Internet of Things

Embedded Security Guidance

May 2023

Center for Internet Security®

Abstract

The Internet of Things (IoT) is constantly evolving, and it continues to become more integrated into societies and industries around the world. With the expanding collection of diverse technologies and devices, properly securing these technologies has become a pressing challenge that needs to be addressed. This publication streamlines the selection and hardening process for IoT vendors by establishing a framework for protocol standards across constrained devices to provide built-in security on these technologies during the design, development, and manufacturing processes.

Acknowledgements

The authors acknowledge and appreciate the key contributions from individuals and organizations that have improved the thoroughness and quality of this document.

Special acknowledgements to Göran Selander from Ericsson for his review on the EDHOC and CoAP protocols as well as Hannes Tschofenig from Arm for his extensive review of the protocols and security recommendations. Additional acknowledgements to Nancy Cam-Winget from Cisco and Brian Haberman from Innovations Lab for their review of the various IoT protocols and aspects of embedded security, Bret Jordan at Afero for editorial review and technical content suggestions, Rui Ferreira Costa from the United Nations International Computing Centre (UNICC) for his review of the Wi-Fi protocol, Eric Spoerl (CIS) for his contributions and review of the key management content, Bret Jordan at Afero, and Stephanie Polczynski at the U.S. Department of Defense.

Additional thanks to Phil White (CIS), Angelo Marcotullio (CIS), John Gilligan (CIS), and Brian Contos, Joe Costantini, and Trent Altman from Phosporous for their reviews and contributions.

Contents

Introduction

The Internet of Things (IoT) is a rapidly growing and expanding market that is projected to reach more than 27 billion connected devices by 2025 (Sinha, 2021). As these devices become more integrated into networks across numerous industries around the world and fundamentally ingrained into the daily lives of every citizen, it is important to understand how these devices can be properly secured. The release of President Joe Biden's "National Security Memorandum on Improving Cybersecurity for Critical Infrastructure Control Systems" on July 28, 2021, highlights the importance of addressing built-in security for IoT. From this memorandum and Executive Order (EO) 14028, NIST is working with the Federal Trade Commission (FTC) and other agencies "to initiate pilot programs informed by existing consumer product labeling programs to educate the public on the security capabilities of IoT devices and software development practices" (NIST, 2021).

Numerous efforts have been initiated that attempt to address IoT security from an enterprise perspective, however, to achieve built-in security, resources, guidance, and materials must be provided to aid IoT vendors. One of the key areas that needs to be secured is the protocol that will be used by all of these devices. To address this the Center for Internet Security® (CIS®) has identified and analyzed the most commonly used IoT protocols from a security standpoint and produced informational guidance and recommendations for their use..

Through the analysis and recommendations found in this document, CIS aims to provide the guidance needed to ensure the communication protocols used in IoT have built-in security. This publication streamlines the selection and hardening process for protocol selection to enable built-in security at the protocol level, which will be vital to the security of systems around the world.

To assist IoT vendors in selecting secure protocols several classifications have been developed to illustrate the constraint a device has in terms of memory, input/output bandwidth or capacity, and battery life. Understanding the level of constraint may help in determining the complexity of the feature set in a device, the set of protocols that are feasible to operate, the level of security that may be built in, and how the device might communicate to other devices, a gateway, or to the cloud.

IoT Introduction, Classification, and Architecture

IoT devices can be classified by device type, functionality, type of data handled, and their degree of constraint. Knowledge of how devices can be classified is vital for considering how to apply built-in security and which application layer and encryption protocols can be supported. Security on IoT devices is not a distinct layer but rather has to be appropriately implemented across all layers of the protocol stack. Broadly, IoT devices are categorized based on their ability to communicate. The first class, based on level of constraint, consists of devices that do not communicate directly with the server but rather transmit to the server through a gateway or hub. These devices, similar to those depicted on the right side of Figure 1, often use protocols like ZigBee, Bluetooth Low Energy (BLE), Zwave, or Thread as they are often limited by or constrained by battery power. The second class of devices has the ability to communicate directly with central or cloud servers for data storage and processing. These devices also support protocols such as Internet Protocol version 4 (IPv4) or 6 (IPv6). These devices are not typically constrained by battery power as they have constant mains power.

IoT devices are also classified as either constrained devices or unconstrained devices based on their hardware capabilities. Small devices with limited CPU, memory, and power would fall under "constrained devices" and are categorized into Class 0, Class 1, and Class 2 devices (Borman et al., RFC7228, 2014). Table 1 provides further descriptions.

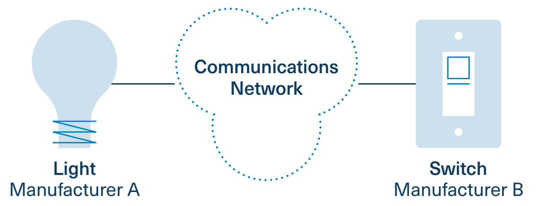
Table 1. Classes of Constrained Devices (Borman et al., RFC7228, 2014)

Classes	Ram	Flash
Class 0, C0	< 10KB	< 100KB
Class 1, C1	Approx. 10 KB	Approx. 100KB
Class 2, C2	Approx. 50KB	Approx. 250KB

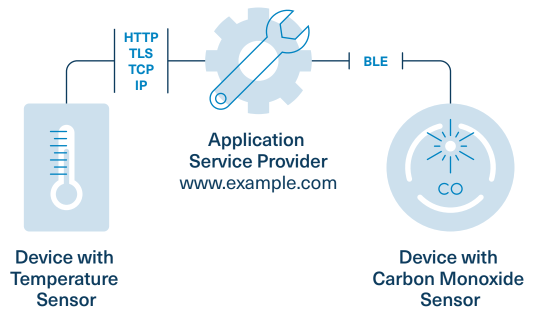
The classification level of a device and the selected protocols used by the device aid in determining architectural requirements that need to be considered. Depending on the selected protocol stack, a device may be capable of connecting directly to the internet, or it could require a gateway to transition from one protocol stack to another. In either case, the guidance provided aims to minimize the attack surface for the device, identify recommendations on how to secure the protocol stack, recommend secure coding practices, identify encryption protocols and key management techniques that could be used, and assist with access controls.

Gateway or hub devices play an important role in IoT and are used to convert transmissions from one protocol to another, for example BLE to WiFi. The gateway function may or may not require standalone device; with more sophisticated IoT devices, it can be a function provided in the device itself. It is also possible to maintain end-to-end encryption through a gateway or proxy when encryption is performed at the application level such as is possible using OSCORE [RFC 8613]. Even though a gateway may terminate transport level encryption capabilities, the object level can remain secure depending on the protocol design. An IoT architecture is the end result of these steps. Figure 1 below represents the three major patterns of connectivity for IoT devices as described in RFC 7452. The first is device-to- device communication, which is the simplest pattern out of the three (i.e., a smart light bulb connected to a light switch through a wireless network). The second communication pattern is device-to-cloud communication, which connects two or more devices or sensors though a cloud service. This communication is typically transported through the protocols that will be described in this document. The final pattern that is depicted in Figure 1 is a backend data sharing pattern. This allows for data to be analyzed from other sources.

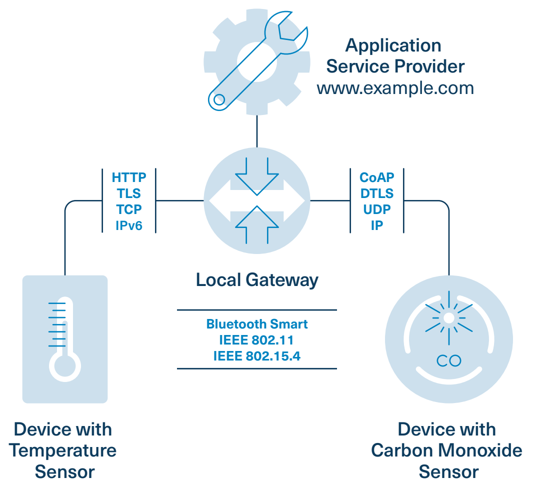
Figure 1. IoT Communication Patterns and Data Sharing Pattern (RFC 7452)



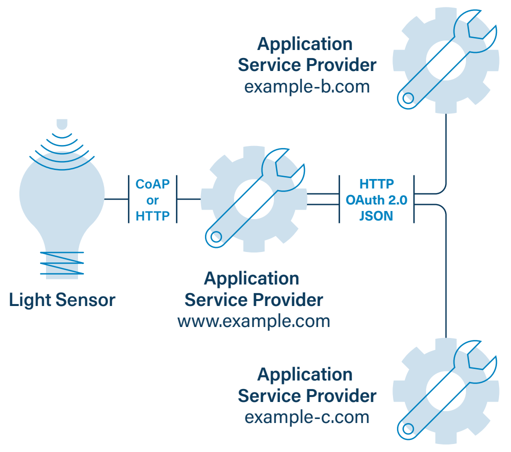
Device-to-device communication pattern


HTTP
TLS
TCP
IP
BLE
Application Service Provider
www.example.com
Device with Temperature Sensor
CO
Device with Carbon Monoxide Sensor

Device-to-cloud communication pattern



Device-to-gateway communication pattern


Application Service Provider
example-b.com
CoAP
or
HTTP
HTTP
OAuth 2.0
JSON
Light Sensor
Application Service Provider
www.example.com
Application Service Provider
example-c.com

Back-end data sharing pattern

Terminology for Constrained Devices in IoT

In the IoT landscape, devices may connect to each other via one of three main network architectures, or topologies. These topologies are described as a point-to-point network (a motion sensor talking to a light bulb), a star or hub-and-spoke network (IoT devices talking to a central control point or cloud service), and a mesh network (IoT devices the relay traffic through the nearest peer, e.g., ZWave or Zigbee) (see topology descriptions here). Prior to identifying the best network topology for a given product, the capabilities, functionality, and available resources of the device must first be determined. Following that, any connectivity and deployment consideration along with the security and privacy requirements must be addressed.

Constrained Device: An IoT device that is limited by the amount of computational power, RAM, flash, or power that it has often due to cost constraints and/or physical constraints such as size and weight In addition some networking level features like multicast or IPv6 may be lacking. Due to the extremely tight tolerances of these devices, care should be given in selecting functional requirements, bandwidth and connectivity requirements, security and privacy requirements, along with things like encryption.

Constrained Network: An IoT network architecture for which the features of a classical LAN or Internet connection are not attainable due to insufficient resources of some type.

Table 2 categorizes the level of constraint for devices into classes to aid in the analysis and decision points when selecting protocol stacks as well as appropriate security options. Protocol stacks examined in the Common IoT Stacks Section will reference the class level of device to determine if a feature would be suitable.

Table 2. Range of constraints on IoT Constrained Node Networks (HCI Tech, 2020 & NISTIR 8200)

Class	Protocol Stack	Security
Class 0	Not designed to support	Not designed to support
Class 1	Designed for constrained nodes	Authentication, confidentiality, and encryption
Class 2	Designed to have more capability to run network protocols and applications	Authentication, confidentiality, and encryption

Commonly Deployed IoT Protocols

IoT security must be looked at from a comprehensive, wide-ranging perspective. IoT devices, especially constrained ones, consist of a plethora of different products, sensors, and technologies that require numerous protocols for ensuring strong functionality throughout their lifecycle. IoT devices communicate using these protocols; to effectively understand how devices connect, it is imperative to first understand the device's IoT protocols. These protocols are configured in protocol stacks, including physical, link, network, transport, session, presentation, and application layers.

Therefore, it is important to note that a major facet of determining the appropriate protocol stack for a device is to first establish how constrained the device is by determining what class it falls into and the connectivity requirements. However, is it important to stipulate that cost will be another primary consideration. Protocols that are capable of operating at high speed and sending high quantities of data may require additional hardware resources that can increase the cost compared to other protocols. The classes specified stipulate security solutions for technical components based upon their capabilities for implementing security. They also help to "define different procedures of managing security and to stipulate compensating measures" (Zero Outage, 2020). Listed below in alphabetical order are application layer and network protocols most often employed today.

Application Layer Protocols

- AMQP (Advanced Message Queuing Protocol)
- CoAP (Constrained Application Protocol)
- DDS (Data Distribution Service)
- HTTP (Hypertext Transfer Protocol)

- MQTT (Message Queuing Telemetry Transport)
- QUIC (Quick UDP Internet Connections)
- WebSocket
- XMPP (Extensible Messaging and Presence Protocol)

Network Protocols

- Bluetooth
- LoRaWAN (Long Range Wide Area Networks) /Low Power Wide Area (LPWA)
- Wi-Fi
- Zigbee
- Z-Wave

Most Common Protocols Expanded

The IoT protocols mentioned above are some of the most common used in the industry. It is important to know the key functions of the protocols as well as the industry and devices they support. Below is a more detailed breakdown of the most common protocols.

Application Layer Protocols

AMQP

The Advanced Message Queuing Protocol (AMQP) is an open standard messaging protocol ideal for supporting business processes. AMQP passes business messages between applications or organizations and has been successfully implemented in “telecommunications, defense, manufacturing, internet and cloud computing, and many additional market segments” (OASIS, n.d.). Compared to other message protocols, AMQP has a larger packet size. It also has a header size of 8 bytes, which makes it not as suitable for more constrained devices. AMQP is built on top of TCP, and security is provided through TLS.

CoAP

Constrained Application Protocol (CoAP) was created by the Constrained Resource Environments (CORE) IETF group (IETF, n.d.). CoAP “provides a request/response interaction model between application endpoints” that tracks each message until it is delivered to its intended receiver (Shelby, et al., 2014). Specialized for use with constrained networks, CoAP packets are small and simple to generate. As such, they do not consume extra RAM on constrained devices. This protocol is designed for machine-2-machine (M2M) applications, which are typically seen in the banking industry, smart energy, and building automation. There are various ways to build the CoAP architecture. CoAP can be built on top of UDP with security provided through DTLS or on top of TCP with security through TLS. Alternatively, security can be provided by EDHOC and OSCORE independently of transport, including UDP and TCP.

DDS

Compared to CoAP, MQTT, and AMQP, the Data Distribution Service (DDS) M2M standard protocol is more versatile. DDS "can manage tiny devices, connect large, high-performance sensor networks, and close time-critical control loops," (Object Management Group, Inc., n.d.) while also serving and receiving data from the cloud. Used by many industries, including transportation, smart energy, medical devices, industrial automation, simulation and test, smart cities, military, and aerospace, DDS "minimizes latency, maximizes scalability, increases reliability, and reduces cost and complexity" (Object Management Group, Inc., n.d.). DDS can be built on top of TCP or UDP, and security can be provided through TLS, DTLS, or DDS security. Security for DDS is not as reliable when messages are sent via IP multicast.

HTTP

The Hypertext Transfer Protocol (HTTP) is an application layer protocol that transfers files such as text, video, images, sound, and other media files over the internet. Although it is the foundation protocol for data communication over the web, HTTP is not always the preferred protocol because of its cost, battery life, energy savings, and other constraints (Uppalapati, 2019). HTTP can be built on top of TCP or UDP, and transport security can be provided through TLS or QUIC depending on the HTTP version. HTTP over TLS is known as HTTPS. Where HTTP falls short is in its operational costs and energy-saving capabilities. It can handle larger packet sizes, which can cause issues when implemented in constrained devices that need a protocol with smaller packet sizes.

