White Paper

Secure Internet Communication for IoT Applications

Brad Rex, Sr. Product Marketing Manager, IoT Infrastructure Business Unit, Renesas Electronics Corp.

January 2020

Abstract

Comprehensively securing connected devices is more important today than ever before, yet there is no one-size-fits-all approach to internet security. Each embedded solution has different requirements, and no security protection can safeguard against all the vulnerabilities that exist. Embedded developers need to strike a balance among competing approaches and priorities, including development costs and time commitments. This white paper examines various approaches to securing internet communication and the challenge of implementing these security methodologies correctly.

Growing Threat Landscape

Today, even the most basic items in our lives, from coffee mugs to pill containers, can be connected to the internet and cloud environments. This means that much of our daily environment has become a potential attack surface. As cyber threats to connected devices grow more powerful and malicious, almost every aspect of everyday life could be compromised by hackers and malware.

Securing embedded applications to protect data and intended functionality is now a primary consideration for developers, as any communication via the internet must be secured.

Importantly, embedded security cannot be an afterthought. It must be planned for and designed in from the beginning. It may be possible to add it later, but this after-the-fact approach significantly increases the expense of embedded security. Or, depending on the security vulnerability, it may not be possible to adequately address the security flaw at all once the device has been released.

What's needed is a security strategy that offers multiple layers of defense to implement in-depth, comprehensive protections for internet communication that takes advantage of the latest security advances in both hardware- and software-based security.

Methodologies for Secure Communication

A proactive approach to securing communication between internet-enabled devices requires multiple tiers of protection. To safeguard a device against multiple security threats requires securing the device's identity through hardware-based key generation. This identity should be securely stored in internal flash, leveraged to create trust, and provide privacy when added to designs and configured for target applications.

Establishing a strong device identity allows every device to be singularly identified and authenticated as unique. Once established, strong device identity addresses core embedded security requirements in a number of ways, as device identity can ensure authentication and identification when devices are connected to one another.

Serialized approaches to device identity, based on product serial numbers or individual device IDs, is a simple way to establish unique device identities, but asymmetric cryptographic approaches offer stronger security and a greater range of use cases.

Key-based encryption. The first step in creating cryptographic device identity is key generation. The keys can be either generated inside the device or they can be generated externally in a secure facility and injected into the device. Once the device keys are generated or injected, an entity called the Certificate Authority (CA) issues digital certificates. A CA can be either public (located in the Cloud) or private (located on-premises and typically hosted on a secure server). Once the device identity is created and programmed on the device, it must be securely stored in secure memory which can only be accessed by "secure code" to prevent the device identity (keys and certification) from being erased or reprogrammed.

Asymmetric cryptographic algorithms require the creation of a public key and a private key and are part of a cryptology methodology called public key infrastructure (PKI) that offers authentication via digital certificates. The public key can be made public to anyone, while the private key must be known only by the party who will decrypt the data encrypted with the public key.

When a device with a secure and singular identity connects to the network, it must authenticate and establish **trust** between other similarly identified devices, services and users, enabling trusted members of the system to securely communicate and exchange encrypted data and information. **Privacy** is another benefit of unique device identity, as data exchanged on secured networks can include personal, sensitive and financial information that must be kept private and secured – often under regulatory compliance. Device **integrity** applies to both the devices and data being transmitted within the trusted ecosystem. The integrity of a device starts with proving it is what it says it is. With strong device identity, devices are ensured as legitimate – reducing counterfeit products and protecting a company's brand.

Securing data in flight. While device identity is the foundation for secure products, secure manufacturing and secure communication, securing data as it moves across the network requires a different kind of security.

Transport Layer Security (TLS) is a cryptographic protocol that provides communications security over a computer network. It continues to evolve new standards with stronger security measures to keep pace with the emerging threat landscape. Both asymmetric and symmetric encryption are utilized to provide a high degree of security without compromising encryption and decryption speeds. TLS provides privacy and reliability between two communicating applications by enabling:

- **Encryption:** The messages exchanged between communicating applications are encrypted to ensure that the connection is private, with symmetric cryptography used for data encryption.
- **Authentication:** A certificate-based mechanism is used to verify identity.
- **Integrity:** To ensure that a connection is reliable and ensure message integrity, Message Authentication Code (MAC) mechanisms detect message tampering and forgery.

Datagram Transport Layer Security (DTLS) is a communications protocol designed to secure datagram-based communications and prevent eavesdropping and tampering. It is based on the TLS protocol and provides a similar level of security. It can be used across web browsing, mail, instant messaging and VoIP. DTLS employs the Use Datagram Protocol (UDP), while TLS uses Transmission Control Protocol (TCP). However, DTLS offers lower overhead and reduced latency than TLS, making it a good choice for time-sensitive transmissions. DTLS is one of the security protocols used for Web Real Time Communication (WebRTC) technology for web browsers via simple JavaScript APIs.

