element14 ESSENTIALS

Op Amps III: IoT Security

1. Introduction

In recent years, IoT (Internet of Things) has emerged as a network of connected physical devices using the Internet for seamless data exchange. While real-time access to information enables opportunities for more autonomous operation of physical assets, the exchange of a large volume of data increases security risks, including data and privacy breaches, product tampering through remote control, and data attacks. Security remains a top concern with the IoT. Moreover, the importance of IoT security extends to the home. It is easy to imagine a situation in a smart connected home where private information, home appliances, or security locks can get hacked. Security issues pose increased risks to manufacturers in the form of compromised products or services and product recalls, leading to a decline in the reputation of the company in concert with reduced revenues. This Essentials module covers the fundamentals of IoT security, including types of security threats, standard security algorithms, error correction, cryptography, and security from end-nodes through applications.

2. Objectives

Upon completion of this module, you will be able to:

- Explain why security is essential in IoT
- List the main types of security threats
- Identify the standard security algorithms
- Describe the difference between symmetric and asymmetric cryptography
- Discuss IoT communication protocols based on their security features

3. Overview

- Cybercrime, cyberwar, and cyberterrorism
- Data and privacy breaches
- Botnets, ransomware, and other malware
- DDoS attacks and other thypes of sabotage
- Product malfunction through remote control
- Theft of intellectual property (IP)

The Internet of Things (IoT) is now an everyday reality. From utility infrastructures, industrial control applications, and medical equipment to smart-home appliances, fitness bracelets, and bike-sharing services – not to mention smartphones and connected vehicles – today's IoT is all around us.

Table 1: Typical IoT security risks

Despite the growth of IoT, connected devices are potential targets for those seeking unauthorized access to the network, malicious control of the device, or theft of collected data. The increasing complexity of IoT ecosystems compounds these dangers, since devices supplied by many vendors and offering varying levels of security can lead to unexpected weaknesses, unanticipated results, and unsafe operation. For example, any type of IoT-connected equipment, be it an air compressor, a washing machine, or a passenger vehicle can be remotely controlled or triggered to operate in an unsafe manner. What's more, the smart grids that govern the distribution of energy and water can, if tampered with or illicitly accessed, pose a significant threat to human health and safety.

It's one thing to know that every IoT device needs a baseline of protection, but quite another to know what's the best way to implement that security. After all, not every device faces the same risk profile – a smart action figure, connected to a home WiFi network, is different from a control mechanism in a nuclear plant – and there's the ongoing need to balance the type of protection with the cost of implementing and maintaining that protection.

One place to start when defining security for IoT devices is to consider the operating environment. How will devices interact with the systems around them, and what specific risks are associated with those interactions? For example, which IoT devices will upload data to the cloud, and which cloud service will receive the data? Who will control each device? What hardware will be used to drive devices, and what software will be allowed to run, and when? Will the IoT device co-exist with potentially sensitive equipment and applications? By characterizing each device as part of a larger ecosystem, and anticipating the threats within that particular ecosystem, it can be easier to know what kind of protection will work best, and how to deploy it.

4. Classification of Security Threats

Security threats are typically classified in the following categories:

Confidentiality Breach: This occurs when the confidential and private information of a subscriber is available to a third party without the subscriber's consent. For example, if a GPS tracker installed in someone's vehicle gets compromised, the location data retrieved from the tracker can be used to track the person and can pose a serious threat to the person's privacy.

Theft of Service: Hackers can enable unauthorized access to services by using security flaws or weaknesses in implemented protocols. A web API that does AI analytics on the provided data can be exploited and used by an attacker if its API gets exposed due to inadequate security.

Data Integrity: An unauthorized user can drop unwanted messages into the network or can take over deployed devices. IoT nodes involving sensors for critical applications in a manufacturing plant can be targeted by hackers to disrupt the plant operation.

Availability: Hackers manage to hack into a network of devices and target to overflow a server or cloud application with random requests from the connected nodes. Such an attack is known as a "Denial of Service" attack, which can ultimately crash a server or an application.