MQTT

Similar to CoAP, Message Queuing Telemetry Transport (MQTT) is an open standard messaging protocol designed for lightweight M2M communications (Jaffey, 2014). MQTT was created to operate with low-bandwidth, high-latency, or unreliable networks, and it ensures reliability while minimizing network bandwidth, making it an optimal protocol for constrained IoT devices. Allowing device-to-cloud and cloud-to-device messaging, MQTT is suitable for a variety of industries, including automotive, logistics, manufacturing, smart home, oil and gas, consumer products, and transportation. MQTT is built on top of TCP and TLS for security. For devices on non-TCP/IP networks, MQTT-SN is a viable option, as it uses UDP as a transport protocol. “MQTT-SN is a publish/subscribe messaging protocol for wireless sensor networks (WSN), with the aim of extending the MQTT protocol beyond the reach of TCP/IP infrastructure for Sensor and Actuator solutions” (MQTT, n.d.).

QUIC

Quick UDP Internet Connections (QUIC) is a secure end-to-end UDP-based transport protocol that “integrates the TLS handshake although using a customized framing for protecting packets” (Iyengar & Thomson, RFC9000, 2021). QUIC allows for low-latency connection establishment, network path migration, and structured communication through flow-controlled streams. While authenticating each packet, QUIC also encrypts as much of the packet as practical (Iyengar & Thomson, RFC9000, 2021).

WebSocket

The WebSocket protocol is a thin, lightweight layer that enables applications to handle two-way communications. WebSocket “provides a mechanism for browser-based applications that need two-way communication with servers that do not rely on opening multiple HTTP connections” (Fette & Melnikov, RFC6455, 2011). This makes it a more attractive protocol for things like gaming applications, messaging apps, and cases where you need near real-time updates in both directions (i.e., stock tickers). WebSocket is built on top of TCP with transport security provided through TLS.

XMPP

The Extensible Messaging and Presence Protocol (XMPP) is an open IETF standard messaging protocol. XMPP is based on XML language. It allows for a near-real time exchange of data between two or more networks, making it a useful protocol for online gaming, news websites, and Voice over Internet Protocol (VoIP) (Uppalapati, 2019). With XMPP being an open protocol, it is highly scalable for consumer-oriented IoT deployments. Although it has many advantages like scalability, XMPP also has drawbacks. It does not offer Quality of Service or end-to-end encryption, making it not suitable for embedded IoT applications. XMPP is built on TCP, with transport security provided through TLS.

Table 3 provides an abbreviated comparison chart between the application layer protocols described in this section to aid in the selection for constrained devices.

Table 3. Application Layer Protocol Chart

D2D: device to device
D2C: device to cloud
C2C: cloud to cloud

* The level of cost (low, medium, high) was determined by taking into consideration the power draw, the weight, the data rate, the bandwidth of the protocol, and how taxing it will be on infrastructure. This assessment of operational cost will be made more precise with the aid of industry experts.

Application Protocols	Transport Layer	Security Layer	Scope	Operational Cost*	Main Drawbacks
AMQP	TCP	TLS	Device-to-device (D2D), Device-to-cloud (D2C), Cloud-to-cloud (C2C)	Medium	Larger packet size
CoAP	TCP or UDP	TLS, DTLS or EDHOC and OSCORE	D2C, D2D	Low	Message unreliability
DDS	TCP or UDP	TLS, DTLS, or DDS Security	D2D, D2C, C2C	Medium	Heavyweight, High Bandwidth
HTTP	TCP or UDP	TLS or QUIC	D2C	High	Cost, energy saving, battery-life
MQTT	TCP	TLS	D2C	Low	Latency, memory/ power requirements
WebSocket	TCP	TLS	D2C	Medium	Heavyweight
XMPP	TCP	TLS	D2D, D2C	Medium	XML-based protocol, heavy data overhead

Network Layer Protocols

Bluetooth/BLE

Bluetooth and Bluetooth Low Energy (BLE) are standardized protocols that allow a device to send and receive data over a 2.4GHz frequency band. They are useful for products that "stream high-quality audio between a smartphone and speaker, transfers data between a tablet and medical device, or sends messages between thousands of nodes in a building automation solution" (Karr, n.d.). BLE is used with application where the battery is constrained. To overcome the risk of passive eavesdropping, BLE encrypts the data being transferred using AES-CCM cryptography. The BLE protocol also "includes a privacy mode which uses random address to help anonymity," but this isn't necessarily enough for security (Chinnick, 2018). Bluetooth and BLE can meet the unique needs of an array of devices. Bluetooth and BLE offer short-range, low-power wireless transmissions between devices at a low cost.

LoRaWAN

Developed by the LoRa Alliance, LoRaWAN is a Low-Power, Wide=Area (LPWA) network protocol that can wirelessly connect battery operated devices to the internet and that targets "bi-directional communication, end-to-end security, mobility and localization services" (LoRa Alliance, 2021). LoRaWAN is a low-power, low-cost architecture that can support M2M, smart city/home, and industrial applications that can be scaled to a global level to handle a network with billions of devices. LoRaWAN utilizes AES algorithms for security to provide authentication and end-to-end encryption. It is typically used alongside TCP/IP.

Wi-Fi

The most common Wi-Fi bands are 2.4 GHz, 5 GHz, and 6 GHz to send and receive wireless signals to devices. Typically associated with routers that are able to transfer an "internet connection from a public network to a private home or office network," (AVSystem, 2020) Wi-Fi allows smartphones, computers, and other devices to connect to the internet. Security for a private Wi-Fi network will be in the form of Wired Equivalent Privacy (WEP), Wi-Fi Protected Access (WPA), or Wi-Fi Protected Access 2nd generation (WPA2). The standard Wi-Fi connection is typically 100 meters, but the most common range in households is 10-35 meters.

Zigbee

Zigbee is a complete IoT solution that allows Zigbee certified devices and products to connect and communicate. Similar to Bluetooth, Zigbee sends and receives data over a 2.4 GHz frequency band. Zigbee can support up to 65,000 devices, or nodes, which makes it able to support a variety of commercial industries including office, hospitality, medical, education, retail, and manufacturing (Connectivity Standards Alliance, 2021). Also utilized by multiple smart home ecosystem providers like Signify's Philips Hue, Amazon's Echo Plus, and others, Zigbee is a proven, full-stack solution in the smart home industry. Zigbee provides developers with a flexible, low-power option with standardization throughout all layers of stack, all while maintaining a low data rate. Zigbee uses the AES-128 standard for encryption to ensure a secure connection.

Z-Wave

Like Zigbee, Z-Wave is a complete IoT solution that's structured to support both commercial and residential smart buildings. Compared to Zigbee, a benefit of Z-Wave is that its devices and applications operate on the 800-900 MHz range of radio frequencies. This reduces the chance of interference (Mears, 2019). Z- Wave is limited to only 232 devices, but even with this limitation, Z-Wave provides a very reliable network for devices to communicate in smart homes that include smart sensors, smart locks, smart lights, smart hubs, and smart thermostats. Z-Wave uses the AES-128 standard for encryption to ensure a secure connection.

Table 4 provides an abbreviated comparison chart between the network layer protocols described in this section to aid in the selection for constrained devices.

Table 4. Network Layer Protocol Chart

Network Protocols	Security	Operational Cost*	Frequency	Range	Data Rate	Power Draw
BLE	Encryption recommended	Low	2.4GHz	200 ft.	10 kB/s	Low
LoRaWAN	Encrypted	High	150MHz- 1GHz	up to 20 miles	50 kbps	Low
Wi-Fi	Encryption recommended	Medium	2.4GHz, 5GHz, 6GHz	115-230 ft.	10 Gbps	High
Zigbee	Encrypted	Medium	2.4GHz, 915MHz (US)	100-325 ft.	250 kbps (2.4), 40 kbps (915)	Low
Z-Wave	Encrypted	Medium	915MHz (US)	100-325ft.	100 kbps	Low

* The level of cost (low, medium, high) was determined by taking into consideration the power draw, the weight, the data rate, the bandwidth of the protocol, and how taxing it will be on infrastructure. This assessment of operational cost will be made more precise with the aid of industry experts.

Transport Security Protocols

Protocol	Full Name	Description	Category	Supporting / Interoperable Protocols	Class Level Support
TLS	Transport Layer Security	Transport encryption protocol for applications providing integrity, confidentiality, and authentication for communications and data sent over a network. RFC8446 The IETF TLS working group is developing cTLS to reduce overhead for IoT devices.	Transport Encryption	TCP, MQTT, HTTP, AMQP, XMPP	Class 1+
DTLS	Datagram Transport Layer Security	Transport encryption protocol used to secure datagram-based (UDP) communications, based on TLS, providing similar security properties. RFC9147	Transport Encryption	UDP, CoAP	Class 1+
EDHOC	Ephemeral Diffie-Hellman Over COSE	Lightweight authenticated key exchange protocol providing forward secrecy, identity protection, and cipher suite negotiation. For use in constrained environments.	Transport Encryption	UDP, OSCORE, CBOR, COSE, CoAP	Class 0+
OSCORE	Object Security for Constrained RESTful Environments	Provides object level encryption, supporting end-to-end security. RFC8613	Object Security	EDHOC, CoAP, HTTP, CBOR, COSE	Class 0+
EST	Enrollment over Secure Transport	Certificate enrollment protocol over TLS for PKI clients obtaining certificates from certificate authorities (CA). EST plays a similar role to SCEP and obsoletes SCEP. RFC7030	Certificate Management Protocol	PKI, TLS, DTLS, CMS, CMP, SCEP	Class 1+
CMP	Certificate Management Protocol	Transport protocol used to manage any digital certificates within public infrastructure keys. Messages are self contained and therefore independent of the transport protocol. Used in enterprise settings, based on ASN.1. ACME is an alternative with additional use cases. RFC6712	Certificate Management Protocol	PKI, HTTP, TCP, CMS	Class 1+
ACME	Automatic Certificate Management Environment	Protocol that enables automation of key management, including issuance and revocation. Based on JSON and intended for Internet and enterprise use cases. JSON Object Signature and Encryption (JOSE) is used for the message syntax security functions. RFC8555	Certificate Management Protocol	PKI, TLS, EST, BRSKI, TEAP, HTTP	Class 1+

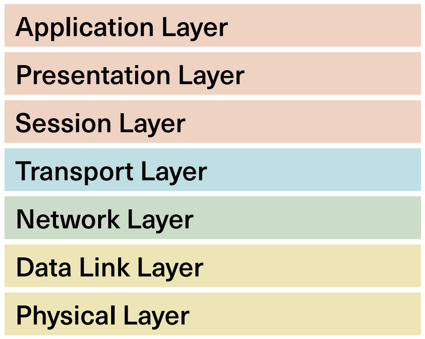
CMS	Cryptographic Message Syntax	CMS is an encapsulation syntax to digitally sign, create a message digest, authenticate, or encrypt message content. Multiple parties may sign a message. CMS is used in CMP and other protocols. RFC5652	Certificate Management Protocol	PKI, HTTP, TCP, IPsec, CMP	Class 1+
802.1x Authentication	802.1x Authentication	Framework used to segment network access control. IEEE 802.1X	Authentication	EAP, RADIUS, PPP, BRSKI, TLS, LEAP	Class 1+
802.1AR	802.1AR	Standard leveraged for authorization and authentication of devices. IEEE Std 802.1AR-2018	Device Identity	EAP, PPP, BRSKI	Class 1+
BRSKI	Bootstrapping Remote Secure Key Infrastructure	Secures connection between a new device and network operation center. RFC8995	Bootstrapping Authentication	TLS, DTLS, EST, PKI, HTTP, CoAP, ACME, SCEP, EAP, TEAP	Class 2+
TEAP	Tunnel Extensible Authentication Protocol	Standardized secure tunnel-based EAP method, and executes other EAP methods within that tunnel. TLS handshake provides authenticated key exchange and establishes a protected tunnel. RFC7170	Authentication	EAP, TLS, OCSP, BRSKI, PPP, RADIUS, Diameter	Class 1+
EAP	Extensible Authentication Protocol	Provides basic response -> request protocol framework for authentication and runs directly over link layer protocols without requiring IP. EAP may be used over wired or wireless and supports multiple authentication methods. RFC3748	Authentication	TLS, DTLS, PPP, IEEE 802, RADIUS, Diameter, BRSKI	Class 1+
RADIUS	Remote Authentication Dial-In User Service	RADIUS "carries authentication, authorization, and configuration information between a network access server which desires to authenticate its links and a shared authentication server." RFC2865	Authentication Framework	TLS, DTLS, PPP, PAP, CHAP, EAP, IEEE 802.1x, UDP	Class 1+
Diameter	Diameter	An authentication, authorization and accounting framework, evolved from RADIUS, to support applications such as network access or IP mobility. RFC6733	Authentication Framework	IPsec, TLS, TCP, DTLS, EAP	Class 1+
SASL	Simple Authentication and Security Layer	A framework for authentication and data security from application to network protocols. RFC4422	Authentication Framework	TLS, XMPP, AMQP	Class 1+

Application Layer Protocols

Common IoT Stacks

The IoT protocol stacks listed to the left are the most commonly seen deployed in the industry. Even so, they are not the only possibilities. The use of these stacks will depend on what service the device is trying to achieve and the communication requirements. Under each protocol stack, recommendations are given as to when the protocol may be best suited along with additional considerations that could make the embedded security more effective. These considerations include encryption, authentication, access control, and secure coding practices, which should all be utilized when working to create the most secure IoT device. Recommendations that apply more generally are in the section that follows the Common IoT Stacks. Additional considerations can be found in the Appendix.

Figure 2. OSI Model Stack



AMQP Protocol Stack and Implementation

Overview

AMQP is an application layer protocol that encapsulates process-to-process communication across IP networks. More simplified, it is a source messaging platform that is lightweight, scalable, and non-reliant on resources like batteries.

Moreover, AMQP is a messaging queue for more “advanced” scenarios. An AMQP-based message queue like RabbitMQ, for example, can be configured to transfer messages according to more complex processes and not simply the standard ones.

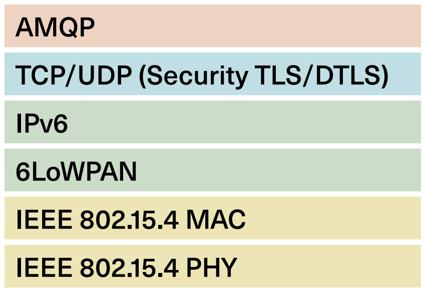


Figure 3. AMQP and Supporting Protocol Stack

Security Recommendations and Constraints Considerations

Security Consideration

AQMP is a reliable and secure protocol that’s most always stacked on the TCP/IP transport layer. Because it is primarily responsible and most functional for network flow control and flow control of messages, it is usually built over IPv6 and 6LoWPAN to optimize transferability in messaging between devices. Security/encryption can be provided through both TLS and DTLS. The TLS Cipher Suites can provide the proper cryptographic algorithm for the desired level of security.

Authentication

SASL (Simple Authentication and Security Layer) (Melnikov & Zeilenga, RFC4422, 2006): SASL provides a structured interface between protocols so as to provide authentication and data security services in “connection-oriented protocols” through replaceable mechanisms.

When to Use

- Applications that need support from protocols such as STOMP or MQTT.
- Supports Class 1 devices and above.

Secure Coding Practices

See Secure Coding Practices section for coding information and resources.

