Abstract
The Industrial Internet of Things (IIoT) is driving investment in new technology and manufacturing methodologies as companies rush to capitalize on a predicted 14 trillion dollar economic gain. A wave of new connected devices is the result, many of which are small endpoint devices running real-time operating systems. These systems often perform critical functions in our factories, electric grid, transportation infrastructure and other essential elements of modern society. The viability of the IIoT therefore depends on the security of the endpoints, the network and all of its subsystems. Unfortunately though, traditional security solutions do not scale down to support the RTOS-based devices that make up the bulk of the IIoT. New solutions and approaches are required.

This paper describes security requirements for IIoT devices and details solutions available today for securing these devices.

I. Introduction

Multiple benefits
IIoT, The Industrial Internet of Things, is being hailed by some as the next great industrial revolution. By some estimates there are more than 60 million machines in factories worldwide and 90 percent are not connected. It is no surprise that companies are looking at this opportunity to create new connected IIoT not just to reduce costs of operation or eliminate downtime, but to add new solutions and services for new revenue streams.

An important area in which the IIoT creates value is the creation of a network of device endpoints: smart, connected sensors and controllers that not only talk to each other, but also monitor and manage a wide range of machines and industrial systems. By combining this connectivity and functionality with analytics, information technologies and operational technologies, owners of industrial plants will obtain major benefits. For example, factories can be designed to adapt in real time to changes during production, or to anticipate and avoid events that might degrade operations. Additionally, predictive maintenance programs can be implemented to eliminate the downtime or catastrophic consequences caused by unanticipated failures of critical system components. By achieving even small percentage gains in plant operations or reductions in unplanned downtimes, these types of upgrades will dramatically improve the profitability of manufacturing operations.
To take full advantage of the improvement opportunities offered by the IIoT, an entire system—from sensors, actuators and motors, up through the controllers—should be connected to information and operational technology systems, and beyond into the cloud. Expanded connectivity will boost efficiencies in operations and integrate the supply chain more tightly and in innovative ways. It will also enable entirely new business models and revenue streams.

**Substantial challenges**

Although connectivity is the key to unlocking the full potential value of the IIoT, it brings a risk of cyberattacks. When systems are connected to the Internet and larger corporate networks, cyberattacks become possible, even likely, from external and internal sources—whether accidental or malicious. The benefits of the IIoT therefore cannot be achieved without multiple layers of security that successfully protect all the networked systems and devices. Secure communication, secure network monitoring and securing code execution at the device level are essential, not optional.

It’s critical for system engineers to address security issues at every layer. Although traditional IT-endpoint security and network-monitoring solutions protect IT business applications, such solutions won’t work for the embedded devices closest to the physical systems. These operational assets must be protected against cyberattacks by integrating security directly into the endpoint devices themselves.

Minimizing vulnerabilities requires both specialized security hardware as well as software. To support enterprise security standards, embedded devices must incorporate the following key features: secure boot code, secure application updates, tightly controlled authentications, and secure communication protocols.

*Figure 1. The factory network is comprised of countless endpoints from sensors, actuators, and controllers – each a critical component that needs security.*
II. What Do We Mean by Security?

Security in its simplest form entails ensuring that authorized operations and actions are allowed, while unauthorized actions are blocked. Most cyberattacks against embedded devices exploit one of the following categories of vulnerabilities:

**Insecure by design:** Devices that use hard-coded passwords, transmit login credentials in the clear, allow remote accesses without authentication, or have other obvious unprotected interfaces.

**Security with significant loopholes:** Devices with built-in backdoors, which use weak or default passwords, permit plaintext storage or transmission of encryption keys, or have similar vulnerabilities.

**Good, but partial security:** Devices that provide strong security against certain types of attacks, but leave other interfaces unprotected. Prime examples are systems that implement TLS only for some but not all communication; and security protocols in which the setup-phase encryption key is exchanged without being encrypted, making them vulnerable to eavesdropping attacks. Other examples include systems that implement secure communication, but don’t incorporate secure boot capability; and systems that have a secure operating system, but fail to secure the application layer.

**Features that are vulnerable to exploitation:** Devices that have weak encryption, exploitable buffer overflows or zero-day vulnerabilities, or cannot withstand brute-forcing of their authentication mechanisms. The hackers depicted in movies and on TV typically take advantage of these types of deficiencies.

Unfortunately, although it’s vitally important to secure networked devices against such vulnerabilities, the reality is that most currently deployed embedded devices cannot withstand even very basic forms of cyberattacks.