5. Standard Algorithms

There are a variety of standard security algorithms. Here is a list of the most important ones:

- **RSA:** RSA, named after its inventors (Ron Rivest, Adi Shamir, and Leonard Adleman), is an asymmetric encryption algorithm. RSA involves public and private keys, computed using two randomly chosen prime numbers. The public key is used to encrypt the message, and the private key is used to decrypt the message.
- **ECDSA (Elliptic Curve Digital Signature Algorithm):** ECDSA is often used to verify the authenticity of messages. A simple message can be signed using a private key and any receiver having the public key of the sender can verify if the message has originated from the sender or not.
- **SHA (Secure Hash Algorithm):** SHA takes data input and produces hash or the message digest. SHA-0 and SHA-1 use 16-Bit hashing, whereas SHA-2 involves a set of two functions with 256-bit and 512-bit technologies, respectively. This algorithm allows taking data of any size and turning it into a string of a specific, predefined size. The

resulting string is called a “Hash,” and the process of applying the hash function to random inputs is called “hashing.” Hashing is used to prove that a set of data has not been tampered with.

- **CRC (Cyclic Redundancy Check):** CRC is often used for error detection and correction in a digital communication network and in storage devices. Blocks of data get padded with short data bytes based on the remainder of a polynomial division of their content. In the receiver side, the CRC bytes are again calculated from the received data and compared with the received CRC. If they do not match, the received data can be discarded or corrected.
- **AES (Advanced Encryption Standard):** AES is a symmetric key block cipher algorithm based on a substitution-permutation network (SPN) and uses three fixed 128-bit block ciphers with cryptographic key sizes of 128, 192, and 256 bits.
- **ECJPAKE (Elliptic Curve Password Authenticated Key Exchange by Juggling):** ECJPAKE is a password-authenticated key agreement protocol without using Public Key Infrastructure (PKI) for authentication. It can establish a private and authenticated channel on top of an insecure network solely based on a shared password.

6. Error Correction and Cryptography

Whenever data gets transferred from a source to a recipient, the major security concerns are data integrity and data security. Data integrity focuses on keeping the data intact during the process of transfer, which is susceptible to communication channel disruption. Error Detection and Correction techniques maintain data Integrity. Parity bit check, CRC check, Hashing, and Polar Codes are a few of the popular techniques used. Data security focuses on encrypting the data to avoid misuse by hackers. Data security uses cryptographic techniques and the science of encoding and decoding (enciphering/deciphering) data. Based on the key used, cryptography can be symmetric or asymmetric.

- **Symmetric cryptography:** the same key is shared between the sender and the receiver. Sharing of the key can pose a security risk while it is shared between the communicating parties. The Diffie–Hellman key exchange method is a secure way of exchanging keys over an insecure channel before communication.
- **Asymmetric or public key cryptography:** a pair of public and private keys is used. Both the public and private keys are mathematically related but different. The sender encrypts the data to be transferred using the public key of the receiver, and the encrypted data can only be decrypted using the private key of the receiver which is never shared with others. Public key cryptography is often used for authenticity.

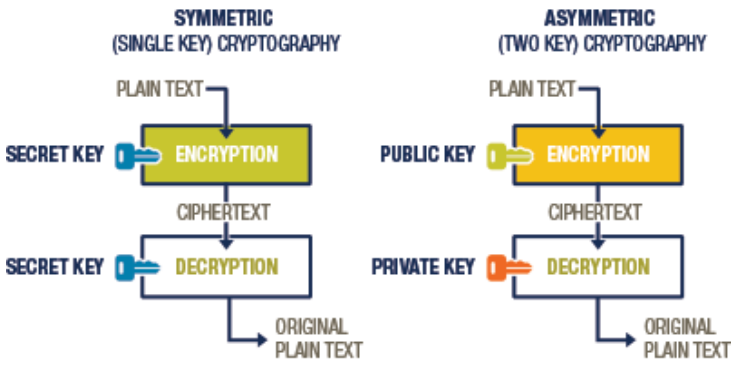

Figure 1: Comparison of Symmetric and Asymmetric Cryptography

Any application that is IoT-enabled has many stages of data transfer involving communication protocols right from the node to the end user interface. In the next section, we will discuss security in each of these stages.

7. Node Level Security

At the device or controller, end developers should consider many security aspects in order to make data more secure. Since the device can be easily accessible, it needs to be protected physically so that no one can tamper with the firmware or access its open communication ports. One-time Programmable ROMs can be used in end devices to avoid hacking; however, a disadvantage with OTPRs is that their firmware cannot be upgraded in some applications. Microcontrollers can encrypt data programmatically, but the encryption algorithms are computation intensive. Some of the more recent processors and controllers provide dedicated encryption/decryption engines. Device-debugging interfaces such as UARTs can be compromised easily and, hence, should be secured against unauthorized access.

Many IoT devices use processors with the support of operating systems, and thus security at the OS level is also critical. Anyone having physical access to the hardware during the manufacturing process, installation, or deployment can pose a threat to security. During production and distribution, ICs are subject to malware injection, key capture, and counterfeiting. To avoid such issues in manufacturing, proper tamper resistant processes and mechanisms should be set up.

