GSMA

IoT SECURITY GUIDELINES

for Endpoint Ecosystems

Supported by

GLOBALPLATFORM® GSA IoTA IoT Alliance Australia IoT WORLD ALLIANCE® MEF MOBILE ECOSYSTEM FORUM

oneM2M simalliance Security | Identity | Mobility

IoT Security Guidelines Endpoint Ecosystem

Version 2.2

29 February 2020

This is a Non-binding Permanent Reference Document of the GSMA

Security Classification: Non-confidential

Copyright Notice

Disclaimer

The GSM Association ("Association") makes no representation, warranty or undertaking (express or implied) with respect to and does not accept any responsibility for, and hereby disclaims liability for the accuracy or completeness or timeliness of the information contained in this document. The information contained in this document may be subject to change without prior notice.

Antitrust Notice

The information contain herein is in full compliance with the GSM Association's antitrust compliance policy.

Table of Contents

1 Introduction

1.1 Introduction to the GSMA IoT Security Guideline Document Set

This document is one part of a set of GSMA security guideline documents that are intended to help the nascent "Internet of Things" (IoT) industry establish a common understanding of IoT security issues. The set of non-binding guideline documents promotes methodology for developing secure IoT Services to ensure security best practices are implemented throughout the life cycle of the service. The documents provide recommendations on how to mitigate common security threats and weaknesses within IoT Services.

The structure of the GSMA security guideline document set is shown below. It is recommended that the overview document 'CLP.11 IoT Security Guidelines Overview Document' [1] is read as a primer before reading the supporting documents CLP.12 [2] and CLP.13 [3] (this document).

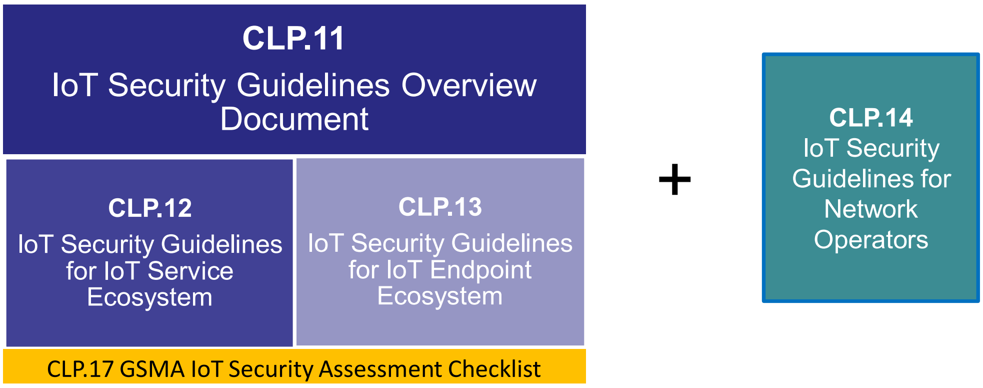

CLP.11
IoT Security Guidelines Overview Document
CLP.12
IoT Security Guidelines for IoT Service Ecosystem
CLP.13
IoT Security Guidelines for IoT Endpoint Ecosystem
CLP.17 GSMA IoT Security Assessment Checklist
+
CLP.14
IoT Security Guidelines for Network Operators

Figure 1 - GSMA IoT Security Guidelines Document Structure

Network Operators, IoT Service Providers and other partners in the IoT ecosystem are advised to read GSMA document CLP.14 "IoT Security Guidelines for Network Operators" [4] which provides top-level security guidelines for Network Operators who intend to provide services to IoT Service Providers to ensure system security and data privacy.

1.1.1 GSMA IoT Security Assessment Checklist

An assessment checklist is provided in document CLP.17 [19]. This document enables the suppliers of IoT products, services and components to self-assess the conformance of their products, services and components to the GSMA IoT Security Guidelines.

Completing a GSMA IoT Security Assessment Checklist [19] will allow an entity to demonstrate the security measures they have taken to protect their products, services and components from cybersecurity risks.

Assessment declarations can be made by submitting a completed declaration to the GSMA. Please see the following process on the GSMA website:

https://www.gsma.com/iot/future-iot-networks/iot-security-guidelines/

1.2 Document Purpose

This document shall be used to evaluate the components of an IoT Service from the IoT Endpoint Device perspective. An Endpoint, from an IoT perspective, is a physical computing device that performs a function or task as a part of an Internet connected product or service. An Endpoint, for example, could be a wearable fitness device, an industrial control system, an automotive telematics unit or even a personal drone unit. All technologies used to drive the physical device shall be evaluated for security risks. The result is a practical set of design guidelines that allow the reader to identify and remediate almost all potential risks to the IoT Service.

The scope of this document is limited to recommendations pertaining to the design and implementation of IoT Endpoint Devices.

This document is not intended to drive the creation of new IoT specifications or standards, but will refer to currently available solutions, standards and best practice.

This document is not intended to accelerate the obsolescence of existing IoT Services. Backwards compatibility with the Network Operator’s existing IoT Services should be maintained when they are considered to be adequately secured.

It is noted that adherence to national laws and regulations for a particular territory may, where necessary, overrule the guidelines stated in this document.

1.3 Intended Audience

The primary audience for this document are:

- IoT Service Providers - Enterprises or organisations who are looking to develop new and innovative connected products and services. Some of the many fields IoT Service Providers operate in include smart homes, smart cities, automotive, transport, heath, utilities and consumer electronics.
- IoT Device Manufacturers - provide IoT Devices to IoT Service Providers to enables IoT Services.
- IoT Developers who build IoT Service on behalf of IoT Service Providers.
- Network Operators who provide services to IoT Service Providers.

1.4 Definitions

Term	Description
Access Point Name	Identifier of a network connection point to which an Endpoint device attaches. They are associated with different service types, and in many cases are configured per network operator.
Attacker	A hacker, threat agent, threat actor, fraudster or other malicious threat to an IoT Service. This threat could come from an individual criminal, organised crime, terrorism, hostile governments and their agencies, industrial espionage, hacking groups, political activists, ‘hobbyist’ hackers, researchers, as well as unintentional security and privacy breaches.
Cellular	Any 3GPP standardised mobile network technology (e.g. GSM, UMTS, LTE (inc LTE-M) and NB-IoT).