CoAP Protocol Stack and Implementation

Overview

CoAP is a battery-friendly open standard web transfer protocol that's designed for M2M applications with constrained nodes and constrained networks. CoAP is often built on top of the constrained network 6LoWPAN, which also supports the fragmentation of IPv6 packets into small link-layer frames (Shelby, et al., 2014). CoAP provides RESTful programming where clients can send GET, PUT, POST, and DELETE resource requests. This is similar to HTTP+TLS, but CoAP is a smaller protocol. The size of this protocol makes it ideal for devices with less than 10 Kib of RAM.

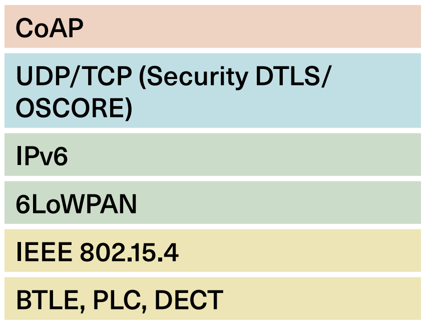


Figure 4. CoAP and Supporting Protocol Stack

Security Recommendations and Constraints Considerations

Security Consideration

CoAP can utilize two different communication security protocols, Datagram Transport Layer Security (DTLS) and Object Security for Constrained RESTful Environments (OSCORE). The latter may be used with handshake protocol Ephemeral Diffie-Hellman over COSE (EDHOC) for public key-based authentication. The more common of the two is DTLS, which is similar to TLS but can secure unreliable datagram traffic. An alternative solution, applicable to OSCORE/EDHOC as well as DTLS, is to instead use the CoAP Echo option [RFC 9175].

The TLS Cipher Suites, which is managed by IANA, lays out what algorithms are specified for use with DTLS and which cipher suites have been recommended through an IETF consensus process.

EDHOC is a lightweight key exchange protocol that can be used in place of DTLS. Suitable for low-power wide area networks, EDHOC reduces message sizes and is known to be able to establish a handshake connection with one-sixth the number of bytes compared to DTLS 1.3 when Raw Public Keys (RPK) authentication is used (Selander, et al., 2021).

The IANA registry for EDHOC Cipher Suites has not yet been published, but initial contents of the registry can be found in the EDHOC Datatracker. EDHOC can provide authentication on dedicated networks and then be redirected through HTTPS, TLS, DTLS, or a third-party platform to facilitate communication with devices not behind the network firewall.

Object-level encryption can be provided via Object Security of Constrained RESTful Environments (OSCORE). Working in very constrained nodes and networks, OSCORE uses CBOR Object Signing and Encryption (COSE) to provide end-to-end protection of data, providing a higher degree of protection than other transport encryption protocols that may expose data at termination points for encryption if proxy connections are in place. OSCORE protects "CoAP and CoAP-mappable HTTP requests and responses end to end across intermediary nodes such as CoAP forward proxies and cross-protocol translators including HTTP-to-CoAP proxies" (Selander, et al., RFC8613, 2019).

Authentication

When implementing CoAP with OSCORE, there are three options for authentication:

1. **PreSharedKey:** A list of pre-shared keys and associated identifiers describes the endpoints with which the node can communicate.
2. **RawPublicKey:** The device has an asymmetric key pair without a signature (a raw public key) that is validated using an out-of-band mechanism. Ituses EDHOC to establish shared secret keys and identifiers to use with OSCORE.

3. **Certificate:** The device has an asymmetric key pair with a certificate (X.509 [RFC 5280], CBOR encoded X.509, CWT [RFC 8392], etc.) that binds it to its subject and is signed by some common trust root. Ituses EDHOC to establish shared secret keys and identifiers to use with OSCORE.

EDHOC supports a mix of different authentication mechanisms. For example, the client may authenticate with RawPublicKey and the server with Certificate.

When to Use

CoAP is designed to meet the needs for constrained devices, which makes a protocol that can be used for Class 1 or above devices.

The smaller packet sizes can establish faster communication for smart home devices (i.e., alarms, thermostats, smart locks) as well as industrial microcontrollers where direct interaction is desired.

CoAP is suitable for devices on a single network and for internet-operative applications that use connected devices/sensors and have resource limitations

Secure Coding Practices

Coding practices depend on how CoAP is being implemented and the type of device that is being used. The implementations can be found here: https://coap.technology/impls.html.

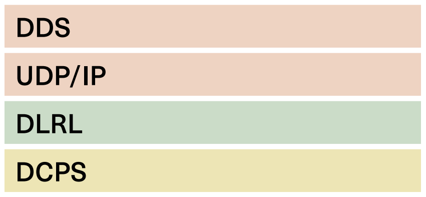
See the Secure Coding Practices section for addition information and resources.

DDS Protocol Stack and Implementation

Overview

DDS is optimized for distributed processing and directly connecting sensors, devices, and applications to each other without any dependence on centralized IT infrastructure. Any DDS device can exchange data with any other DDS device. Data can be exchanged between publishers and subscribers via a unicast, multicast, or broadcast, meaning data from one device can easily reach thousands of other devices. DDS allows networks to scale automatically. Thousands of devices can join a network, auto-configure, and then publish or subscribe to data topics, "or basic unit of information that the DDS system reads and writes" (Aldin Tech, n.d.). While DDS can manage networks with thousands of devices, those devices do need to be on the same local networks unless DDS gateways are deployed. Very often, Industrial Internet of Things (IIOT) network scenarios are dynamic, meaning thousands of devices can leave a network, too. DDS ensures that large numbers of devices can join or leave a network automatically.

Figure 5. DDS and Supporting Protocol Stack



Security Recommendations and Constraints Considerations

Security Consideration

DDS is typically built over HTTP, UDP/IP, DCPS, and DLRL. Its strongest use case is over DCPS (Data Centric Publish Subscribe) given its strong level of encryption and security in delivering information to subscribers. DCPS is a fundamental API for the data-focused layer. A DLRL (Data Local Reconstruction Layer) provides an interface to DCPS functionalities.

For additional security, DDS can also be configured to use TLS for encryption.

Authentication

The authentication plugin implemented in Fast DDS is referred to as DDS:Auth:PKI-DH

- Uses a trusted Certificate Authority (CA) and ECDSA Digital Signature Algorithms to perform "mutual authentication" (eProsima, 2019)
- Uses Elliptic Curve Diffie-Hellman (ECDH) to derive a shared key.

When to Use

DDS is highly configurable, which allows for its use in a large number of scenarios, especially in managing small or micro devices.

With DDS being based on IP Multicast, there are scalability limits that make it difficult to securely implement in large-scale infrastructures because of the known scalability limits of IP Multicast over the internet.

Secure Coding Practices

See Secure Coding Practices section for coding information and resources.

HTTP Protocol Stack and Implementation

Overview

HTTP is rooted in data communication between browsers and servers and is used for transmitting hypermedia documents (HTML). HTTP is designed to be simple and readable, providing feasible testing for developers.

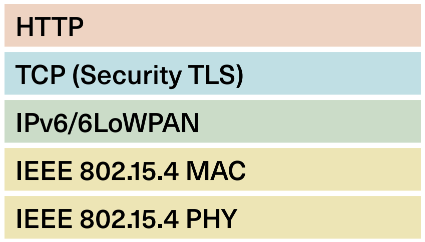


Figure 6. HTTP and Supporting Protocol Stack

Security Recommendations and Constraints Considerations

Security Consideration

Security and Encryption: HTTP security and encryption is provided through TLS.

IANA has listed the TLS Cipher Suites that should be used to secure devices, but vendors are required to regularly check the cryptographic algorithms listed to ensure the expected level of security, as these algorithms become weakened or broken over time.

Authentication

There are several authentication schemes for HTTP, including:

- The ‘Basic” HTTP authentication scheme is not recommended, as it is not secure (Reschke, RFC 7617, 2015)
- TLS Mutual Authentication (mTLS)
- HTTP Proxy-Authenticate as an authentication method to gain access to a resource behind a proxy server
- Character encoding of HTTP authentication

- WWW-Authenticate and Proxy-Authenticate headers (Fielding & Reschke, RFC7235, 2014)
- W3C WebAuthn (https://www.w3.org/TR/webauthn-2/)

mTLS and WebAuthn are consistently rated as the most secure, as they prevent replay attacks and are phishing-resistant due to their use of public/private key pairs to digitally sign challenge and response data.

When to Use

For Class 0 devices, HTTP may be too complex. It can be layered over UDP and TCP, though it could be costlier and less feasible over UDP than CoAP, for instance. The use of UDP is seen for HTTP/3, or HTTP over QUIC, which is discussed in RFC 9114. HTTP is primarily layered over TCP, which calls for more complex congestion control.

Secure Coding Practices

When adding redirects or links to a user-controlled URL, ensure that the scheme is HTTP or HTTPS.

Input:

- Ensure you are validating HTTP header and content,
- “Be sure to include automated post backs from Java, Flash, and any other embedded code.”

Output:

- Contextualize and sanitize all untrusted data.
- Use HTTP POST for transmit authentication credentials.
- See Secure Coding Practices section for addition information and resources.

MQTT Protocol Stack and Implementation

Overview

MQTT is a lightweight and battery-friendly, publish/subscribe messaging protocol for M2M communications. MQTT is highly scalable with its ability to allow device-to-cloud and cloud-to-device messaging, which can allow millions of IoT devices to be connected. Typically built over IPv6 and 6LoWPAN, MQTT can connect devices over unreliable, constrained networks.

To ensure reliability of message delivery, "MQTT has 3 defined quality of service (QoS) levels: 0 – at most once, 1 – at least once, 2 – exactly once" (MQTT.org, n.d.) to define how many times the message should be resent. With the publish/subscribe architecture on MQTT, a broker can also be used. "An MQTT broker, which is the heart of the MQTT Publish/Subscribe protocol, is a server that receives all messages from the MQTT clients and then routes the messages to the appropriate subscribing clients" (HiveMQ, n.d.) Although MQTT has minimal overhead, it is known to have issues with latency. If it is desired to build MQTT over UDP, MQTT for Sensor networks (MQTT-SN) can be implemented. MQTT-SN uses the same publish/subscribe model while reducing the size of message payload, and it removes the need for a permanent connection by using UDP (Cope, n.d.). The use of UDP would allow the MQTT model to use DTLS as the security protocol.

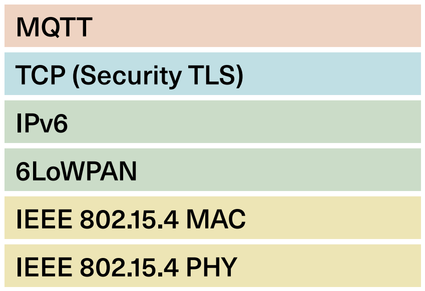


Figure 7. MQTT and Supporting Protocol Stack

Security Recommendations and Constraints Considerations

Security Consideration

Security for MQTT can be provided through TLS. TLS provides a secure communication between the client and the server for the MQTT interface. MQTT-SN security can be provided through DTLS.

- Reference the TLS Cipher Suites description mentioned in the “Security Protocols” section.

There is a potential confidentiality issue resulting from the publish/subscribe nature of MQTT. This happens when it is possible subscribe to a multi-level wildcard in order to receive all messages, which may be a confidentiality concern.

The use of an MQTT broker also contributes to the identity, authentication, and authorization. “The broker is responsible for persisting connections, as well as identifying and authorizing the transfer of data to MQTT clients” (DornerWorks, 2019).

Authentication

The Authentication and Authorization for Constrained Environments (ACE) Working Group states that proof-of-possession keys bound to OAuth2.0 (Hardt, RFC6749, 2012) access tokens are used to authenticate and authorize MQTT Clients when MQTT is built over TLS (Sengul & Kirby, 2021). TLS mutual authentication (mTLS) is the most common authentication protocol in MQTT, which uses the successfully validated client certificate to authenticate the client to the server.

Additional methods for authentication include:

- Client ids
- Usernames and passwords
- Client Certificates

When to Use

- Applications that need support from protocols such as the Simple (or Streaming) Text Orientated Messaging Protocol (STOMP) or MQTT
- Supports Class 0 and Class 1 devices (range of wireless devices)

- Text fields within the MQTT Control Packets are encoded as UTF-8 strings.

Secure Coding Practices

- The character data in a UTF-8 Encoded String MUST be well-formed UTF-8 as defined by the Unicode specification [Unicode] and restated in RFC 3629 [RFC3629].
- A UTF-8 Encoded String MUST NOT include an encoding of the null character U+0000.

Additional best practices can be found in the OASIS Standard mqtt-v5.0.

See Secure Coding Practices section for addition information and resources.

WebSocket Protocol Stack and Implementation

Overview

The WebSocket Protocol has emerged as a superior alternative to existing “bidirectional” communication technologies because of its strong integration of HTTP as a transport layer (Fette & Melnikov, RFC6455, 2011). The protocol primarily provides a system for browser applications to communicate with one other without having to rely on multiple HTTP connections (Fette & Melnikov, RFC6455, 2011).

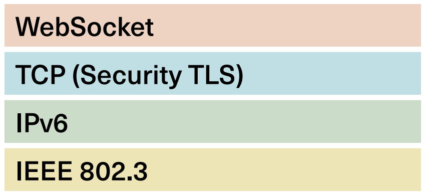


Figure 8. WebSocket and Supporting Protocol Stack

Security Recommendations and Constraints Considerations

Security Consideration

By default, the WebSocket Protocol is assigned to port 80 for regular WebSocket connections and port 443 for WebSocket connections over TLS (Rescorla, RFC2818, 2000). Endpoints or web infrastructures such as proxies are particularly vulnerable targets of attack via WebSockets.

- It is crucial to mask all data from the client to the server. This will ensure that the remote script (attacker) does not have control over how the data being sent appears on the wire and thus cannot construct a message that could be misinterpreted by an intermediary as an HTTP request.
- Confidentiality and general security are provided by running the WebSocket Protocol over TLS. WebSocket implementations must support TLS and should use it when communicating with other networks.

The benefit of TLS depends on the strength of the algorithms negotiated during the TLS handshake. For example, some TLS cipher mechanisms don't provide connection confidentiality. Users should leverage TLSv1.2 or later for secure cipher suites Clients should use only strong cipher suites. If TLSv1.2 is used, guidance in RFC7525 should be followed. The open-source community reports that TLSv1.3 currently does not support WebSocket (github.com/mattermost).

WebSocket Frames:

- WebSocket frames are very specific.
- The bits are binary on a scale of 0-1.
- If bits are set to 0, it is not the last fragment in message. 1 bit indicates the final fragment.
- Extensions interact when establishing socket RSV1, RSV2, and RSV3, which represent 1 bit each (Fette & Melnikov, RFC6455, 2011).

The WebSocket Protocol Registries managed by IANA can be found here.

Authentication

WebSocket key is concatenated by GUID (universal identifier) and is then based in code string.

This protocol doesn't prescribe any particular way that servers can authenticate clients during the WebSocket handshake. The WebSocket server can use any client authentication mechanism available to a generic HTTP server, such as cookies, HTTP authentication, or TLS Mutual authentication.

A WSS URI (WebSocket-secure resource identifier) identifies a WebSocket server and resource name. Itindicates that traffic over that connection is to be protected via TLS, data confidentiality and integrity, and endpoint authentication (Fette & Melnikov, RFC6455, 2011).

When to Use

If an encrypted WebSocket connection is used, then the use of TLS in the WebSocket-secure connection ensures that an HTTP CONNECT command is issued when the browser is configured to use an explicit proxy server.