Ensuring the security of IIoT devices requires addressing all of the issues described above. Ideally, robust design solutions will include adaptable security policy management and the ability to securely update firmware to protect against new types of attacks as they emerge.

**Challenges for Creating a Secure IIoT**

The Industrial Internet of Things encompasses a wildly diverse range of connected devices and systems: from small to large, simple to complex. They span from commercial gadgets to sophisticated systems found in military, utility and processing/manufacturing systems.
Embedded devices are very different from standard PCs or other IT products, but they constitute important and growing elements of the expanding web-connected network. Many of them use specialized real-time operating systems such as ThreadX, µC/OS-III or Nucleus, or a stripped-down version of Linux.

Installing new software on most embedded devices deployed in the field either requires a specialized upgrade process or simply can't be done. Further, in most cases, these ubiquitous devices are optimized to minimize processing cycles and memory usage. Therefore, they don't have the extra processing resources required to support traditional security mechanisms. As a result, standard PC security solutions can't solve the challenges of making embedded devices safe from cyberattacks. In fact, given the specialized nature of embedded systems, PC security solutions won't even run on most embedded devices.

The driving principle for enterprise security is to provide multiple layers of protection. Firewalls, authentication/encryption schemes, security protocols, and intrusion-detection/intrusion-prevention mechanisms are well established, widely adopted enterprise security solutions. Nevertheless, firewalls and intrusion detection features are virtually absent in embedded systems, which typically rely on simple password authentication and security protocols.

Typically, makers of embedded devices have assumed that their products aren't attractive targets to hackers. Other common perceptions have been that networked embedded devices aren't vulnerable to attacks and that authentication and encryption can adequately protect against cyberattacks. These assumptions are no longer valid. Today the number and sophistication of attacks against embedded devices is rising to worrisome levels.

This trend has impacted many new product designs. Whereas cybersecurity has long been a critical focus for large enterprises, it's now a strong focus for most system engineers building sensing and control devices. Fortunately, rather than reinventing the wheel, product developers can apply the security principles used to implement enterprise security.

To ensure security for embedded devices, given their specialized nature, the following concerns must be addressed:

- **Preservation of functionality**
  Embedded control devices are at the heart of the transportation infrastructures, utility grids, communication systems and other elements essential to modern society. Successful cyberattacks on them can have catastrophic consequences. Thus, security solutions must protect both the data stored on networked embedded devices and safeguard the operations they perform.

- **Attack replication**
  After embedded devices are developed, they are mass-produced. If a hacker can find a way to successfully attack one of these devices, that attack can be replicated across all devices of the same type. Thus, a single-point breach can become a mass-failure mechanism.

- **Assumed security**
  Many system engineers have long assumed that embedded devices are not targets for hackers; i.e., they have relied on security by obscurity. That assumption is totally false today. Security should now be considered a top priority for most embedded designs.

- **Upgrade difficulties**
  Most embedded devices are not easily patched. After they are deployed, they run factory-installed software for as long as they remain operational, even if that code has security vulnerabilities.
• **Long life cycles**

Life cycles of embedded devices are typically much longer than for PCs or consumer devices. Devices may be in the field for 15 to 20 years or more. Implementing designs that will withstand the ever-evolving security threats expected over the next two decades is a tremendous system engineering challenge.

• **Deployment outside enterprise security perimeter**

Many embedded devices are either mobile or deployed in application environments. As a result, they may be directly connected to the Internet, totally outside the security protections installed in corporate environments.

## III. Secure Foundations

Cryptographic methods provide the foundation for implementation many of the features used to secure embedded devices including secure data storage, secure communication protocols, and secure boot. These techniques require cryptographic keys, which must be kept secret.

System engineers often implement secure key storage in hardware using a hardware security module (HSM). Most HSMs also offer both crypto-acceleration to offload computation-intensive operations from the main CPU and True Random Number Generation (TRNG). Additionally, some HSMs provide protected code execution that allows security-critical operations to run in a separate memory space that user-space code cannot access. This prevents programs in the user space from tampering with the operation of security-critical features or stealing keys.

The hardware security features of the Renesas Synergy platform (Figure 3), and versions of the Trusted Platform Module (TPM) are examples of secure crypto-processing.