Term	Description
Cloud	A network of remote servers on the internet that host, store, manage, and process applications and their data.
Complex Endpoint	This Endpoint model has a persistent connection to a back-end server over a long-distance communications link such as Cellular, satellite, or a hardwired connection such as Ethernet. See section 3.
Embedded SIM	A SIM which is not easily accessible or replaceable, is not intended to be removed or replaced in the device, and enables the secure changing of profiles.
Endpoint	An IoT Endpoint is a physical computing device that performs a function or task as part of an Internet connected product or service. See section 3 for a description of the three common classes of IoT devices, and examples of each class of Endpoint.
Internet of Things	The Internet of Things describes the coordination of multiple machines, devices and appliances connected to the Internet through multiple networks. These devices include everyday objects such as tablets and consumer electronics, and other machines such as vehicles, monitors and sensors equipped with machine-to-machine (M2M) communications that allow them to send and receive data.
IoT SAFE	IoT SIM Applet For Secure End-2-End Communication. See GSMA IoT.04 [27] for further information.
IoT Service	Any computer program that leverages data from IoT devices to perform the service.
IoT Service Ecosystem	The set of services, platforms, protocols, and other technologies required to provide capabilities and collect data from Endpoints deployed in the field. See CLP.11 [1] for further information.
IoT Service Provider	Enterprises or organisations who are looking to develop new and innovative connected IoT products and services.
Network Operator	The operator and owner of the communication network that connects the IoT Endpoint Device to the IoT Service Ecosystem.
Organizational Root of Trust	A set of cryptographic policies and procedures that govern how identities, applications, and communications can and should be cryptographically secured.
Service Access Point	A point of entry into an IoT Service's back end infrastructure via a communications network.
Subscriber Identity Module	The smart card used by a mobile network to authenticate devices for connection to the mobile network and access to network services.
Trust Anchor	In cryptographic systems with hierarchical structure, a trust anchor is an authoritative entity for which trust is assumed and not derived.

Term	Description
Trusted Computing Base	A Trusted Computing Base (TCB) is a conglomeration of algorithms, policies, and secrets within a product or service. The TCB acts as a module that allows the product or service to measure its own trustworthiness, gauge the authenticity of network peers, verify the integrity of messages sent and received to the product or service, and more. The TCB functions as the base security platform upon which secure products and services can be built. A TCB's components will change depending on the context (a hardware TCB for Endpoints, or a software TCB for cloud services), but the abstract goals, services, procedures, and policies should be very similar.
Trusted Execution Environment (TEE)	An environment which runs alongside a rich operating system and provides security services to that operating system. There are multiple technologies which can be used to implement a TEE, and the level of security achieved varies accordingly.
UICC	A Secure Element Platform specified in ETSI TS 102 221 that can support multiple standardized network or service authentication applications in cryptographically separated security domains. It may be embodied in embedded form factors specified in ETSI TS 102 671.

1.5 Abbreviations

Term	Description
3GPP	3rd Generation Project Partnership
AC	Alternating Current
API	Application Program Interface
APN	Access Point Name
BLE	Bluetooth Low Energy
BT	Bluetooth
CLP	GSMA's Connected Living Programme
CPE	Customer Premises Equipment
CPU	Central Processing Unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
eUICC	Embedded UICC
FIB	Focused Ion Beam
GBA	Generic Bootstrapping Architecture
GPS	Global Positioning System
GSMA	GSM Association
IoT	Internet of Things
IP	Internet Protocol
ISM	Industrial, Scientific and Medical
LAN	Local Area Network
LPWA	Low Power Wide Area networks
LTE-M	Long Term Evolution for Machines

Term	Description
MCU	MicroController Unit
NB-IoT	Narrowband-Internet of Things
NVRAM	Non-Volatile Random Access Memory
OMA	Open Mobile Alliance
PAN	Personal Area Network
PSK	Pre-Shared Key
RAM	Random Access Memory
ROM	Read Only Memory
SCADA	Supervisory Control And Data Acquisition
SPI	Serial Peripheral Interface
SSH	Secure Shell
SIM	Subscriber Identity Module
SRAM	Static Random Access Memory
TCB	Trusted Computing Base
TTL	Transistor–Transistor Logic
UART	Universal Asynchronous Receiver/Transmitter

1.6 References

Ref	Doc Number	Title
[1]	CLP.11	GSMA IoT Security Guidelines Overview Document <LINK>
[2]	CLP.12	GSMA IoT Security Guidelines for IoT Service Ecosystem <LINK>
[3]	CLP.13	GSMA IoT Security Guidelines for IoT Endpoint Ecosystem <LINK>
[4]	CLP.14	GSMA IoT Security Guidelines for Network Operators <LINK>
[5]	OMA FUMO	OMA Firmware Update Management Object <LINK>
[6]	na	ST-LINK/V2 in-circuit debugger/programmer <LINK>
[7]	na	GSMA Mobile IoT Initiative <LINK>
[8]	na	Nmap Security Scanner <LINK>
[9]	CLP.03	GSMA IoT Device Connection Efficiency Guidelines <LINK>
[10]	FIPS PUB 140-2	Federal Information Processing Standards Publication 140-2 <LINK>

Ref	Doc Number	Title
[11]	na	EMVCo <LINK>
[12]	na	SIM Alliance - Open Mobile API <LINK>
[13]	GPD_SPE_013	GlobalPlatform Secure Element Access Control <LINK>
[14]	GPD_SPE_024	GlobalPlatform Trusted Execution Environment API Specification <LINK>
[15]	GPC_SPE_034	GlobalPlatform Card Specification <LINK>
[16]	ISO/IEC 29192-1	ISO Information technology -- Security techniques -- Lightweight cryptography <LINK>
[17]	TS 33.220	3GPP Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA) <LINK>
[18]	TS 33.222	3GPP Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS) <LINK>
[19]	CLP.17	GSMA IoT Security Assessment Checklist <LINK>
[20]	TS-0003	oneM2M Security Solutions <LINK>
[21]	3GPP TS33.163	3GPP Battery efficient Security for very low Throughput Machine Type Communication (MTC) devices (BEST) <LINK>
[22]	na	TCG Guidance for Securing IoT <LINK>
[23]	na	TCG Guidance for Securing Resource-Constrained Devices <LINK>
[24]	na	OWASP Authentication Cheat Sheet <LINK>
[25]	na	SANS Information Security Policy Templates <LINK>
[26]	na	BSI - German Federal Office for Information Security - Evaluation of random number generators <LINK>
[27]	GSMA IoT.04	Common Implementation Guide to Using the SIM as a ‘Root of Trust’ to Secure IoT Applications <LINK>