- For example, some corporate networks only allow internet access via a HTTP proxy. In this case, the best transport for CoAP would be the WebSocket Protocol (Fette & Melnikov, RFC6455, 2011).

The WebSocket protocol provides communication between a client and a server after upgrading the HTTP connection (Fielding & Reschke, RFC7230, 2014). Another scenario for CoAP over WebSockets is a CoAP application running inside a web browser without access to connectivity other than HTTP and WebSockets (Tschofenig & Fossati, RFC7925, 2016).

Secure Coding Practices

See Secure Coding Practices section for coding recommendations and resources.

On closing the Connection:

- Payload data and payload length have a max of 64 KB in single communication but are able to load a lot of information.
- For payload data, if an unknown opcode is received, the receiving endpoint connection must fail.
- Endpoint indicates a closing frame with an opcode.
- The server and client may have cipher suite mechanisms that trigger a connection or trigger a failed connection, but many of those are built into the TLS handshake (Melnikov and Fette, RFC6455, 2011).

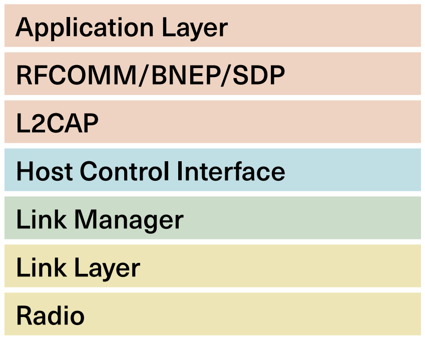
Network Layer Protocols

BLE Protocol Stack and Implementation

Overview

Bluetooth Low Energy (BLE) is a type of Bluetooth that uses less power consumption and transmits data over 40 channels in the 2.4 GHz ISM frequency band. BLE connections employ a key pairing method and are becoming one of the most common wireless standards. Bluetooth LE was introduced in Bluetooth 4.0, improved in Bluetooth 4.1, and developed even further in later versions.

Figure 9. BLE and Supporting Protocol Stack



Security Recommendations and Constraints Considerations

Security Consideration

Extrapolating the high-level issue from all of these attacks is the capability for an attacker to intercept, read, modify, and inject their own traffic between two devices' communication over BLE, bypassing the cryptographic and authentication layers built in." (https://www.openpath.com/blog-post/bluetooth-security-vulnerabilities).

The transmission of IPv6 over Bluetooth LE links or IPv6 over IEEE 802.15.4 has similar requirements and concerns for security.

Using IPv6 over Bluetooth is a strong pairing to consider for security and speed. Combining Bluetooth low energy and IPv6 optimizes small, low-power devices that can communicate directly with each other since using IPv6 allows each device to have its own IP address that can connect directly to the internet.

The security considerations for Bluetooth and BLE include vulnerabilities to spoofing, man-in-the-middle, and key negotiation attacks.

Authentication

- Authentication is provided through Connection Signature Resolving Key (CRSK), which "is a 128-bit key used to sign data and verify signatures on the receiving device" (Renesas Electronics Corporation, n.d.).
- The pairing of BLE devices involves authenticating the identity of two devices, encrypting the link using a short-term key (STK), and then distributing long-term keys (LTK) used for encryption (Duque, 2018).
- Key management, or encryption in Bluetooth LE, is provided by the Security Manager Protocol (SMP).
- **Numeric Comparison:** Devices with the capability to display and confirm information
- **Just Works:** At least one device has no input/output capability

When to Use

BLE has a low bandwidth, which makes it ideal for collecting data from sensor devices. On the other hand, the low bandwidth makes BLE not suitable for large data transfer applications. BLE also supports federation and allows devices from various manufacturers to communicate, making it suitable for home automation devices and medical devices.

Secure Coding Practices

- **Passkey Entry Pairing:** One device supports entering a six-digit number (passkey)
- **Out-of-Band (OOB):** Both devices possess an extra communication channel, such as NFC or WLAN

See Secure Coding Practices section for addition information and resources.

LoRaWAN Protocol Stack and Implementation

Overview

LoRaWAN is a low-power, wide-area networking protocol that manages communication between end-node devices and network gateways over a wide range at a low bit rate. It allows for a "single-hop" between an end-device and one or more gateways. LoRa supports the 900 MHz ISM bands. It's attractive for use in smart cities and industrial areas because it provides long-range communications but consumes very little power.

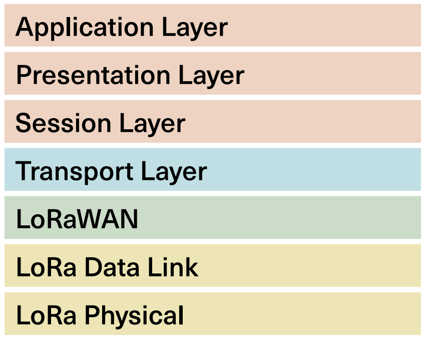


Figure 10. LoRaWAN and Supporting Protocol Stack

Security Recommendations and Constraints Considerations

Security Consideration

A LoRaWAN network can have more than one application server. It is typically built over a radio frequency or LoRa radio, which is a modulation technique that wirelessly connects devices to the internet.

- "LoRaWAN security uses the AES cryptographic primitive combined with several modes of operation: CMAC2 for integrity protection and CTR3 for encryption" (LoRa Alliance, 2016).

Authentication

Each LoRaWAN device is personalized with a unique 128-bit AES key and a globally unique identifier (EUI-64-based DevEUI), which are both used in the authentication process.

- Authentication is accomplished using these pre-shared keys.

When to Use

LoRa devices and the LoRaWAN standard are flexible for rural or indoor use cases in a wide range of industries including smart cities, homes and buildings, communities, agriculture, metering and utilities, healthcare, environment, and supply chain and logistics (Semtech, 2021).

LoRa Devices Classification: There are three classes of end-devices in a LoRa network. The classes are defined as Class A, Class B, and Class C. These can address a range of devices and are bi-directional in nature for communication. Class A and Class B allow bi-directional communication using less power.

Class A is required for all LoRaWAN devices, whereas class B and class C are extensions that may optionally be used as appropriate. Class A communication is initiated by the IoT device, are often battery powered with the lowest energy consumption of the three classes. Class B adds the ability to schedule time periods to receive beacon messages, enabling communication initiated from the gateway during these set intervals. The battery usage remains low, but is increased from class A due to the additional scheduled up time for the device. Class C, on the other hand, can receive windows that are almost always open, mandating more power to operate. LoRaWAN devices can support all three.

- LoRaWAN can be used when there is no other network connectivity and when power is constrained.

Secure Coding Practices

See Secure Coding Practices section for coding recommendations and resources.

Wi-Fi HaLow Protocol Stack and Implementation

Overview

Wi-Fi HaLow is a wireless networking technology that allows devices such as computers, mobile devices, and other equipment to interact with the internet. It allows these devices—and many more—to exchange information with one another, creating a network (Wi-Fi Alliance, n.d.).

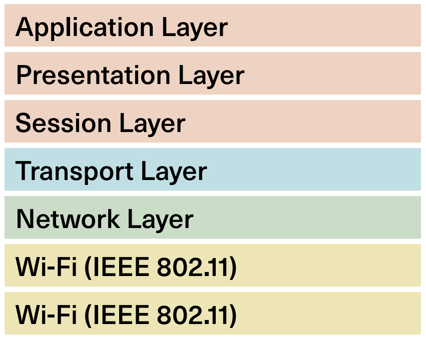


Figure 11. Wi-Fi and Supporting Protocol Stack

Security Recommendations and Constraints Considerations

Security Consideration

- Wi-Fi HaLow (IEEE 802.11ah protocol) is an improvement on traditional Wi-Fi (802.11) because it offers greater range and power efficiency with increased network density capabilities and throughput. Wi-Fi 802.11 was the original wireless identification, but it is not as suited for lower power connectivity. Wi-Fi HaLow 802.11 ah operates a spectrum below Wi-Fi and is suited to low-power connectivity and long ranges.
- Wi-Fi HaLow can be used in the IoT space due to its low-power connectivity. It has become useful in fields such as sensor networks and wearables.
- WPA2 (Wi-Fi Protected Access2) and WPA3 will provide the most secure option for a network layer since it is able to encrypt data to protect wireless connections from external threats.

Authentication

There are two link-level types of authentication:

1. **Open System:** Open System authentication allows any user to authenticate to the access point. This method can be used with no encryption or with (WEP) encryption
2. **Shared Key:** Only those wireless clients that have the shared key can connect. Shared Key authentication can be used only with WEP encryption

When to Use

Traditional Wi-Fi has limitations such as range and power usage, though it is more secure than Bluetooth. With its short range and high-power consumption, it has more limited use in IoT. Wi-Fi HaLow has greater range and power efficiency, but it has not been adopted by industry because it offers low security specific to authentication options. (Calhoun et al., RFC5416, 2009). For security, Wi-Fi HaLow requires the most advanced WPA3 security for encryption.

Secure Coding Practices

- Mask default name on server.
- Use Wireless network encryption.

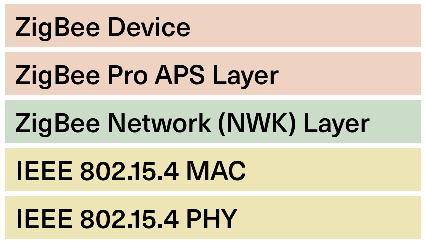
See Secure Coding Practices section for addition information and resources.

ZigBee Protocol Stack and Implementation

Overview

As represented in the ZigBee stack, Zigbee provides a complete solution for IoT protocols. ZigBee-certified products are able to connect and communicate using the same language, making it suitable for smart home and buildings. ZigBee uses small, low-power, low data-rate digital radios based on IEEE 802.15.4, making it a solution for applications that require long battery life and secure networking (Pahwa, 2019).

Figure 12. ZigBee 3.0 Supporting Protocol Stack



Security Recommendations and Constraints Considerations

Security Consideration

Security is embedded into the protocol, and encryption support proceeds through Advanced Encryption Standard (AES)-128 at the Network Layer.

- “AES-128 uses a 128-bit key length to encrypt and decrypt a block of messages,” which makes it optimal for smaller devices where power and latency are a concern (Bernstein & Cobb, 2021).

Authentication

Advanced Encryption Standard (AES) with a modified Counter using CBC-MAC (CCM) mode, also known aa AES-CCM*, combines data encryption, data authentication, and data integrity.

- The AES-CCM encrypts the data and generates an associated Message Integrity Code (MIC), which is sent to the receiver along with the frame.

- "The receiver uses the AES-CCM* to decrypt the data and generate its own MIC from the received frame to be compared with the received MIC" (Rudresh, 2017).

When to Use

ZigBee is used as a main communication protocol for home and building automation devices that perform tasks such as turning lights on and off, controlling thermostats, and monitoring security cameras.

- ZigBee allows for smart home and building devices to communicate over a network with the controller.
- ZigBee is suitable for Class 1 or higher devices, and it can only communicate with other ZigBee devices.

Secure Coding Practices

See Secure Coding Practices section for coding recommendations and resources.

Z-Wave Protocol Stack and Implementation

Overview

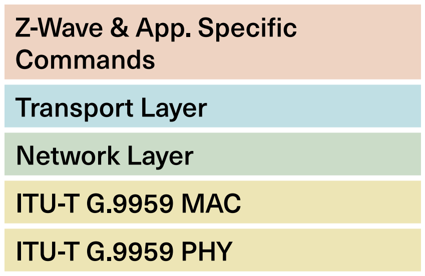
Very similar to Zigbee, Z-Wave provides a complete solution that allows devices to connect and communicate using the same language. A key difference seen in the Z-Wave stack is that it's recommended to define the MAC and PHY layers by ITU-T G.9959 and the channels in the 900MHz ISM frequency. These layers contain the "specification for short-range, narrow-band digital radiocommunication transceivers" (International Telecommunication Union, 2012).

Z-Wave has three standards: Z-Wave Classic, Z-Wave Plus, and Z-Wave Long Range.

Z-Wave Plus, which is the certification program "designed to help consumers identify products that take advantage of the 'Next Gen' Z-Wave hardware platform," contains more modern specifications and other essential features compared to its predecessor, Z-Wave Classic (The Z-wave Alliance, 2021). Not only does Z-Wave Plus offer consumers with the best performing Z-Wave products, but it also improves battery life by 50%, range by 67%, and bandwidth by 250%. These aspects of Z-Wave Plus make it more suitable for IoT devices while still allowing interoperability between all Z-Wave devices.

Z-Wave Long Range contains many of the specifications that Z-Wave Plus has, but it increases the wireless range roughly 4x and the maximum nodes on the network 10x. Additionally, it has a much lower power consumption (Budd, 2020).

Figure 13. Z-Wave and Supporting Protocol Stack



Security Recommendations and Constraints Considerations

Security Consideration

- Confidentiality and integrity protection are provided through AES-128 at the network layer.
- "Authentication is based on the cipher block chaining message authentication code (CBC-MAC) technique that can calculate a message authentication code (MAC) from a block cipher algorithm such as AES" (Fouladi & Ghanoun, *n.d.*).

Authentication

- IEEE MAC frame using the 128-bit key

When to Use

- Z-wave is also used as a main communication protocol option for smart home devices.
- Like ZigBee, Z-Wave is a mesh network that allows signals to hop from one ZigBee device to the next without each device needing to be connected to a Wi-Fi network (Mears, 2019).
- Z-Waves interoperability makes it favored over ZigBee when interoperability is a concern. "Every Z-Wave certified device WILL work with every Z-Wave certified controller" (Blank, n.d.).
- Z-wave provides a lower power alternative compared to Wi-Fi, and it has a much larger range than BLE, making it a great option to control devices in homes of any size.

Secure Coding Practices

See Secure Coding Practices section for coding recommendations and resources.

Recommendations

This section of the report presents recommendations of the CIS working group for IoT security. The recommendations mapped out here are aimed at the vendor to assist with improving security in devices by design. Resources such as this document are necessary building blocks for vendors to build in security. The integrity of a device depends critically on a trusted security stack and having options to secure devices. IoT devices have already become a platform for attacks and major security threats. IoT is also transforming sectors like healthcare and energy, and it will soon lead them. When we think about the vast amount of personal and proprietary data within those devices, it can be overwhelming to think about the risks that exist. Our objective is to provide guidance that will educate vendors about security options that they can use to secure their IoT devices such that they are adapted to today's threats upon delivery to customers.

Essential security measures such as encryption, signature verification, integrity, multi-factor authentication, and secured access control should always be primary options when securing an IoT device. However, these options may not be as feasible for more resource- or interface constrained devices. Therefore, it is imperative that IoT device configuration options be made available, tested, and hardened. As a result, many of the security functions designed for more general-purpose computing devices are more difficult to implement on IoT devices.

Transport Security Recommendations

As previously discussed in context of each protocol stack option, TLS and DTLS transport encryption may be difficult to implement on certain heavily resource-constrained IoT devices. CoAP built on top of the security protocol EDHOC, for example, is better suited for more constrained devices than TLS and DTLS. It allows those constrained devices to join the IoT network even though they're constrained with limited resources and interfaces. MQTT is a reliable protocol, as it has three levels of Quality of Service (QoS) that are designed to work on devices with constrained resources. With security not being built in, a TLS/DTLS stack can be tailored to fit the needs of a specific application, which may not work for low-power devices that lack resources to run MQTT (Hübschmann, 2021).

