process also needs to be properly implemented so that it is not used as an attack vector that allows anyone to install malware on the device. Making a device accessible through a network, merely to provide patching capability, can introduce more risk than it mitigates.

Secure Communication

Most engineers think of security as communications protocol, such as SSL/TLS, SSH, and IPsec, as secure communications have been added to many embedded devices. However, while this is a portion of the security threat, other attack vectors provide new avenues. Many IIoT sensor nodes operate in a low power configuration with lower power processors that are not capable of supporting some of the best options, such as TLS or IPsec. Security protocols provide a good starting point for building secure devices.[7] They are designed to protect against packet sniffing, man-in-the-middle attacks, replay attacks, and unauthorized attempts to communicate with the node.

Small IIoT edge sensor devices are often adopted with wireless protocols such as Zigbee, Bluetooth® low energy (BLE), and other wireless and mesh networking protocols. These protocols have some amount of built-in security. However, it is relatively weak. Many exploits have already been published and are well known by sophisticated hackers. Small form factor IIoT devices typically run on very low cost, lower power processors that do not support TLS or IPSec. For small edge devices, DTLS, which is TLS over UDP, can be used for secure communication.

Physical Security

Physical attacks target the actual edge hardware nodes or gateways of an IIoT system and can include breaches at the front-end sensor. These attacks often require physical access to the system, but may also simply involve actions that merely limit the efficacy of the IIoT hardware. Attackers can tamper with nodes to gain control over sensors or other devices within an IIoT environment. They can then extract confidential data and embedded firmware code from the source. Using a malicious node injection strategy, attackers can physically deploy malicious nodes between legitimate nodes into an IIoT network.[8]

To help mitigate these attacks, several hardware forethoughts can be implemented during the design phase. Easy physical probing of signals through leaded devices, exposed copper vias, or unused connectors should be minimized or even abandoned from the design. A silk screen that details components and offers potential hackers additional information

should be removed, unless it is deemed absolutely necessary for the design. Although it can increase system complexity, an industrial conformal coating not only buffers the hardware from the elements, but can also add an additional step to prevent direct probing of the electronics on the PCB.

Any embedded nonvolatile memory contents should be encrypted and write protected within the component. The interface between the microcontroller and DSP device should be within buried trace layers on the PCB. Even if the contents of the embedded memory could be retrieved, the encryption and validity of that data should render it meaningless.

Manufacturers often include debug or test ports. These are usually serial or JTAG and can be used to gain access and control most of the system. Ensure that these ports are functionally disabled or protected in production, because it is insufficient to not populate debug headers, as a determined individual can just populate them or solder their own connections to pins. Authentication before these interfaces are allowed to be used is required if they need to remain enabled in production devices. They can be password protected, but be sure to allow the user the ability to set strong passwords.

Random Number Generation

Cryptographic functions usually require some sort of random number generator (RNG). Random numbers may need to be unpredictable for key generation or they may need to never repeat. Generating random numbers in constrained embedded systems usually presents a significant challenge, due to the lack of resources and entropy.

Many embedded systems have suffered from too little entropy. This can lead to catastrophic breaks, such as in Taiwan's national ID smart cards. Researchers found that many ID cards generated related keys from the same numbers due to a lack of entropy. As a result, they were able to be broken, despite using a strong RNG.[9] Similarly, in 2012, researchers found that 0.38% of RSA keys on public key servers shared weak, random number generation and were able to break them.[10]

It is difficult or nearly impossible to validate the strength of an RNG. RNG design in the past has been fairly ad hoc and poorly understood. However, in recent years, significant progress has been made toward the design and formal analysis of robust cryptographic random number generators.

Modern, robust RNG designs now tend to have three stages.[8] There is an entropy source that provides the raw entropy, an entropy extractor to give the entropy a uniform distribution, and an expand stage to expand the small amount of entropy available.

The first stage is the entropy source. This may be some physical noise source, such as clock jitter or thermal noise. Some processors, such at the ADI Blackfin® DSP, provide hardware with random number generators that can be used for entropy generation.