2 The IoT Endpoint Security Challenge

The security challenge presented by an IoT Service is, in many cases, directly related to the specific characteristics of the IoT Endpoint employed by the service. For example, many IoT Endpoints have the following characteristics which have particular security challenges associated to them:

2.1 Low Power Consumption

- Low power consumption may be required to achieve long battery life (several years) for a remote inaccessible Endpoint without a permanent power supply or because the device has a permanent, but limited, power supply, e.g. solar energy supply.
- Low power consumption Endpoints can usually only undertake computationally simple cryptographic operations (for example the Endpoint may only support the lightweight cryptographic operations defined within ISO/IEC 29192 [16]) due to the high power consumption requirements associated with more advanced cryptographic operations and may only support limited bandwidth communications again limiting cryptographic capability.

2.2 Low Cost

- The business case for many IoT Services demands that the cost of the IoT Endpoint be kept low. This often results in the device containing low processing capability, small amounts of memory and constrained operating system. The net result is that the device may be unable to perform ‘internet-grade’ cryptography.

2.3 Long-Lived (>10 years)

- Many Endpoints, particularly for civic and industrial applications (e.g. a smart gas meter), must be long lived. This presents a challenge because the cryptographic design choices made when the device is designed will have to be robust for the lifetime of the device. For example the processing power per $ available to an Attacker over this 10 year period is likely to have increased 16 times whereas the capabilities of the device is likely to remain static.
- Management of long lifetime devices is also a challenge particularly if a security vulnerability is found that can’t be patched within the IoT Endpoint.

2.4 Physically Accessible

- Many IoT Endpoints are physically accessible to the Attacker. All hardware components and interfaces on these Endpoints are therefore potentially subject to attack and must be secured by the developer.

The net result of the above is in many IoT Services, the IoT Endpoints are not directly connected to wide area communications networks and many IoT Endpoints have no Internet Protocol (IP) capabilities. For example, an IoT Endpoint may use an industrial, scientific and medical (ISM) radio transceiver to transfer data to a local IoT Service Gateway that then takes the data and transmits it to the communications network using IP, complicating the process of securing the end-to-end communication.

Depending on the capabilities of the IoT Endpoint and the security risks associated with it, different security methods with varying degrees of complexity may need to be applied as explained in the rest of this document.

3 The IoT Endpoint Model

Once considered a set of vastly disparate technologies, interacting with the physical world and connecting to a server somewhere on the Internet for guidance and submission of metrics, the IoT Endpoint model has changed dramatically. In modern engineering, IoT technology has collapsed into a predictable model composed of only several variants. The IoT Endpoint is becoming more predictable as well, and is expected to take on one of only several manifestations:

- The Lightweight Endpoint
- The Complex Endpoint
- The Gateway (or "Hub")

In the diagram below some common IoT Endpoint configurations are shown:

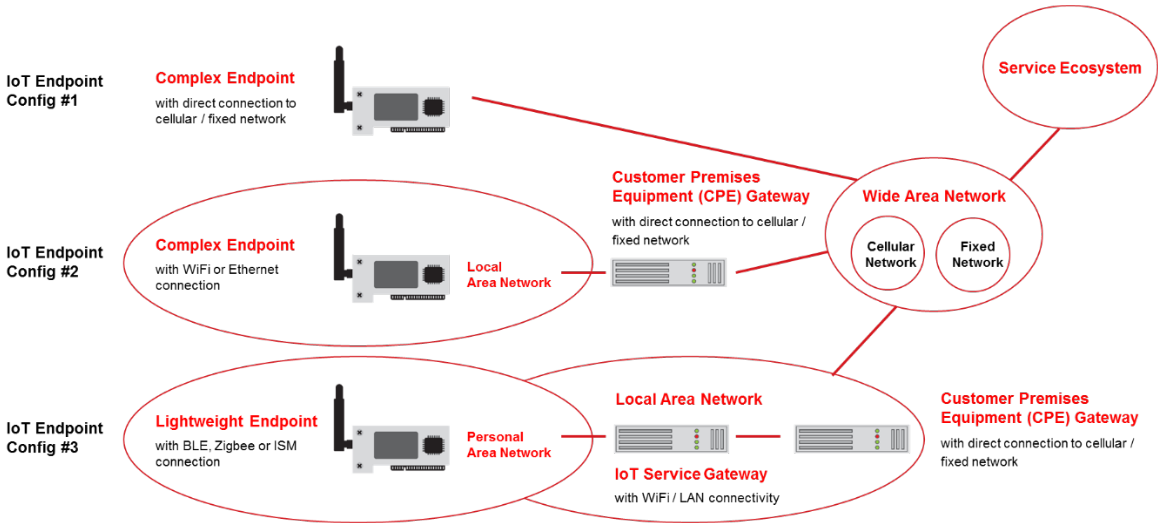

IoT Endpoint Config #1
Complex Endpoint
with direct connection to cellular / fixed network
Service Ecosystem
Customer Premises Equipment (CPE) Gateway
with direct connection to cellular / fixed network
Wide Area Network
Cellular Network
Fixed Network
IoT Endpoint Config #2
Complex Endpoint
with WiFi or Ethernet connection
Local Area Network
IoT Endpoint Config #3
Lightweight Endpoint
with BLE, Zigbee or ISM connection
Personal Area Network
Local Area Network
IoT Service Gateway
with WiFi / LAN connectivity
Customer Premises Equipment (CPE) Gateway
with direct connection to cellular / fixed network

Figure 2 - Example IoT Endpoint Configurations

3.1 The Lightweight Endpoint

This type of Endpoint is typically a sensor or simple physical device, such as a light switch or a door lock that has few functions. Its goal is to serve a singular physical purpose, and to provide metrics to the Service Ecosystem or to the consumer. It commonly uses a very cheap processing unit, possibly an eight-bit microcontroller, and a short-distance Personal Area Network (PAN) or capillary protocol for connectivity, such as Bluetooth Low Energy (BLE), Thread, or Zigbee. It is usually low power, and may operate off of a coin cell battery, solar power, or a small lithium polymer battery. These devices are typically connected to the Service Ecosystem via IoT service Gateways and Customer Premises Equipment Gateway as shown in 'Example Endpoint Config#3' in figure 2.