Security ICs are designed to provide a safe, self-contained environment for staging and executing the authentication tasks that are essential to safe operation in the IoT. The ICs are designed to create a barrier that isolates critical security processes from IoT application software and its associated complexity, so processes can run in a protected, “sandboxed” environment. Security ICs integrate secure, nonvolatile memory into the IC, so keys are securely transported and managed. Specialized key management processes supported

by silicon injection are available for secure manufacturing environments. As a result, security credentials are protected from device creation to decommissioning. The origin of uploaded data can be trusted, the source of commands used by real-time automation systems can be considered reliable, and any private information exchanged with a device stays protected while in transit.

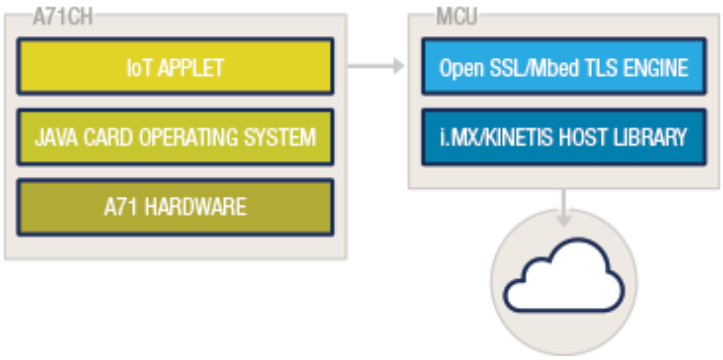

Figure 2: A71CH block diagram

Products such as A71CH and A1006 are good examples of Security ICs. NXP's A71CH Trust Anchor uses a "plug and trust" approach with its microcontrollers to save secret keys locally for asymmetric cryptography solutions. A1006 is a tamper-resistant authenticator IC; it comes with a private key and the corresponding certificate containing the Public Key, a unique identifier and the customer's product fields and usage. The keys are static and unique to each A1006. The Elliptic Curve Digital Signature Algorithm (ECDSA) based on the NIST P-224 curve and the SHA-224 digest hash with the customer's desired certificate authority key are used to sign the certificates digitally. For more information about these products, please visit the Related Components page of this learning module.

Connectivity is a basic ingredient for IoT operation, and when it comes to securely integrating an IoT device to a network, an infrastructure, or a service in the cloud, the first order of business is to protect the credentials that ensure the integrity of that connection and keep data confidential. Now let's look at the network, infrastructure, and cloud services to see how IoT security must be implemented as a coordinated system.

8. Last Mile Communication Security

Wireless technologies are widely used in IoT end nodes. As air is the medium for data transmission, anyone in the range of a Wi-Fi hotspot/modem/Access Point can attempt to connect/eavesdrop making them extremely vulnerable. Hence, security in the last mile (from the telecommunication backbone to a home or business) is critical.

Wi-Fi

Wi-Fi is a wireless Local Area Networking technology that operates on the 2.4GHz or the 5.8GHz ISM band based on the 802.11 standards. Wi-Fi has evolved with many security protocols to provide seamless and secure connectivity for users. Some of them are discussed here:

WEP (Wired Equivalent Privacy): WEP uses the stream cipher RC4 for confidentiality and CRC32 checksum for data integrity. Until 2004, when IEEE deprecated WEP, it was the encryption standard for wireless. WEP used two types of authentication: Open System, and Shared Key. In the Open System, the client has to have the correct key while decrypting and extracting the data, but need not provide any credentials to the access point during authentication and when joining the network. In Shared Key authentication, the client goes through a challenge where the access point decrypts the echo of text sent by it to the client to verify if the client has the right keys. There are many instances where WEP security was penetrated and, hence, not considered secure enough. In 2005, FBI agents were able to crack through using openly available sniffing tools by capturing packets to recover the WEP key. In 2006 A. Bittau, M. Handley, and J. Lackey published a paper titled “The Final Nail in WEP’s Coffin” on insecurity involved in WEP. It describes a vulnerability in WEP which allows an attacker to send arbitrary data on a WEP network after eavesdropping a single data packet. The paper also included techniques to decrypt data packets in real-time.