![Figure 3. Block diagram showing details of Renesas' Synergy HW security modules.](image)

### Asymmetric vs. symmetric encryption

Regardless of whether encryption is implemented in hardware or software, most security protocols employ both symmetric and asymmetric types. IPsec does this unless pre-shared keys are used, and TLS uses asymmetric encryption to securely exchange a secret key at the start of a session. Subsequently, IPsec and TLS both transition to symmetric encryption using the secret key established during the key-exchange process. The reason for this approach warrants some explanation.
Symmetric encryption

Symmetric encryption is simple. It uses a single secret key that is shared by both of the communicating entities or nodes (see Figure 4).

![Figure 4. Because symmetric encryption shares a secret key with the nodes at both ends of the network link, it provides limited security.](image)

The problem, of course, is how can keys be exchanged without someone or something intercepting them?

Asymmetric encryption

Asymmetric encryption uses a key-pair consisting of a public key and a private key (see Figure 5). Each node has its own key pair. The private key must be protected and kept secret, but the public key can be shared with other nodes.

Key-pairs are created in such a way that data encrypted with a public key can be decrypted only with the correct private key, and vice-versa. In other words, if the key-pair is \((\text{PrivK}, \text{PubK})\), data is encrypted and decrypted as follows:

\[
\text{EncData} = \text{Encrypt}(\text{PubKey}, \text{Data})
\]

\[
\text{DecData} = \text{Decrypt}(\text{PrivKey}, \text{EncData})
\]

The resulting decrypted data will be the original data.

\[
\text{Data} = \text{DecData}.
\]

![Figure 5. Asymmetric encryption uses public and private keys to provide a high level of security. Alice, to the left, uses Bob's public key to encrypt data. Bob then uses his corresponding private key – the only key available – to decrypt the message.](image)

All key-pairs are mathematically related to enable encryption/decryption in this manner (Figure 5).
The brilliance of asymmetric encryption is that data encrypted with the public key cannot be decoded with the public key—only with a specific private key. This is accomplished using Modulus math, where equations can easily be run in one direction (with the public key), but not resolved in the opposite direction without the private key.

For a communication session between two nodes, A and B, the key-pair security method works like this. Both nodes A and B generate the public and private key-pair that they will use for communication. Node A’s key-pair is mathematically related, as is Node B’s key-pair. They each can publish or share the public key, but must keep the private key secret.

Consider for example communication from A to B. Node A uses Node B’s public key to encrypt data. Node A then sends it to Node B, which decrypts the message with its private key—the only possible way to decrypt the data. Likewise, when Node B wishes to send data to Node A, it encrypts data using A’s public key, and that data is read at Node A using A’s private key.

It’s critical that the private keys are never exchanged, only the public keys. Again, the mathematical relationship between the public and private keys ensures that data encrypted with the public key can only be decrypted together with the private key. Again, it can be decoded only with the intended receiver’s private key.

In the IIoT market, one problem with asymmetric encryption is that it can easily use 100 times more CPU cycles than symmetric encryption. This processing load causes problems for embedded devices that perform time-sensitive operations and/or have limited system resources.

The solution is to run an asymmetric session only to create an initial encrypted connection for exchanging a secret symmetrical key. Thereafter, that symmetrical key is used for the remainder of the communication session.

Built-in hardware for encryption is often used to accelerate these security processes. Renesas’ Synergy S7/S5 platform provides a “Secure Crypto Engine 7” for the asymmetric algorithms: RSA and DSA. It also handles symmetric algorithms: AES, 3DES and ARC4.

**Security Protocols**

Security protocols are fundamental to securing the IIoT. They ensure that data and commands exchanged are encrypted and cannot be intercepted, spoofed or manipulated. Protocols that enforce strong authentication also protect against control commands being sent to embedded devices from unauthorized sources.

The most common security protocols for protecting Ethernet-based communication in IIoT devices are TLS (SSL), DTLS, SSH and IPsec. TLS solutions are available for most embedded platforms, and this protocol is well understood. As a result, TLS is a good choice for many embedded devices. It’s the security solution most widely deployed today.

However, encryption protocols such as TLS that are implemented higher up in the stack, at the OSI Transport layer, have a drawback. A risk exists that not all communication will be encrypted.

System engineers are responsible for ensuring that TLS is properly used by all applications that communicate sensitive data. They must make it impossible to bypass encryption by implementing communication at a lower level or without using the security protocol. For example, they cannot allow a situation in which an application uses socket programming at the Transport (TCP) layer without using the TLS (SSL) interface. If that happens, the data won’t be secured. This exact scenario recently caused a high-profile security vulnerability in an automotive system.