Examples of Lightweight Endpoints are:

- Wearables
- Home security sensor Endpoints
- Proximity beacons
- Non-Cellular capillary devices

Due to the low cost of lightweight Endpoints, the security technologies available to these devices is minimal. Security technologies that require a significant amount of current draw, cost, or space on the circuit board are not usually available to these systems. However, lightweight Endpoints can still use cost effective and small trust anchors to implement robust security framework.

3.2 The Complex Endpoint

This Endpoint model typically has a persistent connection to a back-end server over a long-distance communications link such as Cellular (inc. LPWA networks) (see 'Example Endpoint Config#1' in figure 2) or connects using Wi-Fi or Ethernet via a Customer Premises Equipment Gateway (see 'Example Endpoint Config#2' in figure 2). The device may have a rudimentary processor in it, even an eight-bit microcontroller, but is capable of running a more robust processing unit as it is either directly connected to an Alternating Current (AC) power source or it contains a battery and has regular access to a battery recharging system. Some Complex Endpoints communicate over capillary protocols, but require more power to run the local application efficiently, such as a streaming audio device.

Examples of Complex Endpoints are:

- IoT connected lighting systems
- Appliances such as refrigerators or washing machines
- Industrial control systems (e.g. SCADA)
- Retro-fit OBD2 Cellular "connected car" tracking and monitoring devices

Complex Endpoints are capable of more current draw, typically implement more robust processors, and have more space on the circuit board available for security technologies. As a result, much more can be done with Complex Endpoints. These devices can use almost any kind of Trust Anchor. As a result, they are easily able to implement a Personalized Pre-Shared Key (PSK) or asymmetric Trusted Computing Base (TCB) model as described later in this document.

3.3 The Gateway (or 'Hub')

A gateway is a device, typically connected to dedicated power source, which typically manages the communication between lightweight Endpoints and the back-end systems that drive them. The gateway manages long-distance communication links, such as Cellular (inc LPWA), satellite, fixed-line, fibre, or Ethernet. It accepts commands from the back-end systems living in the Service Ecosystem, and translates them to messages consumable by the light-weight Endpoints. Endpoint

While the primary function of an IoT Gateway is to route messages to and from lightweight Endpoints, it is also capable of performing critical tasks, such as:

- Device discovery
- Network driver deployment

- Management functionality
- Runtime monitoring
- Authentication and Security such as GBA or TLS setup

While Gateways are technically Endpoints, they may not necessarily be managed by the end-user, and may be managed by the IoT Service Provider or Network Operator (see below). Regardless of this, Gateways may also be engineered as Complex Endpoints to make more efficient use of distributing an uplink to multiple Lightweight Endpoints in a localized network.

Like the Complex Endpoints, Gateways are capable of more processing power, current draw, and typically have more space available on the circuit board. This allows IoT Gateways to implement complex Trusted Computing Base solutions, and technologies such as GBA authentication clients, with relative ease.

These attributes of the Gateway also enables them to incorporate multiple communications technologies to route messages between disparate types of networked devices. This enables communication between endpoints that normally would be unable to exchange messages in an effective manner. In this fashion, Gateways function as an aggregation point for devices within the local ecosystem, allowing them to communicate with each other and, if necessary, the Network and Service Ecosystems.

There are typically two types of Gateway – an “IoT Service Gateway” and “Customer Premises Equipment (CPEs) Gateway”. The difference is explained below:

1. An “IoT Service Gateway” is provided by the IoT Service Provider. It may be owned by the end user but is typically managed by the IoT Service Provider. Such a gateway is typically used as a hub to connect lightweight Endpoints to the service ecosystem (either directly via a fixed/Cellular connection, or via a CPE gateway), where the end-user buys a managed service from an IoT Service Provider.
2. A “CPE Gateway” is provided by a Network Operator. This is typically a broadband router connected to the internet by Cellular or fixed networks. This may be used in residential or enterprise environments. In this configuration, the gateway is usually managed and configured by the Network Operator.

3.4 The Overarching Model

Regardless of which type of Endpoint is being evaluated or designed, they all have similar subcomponent models from a hardware and logistic perspective:

- A Central Processing Unit (CPU) must execute application code
- The CPU must load/store data and executable code from/to persistent storage
- The CPU must compute data in temporal storage
- A Trusted Computing Base must be used to authenticate the environment
- The device must communicate with its IoT ecosystem

It is notable that lightweight Endpoints have less storage and computation capacities than Complex Endpoints or Gateways. They typically have less security capability as well.

The most important aspect of the overarching model is that each type of Endpoint device has one primary job: to define a reliable, high quality, and secure platform for executing a particular application. In other words, similar to more complex computing platforms such as smart-phones, Cloud servers, and mainframes, before a high-quality application can be ran reliably or interact securely with its peers, the engineering team must ensure the hardware presents a *trustworthy* platform to the application.

IoT Endpoints, by nature, participate in a network of other Endpoints. They are not standalone devices that perform an action without the influence or participation of an oversight service. To increase the trustworthiness of a given device, and diminish the potential for liability due to gaps in security or reliability, every Endpoint must be designed with the idea that *trustworthiness* in the entire IoT ecosystem *begins* by the construction of the Endpoint hardware.

With this perspective in mind, it is clear that even the easiest to develop type of Endpoint device must behave in a reliable, high quality, and secure manner because it is expected to participate in a network that could eventually span up to millions of devices in size. The manner in which a single Endpoint behaves will certainly have an effect on its entire IoT ecosystem. As a result, engineers must consider the implications of architecture design far beyond the physical attributes associated with a given embedded device. Engineers must think in terms of the security, reliability, and quality needs of the entire IoT ecosystem.

4 The Security Model

Security in the Endpoint can be assessed from a component perspective. By evaluating each component that is required to build any given Endpoint, the engineer and adversary can build a likely set of attacks that will result in full system compromise without a large amount of effort.