IoT devices should support the most recent version of IPv6 when possible. While IPv4 is still a viable option for IoT devices, IPv6 works more efficiently to transfer data from one IoT device to another and “is capable of sending large packets simultaneously to conserve bandwidth” (Pathak, 2020). It is the latest internet protocol standard, and it provides remote access and management for large vessels of IoT devices (Deering & Hinden, RFC 8200, 2017). While TLS 1.2 and higher are recommended for use, (TLSv1.0 and TLSv1.1 have been deprecated (Moriarty & Farrell, RFC 8996, 2021).) TLS 1.3 is the most current version of TLS, and per the NIST Special Publication 800-52 Rev. 2, it is recommended that agencies create a plan to have their servers fully support this version by January 1, 2024 and no longer allow for SSL to be utilized (McKay & Cooper, 2019). An additional security protocol for devices that are too constrained for TLS or DTLS is Ephemeral Diffie-Hellman Over COSE or EDHOC. It is becoming more commonly used for constrained RESTful environments (CoAP and RESTful HTTP). EDHOC will also provide the symmetric session keys required for Object Security of Constrained RESTful Environments (OSCORE), which provides authenticated encryption for the payload data.

Device Updates

When possible, we recommend supporting the most recent firmware and updating devices as new versions and patches become available. However, it should be recognized that it may not always be feasible to update more current protocols in many cases due to impact on power usage, memory, etc. In addition, vendors must consider the criticality of ensuring secure configuration along with implementing up-to-date firmware; keeping MQTT up to date when configured poorly will still result in an insecure system, for example.

The Software Updates for Internet of Things (SUIT) Working Group is an IETF group that works to define a firmware update solution that is usable on Class 1 devices. RFC9019 addresses the SUIT architecture for a firmware update solution, which includes:

- The internet protocol stack for firmware downloads. Firmware images are often multiple kilobytes, sometimes exceeding one hundred kilobytes, for low-end IoT devices. They can even be several megabytes for IoT devices running full-fledged operating systems like Linux. The protocol mechanism for retrieving these images needs to offer features like congestion control, flow control, fragmentation and reassembly, and mechanisms to resume interrupted or corrupted transfers.

- The capability to write the received firmware image to persistent storage (most likely flash memory).
- A manifest parser with code to verify a digital signature or a message authentication code (MAC).
- The ability to unpack, decompress, and/or decrypt the received firmware image.
- A status tracker (Moran et al., RFC9019, 2021).

Secure-by-Design Recommendations

IoT devices should be restricted to allow listed communications and behaviors. They should only communicate with trusted endpoints and not rely on the network security alone to secure communication. While that statement may sound counterintuitive, to confine a device, it is increasingly important as attackers look to exploit vulnerabilities in the IoT field. It is a good practice to restrict a device's interactions with the primary network. An insulated network ensures that devices are secured on the internet but won't be admissible to yielding critical files. This is important under the umbrella of device management.

Overall, these constraints should not limit the amount of configuration of IoT device communications. A user or organization should be able to configure communications between various devices that trust one another to communicate through the various protocol and IoT stack options. Ultimately, a vendor must fully understand the service their device is going to provide and then the most efficient and secure protocols can be chosen from that understanding.

Along with the choosing the best protocols for a device, there are additional elements that can be integrated to bolster the security posture in the devices themselves. Some of additional elements that should be taken into consideration are:

Hardware Root of Trust

Hardware root of trust is the foundation on which all secure operations of a computing system depend. Hardware root of trust "contains the keys used for cryptographic functions and enables a secure boot process" (Rambus Press, 2021). The utilization of a Trusted Platform Model (TPM), which is a computer chip (microcontroller) that can securely store artifacts used to authenticate the platform, can provide root of trust during pre-boot and post-boot operations. “In pre-boot, the TPM helps to secure the boot process against low-level malware and attest/measure integrity, and in post-boot, the TPM can help with multiple use cases, such as root of trust for authentication and sensitive mobile apps like micropayments, as well as network layer security” (Shpantzer et al., 2013).

The current Trusted Computing Group specification of TPM can be found here.

- Device Identifier Composition Engine (DICE) is a security standard created by the Trusted Computing Group (TCG), which works to help increase security in IoT devices. DICE “works by organizing the boot into layers and creating secret unique to each layer and configuration based on a Unique Device Secret (UDS)” (Guerra, 2018). Where Trusted Platform Modules (TPM) may be unfeasible to the constraints of an IoT device, DICE can be utilized to provide “device identity, attestation of device firmware and security policy, and safe deployment and verification of software updates” (Guerra, 2018).

Secure Boot

Secure boot “is a security standard developed by members of the PC industry to help make sure that your PC boots using only software that is trusted by the PC manufacturer,” (IT Connect, 2019). There are two main low-level software that starts before a device is booted, BIOS and UEFI, with UEFI being the more modern secure boot option.

- **BIOS:** Short for Basic Input-Output System, BIOS is a software that resides in a chip on a motherboard that loads when the device starts up. It is responsible for waking up other hardware components in a secure manner.
- **UEFI:** Very similar to BIOS, Unified Extensible Firmware Interface (UEFI) is software that works to kickstart the motherboard and other hardware. “UEFI can run in 32-bit or 64-bit mode and has more addressable address space than BIOS, which means your boot process is faster” (Hoffman, 2017). The current specification of UEFI can be found here.
- Secure boot on constrained devices may be limited to verification of the firmware image and bootstrapping the authentication and encryption with protocols such as BRSKI when secure boot as defined for PCs is not possible.

Integrity Protection

With modern mobile devices using secure boot to ensure they start up as expected, it is important to make sure the device continues to behave in an expected manner following a successful secure boot (Trusted Computing Group, 2020). The TCG Runtime Integrity Preservation (RIP) recommendation addresses the challenge of platform integrity, and it can prevent or detect and remediate runtime state modifications made by unauthorized actors.

The TCG Runtime Integrity Preservation in Mobile Devices reference can be found here.

Another option for integrity protection is a trusted execution environment (TEE). A TEE "is a secure area of a main processor that guarantees optimal protection for highly sensitive data in all its states with respect to confidentiality and integrity" (Wiedmann, 2021). The use of a TEE adds a layer of security on a device and protects the data in use without changing anything in the application itself.

- Hardware support to implement a TEE:
  - Arm's TrustZone
  - MultiZone Security
  - AMD Platform Security Processor (PSP)
  - Intel Software Guard Extension (SGX)
  - Google Titan

Root of trust with a Trusted Platform Module (TPM) or TCG's DICE can be used in conjunction with the TEE to verify policies, configurations, and the hashes of executables to assure the integrity of a device. The capability to assess integrity on IoT devices should increase in time and these resources serve as a guide when exploring these capabilities and assist with their evolution.

Lifecycle of Devices

Businesses and organizations are often slow to replace devices when it comes to its end, posing a great security risk. It is crucial to close out a device in a secure manner. It is recommended that IoT devices require a strong form of authentication or bootstrapping when possible as a prerequisite to connecting to a network.

Every IoT device needs a step-by-step plan to update throughout the lifecycle and to decommission it. Flaws will surface and vulnerabilities will emerge, so manufacturers need a way to consistently update them. This plan should include a way to push firmware updates securely, (See RFC 9019 and RFC 9124 for IoT specific firmware update architecture.) update crypto-libraries on which authentication is based, revoke authentication if needed, and re-enroll certificates as they expire over time. Additionally, the plan also needs to consider how manufacturers can deliver these updates, when necessary, even as devices go offline or move locations (i.e., in the EU, where traveling across countries frequently leads to roaming charges). When the device is ready to be retired, (It is imperative you include in your process when/what will signal its retirement.) you must employ a device management solution to provide a structured, secure service wrap.

Secure Software Development Framework (SSDF)

The National Institute of Standards and Technology (NIST) has developed a set of sound practices for secure software development to help organizations protect their software and produce well-secured software. The SSDF provides recommendations for mitigating the risk of software vulnerabilities. The most recent version of the SSDF can be found here: Secure Software Development Framework (SSDF) Version 1.1.

In addition to the SSDF, it is additionally recommended to create a Software Bill of Materials (SBOM) for software inventory. For IoT devices, it might not be practical for the SBOM to accompany the software. Instead, they might need to be stored signed in a repository for verification.

Secure Coding Practices

Along with the secure coding practices that may be specific to each stack, there are additional general coding practices that should be followed to ensure security. It is important to use safe, trusted code and functions during development. Coding can also provide multiple layers of defense. “[C]ombining secure programming techniques with secure runtime environments should reduce the likelihood that vulnerabilities remaining in the code at deployment time can be exploited in the operational environment” (Schiela, 2018). The resources below contain the top secure coding practices that should be considered during the development of a device.

- OWASP Secure Coding Practices Quick Reference Guide
- Secure Coding Guidelines for Java SE
- SEI CERT Coding Standards

Access Control

Access controls should specify which users are granted access and which operations they are permitted to perform. Each entry in an Access Control List (ACL) specifies a user and their access rights. Access control devices are usually physical hardware that an access control system needs to enforce these rules. Examples include locks, card readers, biometric devices, and controllers. Creating the settings and management framework of access control software is an integral process. When choosing an access control system, much focus is placed on how the software is managed and its user features—and rightly so. But it’s important to also investigate how well the system combines its physical devices so you can improve the user experience, lifecycle, and value of the investment.

There are three primary access control variations for IoT that will broadly define your device:

1 Role-Based Access Control (RBAC):
   - Individualized restrictions: Restricts network access based on a person’s role within an organization compared to other individuals or employees that have access to the network.

2 Attribute-Based Access Control (ABAC):

- Compared to RBAC, ABAC has a much greater number of possible control variables. Access control draws on a set of characteristics, called attributes, that include user attributes, environmental attributes, and resource attributes.
    - User attributes include things like the user's name, role, organization, ID, and security clearance.
    - Environmental attributes include the time of access, location of the data, and current organizational threat levels.
    - Resource attributes include things like creation date, resource owner, file name, and data sensitivity. (Solarwindssoftware, 2019)
- ABAC is more suitable for IoT environments as administrators to choose the best combination of a range of variables to build a robust and comprehensive set of access rules and policies.

3. Capability-Based Access Control (CapBAC):
    - CapBAC accounts for unique characteristics of the IoT (e.g., volume of devices, limited device-level resources).
    - Capability is a token or key that holds a privilege assigned to its holder. "When the requester intends to perform the action associated with the token it needs to send the request and the token to the gatekeeper/provider, which entity only needs to check the validity of the token" (Vilmos, 2021).
    - No user list or access rules need to be maintained at the point of access control. The task may also be performed by a resource-constrained device itself.

Access Control Examples

One of the most important devices in a physical access control system is the door controller, which connects card readers and software applications. The controller should be able to continue managing its tasks locally, and it must support all the necessary network security and performance requirements.

Another type of access control system, AnyConnect Access Control, is optimal for Wi-Fi, LTE, and 5G. It grants access to only the right users over public networks and offers different access levels, simplifying access control management. AnyConnect Access Control is secure and enables you to be GDPR compliant. AnyConnect Access Control is a cloud-based access control that is resilient to network and state, including online/offline transitions (AnyConnect, 2020).

Access Control Implementation Methods

MUD

Manufacturer Usage Description (MUD) enables IoT devices to communicate with the networks they connect with, bridging the gap between manufacturer and user. IoT devices represent some of the most vulnerable devices that can be exposed to Distributed Denial-of-Service (DDoS) attacks. MUD provides a framework to build implementation to address security problems and secure every device or server your system/device will connect to. One stipulation: MUD should not replace basic security measures for any device or network. It is simply another layer of added security. MUD's value lies in how it creates communication points among all devices and uses it as a pivot point to compromise all devices. Similar to a JSON file, MUD emits a file URL released by a DHCP packet. The DHCP packet then extracts it and sends it to the managing system or manufacturer. It is also crucial to have a threat-signaling layer on top of MUD for extra security.

Threat signaling using MUD from a user perspective:

- View all devices on the network with the category of MUD-capable devices
- Expand profile of device/implement MUD file for that device
    - MUD will intercept DNS requests from the network to see if there is a threat
    - Device will attempt to communicate with malicious domain
    - Local DNS will set up Quad9, which is a recursive DNS resolver that aims to protect users from malware and phishing
    - Will reach out to Quad9 threat API
    - Will show who has blocked the domain

Profiles can be developed by the manufacturer or a third-party integrator to list the behavior of devices, including "the expected network access, such as port and protocol information, but can also include other behaviors of the system such as expected interaction models" (Moriarty, 2020, p. 60).

Resources for MUD can be found at the NIST NCCOE MUD Resource Page and includes implementations as well as device MUD files representing the allowed and expected traffic and behaviors for specific devices. https://www.nccoe.nist.gov/mud-related-resources

XACML

eXtensible Access Control Markup Language (XACML) has been developed by the Organization for the Advancement of Structured Information Standards (OASIS) to standardize the language used for access control. One of the objectives of XACML is to specify a common language that promotes interoperability between access control implementations by multiple vendors. The most recent version of XACML can be found here: eXtensible Access Control Markup Language (XACML) Version 3.0.

Key Management

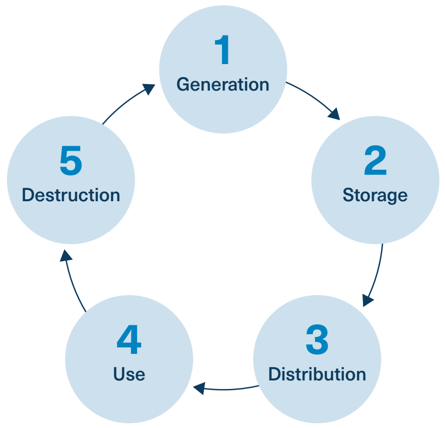


Figure 14. Key Management Lifecycle

Key management is a critical aspect to device and network security that involves "the handling of cryptographic keys and other related key information during the entire lifecycle of the keys," including their secure generation, storage, distribution, use, and destruction (Barker, p. 12, 2020). These stages of key management are the key management lifecycle:

Symmetric Encryption

In symmetric encryption, the same key is used for encrypting and decrypting data. Symmetric encryption may be accomplished through a stream cipher where data is encrypted one byte at a time or more commonly through a block cipher where data is encrypted in larger chunks such as 128-bit blocks (Paar & Pelzl, p. 30, 2009). Modern symmetric encryption can have key lengths as short as 128 bits or as long as 2,040 bits, but the most common key lengths used are 128 and 256 bits in order to balance encryption power and speed (IBM, 2021a). At the time of writing, popular non-deprecated examples of symmetric encryption algorithms include AES, Blowfish, Twofish, CAST5, and IDEA (Callas, et al., RFC 4880, 2007). Deprecated examples that should not be used include RC4, DES, and 3DES (Popov & Microsoft Corp., 2007; NIST, 2005; NIST, 2017).

Symmetric encryption has specific use-cases. Its relative speed makes it useful for encrypting large amounts of data, but symmetric encryption has no way to establish a secure channel between participants who have never met, which means that the difficulty involved in securely transferring keys limits usefulness on its own.