IPsec is a secure replacement for the IP protocol, which resides at the network level (see Figure 6), is often the preferred security solution because it prevents this type of security breach. IPsec provides secure communication for all data above the network layer, as long as no alternative unencrypted IP-interface is available. However, IPsec is not as widely supported as TLS.
Determining the best security protocol to use to protect an embedded device depends on a number of factors. Among them are available computing power, interoperability requirements, and which security protocols the platform supports.

As previously mentioned, TLS now is widely used in embedded devices. Currently, IPsec is gaining in popularity. DTLS (TLS over UDP) is another good choice, especially for very low-end wireless sensors that have limited bandwidth and processing power.

<table>
<thead>
<tr>
<th>OSI Layer</th>
<th>Protocol Example</th>
<th>Network Security Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applications</td>
<td>7. Application: HTTP, FTP</td>
<td>Firewall (e.g. application protocol filtering)</td>
</tr>
<tr>
<td></td>
<td>6. Presentation: HTML</td>
<td>Authentication (passwords, keys, certificates)</td>
</tr>
<tr>
<td></td>
<td>5. Session: Telnet, SMTP</td>
<td></td>
</tr>
<tr>
<td>Network services</td>
<td>4. Transport: TCP, UDP, SSH+TLS</td>
<td>Firewall, IP address, port + protocol filtering</td>
</tr>
<tr>
<td></td>
<td>3. Network: IP, IPsec</td>
<td></td>
</tr>
<tr>
<td>Infrastructure</td>
<td>2. Data link: Ethernet, WiFi</td>
<td>Firewall, MAC address filtering</td>
</tr>
<tr>
<td></td>
<td>1. Physical: Broadband access</td>
<td></td>
</tr>
</tbody>
</table>

Determining the best protocol to use in an IIoT device depends on interoperability requirements, security use cases and the capabilities of the particular IIoT device, among other factors. One recommendation is Icon Labs’ Floodgate TLS, which is available for the Renesas R-IN32 industrial network device platform. Another is the Renesas Synergy platform, which supports both TLS and the NetX implementation of IPsec.

Small IIoT network-edge devices with low bandwidths (sensors, etc.) use wireless protocols such as ZigBee or Bluetooth Low Energy (BLE). Those protocols have been demonstrated to have weak built-in security. Such edge devices typically run on low-cost, lower-power processors. They don’t support the security protocols commonly utilized in TCP/IP networks.

There are two options for increasing the security features of such devices: implement application-level security, or use DTLS (TLS over UDP). DTLS is ideal for low-bandwidth communication channels where battery life is critical. It minimizes the number of packets transmitted by using UDP instead of TCP.

Several other important security measures can be used to protect embedded devices against common attacks: application-level encryption, session tokens, and packet-sequence numbers. Newer versions of some protocols, such as BLE 4.2 privacy, address many security weaknesses.

Even if current system designs don’t employ application-layer protection features, they can be added in the future as part of a defense-in-depth security strategy.

**Figure 6. This chart shows the OSI layers and some of the protocols used in each.** The Transport layer, for instance, handles connections at a ‘socket’ level; i.e., the ‘socket’, ‘bind’, ‘listen’ and ‘accept’ TCP library calls.
Access control

Access-control solutions aim to ensure that only authorized users are allowed to control the operations of an IIoT device. A user may be a person using a Human Machine Interface, or another machine communicating with the IIoT device. Access control necessitates a good authentication mechanism. For commands originating from an HMI, that mechanism might be a password or biometric data. For commands received over a network interface, a certificate, token or password-based authentication mechanism can be used.

Firewall, packet awareness, application data filtering

Embedded firewalls reside directly on the embedded device, integrated directly within the TCP/IP stack. Like familiar network firewalls, they control which packets the device can process. A firewall filters packets at the Network (IP) layer. Packet awareness blocks bad packets before being passed up the protocol stack to the application. This capability provides an effective defense against cyberattacks.

An embedded firewall can implement several filtering stages, depending on the embedded device’s requirements (Figure 7).