Using the Overarching Endpoint Model defined above, the components used can be evaluated from a high level. The high level perspective of each component will direct an analyst to technologies that are commonly used, and likely to be improperly secured. By prioritizing these components from the least level of expertise, equipment, and cost required to succeed, an analyst or adversary can build an attack model that will quickly assess any given Endpoint for security flaws.

In the Endpoint Ecosystem, there are several threat surfaces that will be investigated by adversaries depending on their resources, access to infrastructure, and expertise. These threat surfaces are:

- Network Communications
- Accessible network services
- Console access
- Local bus communications
- Chip access

4.1 Network Communications Attacks

The first and simplest step in attempting to compromise an IoT Endpoint typically involves weaknesses in the communications model. Analysts will observe whether the communication model incorporates communications security best practices. If the analyst can easily capture login credentials, communications tokens, or other identifiers that the Service Ecosystem will use to identify the Endpoint, they have compromised the device.

This strategy can waver from extremely simple to extremely difficult. The reason for this is the analyst or adversary's access to the plaintext data passing over the communications channel. A sufficiently outfitted analyst will already have technology to intercept communications for BLE, 802.15.4, and other popular protocols. Since observing, or performing a man-in-the-middle attack against an Endpoint's communications typically requires little to no change to the Endpoint, the adversary is in a highly beneficial position. It requires very little effort and work to implement this type of attack.

However, if the communications model uses best practices to enforce the confidentiality and integrity of data, the adversary will have an exponentially difficult time accessing valuable secrets. This will cause the adversary to move on to the next easiest attack model.

4.2 Accessible Network Services Attacks

The next step in attacking an IoT Endpoint is an evaluation of the network services that are open. In the first step, outbound messages emanating from the Endpoint are captured to identify whether immediately usable secrets are accessible in the messages. This allows an adversary to reduce the amount of work required in extracting secrets from the Endpoint, itself. If the outbound communications security model is sound, network services are scanned to evaluate whether the Endpoint's operating system can be accessed, or instrumented, from the network.

An assessment will be performed with a tool such as NMap [8] to determine whether network ports are open. If the network topology isn't IP capable, which is common in BLE or IEEE 802.15.4 networks, the adversary can still use readily accessible tools to connect to the Endpoint over the appropriate radio protocol.

The adversary will then attempt to send messages to the Endpoint to determine if the Endpoint can be manipulated into either executing commands, or providing remote-console access to the operating system. A common method is assessing whether a network login interface, such as Secure Shell (SSH) or telnet is available. If default login credentials are used, the adversary may be able to log into the Endpoint. This will allow the adversary to manipulate the local operating system, and potentially abuse local vulnerabilities to escalate privileges and extract secrets from the device.

Another common example includes the abuse of poorly designed web services, where commands can be injected over Common Gateway Interface (CGI) scripts that don't adequately strip control characters from user input fields, resulting in code execution on the local operating system.

4.3 Console Access Attacks

Console access isn't exactly an attack, it's a strategy. Typically, consoles need to be enabled on Endpoints to provide developers and Quality Assurance (QA) technicians with the ability to diagnose anomalies in hardware or software. However, the information provided from a console is very valuable to an adversary. In addition, a console may provide an adversary with the ability to log into the Endpoint system both locally and remotely.

Typically, local hardware consoles can be found on Endpoint devices by:

- Looking for a 5 pin header on the circuit board indicating a TTL serial port
- Looking up the specifications for the CPU or MCU and identifying the UART pins

A multi-meter can be used to identify a TTL port, as the pins will adhere to the typical voltage specification for TTL. Alternatively, a logic analyser can be used to guess the baud-rate of any serial data traversing hardware pins. The analyst will quickly be able to discern whether a console is available on the local hardware.

In many cases, simply accessing a console port allows an analyst to have direct access to a command prompt on the Endpoint device. In other cases, login credentials are required, but are typically guessable. If another individual on the Internet has identified the login credentials, and all Endpoint login credentials are the same, all an analyst has to do is perform a Google search online to see if someone else has posted the credentials.

Remote console access can be acquired through diagnostic networking protocols, console access protocols (e.g. SSH or telnet), or other means. These access methodologies should be evaluated to determine whether an adversary can manipulate the access channel, thereby granting the adversary access to a remote console.

4.4 Local Bus Communications Attacks

If a command prompt cannot be obtained through a console, the adversary or analyst will need to start inspecting hardware to determine if the Endpoint is easily compromised. This takes many different forms, but there are easy steps that will be taken:

- Is writable media present and alterable
- Are cryptographic secrets passed in the clear over hardware busses
- Can messages be injected into the hardware circuit that influence the behaviour of the application or operating system in the adversary’s favour

The simplest attack is identifying whether writable media is present. This could be media that is simple to alter, such as a writable external memory (SD/MMC) card. Or, an NVRAM chip or EEPROM can be altered with application or configuration changes to allow command prompt access, or access to securely stored tokens.

If this vector is properly secured, the analyst will determine if cryptographic secrets are passed in the clear over hardware busses. This could involve using a logic analyser to intercept messages between an EEPROM and a CPU, a microcontroller and a SPI-connected network adapter, or other attacks. These attacks can range from extremely simple and fast, to complex and expensive, depending on the complexity of the attack and the technology abused.

If the adversary cannot intercept valuable secrets using the above method, they can attempt to inject messages into hardware busses to change the behaviour of an application running on the Endpoint. This is a difficult attack that requires a high degree of expertise, equipment, and the ability to evaluate the application-specific data and its context.

4.5 Chip Access Attacks

If the attacks above are too complex or expensive, the adversary must move on to even more complex attacks against the hardware. This typically involves abusing the security of the chip, or the various components on the circuit board. This may include:

- Decapping the microcontroller or CPU
- Extracting secrets from internal EEPROM or NVRAM
- Intercepting internal SRAM messages
- Performing X-Ray analysis or FIB reverse engineering

All of these attacks require a high degree of skill, electronic engineering knowledge, and expensive equipment. While most organizations will not need to fear an Attacker utilizing these methodologies to reverse engineer their products, it is still an important possibility to consider. The reason is that these attacks only need to be performed *one time* if the Endpoint devices are not provisioned with unique cryptographic secrets.