Random numbers for crypto need to have a uniform statistical distribution. All entropy sources have some amount of bias, and this bias needs to be eliminated before using it for a cryptographic application. This is done using an entropy extractor, which takes nonuniformly distributed input with high entropy and generates a uniformly distributed output with high entropy. This comes at the cost of some entropy loss, as the entropy extractor requires more entropy input into it than it can output. As a result, many more bits need to be collected from the entropy source and distilled into a small, high entropy number that can be used to seed a cryptographically secure pseudo-random number generator.[11, 12]

Exploiting Errata

Nearly all IIoT nodes are operated with some form of embedded firmware or algorithms. Functionally, this firmware may operate just fine with no apparent issue in its capability to perform its requirements. However, all software has some level of bug or erratum that permits a small percentage of abnormal operation that may cause security problems. For instance, a 99.99% erratum-free firmware will rarely, if ever, cause any operational problems. But this small, 0.01% erratum may be able to be exploited by an intruder to force the operation of the node to fail 100% of the time for that particular mode of operation. Software bugs arise from complexity, which is necessary for any system to do anything useful. Software bugs and vulnerabilities exist in essentially all systems.

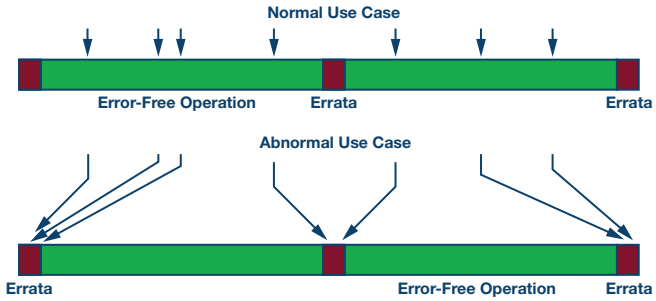

Figure 5. Exploiting small errata to force a failure 100% of the time.

Design for Security

Security must be a consideration of the system design from the beginning. It should be a part of the design process, not something that is bolted on at the end of the project. Security is not about adding security features; it is about managing risk. Secure design methodologies are essential for any IIoT system development.

Existing secure design practices still apply. Use threat modeling to identify risks and to choose appropriate risk mitigation strategies. Identify the entry points to a system in order to identify the highest risk areas in a system. Most attack vectors are through external interfaces, so review the design implementation for security vulnerabilities. Handle unknown data carefully and validate all input—validation and security should not just be limited to the entry points. Defense in depth is important, meaning layers of security are needed in case the outer layer is breached

Many processors provide different levels of privilege. ARM® has Trustzone and the ADI Blackfin DSP provides both user-level execution mode and privileged execution mode. Execute as much code as possible in the lowest level of privilege possible to keep the most important code within a privileged mode. Security requirements for IIoT devices must take into consideration the cost of a security failure, the likelihood of an attack, primary attack vectors, and the cost of implementing a security solution.

Conclusion

Many of these recommendations conflict with each other and with the other design goals of the system. Providing security usually involves some sort of trade-off, often with cost, functionality, or usability. Some trade-offs are very effective and inexpensive, while others have high cost and little impact. Security needs to be balanced against the other needs of the design and should be determined in an application specific basis through a secure design process.

To assist with securing the IIoT, ADI offers several processors that provide hardware-based security enhancements that can help push the boundary of what is possible in edge nodes. The ADF7023 RF, low power transceiver offers internal AES encryption by using an ISM band with many different available modulation schemes.

The embedded transceiver within the ADuCM3029 provides AES and SHA-256 hardware acceleration and a true random number generator, along with multiparity protected SRAM. The ADSP-BF70X Blackfin family of digital signal processors provide embedded, one-time programmable memory for secure key storage and fast secure boot, providing a strong guarantee that the system will return to a known good state after compromise.

Rollback protection in the Blackfin DSP with a hardware-based, increment-only counter allows firmware to be updated to fix vulnerabilities when they arise. This, coupled with the immutability of the key storage, provide the capability to create a robust and resilient edge node. In addition, the Blackfin DSP provides crypto hardware accelerators, a hardware-based true random number generator, separation of privileged and unprivileged code execution, an MMU, and the ability to restrict access for the many DMA channels to allow for parallel and power efficient secure DSP at low cost.