- **Stateful Packet Inspection (SPI)** filtering accepts or discards packets based on the state of the connection. It provides protection against “Christmas tree attacks”, SYN flood attacks, and other assaults that attempt to exploit the stateful nature of protocols such as TCP.
- **Static filtering** blocks packets by IP address, port number or protocol type. It can enforce network segmentation and is particularly useful in static networks where an IIoT device communicates only with a small, defined set of other devices. Because any communication from an unknown device might be the result of a cyberattack, the filter will reject it.
- **Threshold filtering** protects against packet flooding and other Denial of Service (DoS) attacks.
- **Protocol- and application-specific filtering** can protect networked embedded devices against application-specific attacks. These filtering methods can be added to implement access control in legacy protocols that lack security control mechanisms.

Adding protocol-specific filtering would be a good security strategy to apply to an embedded device that uses the ModbusTCP industrial control protocol, for example. ModbusTCP doesn’t provide access control; it processes any command it receives, regardless of who or what sent it. The security upgrade can include a whitelist of authorized IP addresses. This would enable the filter to block Modbus commands sent from any source other than the trusted ones on the list.

Figure 7. A firewall can be configured with a set of filtering rules that define which packets to block and which ones to pass. Access decisions can be based on the sender’s IP address, the port to which the packet is sent, or the protocol used. Filter-type firewalls are effective for enforcing access policies, but they cannot protect embedded devices against spoofed packet headers. Implementing that protection requires certificate-based or other application level authentication.
Intrusion detection

As previously mentioned, the majority of currently deployed embedded devices lack basic security features, making it relatively easy for hackers to launch malicious assaults. As a result, more cybercriminals are beginning to specifically target them.

Cyberattacks typically develop over a period of time. Hackers first probe a network looking for, finding, and exploiting any vulnerabilities they discover. Then, leveraging an exploited device, the criminals move deeper into the network, beginning another vulnerability search and exploitation from that beachhead. To stop such attacks, it’s first necessary to detect them.

Intrusion Detection Systems (IDS) are commonplace security safeguards in enterprise networks and PCs. An IDS, as the name implies, sends an alarm and/or initiates countermeasures whenever a system is determined to be under attack or being probed. IDS solutions differ in design and capabilities, and can detect many different types of attacks, regardless of form. Unfortunately, many of them aren’t suitable for use in embedded devices.

Embedded firewalls, though, can provide a fundamental intrusion-detection function by monitoring and reporting anomalous activity to a security management system. Here is a simple example: An IIoT device that doesn’t support SSH begins receiving messages on the default SSH port. That activity is probably an attempt to connect to the device by an unauthorized, possibly malicious, source. A firewall that can report this information can generate an early attack warning, which is a critical capability too often missing in even the networked embedded devices today.

All too often, cyberattacks remain undetected for months while hackers probe a network, discover vulnerabilities, ex-filtrate data and plan for their final attack on the system. Adding IDS capabilities (Figure 8) to embedded devices is very prudent and highly recommended wherever feasible.

Prompt detection and alerts about suspicious activity allow system administrators to take action to block attacks, quarantine compromised systems, and protect their networks. When embedded devices are upgraded to support basic IDS capabilities, they will no longer be easy targets for hackers.

Secure Boot

To ensure that an IIoT device runs only code authorized by the manufacturer, it must support a secure boot and have a secure firmware update capability. Properly implemented, these features eliminate an entire class of vulnerabilities.

Figure 8. Secure signing produces a cryptographic signature that allows the networked embedded device to validate the code it receives.
Secure boot begins with a first-stage bootloader that is programmed into a protected or non-writable storage location on the device. It collaborates with a second-stage bootloader, which can be more complex than the first-stage unit and stored in reprogrammable memory.

After the first-stage bootloader calculates the hash value of the second-stage bootloader, it verifies that the hash is correct by comparing it to a signed signature value (see Figure 9) of the second-stage bootloader. The second-stage bootloader then repeats this process to verify that the operating system and applications are valid. Embedded devices that use a monolithic RTOS perform this procedure in a single step. In Linux-based devices that have separately loadable applications, this process can be repeated to validate each application in the system before it is loaded.

After a layer of code is validated, it becomes trusted and the validation process proceeds to the next layer in the chain.

Figure 9. Secure boot validates that the code on the device is from the OEM and has not been altered. This verification is performed using the public key—the only key that can decrypt the signature needed to access the reference hash digest.

Secure boot relies on signed code images to enable validation of the image during the secure boot process. Be aware that the roles of the public and private keys are reversed for signatures, when compared with asymmetric encryption. The device supplier signs the code images using its private key. IIoT devices use the corresponding public key to verify that the authorized OEM produced the new software.