If they are not provisioned with unique cryptographic secrets, one attack in this class will extract secrets that can affect the entire product line. That is a significant risk, because if the data is released to the public for any reason, the technology will be subject to attack and abuse until a patch is released, if one can be released.

5 Frequently Asked Security Questions

Endpoint security is broken down into recommendations by priority in this document. But, for practical use, it is more beneficial to evaluate recommendations from a practical starting point. Engineers typically start building a list of recommendations based on a technological or business-influenced goal. This section outlines common goals from an Endpoint perspective, and which recommendations are relevant toward achieving those goals.

5.1 How do we Combat Cloning?

Protecting intellectual property is an important goal for modern businesses. The hardware, firmware, and communications technologies used to build an Endpoint product take time, expertise, and finances that companies don't want to easily build another less scrupulous company's brand or business. However, no matter what a company does, someone can use the exact same hardware components to build a look-a-like “rip off” or “clone” of a given product. There is nothing the company can do to prevent this outside of legal contracts and partnerships. However, there are cost-effective ways to prevent someone from *using* such a clone.

Building authentication into the Endpoint communications will ensure that each Endpoint is cryptographically proven to be manufactured by the IoT Service Provider. Whenever the back-end services, or a peer Endpoint, communicates with an Endpoint device, it will be able to differentiate between a valid Endpoint and a *clone* by forcing the Endpoint to authenticate itself. If the device cannot do so, the peer or service can reject the Endpoint. This requires the following recommendations to work:

- Authenticating an Endpoint Identity
- Improperly Engineered or Unimplemented Mutual Authentication

5.2 How should I Secure the Endpoint Identity?

In order to properly authenticate an Endpoint, the engineer must be able to trust the Endpoint’s cryptographic identity. This is more complex than it seems, and requires a combination of process, policy, and technology to achieve the goal. This is further elaborated in the *Implement a Trusted Computing Base* recommendation, but the way in which authentication tokens are encoded on an Endpoint will determine how secure the overall system is.

In many Endpoint architectures, an adversary can simply copy cryptographic tokens (if there are any) from the target device in order to impersonate it. If each Endpoint manufactured by the IoT Service Provider utilizes the same set of cryptographic tokens, the adversary may be able to impersonate any device simply by compromising one single set of tokens.

Thus, building the proper TCB requires the following recommendations:

- Implement a Trusted Computing Base
- Utilize a Trust Anchor
- Use a Tamper Resistant Trust Anchor
- Utilise an API for the TCB
- Use a Proven Random Number Generator
- Use Tamper Resistant Product Casing
- Enforce Confidentiality and Integrity to/from the Trust Anchor

5.3 How do I Reduce the Impact of an Attack Against the Trust Anchor?

It is also important to note that the way a device is manufactured and provisioned has a drastic effect on the security of an Endpoint in production. The manufacturing process will determine whether Endpoints are securely encoded with keys. The fulfilment and provisioning process will determine how an Endpoint is associated with a particular consumer, and whether the device can be compromised before or after an association is made.

- Consider Supply Chain Security
- Personalize Each Endpoint Device Prior to Fulfilment
- Uniquely Provision Each Endpoint
- Privacy and Unique Endpoint Identifiers

5.4 How do I Reduce the Probability of Endpoint Impersonation?

After cloning of devices for business reasons, a desirable attack from an adversary’s perspective is the impersonation of a person or a *particular* device. This may or may not be directly associated with the attack of a particular individual. It could simply be the impersonation of a device for the purposes of bypassing a security control, such as a Bluetooth-enabled digital lock.

Regardless of the rationale, combatting this attack can be achieved by using a TCB, personalization, authentication, and also:

- Perfect Forward Secrecy
- Locking Critical Sections of Memory

5.5 How do I Disallow the Ability to Impersonate Services or Peers?

Every IoT network is composed not only of Endpoint devices, but of network services and peers. The Endpoints must be authenticated by the services, but the services must also be authenticated by the Endpoints. This ensures that critical services, such as application updates, cannot be subverted to further compromise the network.

- Endpoint Communications Security
- Perfect Forward Secrecy
- Use a Proven Random Number Generator
- Over the Air Application Updates
- Improperly Engineered or Unimplemented Mutual Authentication
- Unauthorized Metadata Harvesting

5.6 How do I Disallow Tampering of Firmware and Software?

Once a root of trust has been established, the Endpoint can authenticate from a trustworthy component. Doing so allows the Endpoint to establish a base of trust and ensure that the next stage application has not been altered either unintentionally (through faulty NVRAM, for example) or intentionally by an adversary. Accomplish this by:

- Minimum Viable execution Platform (Application Roll-Back)
- Cryptographically Sign Application Images
- Bootloading Outside of Internal EEPROM
- Locking Critical Sections of Memory
- Insecure Bootloaders
- Use Tamper Resistant Product Casing

5.7 How do I Reduce the Possibility of Remote Code Execution?

If tampering with physical firmware or software doesn’t yield adequate results, the adversary may move on to more complex attacks, such as code execution against the bootloader or applications that communicate over bus or network interfaces. If all peers in the network are authenticated, as depicted earlier in this chapter, it will be far more challenging for an adversary to inject malicious content. Yet, most devices require some semblance of public communications to interact with devices from other organizations. Therefore, they may not be able to adequately enforce restrictions on the origin of data.

Thus, the ingress of data into the computer system from both remote and physical interfaces must be heavily scrutinized. To limit the potential for exploitation of an application, and limit exposure once an application *is* compromised, consider the following:

- Enforce Memory Protection
- Use Internal Memory for Secrets
- Over the Air Application Updates
- Run Applications With Appropriate Privilege Levels
- Enforce a Separation of Duties in the Application Architecture
- Enforce Language Security
- Enforce Operating System Level Security Enhancements
- User Interface Security

- Third Party Code Auditing

5.8 How do I Disallow Unauthorized Debugging or Instrumenting of the Architecture?

An Attacker with architectural knowledge and access to debugging tools will typically attempt to instrument standard debugging and diagnostic utilities to gain access to system secrets, or to alter or inject beneficial code. Restricting an adversary’s ability to do this will diminish the potential for fast and stealthy attacks that may not be detected by a consumer.