WPA (Wi-Fi Protected Access): The WPA protocol adopts TKIP (Temporal Key Integrity Protocol) and employs a per packet key leading to the generation of a new 128-bit key for each packet and thus prevents the types of attacks that were happening with WEP. It also implements a Message Integrity Check (MIC) in place of CRC and protects from attackers who try to modify packets and resend them. WPA2 replaced WPA with the implementation of CCMP, an AES-based encryption mode with strong security. WPA2 uses 192-bit encryption and individualized encryption for each user. Implementation of this protocol requires a testing and certification form from the Wi-Fi Alliance and is mandatory for all devices to bear the Wi-Fi trademark. In 2011 Stefan Viehböck’s “Brute forcing Wi-Fi Protected Setup” paper revealed a significant flaw in routers equipped with WPS where attackers can use the eight-digit PIN for WPS to recover WPA/WPA2 passwords. Another attack was Hole196, where an attacker from within the network having a Group Temporal Key (GTK) conducted a denial-of-service attack. Within security attacks to Wi-Fi networks, the recent one is the KRACK (Key Reinstallation Attack) attack which got published in CCS’17 by M. Vanhoef and F. Piessens. This paper describes a key reinstallation attack that can be carried out against the four-way handshake used by Wi-Fi networks to generate fresh session keys. The attack is achieved by tricking the victim to reinstall an already-

in-use key by manipulating and replaying handshake messages. Manufacturers later addressed this by releasing software patches for the Wi-Fi devices, but not all devices were lucky enough to get one.

EAP (Extensible Authentication Protocol): an authentication protocol framework specifying methods for secure key distribution and usage. Many different authentication methods, like dynamic key distribution and digital certificates, can be used within the framework. Wi-Fi has adopted various EAP implementations like EAP-MD5, EAP-MSCHAPv2, LEAP, EAP-TLS, EAP-TTLS, and MSCHAPv2 for implementation along with WPA2. To be released in later 2018, WPA3 will replace WPA2 and is supposed to increase security further, addressing a few of the possible attacks. The data transfer between the access point and the client will be entirely encrypted whether the network is public or open. It will also have a particular 192-bit encryption as suggested by the US government for military or government use purposes.

LORA/LORAWAN

LoRa is a long-range, low-power wireless communication technology using LPWAN in the ISM band. It fits perfectly with battery-powered IoT end nodes. LoraWAN is a protocol specification for a media access control layer for managing communication between LoRa nodes and gateways. The gateways are connected to cloud services using IP connectivity and nodes can send data to multiple gateways.

LoRa uses two layers of encryption. They are:

- A unique 128-bit Network Session Key shared between the end-device and network server
- A unique 128-bit Application Session Key (AppSKey) shared end-to-end at the application level

All the authentication and keys use the AES algorithm. Encryption at the application and network level enables data transmitted to be completely private, even from the network operator. The keys can be activated by Personalization (ABP) during production or commissioning or can get activated and modified Over the Air (OTAA). OTAA is considered more secure as compared to ABP, as the keys are negotiated during the joining process.

A LoRaWAN device comes with a 128-bit AppKey and an EUI-64 based DevEUI global unique identifier. The AppKey and DevEUI are used during the device authentication process. From the AppKey, two AES-128 session keys are generated, namely NwkSKey and AppSKey, and are used to encrypt traffic between nodes and servers. NwkSKey verifies packet authenticity and integrity, while AppSKey is used to encrypt and decrypt application payloads.

In 2017, a paper published by E. Aras, G. Ramachandran, P. Lawrence, and D Hughes discussed the different types of attack that can be carried out in a LoRa network. If anyone has physical access to end nodes, the device can be compromised easily by intercepting the communication lines between the host microcontroller and LoRa module. Jamming can also be used to flood messages in a particular frequency band used by LoRa, leading to interference and data loss.

Bluetooth Low Energy (BLE)

Bluetooth Low Energy is a Personal Area Network (PAN) technology, targeting battery-operated devices. It uses the same frequency band (2.400–2.4835 GHz ISM) as classical Bluetooth.

Security Mode 1: This mode enforces security using encryption, and contains four levels:

- Level 1 - No Security (No authentication and no encryption)
- Level 2 - Unauthenticated pairing with encryption
- Level 3 - Authenticated pairing with encryption
- Level 4 - Authenticated LE Secure Connections pairing with encryption

Security Mode 2: This mode enforces security using data signing, and contains two levels:

- Level 1 - Unauthenticated pairing with data signing
- Level 2 - Authenticated pairing with data signing

Each connection starts without a security mode by default, but later gets upgraded to any security level through authentication procedures.

In September 2017, Armis Labs published the details of an attack named BlueBorne, which allowed a hacker to take control of Operating-System-based devices like mobile, desktop, and IoT devices. The attack was carried out by locating active Bluetooth devices in the vicinity including devices set in non-discoverable mode. After locating active devices, the attacker fetched the MAC address of the device, and by probing in to find the operating system used by the target device, the attacker exploited vulnerabilities in the implementation for the target device. This was addressed by device manufacturers with the release of respective patches.