Secure firmware update

Secure firmware update, like secure boot, validates that the OEM signed the new code images during the upgrade process. If downloaded images fail this validation process, they are discarded and the upgrade is not performed. Only images that pass this validation process are accepted and saved to the device; i.e., written to flash, replacing the previously stored image.

Machine-to-machine authentication methods are another aspect of secure firmware updates. Used by the IIoT device to authenticate the upgrade server before downloading the new firmware image, they add an additional layer of protection against hacking.

The infamous “Heartbleed” vulnerability is a vivid example of a high-profile vulnerability that impacted a very large number of embedded devices. It shows that implementing security for remote updates is critical. In the absence of a secure, remote software update capability, either a technician has to go to the site to do it, or the device has to be returned to the manufacturer, or the device’s users will have to perform the installation.

Data at Rest (DAR) protection

IIoT devices typically are located “in the field”, unlike enterprise servers, which are locked away in data centers. This raises the distinct possibility of theft or physical attack. Therefore, sensitive data stored on embedded devices should be encrypted to thwart physical attacks. Data thieves cannot be allowed to read unprotected data from a flash drive in the device. Data at Rest (DAR) security measures ensure that those activities will fail.
DAR protection is achieved by encrypting the sensitive data on the embedded device. Most IIoT devices don’t have sufficient computing power to support full disk encryption. Nevertheless, sensitive data can and should be protected.

Although the optimum DAR security measures will vary, in certain situations it must be extremely robust. This is especially true for devices that process credit card numbers or patient information, for example. The data they contain must always be encrypted. The encryption key should be kept in protected memory on the device using a hardware security module. Alternatively, it should be stored in a secure location such as on an encrypted USB drive or a network server, and retrieved as needed for accessing the data.

To be effective, DAR solutions must ensure that sensitive information can’t be stored in raw form on a hard drive, flash drive or other persistent storage media. Data should always be encrypted before it is written to a file. Thereafter, encrypted files should be decrypted in memory and remain in RAM—and never written back to the file system without being re-encrypted.

V. Security Management and Visibility

Increasing numbers of embedded devices are deployed on large enterprise networks that implement sophisticated security-management and security-monitoring features. When an embedded device with no management agent is placed on such network, it shows up on the management console as a rogue system. The management system cannot manage the device or even intelligently display any information about it. This is an unacceptable situation that violates security standards.

Many enterprises mandate that every device on the network be able to report its properties and status (situational awareness). It must also be able to accept changes to its security policies sent from a security management system. This requirement provides several operational advantages:

- All devices on the network can be tracked and managed. Knowing what devices are on the network is the first step towards robust cybersecurity.
- Centralized policy management enables consistent security policies throughout the network.
- Centralized policy management also simplifies security implementations because it dramatically reduces the time required to manage policies.
- Centralized reporting of events increases the effectiveness of security procedures and processes. After an attack is detected against one node, policy changes can be immediately sent throughout the network to prevent the attack from spreading.
- Centralized activity reporting can detect suspicious behavior. Embedded systems have limited resources to analyze the packets they are processing. A security management system, however, can collect a list of all IP addresses communicating with an embedded device and then correlate those addresses to detect communication with known bad hosts. It can even determine if the enterprise system is sending packets to known malicious IP addresses.

Because more and more cyberattacks are being launched against embedded devices, it is important to integrate them with management systems to detect, report and mitigate malicious activities.

VI. Putting It All Together – A Framework For Device Security

No single “one size fits all” security solution exists for networked embedded devices in IIoT markets. Security requirements must take into consideration the cost of a security failure (economic, environmental, social, etc.), the likelihood of attacks, the available attack vectors, and of course, the cost of implementing and maintaining the appropriate security solutions. Protection against cyberattacks should be a standard feature, not just an option. With the diverse nature and deployment of IIoT devices, and the reality that security perimeters can—and are likely to be—penetrated, device suppliers cannot be certain that their products will be installed in secure networks.
The Floodgate solution

Icon Labs and Renesas Electronics have collaborated to develop secure platforms for both IIoT and IoT applications. The Synergy security portfolio delivers comprehensive protection from threats. Icon Labs’ Floodgate Security Framework has been integrated into the Renesas Synergy Platform for IIoT. The Floodgate Security Framework is also integrated with the Renesas R-IN32M3 platform for industrial applications.