Asymmetric Encryption

Asymmetric encryption uses different keys for encrypting and decrypting data. The mathematical relationship between the keys allows for one key in the pair to encrypt and the other key in the pair to decrypt (Krzyworzeka, p. 39, 2016). Modern asymmetric encryption can have key lengths as short as 1,024 bits or as long as 4,096 bits, with the most common key lengths used being 2,048 and 4,096 bits in order to balance encryption power and speed (Barker & Roginsky, pp. 12-15, 2019). As of the time of writing, popular non-deprecated examples of asymmetric encryption methods include Diffie-Hellman, DSS, ECC, and RSA (Barker & Roginsky, 2019). With each of those examples, however, key size is crucial—many kinds of asymmetric encryption are deprecated if key sizes fall below 1,024 or even 2,048 bits (Barker & Roginsky, pp. 12-15, 2019). Asymmetric encryption has specific use-cases. It can establish secure communication over an insecure channel between participants who have never met. Their entire communication can't be decrypted even if it's intercepted, but it is much slower at encrypting data than symmetric encryption, limiting its usefulness on its own (Krzyworzeka, p. 39, 2016). With that being said, asymmetric encryption can be used for both authentication and confidentiality because it can be used to create digital signatures as well as encrypt data.

Symmetric and Asymmetric Encryption in IoT Devices

By utilizing both symmetric and asymmetric encryption when communicating with IoT devices, the strength of each encryption method can be used to mitigate the weakness of the other to establish a strong holistic encryption solution. There are two states of data that need encryption solutions, data at rest and data in transit. For data at rest, which is any data that is being stored in any computer (including IoT devices, cloud storage, on-site databases, a host computer, etc.), symmetric encryption is the only encryption solution required. For data in motion, although symmetric encryption could theoretically handle all communication between known participants, it is much more viable to combine asymmetric and symmetric encryption in practice. By using asymmetric encryption to establish secure communication between participants who have never met over an insecure channel, using that encrypted channel to share a symmetric key,and using that symmetric key to encrypt the rest of the data, participants can leverage asymmetric encryption to enable symmetric encryption over a distance, resulting in the most secure andefficient transfer of data. Device classification and device constraint levels will also determine what encryption algorithms can be used:

Table 5. Supported Encryption Algorithms Based on Device Class

Classes	Supported Encryption Algorithms
Class 0, C0	Difficult to support encryption algorithms (Choi & Park, 2017); potential use of AES-128
Class 1, C1	AES-128/256/384, RSA 1024/2048, and SHA-1/256/384 (Choi & Park, 2017)
Class 2, C2	AES-128/256/384, RSA 1024/2048, and SHA-1/256/384 (Choi & Park, 2017)

Encryption and Key Management Best Practices

- Best practice for IoT key management requires establishing perfect forward secrecy as well as deciding on symmetric and asymmetric algorithms and key sizes depending on the security requirements and capabilities for each IoT device. The goal for each device is to use the information available in this document to establish a high level of security for the lowest possible impact on device operations.
- In perfect forward secrecy, breaking a device's encryption only breaks the encryption for that session, not for all time (IBM, 2021b). In the basic example under *Symmetric and Asymmetric Encryption in IoT Devices*, asymmetric encryption is used once to establish a secure channel for the transmission of one or more symmetric keys. If that initial asymmetric encryption is ever broken, however, then all communication past, present, and future is exposed in perpetuity. In order to prevent that single point of failure and establish perfect forward secrecy, IoT devices must establish communication using new asymmetric encryption and pass information using new symmetric encryption *every session*. In that case, even if either level of encryption is broken, only that session is compromised. The disadvantage is that establishing new asymmetric and symmetric keys for every session puts a heavier load on devices, and so it is crucial that IoT devices use the lightest protocols possible while still maintaining that high level of security.

Key Management Protocol Guidance

RFC 4962, *Guidance for Authentication, Authorization, and Accounting (AAA) Key Management*, offers best practices for key management protocol that should, in general, be studied whenever new protocols are up for consideration. Key management often involves a collection of protocols, each of which can be incredibly complicated, to provide a holistic solution; therefore, careful study should be made before implementation (Housley, et al., RFC 4962, 2007). In order to be considered a AAA key management protocol:

- Protocols must be cryptographic algorithm independent (algorithm agility),though it does not need to support both symmetric and asymmetric encryption. Otherwise, a flaw found in one dependent algorithm can deprecate an entire protocol.
- Sessions must have independent session keys.

Additional considerations can be found in RFC 4962.

- For recommendations and security considerations that do not meet the level of an absolute requirement or for more context on those requirements, please see RFC 4962.

General Hardening Practices

While there are many IoT specific security measures that can be embedded on a device, general hardening practices should still be used. Some of these practices include:

- Removing any unnecessary services
- Access control
- Data encryption
- Patching and updates
- Allow listing for software and protocols supported
- Auditing
- Logging
- Strong authentication, not subject to replay or phishing attacks

Hardening practices will work to ensure that devices are secure and that there is the lowest likelihood of a breach through a device. The CIS Benchmarks and IoT companion guide for the CIS Critical Security Controls (CIS Controls) v8 also provide strong recommendations for hardening and securing an organization's assets. The CIS Benchmarks provide more than 100 configuration guidelines across 25+ vendor product families to safeguard systems against today's evolving cyber threats. These Benchmarks can be found here. The IoT companion guide for the CIS Controls provide a holistic set of control recommendations in association with a variety of devices within IoT. It can be found here.

Industry Recommendations

Healthcare Sector

In the healthcare sector, it is not a stretch to say that hospitals and other medical systems alike could be flooded with vulnerable IoT devices. Top IoT threats continue to evolve and target IoT devices with new techniques. There is a natural heightened awareness around securing medical devices given that 80% of healthcare organizations were targeted with cyber attacks in recent years and that 25% of cyber attacks against hospitals involved IoT (Palo Alto Network, 2021). Additionally, 41% of attacks exploit device vulnerabilities, as attacks on IT networks target network-connected devices in an attempt to exploit known weaknesses. These devices are historically managed by hospitals or clinical engineers, but it is crucial going forward that these devices be managed by IoT security experts and systems (Unit 42 IoT Threat Report, 2020).

DICOM (Digital Imaging and Communications in Medicine), for example, is an operational technology (OT) protocol for managing different types of medical imaging devices and related data, which is why it is used in the majority of healthcare facilities. DICOM ports are especially vulnerable to being exposed online. Attacks on DICOM ports can disrupt critical business operations. DICOM can be best secured over the TCP network layer for speed, efficiency, and manageable cost.

The healthcare industry's first step should always include basic security hygiene, such as:

- Secure devices with product integration:
  - Next-generation firewalls
  - Network access control
  - Wireless management solutions
- Network segmentation

- Regular audits of devices carrying personal and confidential information along with the retirement of such assets if they're not being managed
- Ensuring multi-factor authentication on all devices
- Key Management

However, industry-specific standards in sectors require different standards, and given the amount of personal data that is stored in medical IoT as well as the critical information that is needed to treat patients, this is perhaps the most crucial area of IoT to secure properly not only for improving operational activity to improve patient care but also for protecting patient history. Security still remains the greatest barrier to adoption since most medical devices are not designed with a focus on security. Open gateways in medical IoT can bring on major data breaches. An architectural approach to securing devices should include network segmentation. Dividing the network in separate segments will allow security managers to isolate and better protect certain information. Additionally, implementing an intelligence application to display current vulnerabilities, any firmware patches that need updating, and an overall outline in other gaps (OSSs for example) will provide a robust and well-rounded understanding of related risk. The procurement process should also have a great deal of focus in that vendors should approach that stage with the client or buyer as a highly collaborative effort. Vendors should take the time to understand current security measures in the case there is room to incorporate those into the new product. Additionally, there should similarly be a collaborative effort in the supplying process between vendor and manufacturer. Vendors should work to influence the manufacturer to list out known vulnerabilities so that issues can be communicated early and often to clients.

Energy Sector

IoT devices are becoming ever more crucial for the energy sector. Similarly, the energy sector contains a massive amount of sensitive information, with systems relying on IoT constituting a potential threat to vulnerable exposed devices. There are five primary use cases to consider for the energy sector when it comes to IoT and ramping up wireless connectivity:

- Optimization of renewable energy resources
- Empowering microgrids
- Enabling predictive protocols for disaster management

- Smart meter technology
- Proactive repair mechanism

Being able to trust the input the sensor/IoT device is providing into your SCADA system is key. Maintaining the security of those communications in transit is crucial, but it's also crucial to have confidence that the data in transit is what you were expecting before you make operational decisions. Wider architectural considerations also apply, such as the integrity of an IoT sensor in your architecture. Are they trusted or untrusted? Any connections to third party-owned devices at these levels add complexity given that attacks could surface and affect IoT sensors connected to your architecture, especially at lower levels.

Additional Considerations/IoT Terminology Appendix

Topology of Networks

Another consideration for IoT security is the topology of the network into which devices are being integrated. In addition to the architecture of the device, as discussed throughout this document, the network topology can also provide security benefits. The three main topologies that are driving IoT networking standards are point-to-point network, star or hub-and-spoke network, and mesh network.

Point-to-Point Network

A point-to-point network establishes a direct connection between two network nodes or devices. These types of networks are simple and low cost, but they cannot scale given its one-to-one nature. An example of this type of network is a Bluetooth link between a cell phone and an ear piece.

Star/Hub-and-spoke Network

A star network, also known as a hub-and-spoke network, consists of one central hub (e.g., a gateway node) to which other nodes (e.g., sensor nodes) are connected. All peripheral nodes are able to communicate with the others by transmitting to, and receiving from, the central hub only. A star network has consistent, fast, and predictable performance with a data packet typically only traveling one hop to reach its destination. Another security benefit to a star network is that it is easy to isolate a device if a fault is detected because each device has a single link to the hub. A star network has a similar disadvantage to the point-to-point network with range being limited.

Mesh Network

A mesh network is the most complex out of the three topology types. A mesh network consists of three major components: the gateway, the repeater, and the endpoints. Devices in a mesh network are connected directly in a non-hierarchical way to route data across a network. With nodes being connected to one another, the network never relies on a single node, thus reducing the risk of connectivity failure. Range for a mesh network is much larger compared to the other topologies because the network range is not limited to the transmission range of one device. This makes a mesh network ideal for buildings and campuses or other applications where scalability is needed. A mesh network can scale to thousands of nodes, providing a high density of coverage with a broad assortment of sensors and actuating devices (Pacelle, 2014). With the complexity of a mesh network, there is a higher network latency due to the multiple hops that are typical from the sensor to gateway.

TCP/UDP in IP Stacks

Transmission Control Protocol/Internet Protocol (TCP/IP) is the core standard protocol for internet-based communications. Its main purpose is to deliver data packets between the source application or device and the destination using methods and structures that "place tags," such as address information into packet headers to be used at the transport layer (Socolofsky & Kale, RFC 1180, 1991). Some wireless systems "break" TCP/IP in order to lower the overhead of the on-air signals. While very different, TCP and User Datagram Protocol (UDP) are transport layer protocols with similar capabilities. Chief among them is that they both use the IP protocol. What separates them, however, is that UDP is a connectionless protocol, while TCP is connection-oriented with built-in error recovery and retransmission (similar to a telephone connection). UDP, on the other hand, does not correct or recover errors in the message. Any error detection and recovery are the responsibility of the receiving application. Hypertext Transfer Protocol, File Transfer Protocol, Simple Mail Transfer Protocol, Post Office Protocol, Internet Mail Access Protocol, and many other common internet application protocols use TCP. UDP is used by Domain Name System, Dynamic Host Configuration Protocol, Trivial File Transfer Protocol, Simple Network Management Protocol, Routing Information Protocol, and Voice over Internet Protocol. It should be noted that UDP is increasingly used for transport due to performance gains using protocols such as Quick UDP Internet Connection.

Gateway	A gateway is a device that receives information from positions on the network and transmits information to another network. Gateway devices play an important role in IoT; in the case that multiple protocols are mixed, wireless or wired, a gateway is almost always necessary (SBT, 2020). The gateway is the stopping point for online communications, the hub through which data is sent back and forth. A set of very constrained devices may be restricted to an accommodating protocol stack for communications, and then a gateway may be used for management, security, and interfacing with a less constrained network.
DHCP	Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) both work across the client-server architecture. DNS is a decentralized system that assigns domain names to an IP address and vice versa, preventing the user from having to remember IP addresses. DHCP "dynamically assigns IP addresses and other configuration options to devices in a network" (Schneider, 2022).

Standards Organizations

IETF	The Internet Engineering Task Force (IETF) is the leading platform that develops and publishes "open standards through open processes with one goal in mind: to make the internet work better," (ietf.org, n.d). IETF is the go-to resource to land on if you're looking to track how the internet and internet security have evolved as well as to find invaluable technical documents that map out benchmarks and standards for internet security.
IANA Registries	Internet Assigned Numbers Authority (IANA) registries are used to identify the current set of recommended algorithms, key sizes, and other protocol parameters. These registries keep track of IP addresses, domain names, and protocol parameter identifiers that are used by internet standards (iana.org). These registries are key in providing security recommendations for the IETF protocols and others that use the IANA registry.
NIST	The National Institute of Standards and Technology (NIST) is a non-regulatory agency of the U.S. Department of Commerce. The mission of NIST is "to promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve our quality of life" (NIST, 2021). NIST works to supply industry, academia, government, and other users with standards to measure equipment and procedures, quality control benchmarks for industrial processes, and experimental control samples.
OASIS	The Organization for the Advancement of Structured Information Standards (OASIS) works to create open source standardizations "for reference in international policy and procurement" (OASIS Open, 2021). The focus of the OASIS standards are cybersecurity, IoT, cloud computing, as well as other areas.

Connectivity Standards Alliance	Formerly the Zigbee Alliance, the Connectivity Standards Alliance is an organization of companies working to create, maintain, and deliver “open global standards for Internet of Things (IoT)” (Connectivity Standards Alliance, 2021). This alliance will work to continue to develop Zigbee technology and work to produce standards that are “reliable by nature, secure by design, and compatible at scale” (Connectivity Standards Alliance, 2021). CSA is also responsible for the development of matter and will be covered in a future document. Matter is a protocol to connect devices and systems with the aim of providing seamless and reliable security.
Z-Wave Alliance	The Z-Wave Alliance “is comprised of industry leaders throughout the globe that are dedicated to the development and extension of Z-Wave as the key enabling technology for ‘smart’ home and business applications” (Z-Wave Alliance, 2105). Their goal is to create advanced, practical, interoperable wireless products and services that can be implemented in residential and light commercial environments.
IEEE	The Institute of Electrical and Electronics Engineers (IEEE) “is the world’s largest technical professional organization dedicated to advancing technology for the benefit of humanity” (IEEE, n.d.). IEEE produces publications and technology standards, including the “IEEE 802 standards for local, metropolitan, and other area networks, including Ethernet and Wireless LAN (commonly referred to as Wi-Fi)” (IEEE, n.d.)
LoRa Alliance	LoRa Alliance is an organization “that has become one of the largest alliances in the technology sector, committed to enabling large-scale deployment of Low Power Wide Area Networks (LPWAN) IoT through the development and promotion of the LoRaWAN® open standard” (LoRa Alliance, 2021).

ETSI	The European Telecommunications Standards Institute (ETSI), is an organization that supports the developing and testing of technical standards for ICT-enabled systems, applications, and services. Through collaborative efforts, ETSI publishes standards to allow up-to-date security measures to be streamlined. For IoT related standards from ETSI, see Cyber Security for Consumer Internet of Things: Baseline Requirements.
ISO	International Standard Organization (ISO) provides consensus global standards through a rigorous review and collaborative process providing frameworks for policy on security and privacy in IoT, for example with ISO 27400:2022.