SIGFOX

Sigfox is an LPWAN, ultra-narrowband communication technology using the 868- and 902-MHz ISM frequency band.

Sigfox Ready devices come with a unique symmetrical authentication key. Messages to/from the device contain a cryptographic token computed from the

authentication key, which verifies the authenticity of the origin of the message and helps in data integrity. The authentication key is stored locally in the end device, and as it is unique for every Sigfox ready device, compromising any node will not affect others. Communication between the Sigfox core network and application servers relies on Internet protocols such as VPN or HTTPS. The Sigfox network handles message replay attempts by employing a sequence counter to each message. The authentication token covers the integrity of the message counter.

By default Sigfox messages are not encrypted, as this would increase the computational overhead. However, if the application requires it, Sigfox allows users to implement their own encryption technique or implement the solution provided by the Sigfox network.

Communication between a Sigfox base station and the Sigfox cloud uses VPN and SSL encryption. Each message originating from Sigfox end nodes contains a unique signature generated from a locally stored key, thus providing the authenticity of the message. The message also includes a unique packet number which saves the network from message replay. While transmitting from the end node, it transmits the same message three times in three different frequencies to ensure the message is delivered to the Sigfox cloud; this also protects it from jamming, as the frequency of transmission is random.

Thread

Thread is an IPV6-based 6LoWPAN communication technology, designed for smart home applications.

ECJPAKE (Elliptical Juggling Password-Authenticated Key Exchange) is used as a primary security measure in the Thread Network with NIST P-256 elliptic curve. It uses the Diffie-Hellmann elliptic curve algorithm for key agreement and Schnorr signatures as a NIZK (Non-Interactive Zero-Knowledge) proof mechanism to authenticate two peers and to establish a shared secret between them based on the passphrase.

A Thread network also uses a network-wide key at the MAC layer to protect the data frames, and the key gets shared with all nodes in the network. It saves the network from eavesdropping by external devices without the knowledge of the key. Though it is used to differentiate between the authenticated and non-authenticated nodes in the network, if one device gets compromised the key will also get exposed as the key is shared with all devices. The key is delivered to new joiners using a Key Encryption Algorithm. Additionally, it uses SSL, TLS and DTLS encryptions and, since the Thread network is based on IPV6, all communications are AES encrypted.

ZIGBEE
ZigBee implements the IEEE 802.15.4 defined security model, which includes controlling access to network devices (authentication), encryption (symmetric-key cryptography) and message integrity checks (MIC), ensuring transmitted frames are safe. Its symmetric key cryptography uses three different types of keys for the association to a network or group of devices or peer-to-peer communication:

- **Master Key:** pre-installed by manufacturers to the device
- **Link Key:** used to encrypt point-to-point communication data at the application level and confined to the nodes. This key is different for each pair of nodes in the communication, and it minimizes the risk of distribution of the master key in the network.
- **Network Key:** used at the network level and known to all the nodes in the network

KillerBee is a Python-based framework that performs attacks and exploits the vulnerabilities of ZigBee-based devices by sniffing keys. It can also be used for injecting traffic, packet manipulation, and decoding packets. In 2016, a paper was published explaining the attack targeted on Philips Hue light bulbs. Researchers attacked the bulb with a worm and exploited the hard-coded symmetric keys which let them make it turn on/off. ZigBee claimed that the vulnerability was not part of the ZigBee standard, but rather an internal implementation error made by Philips.

Z-Wave
Targeted towards Home Automation or Smart Home products, Z-Wave is a low-power RF communication technology for low latency and small data packets, and supports mesh networking.

In 2016 the Z-Wave Alliance introduced the S2 Security framework as a replacement for their initial S0 with an improvement in the key exchange mechanism. The S2 framework uses an Elliptic Curve Diffie-Hellman (ECDH) algorithm for key exchange. It uses an AES algorithm to generate all the keys. All new devices coming with Z-Wave support are mandated to have S2 Security framework as per guidance by the Z-Wave Alliance.

Researchers found a vulnerability in the S2 security framework where they were successful in downgrading the security framework used by S2-based devices to S0, so the device can be hacked using a vulnerability that exists in S0. The cause of this was the backward compatibility of both versions. Devices that supported both versions of key-sharing mechanisms could be forced to downgrade the pairing process from S2 to S0.