**Figure 10.** Icon Labs’ Floodgate Security Framework is incorporated into the Renesas Synergy and R-IN32 platforms.

The Floodgate Security Framework provides the software building blocks for implementing the security countermeasures discussed through this whitepaper. This provides a complete, integrated solution for IoT device cybersecurity. Depending on the requirements of the device, individual modules can be used to implement specific protections such as secure boot or intrusion detection.

**Figure 11.** The Floodgate Agent enables the integration of embedded devices into networks that use security management systems for event reporting and policy management.

Integration with Enterprise Security management systems extends security monitoring and managing to IoT devices. This enables IoT devices to report cyber-attacks and for system administrators to upgrade security policies, upgrade firmware and manage IoT endpoints.
Platform Security with the Renesas Synergy Platform

The first step to achieving IIoT security is to ensure that security is built into the embedded device itself. A prime example of an off-the-shelf, built-in security capability is the Renesas Synergy Platform. The S7/S5 platform has a “Secure Crypto Engine” with the following hardware blocks.

- **True Random Number Generator** — A hardware block that produces encryption keys that are impossible to predict
- **3DES** — A symmetric key block cipher (sender and receiver use same key)
  - **AES** — Hardware executing a symmetric-key algorithm that uses a substitution-permutation approach to encrypt and decrypt data. AES works with 128/192/256-bit encryption keys. Additionally, it offloads the embedded device's core processor for better application performance.
  - **ARC4 (or RC4)** — A symmetric, pseudorandom key stream that encrypts plaintext digits to form a secure cipher stream
- **RSA/DSA** — The asymmetric encryption method widely used for secure data transmission that employs public and private keys. The tight security that RSA provides is based on the practical difficulty of factoring the product of two large prime numbers.
  - **Hash Blocks** — Computations primarily used for digital signatures and message authentication. The processing is similar to what CRC or checksums perform, but is very difficult to forge. Hash blocks can also index data, detect duplicate data or uniquely identify files.
  - **TRNG** — A true random number generator
  - **SHA1/224/256** — Cryptographic hash functions specified by the National Institute of Standards & Technology

![Figure 12. Renesas Synergy security implemented in the on-chip hardware.](image)
The Floodgate Security Framework uses the Synergy crypto acceleration hardware to implement secure boot and provide cryptographic support for TLS, Data at Rest (DAR) protection, and certificate based authentication. The embedded firewall, intrusion detection and management agent are also supported on the Synergy Platform.

**Figure 13.** MCUs in the Renesas Synergy product family feature accelerators for symmetric cryptography, a HASH calculator and a true random number generator. They have the ability to limit JTAG accesses. The S7 and S5 versions add accelerators for asymmetric cryptography and asymmetric key generation.

<table>
<thead>
<tr>
<th>Unique ID</th>
<th>True Random Number Gen</th>
<th>Crypto HASH Functions</th>
<th>Symmetric Key Crypto</th>
<th>Asymmetric Key Crypto</th>
<th>Secure Key Storage</th>
<th>Read-Out Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>S7</td>
<td>Best</td>
<td>Best</td>
<td>Better</td>
<td>Good</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td>Best</td>
<td>Best</td>
<td>Better</td>
<td>Good</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>Best</td>
<td>Best</td>
<td>Better</td>
<td>Good</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>Best</td>
<td>Best</td>
<td>Best</td>
<td>Limited</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Network security using the Renesas R-IN32M3**

The Renesas R-IN32M3 ASSP is specifically designed to enhance the processing of Ethernet data. The chip incorporates a HW-RTOS that helps maximize overall performance of embedded devices. It integrates optimized controller hardware for EtherCAT and CC-Link, as well as a general hardware accelerator circuit that processes all other Ethernet frames. These accelerator functions combine to greatly facilitate Ethernet processing, thus enabling the embedded system’s core CPU to better handle security processing.

Two previous whitepapers discuss the features of the R-IN32M3 ASSP: Accelerating Industrial Ethernet Communication in Hardware and HW-RTOS and Improved RTOS Performance by Implementation in Silicon. The latter discusses how the on-chip HW-RTOS reduces both CPU load and jitter.

The R-IN32M3’s Ethernet accelerator and HW-RTOS cooperate to increase frame-processing rates. They reduce Ethernet frame loading and jitter by decreasing CPU loading at any given time. The Ethernet acceleration blocks of the Renesas R-IN32 ASSP require no user intervention. The checksum accelerator checks length and errors, and checksums the frame. The MAC-DMA and Buffer Manager combine to quickly move the frame out of the FIFO to a bigger buffer, where the data can be further processed. The Header EnDec analyzes the frame’s header, aligns it and puts it into user RAM.
Figure 14. Integration of the Floodgate Security Framework with the R-IN32 provides a strong foundation for building secure industrial control devices.