- Use a Tamper Resistant Trust Anchor
- Logging and Diagnostics
- Locking Critical Sections of Memory
- Anomaly Detection
- Use Tamper Resistant Product Casing
- Disable Debugging and Testing Technologies
- User Interface Security

5.9 How should I handle Side-Channel Attacks?

When an adversary is out of typical options, they will look to more esoteric attacks in order to extract secrets from a device. These attacks evaluate how the hardware behaves in order to ascertain whether a pattern in behaviour can equate to a value, such as a one or zero, or a particular instruction. This, over time, will give the analyst the ability to reverse engineer the data being processed by the embedded system.

Also, the adversary can use expensive analysis technology to extract secrets from the device, or to build extremely small circuits that bridge connections through security layers in the silicon. While these attacks are extremely difficult to combat against, there are some things that the implementer can do to dissuade attacks:

- Personalize Each Endpoint Device Prior to Fulfilment
- Use Internal Memory for Secrets
- Use Tamper Resistant Product Casing
- Tainted Memory via Peripheral-Based Attacks
- Implement Environmental Lock-Out Thresholds
- Enforce Power Warning Thresholds
- Device Decommissioning and Sunsetting
- Defeating Shadowed Components and Untrusted Bridges
- Defeating a Cold-Boot Attack
- Combating Focused Ion Beams and X-Rays

5.10 How should I Implement Secure Remote Management?

Remote management is a critical part of the IoT Endpoint lifecycle that must be safeguarded to ensure the channel used for management and administration cannot be abused. This is not only an issue with unknown third-party adversaries. Internal abuses can occur as well, either within the consumer’s circle, or within the IoT Service Provider.

- Endpoint Password Management

- Remote Endpoint Administration
- Logging and Diagnostics
- Perfect Forward Secrecy
- Use a Private APN

5.11 How do I Detect Compromised Endpoints?

Depending on the architecture of the Endpoint, it may be nearly impossible to determine if the hardware or firmware has been tampered with if the device is behaving normally. However, a compromised device can be detected by anomalous behaviour as long as the infrastructure is tracking, logging, and alerting when abnormalities are detected. Consider the following recommendations:

- Anomaly Detection
- Use Tamper Resistant Product Casing
- Enforce Power Warning Thresholds

5.12 How do I Securely Deploy a Device Without a Back-End Connection?

There are certain times where a connection to a back-end environment is neither available nor desired. In these environments, security becomes more of a challenge because of the obvious inability to manage security keys, identities, and dynamic authentication mechanisms. However, a fair level of security can be achieved. Consider the following:

- Implement a Trusted Computing Base
- Defining an Organizational Root of Trust
- Personalize Each Endpoint Device Prior to Fulfilment
- Perfect Forward Secrecy
- Authenticating an Endpoint identity
- Environments Without Back-End Connectivity

5.13 How do I Ensure my Consumer’s Privacy?

Consumer privacy is a complex issue that requires an in-depth analysis of not only the Endpoint technology, but the entire IoT product or service. Each component in the overall system must be analysed for potential gaps in privacy. Review the following recommendations to gain more insight on enforcing privacy:

- Perfect Forward Secrecy
- Endpoint Communications Security
- Privacy Management
- Privacy and Unique Endpoint Identities
- Utilize a Private APN
- Unauthorized Metadata Harvesting
- Non-Obvious Security Risks (Seeing Through Walls)
- Lawful Interception

Note: Detailed recommendations on how to approach data privacy is contained in Annex A of the GSMA IoT Security Guidelines Overview Document [1].

5.14 How do I Ensure User Safety While Enforcing Privacy and Security?

Safety is a topic that must be considered in context with the application, its purpose, the intended environment(s) in which the application will live, the type of consumer, and the communications technology used. There are often times it may seem that trade-offs should be made between safety and security. This may not be true, however. Instead, the architectural model may need to be shifted in order to maintain both safety and security. Where possible, security should not be discarded in favour of safety. Both should be enforced, where ever possible. While this is a philosophical recommendation, it is important that safety be constantly reviewed by the engineering team. Consider the following recommendations to start a discussion about safety in IoT:

- Safety Critical Analysis
- Intentional and Unintentional Denial of Service
- Lawful Interception
- Consider Supply Chain Security

5.15 What Issues Should I Not Expect To Resolve?

In every system there are risks that cannot be resolved due to the laws of physics, cost, or simply a lack of technological solutions. Some of these issues are noted here:

- Intentional and Unintentional Denial of Service
- Defeating Shadowed Components and Untrusted Bridges
- Non-Obvious Security Risks (Seeing Through Walls)
- Combating Focused Ion Beams and X-Rays
- Consider Supply Chain Security
- Lawful Interception

6 Critical Recommendations

When developing a secure Endpoint, the following recommendations should always be implemented. The following critical recommendations define a secure Endpoint architecture. Without these recommendations, the Endpoint will have an incomplete security profile that will be abused by an adversary.

6.1 Implement an Endpoint Trusted Computing Base

The first step in securing any embedded system is the definition of the Trusted Computing Base (TCB). In the context of an Endpoint (or similar embedded devices), a TCB is a suite composed of hardware, software, and protocols that ensures the integrity of the Endpoint, performs mutual authentication with network peers, and manages communications and application security.

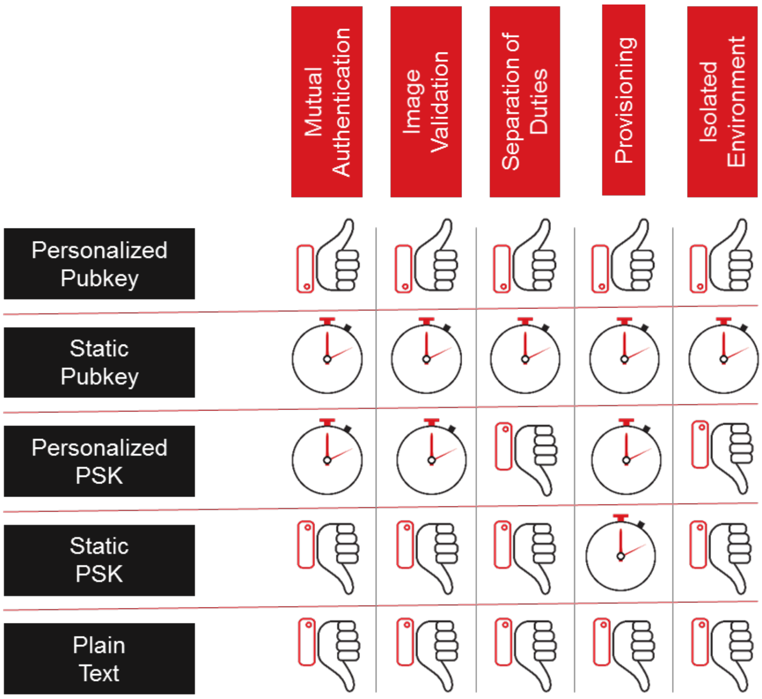
The core of the TCB is the trust anchor, a secure hardware technology that stores and processes cryptographic secrets such as Pre Shared Keys (PSK) or asymmetric keys. Trust anchors, such as an UICC with IoT SAFE capability (see GSMA IoT.04 [27]), can be used to authenticate not only peers during network communications, but can be augmented to store data useful for Endpoint application security.