Near Field Communication (NFC)
NFC technology operates at a frequency of 13.56MHz, supporting up to 420kbps of data transfer speed, and a short communication range of less than 10cm. NFC

is integrated into devices like smartphones and handheld devices and is used for authentication and pairing for contactless payment transactions.

NFC manufacturers can use any encryption algorithm to handle security attacks. Data integrity is handled by using the NFC forum signature in the tags, which are similar to digital signature to authenticate tags. Though NFC uses RF communication, the maximum 10cm limitation of the reader from the device in similar orientation provides built-in security.

9. Gateway Server Communication Security

End nodes connected in a personal area network (PAN) not having IP connectivity can't send data directly to web servers or the cloud. They need gateways that can connect the PAN to the Internet by routing the traffic. If devices are connected using Ethernet or Wi-Fi, this is not an issue, though. Since the gateway connects to the servers using TCP/IP or a UDP connection, the communication must also be protected. In this section, let's walk through some protocols used for gateway-server secured communication.

HTTP/HTTPS
The Hypertext Transfer Protocol (HTTP) is an application layer protocol working on the client-server model. It is used to transfer hypermedia documents such as HTML using commands like GET, POST, PUT and DELETE to exchange data.

HTTP traffic by default uses plain text; hence, anyone having physical access or sniffing tools can get hold of the information in the packet. To mitigate such issues, an advanced secure version HTTPS emerged, which encrypts all the traffic using TLS (Transport Layer Security) or SSL (Secure Sockets Layer). In HTTPS, the server issues an X.509 certificate which has to be used by the client, and it usually contains the public key for encryption. Data from the client is encrypted using the public key of the server and can only be decrypted by using the private key, which the server never discloses. A website or web service has to be served entirely over HTTPS rather than just the portion containing sensitive information to avoid exposure to attacks.

HTTP is prone to distributed denial-of-service (DDoS) attacks, where an attacker floods the server with traffic using HTTP GET or POST calls from interconnected and internet-connected devices. This requires the attacker to have a deep understanding of the targeted server or the web application. Attacks using HTTP GET requests are more straightforward, whereas an attack with POST requests requires fewer resources for the attacker. A POST attack may include parameters to trigger resource-intensive server-side processing.

In 2014, the POODLE (Padding Oracle On Downgraded Legacy Encryption) attack was published, which takes advantage of the protocol negotiation feature built into SSL/TLS and the SSL3.0 block padding vulnerability. An attacker can

carry out a man-in-the-middle (MITM) attack and force the server to get into SSLV3 and carry out the POODLE attack. The vulnerability exists in Cipher Block Chaining mode. SSL 3 uses padding to fill extra space if the data in the last block is not a multiple of the Block Cipher's size. An attacker can decipher an encrypted block by changing the padding values and checking out the response from the server. With a maximum of 256 bytes of SSL 3 requests per byte, the attacker can decrypt the values.

MQTT

Message Queuing Telemetry Transport (MQTT) is an ISO-standard publish-subscribe messaging protocol. It is designed for bandwidth-constrained networks, and its code footprint for implementation is very low. MQTT works on top of TCP/IP. To operate, a broker needs to be implemented on the cloud server. Devices connected through MQTT either subscribe or publish to different topics, which are managed by the broker.

Based on priority, MQTT provides three types of Quality of Service (QoS):

- At most once delivery
- At least once delivery
- Exactly once delivery

By default, MQTT doesn't apply any encryption to the payloads, and so it relies on TLS and SSL or payload encryption. If the username and password authentication are used to authenticate users or nodes in the network, the use of TLS should be considered. If TLS is not possible due to constrained resources, a user can look into encrypting the payload being transmitted using MQTT. If not the payload, then at least the authentication credentials should be encrypted by using any encryption method like hashing.

COAP

Constrained Application Protocol (COAP) is a lightweight RESTful protocol specifically for M2M communication for IoT applications. COAP security uses DTLS (Datagram Transport Layer Security), and by default it uses 3072-bit RSA equivalent keys.

10. Server / Cloud level Security

IoT devices are connected to web services that run on dedicated servers or cloud networks. Data generated from the connected devices is stored on cloud servers. This data is used to make decisions and presented to the user using web apps or mobile apps. The security of private data stored in databases, servers, and applications is of utmost importance when designing IoT applications.

A Cloud Server for IoT is no different from traditional servers in terms of their hardware and hosting environment. An IoT server might process millions of bytes

of data sourced from a large number of nodes connected to the server. As the amount of data to be processed is huge, an IoT server can often be distributed and can use specific databases and scripting languages. Many service providers already provide secured Servers as a Platform-As-A-Service (PAAS) or Software-As-A-Service (SAAS). PAAS is a complete development and deployment environment in the cloud, whereas SAAS provides software to end users on a license basis.