The embedded firewall, intrusion detection module and management agent are integrated with the R-IN32, enabling secure, fast packet handling of industrial protocols. Floodgate TLS, secure boot and secure update capabilities provide secure code execution and secure communication. The Floodgate Framework provides the features required to build IIoT devices that comply with industry security standards such as the IEC 62443 standard for industrial systems and the FDA guidelines for medical devices.

Figure 15. The Floodgate Security Framework provides key security features for the R-IN32 and Synergy HW platforms from Renesas.

<table>
<thead>
<tr>
<th>Security Feature</th>
<th>RIN32 support</th>
<th>Renesas Synergy support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security protocols</td>
<td>TLS</td>
<td>TLS and IPsec</td>
</tr>
<tr>
<td>Firewall</td>
<td>Floodgate Firewall</td>
<td>Floodgate Firewall</td>
</tr>
<tr>
<td>Intrusion Detection</td>
<td>Floodgate IDS</td>
<td>Floodgate IDS</td>
</tr>
<tr>
<td>Secure boot</td>
<td>Floodgate Secure boot</td>
<td>Floodgate secure boot</td>
</tr>
<tr>
<td>Secure firmware update</td>
<td>Floodgate Secure update</td>
<td>Floodgate Secure update</td>
</tr>
<tr>
<td>Data at Rest protection</td>
<td>Floodgate Data at Rest protection</td>
<td>Floodgate Data at Rest protection</td>
</tr>
<tr>
<td>Secure Key storage</td>
<td>Floodgate SW secure key storage</td>
<td>Synergy HW secure key storage</td>
</tr>
<tr>
<td>Crypto algorithms</td>
<td>Floodgate SW crypto</td>
<td>Synergy HW crypto</td>
</tr>
<tr>
<td>Security management</td>
<td>Floodgate agent</td>
<td>Floodgate agent</td>
</tr>
</tbody>
</table>
Code Security Using the RZ/T1 Processor

The powerful RZ/T1 processor incorporates a R-IN32 block dedicated for Industrial Ethernet. It has an Intelligent Cryptographic Unit (ICU) to accelerate cryptography processing using AES-128 ECB/CBC encryption/decryption.

The ICU contains data OTP’d at time of production.

1. A customer Unique ID (UID - not necessarily secret) plus another internal secret key (BOOT Message Authentication Code)

2. A permanent (non-changeable) initial boot program

The initial boot program decrypts a user-defined secondary loader located in external ROM and verifies its “Key Data” using a command that invokes the ICU’s UID and secret key. It then executes the user’s loader (from RAM). This secondary loader should then process the user application code with its Key Data using the ICU in the same way.

An application note explaining how to generate encrypted ROM code, with example user-defined loader and application program is available from Renesas*.

* RZ/T1 Group Development Environment for Secure Booting (Secure Chip Producing Program).

**Figure 16. The RZ/T1 has an ICU used to decrypt and verify code before execution. It includes a permanent initial stage bootloader, a customer Unique ID paired with a secret BOOT_MAC code, all OTP’d by Renesas at production. The user’s Key Data in loader and application must mathematically have the right values to work with the UID and secret key embedded in the ICU in order to be authenticated.**
VII. Conclusion

System engineers face greater challenges today when developing IIoT-capable, network-connected embedded devices. Besides the usual issues—programming languages, sensors, drivers, middleware, operating systems, process management, etc.—they must deal with security issues, encryption standards, networking protocols and new technologies.

Solutions from Renesas, Icon Labs and other firms now help reduce development problems by offering proven products that address IIoT-specific endpoint design issues. By taking advantage of what’s now available in IIoT platforms—especially products that implement security functions—system engineers can focus more closely on leveraging their expertise to add targeted value to their designs. This approach enables new and upgraded products to advance to production quicker.

It is obvious that the viability of the IIoT therefore depends on the security of the endpoints, the network and all of its subsystems. The different security concepts and solutions presented in this paper, while not an exhaustive list, are all relevant to embedded, endpoint devices. Protection against cyberattacks should be a standard feature, not just an option.

alan.grau@iconlabs.com – www.iconlabs.com

carl.stenquist@renesas.com – www.renesas.com

wil.florentino@renesas.com – www.renesas.com