Secure IoT and Cloud Communication

The recent growth of the Internet of Things (IoT) and cloud computing have introduced new requirements for secure digital communication.

The IoT comprises a sprawling set of technologies and connected smart devices such as sensors and actuators that intelligently link together to enable new forms of communication between and among things and people. In the process, IoT devices generate massive amounts of data, and cloud computing provides a pathway to enable data to travel to its intended destination, or to perform computations that are beyond the often-limited processing power of IoT devices.

IoT use cases include smart homes, smart cities, wearables, e-health, agriculture and energy management. These smart networks have the capability to collect and analyze information and even make decisions without any human interaction. In these scenarios, digital communication security is of utmost importance given the impact that could result from a malicious unauthenticated device in an IoT system. While the security requirements of an IoT network (the need for confidentiality, integrity or authentication) are largely defined by the type of applications it serves, traditional authentication or cryptographic approaches may not be possible with resource-limited IoT devices and networks.

IoT devices need to authenticate and be provisioned to use cloud services. Most cloud providers today use combinations of HyperText Transfer Protocol (HTTP), Message Queuing Telemetry Transport (MQTT), and Constrained Application Protocol (CoAP) for data transmissions secured via TLS/DTLS using provisioning certificates and keys generated against device credentials or through side-channel mechanisms.

HTTP is the underlying protocol used by the World Wide Web. It was designed for communication between web browsers and web servers, but it can also be used for other purposes. However, HTTP was originally designed as a clear text protocol, and is vulnerable to eavesdropping and man-in-the-middle attacks. HyperText Transfer Protocol Secure (HTTPS) is an extension of HTTP and is used for secure communication over computer networks and the internet. HTTPS augments HTTP with TLS encryption to ensure all data exchanged between networked devices is encrypted in both directions.

This is achieved through PKI and the use of X.509 certificates to attach cryptographic key pairs to the identities of trusted devices, individuals, companies and websites. If the certificate presented by an HTTPS website has been signed by a publicly trusted certificate authority, users can be assured that the identity of the website has been validated by a trusted third party. Each key pair includes a private key, which is kept secure, and a public key which can be widely distributed. The private key acts as a decoder to allow its possessor to decrypt messages encrypted by the public key. Senders can also use private keys as a way to digitally sign messages. Even though the internet itself is an insecure network, the cryptographic system establishes a safe connection within the network.

HTTP-based protocols are too heavyweight for most IoT devices, and request-response speed often isn't fast enough. Much more lightweight, **MQTT** is a client-server publish-subscribe messaging transport protocol that is open and simple to use. MQTT is designed for constrained devices such as sensor nodes and other IoT devices, as well as low-bandwidth, high-latency or unreliable networks. These characteristics make MQTT ideal for use in environments, such as communication in Machine-to-Machine (M2M) and IoT scenarios, where a small code footprint is required, and/or network bandwidth is at a premium. MQTT supports TLS and, if implemented, can encrypt the complete communication between client and broker, preventing hackers from intercepting data in motion.

CoAP is a specialized web transfer protocol for use with constrained nodes and networks in the IoT. The protocol is designed for M2M applications such as smart energy and building automation. CoAP relies on UDP transport to transfer data and DTLS security aspects to protect the information.

Challenges of Securing Connected Devices

Although both MQTT and CoAP were designed for the special requirements of the IoT, and both support popular security protocols, they can be easily attacked if not secured correctly. Networks based on these protocols are particularly susceptible to the following security threats.

Distributed Denial of Service (DDoS) Attack. One of the most common and disruptive forms of cyberattack, a DDoS impedes the normal operations of a targeted server, service or network by overwhelming the target with a flood of internet traffic. These attacks shut down the system or network and prevent authorized users from accessing data or services. These attacks usually result from hackers breaching thousands of networked devices and then orchestrating their messaging ability to bombard a central server until it fails from exhausting its compute resources. Inadequately secured IoT devices are often implicated in DDoS attacks. The massive attack on Dyn in October 2016 that brought down much of the internet was perpetrated through hacked networked devices such as security cameras and digital video recorders.