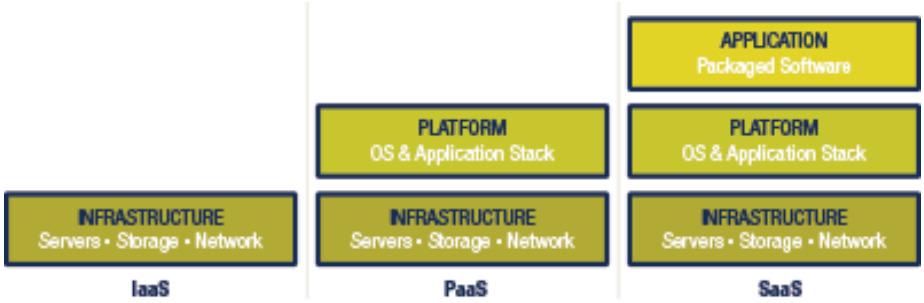

Figure 3: Comparison of Iaas, PaaS and SaaS

As the servers are often located in data centers and are physically inaccessible, SSH (Secure Shell) is used by system admins to connect to remote servers. If an attacker manages to get SSH access to the server, he can threaten a serious data breach. SSH keys based on public key cryptography can be used to make the communication secure.

A server typically opens up different ports from the server to enable multiple services. A high number of opened ports results in a larger attack surface that can be exploited by an attacker. Some of the open ports could be admin ports needing additional security. A firewall which limits the exposed services and ports has to be used with a server to prevent unwanted access.

SSL or TLS has to be used with exposed web services. All the traffic to the server gets encrypted, which helps in avoiding attacks like “Man in the Middle.” The system admins should also carry out service and file auditing at regular intervals to check if everything in the server is running as expected, or if an attacker has compromised any service or application.

Virtual Private Networks (VPNs) can be used by server admins to make the communication between the server and the admin private and secured. VPNs help in securing the connection between machines and presents the connections as if they are on a private local network. It again adds an extra layer of security.

11. Application Level Security

PC, mobile, and web applications are usually exposed to end users, and thus become an easy target surface for attackers. For native applications such as

Android/iOS/ Windows, security has to be considered strictly on the device locally. An attacker may try to decompile source codes and try to patch the software with malicious code. If an application has user input and database interaction, it has to be protected against SQL injection. User inputs can be primary targets for hackers, so an appropriate filtration algorithm has to be deployed.

Programming languages such as Java, Android, PHP, and .NET also offer security features and should be considered by a developer when designing the app. A secure authentication system for identification and authentication of the user and other apps accessing the application has to be implemented with a username and password. The password and other user credentials should be encrypted and stored in a database or files. Data transferred from the application to cloud services can travel through open public networks (like Wi-Fi in a coffee shop) and might not have encryption. Hence, any API exposed to applications should use SSL or TLS. Two-phase authentication can also be deployed for user login activities to the application. Native applications should also be protected against buffer overflow attacks.

An API used by mobile and the native application has to be well encrypted, and a good authentication service such as OAuth 2.0 has to be implemented. The applications should support OTA (Over the Air) updates so that they can be patched as soon as a new vulnerability is found. Data stored by such applications locally has to be encrypted.

12. Application Example

To conclude this learning module, let's consider an IoT security example. We will discuss security in a popular IoT platform, Amazon's AWS, focusing on node security. The secured AWS IoT node will have:

A71CH Security IC supporting the following ECC (Elliptic-Curve Cryptography) functionalities:

- Elliptic Curve Digital Signature Algorithm
- Elliptic Curve Diffie-Hellman or Elliptic Curve Diffie-Hellman Exchange
- Secure storage, generation, insertion, or deletion of key pairs (NIST-P256 elliptical curve)

A71CH can be used with end nodes to have a secure connection to any IoT services. The A71CH will be used to hold the credentials of the communication.

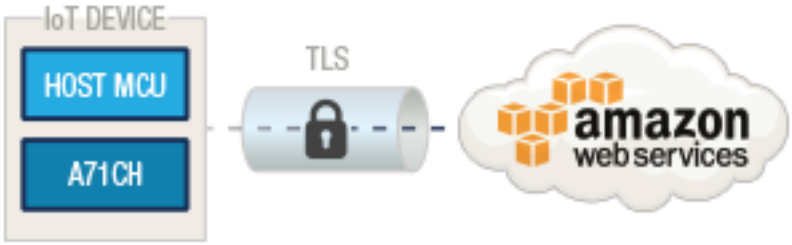

Figure 4: IoT Device with A71CH secure element and AWS Cloud

OEM and AWS Cloud Credentials
The IoT device manufacturer shall have an intermediate CA certificate with the keys. They can use their own Public Key Infrastructure or a third party CA to issue the intermediate CA. The intermediate CA Certificates are used to register the device to the AWS cloud as well as for validation of the device.