References

Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., & Levkowetz, H. (2004, June). *RFC 3748 - Extensible Authentication Protocol (EAP)*. IETF. Retrieved May 16, 2022, from https://datatracker.ietf.org/doc/html/rfc3748.

Aboba, B., Simon, D., Microsoft Corporation, Eronen, P., & Nokia. (2008, August). Extensible Authentication Protocol (EAP) Key Management Framework. IEFT. https://datatracker.ietf.org/doc/html/rfc5247.

AQMP Forum. (2021). Aqmp.Org. https://www.amqp.org/product/architecture.

AVSystem. (2020, March 4). IoT Standards and protocols guide — protocols of the Internet of Things. https://www.avsystem.com/blog/iot-protocols-and-standards/.

Barker, E. (2020, May). NIST SP 800–57 Part 1 Rev 5: Recommendation for Key Management. NIST. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf.

Barker, E., & Roginsky, A. (2019, March). NIST SP 800–131A Rev 2: Transitioning the Use of Cryptographic Algorithms and Key Lengths. NIST. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf.

Barnes, R., Hoffman-Andrews, J., McCarney, D., & Kasten, J. (2019, March). *RFC 8555 - Automatic Certificate Management Environment (ACME)*. IETF. Retrieved May 16, 2022, from https://datatracker.ietf.org/doc/html/rfc8555.

Bernstein, C., & Cobb, M. (2021, September 24). *Advanced Encryption Standard (AES)*. SearchSecurity. Retrieved October 13, 2021, from https://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard.

Blank, E. (2021, October 25). *Z Wave Vs ZigBee: Which Is Better For Your Smart Home?* The Smart Cave. Retrieved November 15, 2021, from https://thesmartcave.com/z-wave-vs-zigbee-home-automation/.

Borman, C. B. (2014, May). Terminology for Constrained-Node Networks. IETF. https://datatracker.ietf.org/doc/html/rfc7228.

Brockhaus, H., Fries, S., & von Oheimb, D. (2021, February 22). Lightweight Certificate Management Protocol (CMP) Profile. IETF. https://www.ietf.org/archive/id/draft-ietf-lamps-lightweight-cmp-profile-05.html.

Brockhaus, H., von Oheimb, D., Fries, S., & Siemens. (2022, May 13). Lightweight Certificate Management Protocol (CMP) Profile draft-ietf-lamps-lightweight-cmp-profile-12. IETF. https://datatracker.ietf.org/doc/html/draft-ietf-lamps-lightweight-cmp-profile.

Budd, R. (2020, October 2). *The Differences Between Z-Wave Versions Made Easy*. Wltd. Retrieved March 3, 2022, from https://wltd.org/posts/the-differences-between-z-wave-versions-made-easy.

Buypass. (n.d.). Buypass Go SSL - free TLS/SSL certificate, based on ACME. Retrieved June 9, 2022, from https://www.buypass.com/products/tls-ssl-certificates/go-ssl.

Calhoun, P., Montemurro, M., & Stanley, D. (2009, March). *rfc5416*. IETF. Retrieved November 15, 2021, from https://datatracker.ietf.org/doc/html/rfc5416.

Callas, J., PGP Corporation, Donnerhacke, L., IKS GmbH, Finney, H., Shaw, D., & Thayer, R. (2007, November). OpenPGP Message Format. IETF. https://datatracker.ietf.org/doc/html/rfc4880.

Chinnick, J. (2018, January 3). *Maximizing BLE Security & Privacy Features*. Nuvation Engineering. Retrieved May 3, 2022, from https://www.nuvation.com/resources/article/maximizing-ble-security-privacy-features.

Choi, E. Y., & Park, H. (2017, September). A Study on Encryption Key Management Technology in a Light IoT Device Environment. *International Journal of Innovative Research in Technology & Science*, *5*(5).

Cisco. (2016). PKI: Simplify Certificate Provisioning with EST. https://www.cisco.com/c/dam/en_us/about/doing_business/trust-center/docs/public-key-infrastructure-provisioning-with-est.pdf.

Classification of IoT Devices. (2021). CISO Platform. https://www.cisoplatform.com/profiles/blogs/classification-of-iot-devices.

CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets. (2020). IETF. Retrieved from https://datatracker.ietf.org/doc/html/draft-ietf-core-coap-tcp-tls-03.

Connectivity Standards Alliance. (2021, April 15). The Alliance in Commercial. https://zigbeealliance.org/market-uses/commercial/.

Connectivity Standards Alliance. (2021, May 13). *The Zigbee Alliance Rebrands as Connectivity Standards Alliance*. Retrieved November 16, 2021, from https://zigbeealliance.org/news_and_articles/connectivity-standards-alliance/.

Cope, S. (2021, July 25). *Introduction to MQTT-SN (MQTT for Sensor Networks)*. Steves Internet Guide. Retrieved November 15, 2021, from http://www.steves-internet-guide.com/mqtt-sn/.

Cox, T., & Furlong, J. (2019, October 24). Key Management Interoperability Protocol Usage Guide Version 2.0. OASIS Open. https://docs.oasis-open.org/kmip/kmip-ug/v2.0/cn01/kmip-ug-v2.0-cn01.html.

Cressler, C. (2021, September 7). *AMQP vs MQTT: Comparing Instant Messaging Protocols*. Cometchat. Retrieved October 13, 2021, from https://www.cometchat.com/blog/amqp-vs-mqtt-comparing-instant-messaging-protocols.

Deering, S., & Hinden, R. (2017, July). *rfc8200*. IETF. Retrieved November 15, 2021, from https://datatracker.ietf.org/doc/html/rfc8200.

Diffie, W., & Hellman, M. (1976). New Directions in Cryptography. IEEE Transactions on Information Theory, 22(6), 644–654. https://ee.stanford.edu/~hellman/publications/24.pdf.

DigiCert. (n.d.). Set up ACME agent-based automation for hosts. Retrieved June 9, 2022, from https://docs.digicert.com/certificate-tools/Certificate-lifecycle-automation-index/automation-user-guide/install-automation-agent-webserver/.

DNS Protocol. (2021). NS1. https://ns1.com/resources/dns-protocol.

DornerWorks. (2019, June 17). *The 3 Basic Concepts of MQTT Security*. IoT For All. Retrieved May 3, 2022, from https://www.iotforall.com/mqtt-security-three-basic-concepts.

Duque, A. (2018, October 18). *Deep Dive into Bluetooth LE Security - Rtone IoT Security*. Medium. Retrieved November 15, 2021, from https://medium.com/rtone-iot-security/deep-dive-into-bluetooth-le-security-d2301d640bfc.

Dynamic Host Configuration Protocol (DHCP). (2021, September 2). GeeksforGeeks. https://www.geeksforgeeks.org/dynamic-host-configuration-protocol-dhcp/.

Earls, A. R. (2016, November 14). XMPP: IoT protocol winner, or second place to MQTT? IoT Agenda. https://internetofthingsagenda.techtarget.com/feature/XMPP-IoT-protocolwinner-or-second-place-to-MQTT.

Entrust. (n.d.). Certificate Services Support. Retrieved June 9, 2022, from https://www.entrust.com/knowledgebase/ssl/how-to-use-acme-to-install-ssl-tls-certificates-in-entrust-certificate-services-apache.

eProsima. (2019). *8.1. Authentication plugin: DDS:Auth:PKI-DH — Fast DDS 2.2.0 documentation*. Retrieved December 13, 2022, from https://fast-dds.docs.eprosima.com/en/v2.2.0/fastdds/security/auth_plugin/auth_plugin.html.

Fajardo, V., Arkko, J., Loughney, J., & Zorn, G. (2012, October). RFC 6733 - Diameter Base Protocol. IETF. Retrieved June 17, 2022, from https://datatracker.ietf.org/doc/html/rfc6733.

Farrell, S. (2018, May). *rfc8376*. IETF. Retrieved November 15, 2021, from https://datatracker.ietf.org/doc/html/rfc8376.

Fette, I., & Melnikov, A. (2011, December). rfc6455. IETF. https://datatracker.ietf.org/doc/html/rfc6455.

Fielding, R., & Reschke, J. (2014, June). *rfc7230*. IETF. Retrieved November 15, 2021, from https://datatracker.ietf.org/doc/html/rfc7230.

Fremantle, P. & WSO2. (2015, October). *A Reference Architecture For The Internet of Things*. WSO2. Retrieved December 3, 2021, from https://wso2.com/whitepapers/a-reference-architecture-for-the-internet-of-things/.

Garcia-Carrillo, & Marin-Lopez, R. (2022). EAP-based Authentication Service for CoAP draft-ietf-ace-wg-coap-eap-08. *ACE Working Group*. https://datatracker.ietf.org/doc/pdf/draft-ietf-ace-wg-coap-eap-08.

Garcia-Morchon, O., Philips, Kumar, S., Signify, Sethi, M., & Ericsson. (2019, April). Internet of Things (IoT) Security: State of the Art and Challenges. IETF. https://www.rfc-editor.org/rfc/rfc8576.

Gregersen, C. G. (2021, January 29). Websocket vs. MQTT vs. CoAP: Which is the Best Protocol? Nabto. https://www.nabto.com/websocket-vs-mqtt-vs-coap/.

Guerra, M. (2018, July 12). *What is DICE architecture and DICE microcontroller?* Microcontroller Tips. Retrieved March 7, 2022, from https://www.microcontrollertips.com/what-is-dice-architecture-faq/.

Hardt, D. (2012, October). *rfc6749*. IETF. Retrieved November 15, 2021, from https://datatracker.ietf.org/doc/html/rfc6749.

Helpful IoT Terms & Definitions. (2020, January 19). SBT Solutions. https://sbtalliance.com/helpful-iot-terms-definitions/.

HiveMQ. (n.d.). *The Public MQTT Broker by HiveMQ*. Retrieved March 3, 2022, from https://www.hivemq.com/public-mqtt-broker/.

Hoffman, C. (2017, November 27). *What Is UEFI, and How Is It Different from BIOS?* How-To Geek. Retrieved May 24, 2022, from https://www.howtogeek.com/56958/htg-explains-how-uefi-will-replace-the-bios/.

Hoffman, P., VPN Consortium, Schaad, J., & Soaring Hawk Consulting. (2010, June). RFC 5911: New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME. IETF. https://datatracker.ietf.org/doc/html/rfc5911.

Housley, R. & Vigil Security. (2009, September). RFC 5652: Cryptographic Message Syntax (CMS). IETF. https://datatracker.ietf.org/doc/html/rfc5652.

Housley, R., Vigil Security, Aboba, B., & Microsoft. (2007, July). Guidance for Authentication, Authorization, and Accounting (AAA) Key Management. IETF. https://www.ietf.org/rfc/bcp/bcp132.html.

IBM. (2021a, March 3). Cryptographic algorithm and key length. https://www.ibm.com/docs/en/sgklm/3.0?topic=overview-cryptographic-algorithm-key-length.

IBM. (2021b, March 3). Perfect forward secrecy. https://www.ibm.com/docs/en/sss/3.1.1?topic=reference-perfect-forward-secrecy.

IEEE. (n.d.-a). *IEEE at a Glance*. Retrieved November 16, 2021, from https://www.ieee.org/about/at-a-glance.html#standards.

IEEE. (n.d.-b). *Mission & Vision*. Retrieved November 16, 2021, from https://www.ieee.org/about/vision-mission.html.

IETF. (n.d.). Constrained RESTful Environments (core). Retrieved October 11, 2021, from https://datatracker.ietf.org/wg/core/about/.

Information Commissioner's Office. (2021, March 30). What types of encryption are there? https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/encryption/what-types-of-encryption-are-there/.

Interagency International Cybersecurity Standardization Working Group (IICS WG). (2018, November 29). *NISTIR 8200, Status of International Cybersecurity Standardization for IoT | CSRC*. https://csrc.nist.gov/publications/detail/nistir/8200/final.

International Telecommunication Union. (2012). Short range narrow-band digital radiocommunication transceivers – PHY and MAC layer specifications. Recommendation ITU-T G.9959. Published.

Internet Security Research Group (ISRG) & Internet Engineering Task Force (IETF). (n.d.). Automatic Certificate Management Environment (ACME). Github. Retrieved June 7, 2022, from https://github.com/ietf-wg-acme/acme/.

IoT Solution Developer Protocols Guide. (2021). T-Moblile. https://www.tmobile.com/content/dam/tfb/pdf/IoT-Solution-Developer-Protocols-Guide.pdf?icid=TFB_TMO_P_19IOT_E3UOLES40BRO6VPW19260.

IoT Standards and Protocols. (2020, January 2). Postscapes. https://www.postscapes.com/internet-of-things-protocols/.

IoT Standards and protocols guide — protocols of the Internet of Things. (2021). AVSystem. https://www.avsystem.com/blog/iot-protocols-and-standards/.

IoT technologies and protocols. (2020). Azure. https://azure.microsoft.com/enus/overview/internet-of-things-iot/iot-technology-protocols/.

IT Connect. (2019, December 9). *How to Enable Secure Boot*. Retrieved May 24, 2022, from https://itconnect.uw.edu/wares/mws/mgmt/setup-computer/secure-boot/.

Iyengar, J., & Thomson, M. (2021, May). *rfc9000*. IETF. Retrieved November 15, 2021, from https://datatracker.ietf.org/doc/html/rfc9000.

Jaffey, T. (2014). MQTT and CoAP, IoT Protocols | The Eclipse Foundation. Eclipse Foundation. https://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php.

Jones, Q. (2020, July 22). *IoT Architecture: Topology and Edge Compute Considerations*. Digi. Retrieved November 15, 2021, from https://www.digi.com/blog/post/iot-architecture-topology-and-edge-compute.

Joshi, N. J. (2019, December 7). Six types of IoT network protocols. Allerin.

Karr, S. (n.d.). Bluetooth Technology Overview. Bluetooth® Technology Website. Retrieved September 28, 2021, from https://www.bluetooth.com/learn-about-bluetooth/techoverview/.

Krzyworzeka, N. (2016). Asymmetric Cryptography and Trapdoor One-Way Functions. AUTOMATYKA, 20(2), 39–51. https://journals.agh.edu.pl/automat/article/view/2774/1885.

LIFECYCLE-ORIENTED IT STACK. (2021). Zero Outage. https://zero-outage.com/thestandard/security/security-taxonomy-for-iot/lifecycle-oriented-it-stack/.

LoRa Alliance. (2017, February). LoRaWAN Security Full End-to-End Encryption for IoT Application Providers. Retrieved from https://lora-alliance.org/wp-content/uploads/2020/11/lorawan_security_whitepaper.pdf.

LoRa Alliance. (2021, November 9). *Homepage*. Retrieved November 16, 2021, from https://lora-alliance.org/.

LoRa Alliance. (2021, September 20). What is LoRaWAN® Specification. https://loraalliance.org/about-lorawan/.

LoRaWAN Architecture. (2021). The Things Network. https://www.thethingsnetwork.org/docs/lorawan/architecture/.

MANI, S. K. M. (2018). A System for Clock Synchronization in an Internet of Things. Networking and Internet Architecture, 2–18.

McKay, K. A., & Cooper, D. A. (2019). Guidelines for the selection, configuration, and use of Transport Layer Security (TLS) implementations. NIST Special Publication 800–52. Published. https://doi.org/10.6028/nist.sp.800-52r2.

Mears, K. (2019, April 17). Zigbee vs ZWave: Which Is The Best Choice For Your Smart Home? Smarthome. https://www.smarthome.com/blogs/buyers-guides/zigbee-vs-z-wavewhich-is-the-best.