References

[1] Mirai botnet leaked source code.

[2] Ross Yu. "Security and Reliability Are Key in Wireless Networks for Industrial IoT." Analog Devices, Inc., 2017.

[3] Patrick Nelson. "Organizations Must Isolate IoT from Regular IT, Says Telco." *Network World,* March *2016.*

[4] Brian Girardi "Endpoint Security and the Internet of Things." *CSO*, 2017.

[5] Tristan O'Gorman. "A Primer on IoT Security Risks." *Security Intelligence*, February 2017.

[6] Abhijeet Rane. "IoT Security Starts with Secure Boot." *Embedded Computing Design*, January 2017.

[7] Amitrajan Gantait, Ayan Mukherjee, and Joy Parta. "Securing IoT Devices and Gateways." IBM, May 2016.

[8] Boaz Barak and Shai Halevi. "A Model and Architecture for Pseudo-Random Generation with Applications to /dev/random." Proceedings of the 12th ACM conference on Computer and communications security, November 2005.

[9] Chen-Mou Chang, Daniel J. Bernstein, Chang, Li-Ping Chou, Nadia Heninger, Nicko van Sormersen, Tanja Lange, and Yun-An Chang. "Factoring RSA Keys from Certified Smart Cards: Coppersmith in the Wild." *Springer*, 2013.

[10] Arjen K. Lenstra, Christopher Wachter, James P. Hughes, Joppe W. Bos, Maxine Augier, and Thorsten Kliengun. "Ron Was Wrong. Whit Is Right." *Cryptology ePrint Archive*, Report 2012/064, 2012.

[11] Boaz Barak, Eran Tromer, and Ronen Shaltiel. "True Random Number Generators Secure in a Changing Environment." *Springer*, 2003.

[12] Boaz Barak, François-Xavier Standaert, Hugo Krawczyk, Krzysztof Pietrzak, Olivier Pereira, Yevgeniy Dodis, and Yu Yu. "Leftover Hash Lemma, Revisited." 31st Annual Conference on Advances in Cryptology, August 2011.

About the Author

Ian Beavers is a product engineering manager for the Automation Energy and Sensors Team located at Analog Devices, Greensboro, NC. He has worked for the company since 1999. Ian has over 19 years of experience in the semiconductor industry. Ian earned a bachelor's degree in electrical engineering from North Carolina State University and an M.B.A. from the University of North Carolina at Greensboro. He can be reached at *ian.beavers@analog.com*.

Erik MacLean is a hardware engineer at the Trusted Security Solutions group of Analog Devices in Tampa. He has six years of experience in embedded hardware design and security. He received a B.S.E.E. from the University of Florida in 2009 and an M.S.E.E. from the University of South Florida in 2011. He can be reached at *erik.maclean@analog.com*.

Analog Devices, Inc.
Worldwide Headquarters

Analog Devices, Inc.
One Technology Way
P.O. Box 9106
Norwood, MA 02062-9106
U.S.A.
Tel: 781.329.4700
(800.262.5643, U.S.A. only)
Fax: 781.461.3113

Analog Devices, Inc.
Europe Headquarters

Analog Devices GmbH
Otl-Aicher-Str. 60-64
80807 München
Germany
Tel: 49.89.76903.0
Fax: 49.89.76903.157

Analog Devices, Inc.
Japan Headquarters

Analog Devices, KK
New Pier Takeshiba
South Tower Building
1-16-1 Kaigan, Minato-ku,
Tokyo, 105-6891
Japan
Tel: 813.5402.8200
Fax: 813.5402.1064

Analog Devices, Inc.
Asia Pacific Headquarters

Analog Devices
5F, Sandhill Plaza
2290 Zuchongzhi Road
Zhangjiang Hi-Tech Park
Pudong New District
Shanghai, China 201203
Tel: 86.21.2320.8000
Fax: 86.21.2320.8222