Device Credentials:
Each device intended for cloud connection should have public and private key pairs, as well as its digital certificate issued by the manufacturer. That's where A71CH comes into the picture; it stores these credentials. The device must also have the intermediate CA certificate which can be stored in the A71CH GP storage.

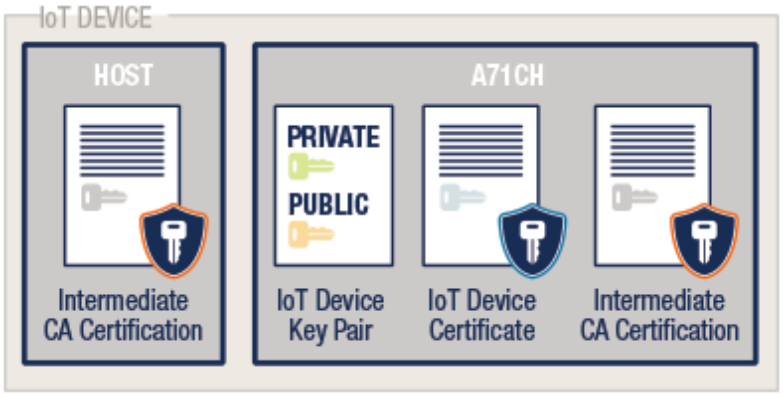

Figure 5: Public and Private Key Pairs

Transport Layer Security protocol (TLS):
Communication between IoT devices and the Cloud can be secured using transport layer protocols like TLS, which helps in providing a secure connection over an unsecured network. In the case of TLS, before the communication between the device and cloud occurs, they negotiate regarding encryption, referred to as a "handshake." The handshake is responsible for authentication and key exchange. The handshake involves negotiation of the TLS version to be used, selection of the cipher suite, the exchange of digital certificates and the sharing of a symmetric shared key for encryption.

The A71CH security IC supports the TLS Handshake Protocol version 1.2 with the following options:

- Pre-Shared Key Cipher Suites for TLS as described in [RFC4279]: A set of cipher suites for supporting TLS using pre-shared symmetric keys (TLS_PSK_WITH_xxx).
- ECDHE_PSK Cipher Suites for TLS as described in [RFC5489]: A set of cipher suites that use a pre-shared key to authenticate an Elliptic Curve Diffie-Hellman exchange with Ephemeral keys (TLS_ECDHE_PSK_WITH_xxx).

Transport Layer Security Software Libraries:
There are several full-featured TLS software libraries that can be used in both server cloud and IoT devices, such as OpenSSL, mbedTLS, and WolfTLS, among others. OpenSSL [OPEN_SSL] is an open-source implementation of the SSL/TLS protocol. It is written in the “C” language, although there exist several wrappers so you can use this library in other languages. It implements all the cryptography functions needed, and it is widely used.

Starting with OpenSSL 0.9.6, an “Engine interface” was added, allowing support for alternative cryptographic implementations. This Engine interface can be used to interface with external crypto devices such as, HW accelerator cards or security ICs like the A71CH. The OpenSSL toolkit including an A71CH OpenSSL Engine is available as part of the A71CH Host software package [A71CH_OPENSSL_ENGINE]. The A71CH OpenSSL Engine gives access to several A71CH features via the A71CH Host Library not natively supported by OpenSSL implementation.

The Engine links the OpenSSL libraries to the A71CH Host API and overwrites some of the native OpenSSL functions in order to include the use of the A71CH crypto functionality such as sign, verify, and key exchange operations or random messages generation that can be used (for instance) during the TLS Handshake protocol. The A7CH OpenSSL Engine is fully compatible with the i.MX6UltraLite embedded platform.

AWS IoT and Just-in-Time Registration (JITR)
As a first step, the A71CH ICs (see Related Components) provided by NXP need to be provisioned by a programming facility in charge of injecting die-individual credentials. The manufacturer needs to register and activate his intermediate CA in the AWS platform. Then, in order to communicate in the AWS IoT network, each IoT device needs to be known via its device certificate. Thus, each IoT device certificate needs to be registered into the AWS IoT platform.