Melnikov, A., & Zeilenga, K. (2006, June). *RFC 4422*. IETF. Retrieved November 15, 2021, from https://datatracker.ietf.org/doc/html/rfc4422.

Moran, B., Tschofenig, H., Brown, D., & Meriac, M. (2021, April). *rfc9019*. IETF. Retrieved March 7, 2022, from https://datatracker.ietf.org/doc/html/rfc9019.

Moriarty, K. M. (2020). *Transforming Information Security: Optimizing Five Concurrent Trends to Reduce Resource Drain*. Van Haren Publishing.

Moriarty, K., & Farrell, S. (2021, March 1). *RFC 8996: Deprecating TLS 1.0 and TLS 1.1*. https://www.rfc-editor.org/rfc/rfc8996.

Moskowitz, R., Heer, T., Jokela, P., & Henderson, T. (2015, April). *RFC 7401 - Host Identity Protocol Version 2 (HIPv2)*. IETF. Retrieved May 16, 2022, from https://datatracker.ietf.org/doc/html/rfc7401.

MQTT. (n.d.). MQTT Specification. MQTT.Org. Retrieved April 27, 2022, from https://mqtt.org/mqtt-specification/.

MQTT.org. (n.d.). MQTT: The Standard for IoT Messaging. Retrieved September 28, 2021, from https://mqtt.org/.

National Security Memorandum on Improving Cybersecurity for Critical Infrastructure Control Systems. (2021, June 28). The White House. https://www.whitehouse.gov/briefingroom/statements-releases/2021/07/28/national-security-memorandum-on-improvingcybersecurity-for-critical-infrastructure-control-systems/.

Neuman, C., Hartman, S., & Raeburn, K. (2005, July). *RFC 4120 - The Kerberos Network Authentication Service (V5)*. IETF. Retrieved May 16, 2022, from https://datatracker.ietf.org/doc/html/rfc4120.

NIST Cybersecurity for IoT Program. (2021). NIST. https://www.nist.gov/programsprojects/nist-cybersecurity-iot-program.

NIST. (2005, June 2). NIST Withdraws Outdated Data Encryption Standard. https://www.nist.gov/news-events/news/2005/06/nist-withdraws-outdated-data-encryption-standard.

NIST. (2017, July 11). Update to Current Use and Deprecation of TDEA. https://csrc.nist.gov/news/2017/update-to-current-use-and-deprecation-of-tdea.

NIST. (2021, March 4). *NIST Mission, Vision, Core Competencies, and Core Values*. Retrieved November 16, 2021, from https://www.nist.gov/about-nist/our-organization/mission-vision-values.

OASIS Open. (2021, January 29). *About Us*. Retrieved January 28, 2022, from https://www.oasis-open.org/org/.

OASIS. (n.d.). Products and Success Stories | AMQP. AMQP. Retrieved September 28, 2021, from https://www.amqp.org/about/examples.

Object Management Group, Inc. (n.d.). How Does DDS Compare to other IoT Technologies? DDS Foundation. Retrieved September 28, 2021, from https://www.ddsfoundation.org/features-benefits/.

Omale, G. (2018, November 7). Gartner Identifies Top 10 Strategic IoT Technologies and Trends. Gartner. https://www.gartner.com/en/newsroom/press-releases/2018-11-07gartner-identifies-top-10-strategic-iot-technologies-and-trends.

OWASP. (2010, November). OWASP Secure Coding Practices Quick Reference Guide. Retrieved July 7, 2022, from https://owasp.org/www-pdf-archive/OWASP_SCP_Quick_Reference_Guide_v2.pdf.

Paar, C., & Pelzl, J. (2009). Understanding Cryptography. Springer Publishing.

Pacelle, M. (2014, April 4). *3 topologies driving IoT networking standards*. O'Reilly Radar. Retrieved May 16, 2022, from http://radar.oreilly.com/2014/04/3-topologies-driving-iot-networking-standards.html.

Pahwa, V. (2019, December 11). *5 Reasons ZigBee is Ideal for Smart Homes*. EInfochips. Retrieved October 14, 2021, from https://www.einfochips.com/blog/5-reasons-zigbee-is-ideal-for-smart-homes/.

Pathak, U. (2020, December 14). *Advantages Of IPv6 In IoT*. Pianalytix - Machine Learning. Retrieved May 2, 2022, from https://pianalytix.com/advantages-of-ipv6-in-iot/.

Popov, A. & Microsoft Corp. (2015, February). Prohibiting RC4 Cipher Suites. IEFT. https://datatracker.ietf.org/doc/html/rfc7465.

Pritikin, M., Cisco Systems, Inc., Yee, P., AKAYLA, Inc., Harkins, D., & Aruba Networks. (2013, October). Enrollment over Secure Transport. IETF. https://datatracker.ietf.org/doc/html/rfc7030#section-4.

Rambus Press. (2021, November 10). *Hardware Root of Trust: Everything you need to know*. Rambus. Retrieved May 24, 2022, from https://www.rambus.com/blogs/hardware-root-of-trust/#:%7E:text=A%20hardware%20root%20of%20trust,must%20be%20secure%20by%20design.

Renesas Electronics Corporation. (n.d.). *3. Pairing and bonding — DA145XX Tutorial BLE security*. Renesas. Retrieved June 13, 2022, from http://lpccs-docs.dialog-semiconductor.com/Tutorial-DA145x-BLE-Security/pairing_and_bonding.html.

Rescorla, E. (2000, May). *rfc2818*. IETF. Retrieved November 15, 2021, from https://datatracker.ietf.org/doc/html/rfc2818.

Rescorla, E., & Modadugu, N. (2012, January). *rfc4347: Datagram Transport Layer Security*. IETF. Retrieved October 13, 2021, from https://datatracker.ietf.org/doc/html/draft-rescorla-dtls-05.

RF Wireless World. (n.d.). *IoT devices | IoT device categories class 0,1,2*. Retrieved April 28, 2022, from https://www.rfwireless-world.com/IoT/IoT-devices.html.

Rigney, C., Willens, S., Rubens, A., & Simpson, W. (2000, June). RFC 2865 - Remote Authentication Dial In User Service (RADIUS). IETF. Retrieved June 17, 2022, from https://datatracker.ietf.org/doc/html/rfc2865.

Rudresh, V. (2018, January 23). *ZigBee Security: Basics (Part 2)*. Kudelski Security Research. Retrieved November 15, 2021, from https://research.kudelskisecurity.com/2017/11/08/zigbee-security-basics-part-2/.

Sarangam, A. S. (2018, October 18). 7 IOT Layers That You Should Know in 2021. Jigsaw. https://www.jigsawacademy.com/4-layers-of-the-internet-of-things/.

SCADA Communication with TCP IP Stack. (2021). Semtech. Retrieved from https://www.researchgate.net/figure/SCADA-Communication-with-TCP-IP-Stack_fig4_266678442.

Schiela, R. (2018, May 2). Top 10 Secure Coding Practices - CERT Secure Coding - Confluence. Carnegie Mellon University Software Engineering Institute. Retrieved July 6, 2022, from https://wiki.sei.cmu.edu/confluence/display/seccode/Top+10+Secure+Coding+Practices.

Schneider, S. (2022, February 11). *Brief Introduction: DHCP and DNS*. Univention. Retrieved May 2, 2022, from https://www.univention.com/blog-en/brief-introduction/2019/03/brief-introduction-dhcp-dns/.

Scott, G., Furlong, J., & Bartell, J. (2020, December 14). OASIS Key Management Interoperability Protocol (KMIP) TC. OASIS Open. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=kmip.

SecureW2. (2022, January 19). What is 802.1X? How Does it Work? Retrieved May 16, 2022, from https://www.securew2.com/solutions/802-1x.

Selander, G., Mattsson, J., & Palombini, F. (2021, September 24). Ephemeral Diffie-Hellman Over COSE (EDHOC) draft-ietf-lake-edhoc-05. IETF. https://datatracker.ietf.org/doc/html/draft-ietf-lake-edhoc-05.

Selander, G., Mattsson, J., Palombini, F., & Seitz, L. (2019, July). *RFC8613*. IETF. Retrieved November 15, 2021, from https://datatracker.ietf.org/doc/html/rfc8613.

Semtech. (n.d.). *Innovation for a Better World*. Retrieved November 15, 2021, from https://www.semtech.com/company/corporate-citizenship/innovation-for-a-better-world.

Sengul, C., & Kirby, A. (2021, May 11). *draft-ietf-ace-mqtt-tls-profile-12*. IETF. Retrieved November 15, 2021, from https://datatracker.ietf.org/doc/html/draft-ietf-ace-mqtt-tls-profile-12.

Shelby, Z., Hartke, K., & Bormann, C. (2014, June). RFC 7252. IETF. https://datatracker.ietf.org/doc/html/rfc7252.

Shpantzer, G., Pescatore, J., Hallum, C., & Cannady, S. (2013). *Implementing Hardware Roots of Trust: The Trusted Platform Module Comes of Age* [Slides]. Trusted Computing Group. https://trustedcomputinggroup.org/wp-content/uploads/SANS-Webcast-Presentation.pdf.

Simon, D., Aboba, B., Hurst, R., & Microsoft Corporation. (2008, March). The EAP-TLS Authentication Protocol. IEFT. https://datatracker.ietf.org/doc/html/rfc5216.

Sinha, S. (2021, September 22). *State of IoT 2021: Number of connected IoT devices growing 9% to 12.3 billion globally, cellular IoT now surpassing 2 billion*. IoT Analytics. Retrieved November 15, 2021, from https://iot-analytics.com/number-connected-iot-devices/.

Socolofsky, T., & Kale, C. (1991). TCP/IP tutorial. *Network Working Group*. https://doi.org/10.17487/rfc1180.

solarwindssoftware. (2020, July 7). *RBAC vs. ABAC Access Control: What's the Difference? - DNSstuff*. Software Reviews, Opinions, and Tips - DNSstuff. Retrieved May 17, 2022, from https://www.dnsstuff.com/rbac-vs-abac-access-control#:%7E:text=The%20primary%20difference%20between%20RBAC,%2C%20environment%2C%20or%20resource%20attributes.

SSL.com. (n.d.). Order Free 90-Day SSL/TLS Certificates with ACME. Retrieved June 9, 2022, from https://www.ssl.com/how-to/order-free-90-day-ssl-tls-certificates-with-acme/.

Su, L. (2021, June 1). *MUD is officially approved by IETF as an Internet Standard, and Cisco is launching MUD1.0 to protect your IoT devices*. Cisco Blogs. Retrieved January 26, 2022, from https://blogs.cisco.com/security/mud-is-officially-approved-by-ietf-as-an-internet-standard-and-cisco-is-launching-mud1-0-to-protect-your-iot-devices.

Thales. (n.d.). *What is Key Management Interoperability Protocol (KMIP)? | Thales*. Thales Group. Retrieved May 23, 2022, from https://cpl.thalesgroup.com/faq/key-secrets-management/what-key-management-interoperability-protocol-kmip.

The Three Software Stacks Required for IoT Architectures. (2021). Eclipse IoT. https://iot.eclipse.org/community/resources/white-papers/iot-architectures/.

The Ultimate Guide to IoT Terminology. (2021). Link Labs. https://www.link-labs.com/35-topiot-terms-you-need-to-know.

Trusted Computing Group. (2020, January). TCG Runtime Integrity Preservation in Mobile Devices. Retrieved May 24, 202, from https://trustedcomputinggroup.org/wp-content/uploads/TCG_RIP_in_Mobile_Devices_2020_web_FC01.pdf.

Trusted Computing Group. (2020, September 17). TPM 2.0 Keys for Device Identity and Attestation. Retrieved May 16, 202, from https://trustedcomputinggroup.org/wp-content/uploads/TCG_IWG_DevID_v1r2_02dec2020.pdf.

Tschofenig, H., & Fossati, T. (2016, July). *rfc7925*. IETF. Retrieved November 15, 2021, from https://datatracker.ietf.org/doc/html/rfc7925.

Tucker, C. T. (2020, May 11). What is DNS? Free Code Camp. https://www.freecodecamp.org/news/what-is-dns/.

Uppalapati, K. (2019, May 16). How IoT Protocols and Standards Support Secure Data Exchange in the IoT Ecosystem? Kellton Tech. https://www.kelltontech.com/kelltontech-blog/internet-of-things-protocols-standards.

Uribe, F. U. (2017, June). The Classification of Internet of Things (IoT) Devices Based on Their Impact on Living Things. Research Gate. https://www.researchgate.net/publication/327248925_The_Classification_of_Internet_of_Things_IoT_Devices_Based_on_Their_Impact_on_Living_Things.

Uribe, F. U. (2018, January 1). The Classification of Internet of Things (IoT) Devices Based on Their Impact on Living Things. SSRN. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3350094.

Vilmos, A. (2021, February 9). *Front-End Access Control: The Capability-Based Access Control for IoT*. IoTAC. Retrieved May 17, 2022, from https://iotac.eu/front-end-access-control-a-new-solution-for-capability-based-access-control/.

W3Techs. (n.d.). Usage Statistics of Site Elements for Websites, September 2021. Retrieved September 28, 2021, from https://w3techs.com/technologies/overview/site_element.

WebSocket Security - Cross-Site Hijacking (CSWSH). (2020, October 1). AppCheck. https://www.google.com/url?q=https://appcheck-ng.com/cross-site-hijacking/&sa=D&source=editors&ust=1632836489050000&usg=AOvVaw37oagh73unR8Ime7vTFz-O.

What is DTLS and how is it used? (2021). Hack Control. https://hackcontrol.org/blog/what-isdtls-and-how-is-it-used/.

Wiedmann, F. (2021, July 1). *What is a trusted execution environment (TEE) and how can it improve the safety of your data?* Piwik PRO. Retrieved May 24, 2022, from https://piwik.pro/blog/what-is-a-trusted-execution-environment/.

Wi-Fi Alliance. (n.d.). *Wi-Fi CERTIFIED HaLow | Wi-Fi Alliance*. Retrieved December 13, 2022, from https://www.wi-fi.org/discover-wi-fi/wi-fi-certified-halow.

Zero Outage - Industry Standard. (2020, June 23). *Approach and Principles*. Zero Outage. Retrieved November 15, 2021, from https://zero-outage.com/the-standard/security/security-taxonomy-for-iot/approach-and-principles/.

ZeroSSL. (n.d.). SSL Protection For Anyone Fast. Reliable. Free. Retrieved June 9, 2022, from https://zerossl.com/.

Zhou, H., Cam-Winget, N., Salowey, J., & Hanna, S. (2014, May 7). *RFC 7170: Tunnel Extensible Authentication Protocol (TEAP) Version 1*. IETF Datatracker. Retrieved December 13, 2022, from https://datatracker.ietf.org/doc/rfc7170/.

Z-Wave Alliance. (2015, January 5). *Alliance Overview*. Retrieved November 16, 2021, from https://z-wavealliance.org/z-wave-alliance-overview/.

Z-Wave Alliance. (2020, September 24). *Z-Wave Plus™ Certification*. Retrieved November 15, 2021, from https://z-wavealliance.org/z-wave_plus_certification/#:%7E:text=Z%2DWave%20Plus%E2%84%A2%20is,or%205th%20Generation%20Z%2DWave.

CIS® Center for Internet Security®

cisecurity.org
info@cisecurity.org
518-266-3460
Center for Internet Security
@CISecurity
TheCISecurity
cisecurity