TCP SYN Flooding. Another form of DDoS attack, a TCP SYN flood manipulates the three-way TCP handshake between client and server to stall the targeted server with incomplete requests to open connection ports. By repeatedly sending initial connection request (SYN) packets without completing the transaction, the attacker is able to overwhelm all available ports on a targeted server, rendering it unresponsive.[1]

Slowloris. Another type of DDoS attack, Slowloris allows an attacker to stall a targeted server using a single machine to open and maintain many simultaneous HTTP requests between the attacker and the target. With a limited number of threads available to handle concurrent connections, the targeted server will wait for the request to complete, which never occurs. When the server's maximum possible connections have been exceeded, denial-of-service will occur.

Insecure MQTT. While MQTT is not inherently secure, there are several mechanisms available for securing MQTT connections, including simple username and password combinations and TLS, which provides an encrypted pipeline for messages on MQTT.
With MQTT, it's important to note that security restrictions are enforced by the MQTT broker or server, and the client nodes must be configured separately. Not only does this introduce added complexity, designers must consider the capabilities of MQTT clients when planning security for the IoT implementation, as sufficient support may be unavailable on simple clients, such as very basic sensors.

Insecure CoAP: The security concerns for CoAP are analogous to those for MQTT, but the ramifications of a poor deployment are more severe. Since some of the size and speed advantages of CoAP are negated by adding DTLS security, some publicly exposed infrastructures have neglected to require it, resulting in thousands of unsecured devices on the internet. Due to the differences between TCP and UDP, unsecured CoAP devices can be exploited to launch DDoS attacks with amplification factors of up to 51,000x.[2]

However, the problem with MQTT is not the protocol, but the fact that many MQTT networks are either misconfigured for security or operate with no security. Particularly in smart home deployments or IoT networks in small organizations, it often falls to the customer or IoT vendor to implement and properly configure security mechanisms to make MQTT communication protected.

Misconfigured MQTT servers without passwords are publicly discoverable on the internet, which allows hackers to infiltrate any smart home or business associated with that server. In addition, because of the one-to-many subscription architecture of MQTT, gaining access to the MQTT server means getting access to all the data crossing the network.
Cybercriminals can easily exploit these configuration flaws and vulnerabilities to conduct reconnaissance, covert data theft and DDoS attacks.

Securing Connected Devices in the Hostile World

Ensuring security for connected designs can be challenging and time-consuming even for experienced developers, with multiple risks and malicious actors awaiting the unprepared. To deliver comprehensive, in-depth security protection for products based on embedded devices requires multiple protocols and safeguards that work together to provide security at many levels.

Renesas has been a leader in embedded security for decades and is well positioned to address the heightened need for security in today's connected products. Renesas offers multiple approaches to embedded security, providing a multi-tiered development infrastructure that provides comprehensive security protection for a wide variety of embedded products.

The Renesas Synergy™ Platform is a comprehensive, qualified development platform that includes production-grade software in the form of the Synergy Software Package (SSP), and a scalable family of pin-compatible MCUs, pre-integrated and pre-tested to provide security at multiple levels. The Synergy platform ensures that IoT applications are built on a secure, robust technology foundation.

In addition, the new Renesas RA family of MCUs delivers an option with more platform flexibility, combining Arm Cortex-M cores and best-in-class embedded system peripheral IP from Renesas. The RA's Flexible Software Package (FSP) provides optimized HAL drivers as well as a baseline software platform built on FreeRTOS and associated middleware. Designed for flexibility, developers can easily incorporate their middleware and libraries of choice.

The Synergy platform and RA MCUs both contain an integrated crypto subsystem called the Secure Crypto Engine (SCE). The SCE provides hardware acceleration for the most prevalently used cryptographic algorithms as well as key generation and a True Random Number Generator (TRNG). Both the Synergy platform and the RA FSP support public key infrastructure (PKI).

Developers also need to ensure that their development platform makes it safe and easy to connect to the cloud. The Synergy SSP delivers support for cloud connectivity with built-in MQTT and TLS modules, and the Synergy cloud connectivity applications provide secure, built-in connectivity to leading cloud environments, including Amazon Web Services (AWS), Google Cloud, and Microsoft Azure. The RA FSP provides similar functionality, leveraging Arm ecosystem software.

Conclusion

Delivering comprehensive, in-depth security protection for connected applications requires integrated technologies that work together to provide defenses at multiple levels. Renesas helps embedded developers meet the challenges of securing connected devices by offering numerous ways to secure devices, services and networks at a deep level.

Renesas offers multiple MCUs[3] that address the concerns discussed in this white paper. Please visit our website to learn more.

References

[1] Hacked Cameras Were Behind Friday's Massive Web Outage

[2] In-the-Wild DDoSes Use New Way to Achieve Unthinkable Sizes

[3] RA Family of 32-bit Arm Cortex-M MCUs
RX Family of 32-bit MCUs
Synergy Platform of 32-bit Arm Cortex-M MCUs + qualified software