Once the trust anchor is chosen and integrated into the Endpoint solution, libraries can be chosen or designed that integrate the trust anchor into the overall TCB suite. The TCB will allow the Operating System and the Endpoint's primary applications to more easily manage the overall security of not just the device, but the network.

It is important that the engineering team choose the correct trust anchor for the solution, as each combination of trust anchor and TCB will result in different level of security. Some combinations and trust anchor implementations will result in a false sense of security.

The most common variations of a Trusted Computing Base, in order of 'least secure' to 'most secure' are:

- None implemented (Plain text)
- Static Pre-Shared Key (PSK)
- Static Public Key
- Personalized PSK
- Personalized Public Key



	Mutual Authentication	Image Validation	Separation of Duties	Provisioning	Isolated Environment
Personalized Pubkey	👍	👍	👍	👍	👍
Static Pubkey	⏱	⏱	⏱	⏱	⏱
Personalized PSK	⏱	⏱	👎	⏱	👎
Static PSK	👎	👎	👎	⏱	👎
Plain Text	👎	👎	👎	👎	👎

Figure 3 - Security assurances provided per each type of TCB.

Consider the figure above. In this diagram, each TCB variant's capabilities are given a weight. A thumbs-down icon denotes that the TCB model cannot accommodate for the security strategy depicted along the top row. A stop-watch icon shows that the security strategy can be used, but will be subject to a break in security within a reasonable amount of time. A thumbs-up icon shows that the security strategy can be implemented soundly, and that the lifetime of the security strategy will likely be long-lived.

While a TCB can be used to secure many aspects of the overall IoT product and service, this document focuses on five core concepts:

- Executable image validation

- The mutual-authentication of network peers
- Separation of Duties within the IoT security architecture
- Provisioning and Personalization
- Isolated Environment security (or connectionless site security)

A TCB that implements *executable image validation* secures the Endpoint device by cryptographically verifying each executable image to be loaded and executed by the device. This process starts at the bootloader, which should cryptographically validate the next stage of execution, typically an operating system kernel. The bootloader could also validate the operating system image, or a firmware application image stored in NVRAM.

A TCB that implements *mutual-authentication of network peers* helps provide a root of trust for the authentication of network components, and cryptographically authenticates itself to network peers. This increases the likelihood that peers on the network represent the identities that they claim to represent. For example, if the network peer claims to offer a firmware-update service, the TCB would authenticate the peer as being a part of the core IoT Service Provider network before accepting firmware updates from the peer.

A TCB that implements *a separation of duties* uses a hierarchy of keys to identify varying components or services within the IoT Service Provider's offerings. For example, one set of cryptographic keys could represent a firmware update service, while a second set of cryptographic keys could represent a "push" service. Since these services have a completely disparate functionalities, they should not use the same cryptographic keys and identities for communication. As such, the TCB should manage and verify each identity to separate one service or function from another. This reduces the ability for an adversary to compromise the entire IoT service infrastructure if *one* of the cryptographic keys is compromised. In other words, if an Attacker compromises the key for the "push service", they will not also have the ability to impersonate the firmware update service.

A TCB that implements *personalization and provisioning* ensures that the Endpoint has an identity that is cryptographically unique from every other Endpoint of its type. It also ensures that all communications identities are safeguarded to reduce the potential for privacy leaks or tracking.

A TCB that implements *isolated environment security* enforces policies and procedures that validate the authenticity of peers and the confidentiality and integrity of data even if there is no back-end service to help in the process. In other words, if communication to back-end services are cut off for an extended period of time, the localized IoT ecosystem will still be able to function with a high degree of security. Though the integrity of isolated environments degrades over time, a well designed TCB that implements *isolated environment security* can strengthen the resilience of the network and lengthen the amount of time that the environment can be considered secure.

In this context, *personalized* is indicative of a unique set of keys that are associated with a specific trust anchor. The personalization process includes the generation and installation of the unique keys, the association of the keys with the unique chip, and the secure dissemination of this information and the relevant metadata to the appropriate authorities.

This ensures that each chip has a unique cryptographic identity. *Static*, in this context, refers to the same set of keys used for every Endpoint.

While TCBs can be used to solve almost any security concern an embedded system may have, there are several core problems that a TCB must be able to solve

- Endpoint application image validation
- Network authentication and/or peer authentication
- A separation of duties
- Provisioning and personalization
- Isolated environment (connectionless site) provisioning and communication
- Randomization

While it is obvious that choosing to not implement a TCB results in a lack of security, there are subtleties to the other common TCB implementations that should be addressed. If these subtleties are not addressed, they may result in substantial gaps in security.

6.1.1 Trust Anchor Key Models

6.1.1.1 Static Keys

A static key implementation, whether it is PSK or asymmetric keys, is defined as a solution where every Endpoint utilizes the same cryptographic secret to solve a given problem. While different keys may be used to solve different core problems, the key is still the same set for each Endpoint.

This model seems secure in that each problem solved by the TCB can be done effectively. However, the lifespan of the overall solution can range from long to extremely short. Depending on the security of the trust anchor and the cryptographic algorithm and key size chosen, adversaries may be able to break the solution almost immediately.

The problem really arises in that a single compromise of the key exposes every Endpoint system to compromise. This devalues the TCB implementation and negates the time and money used to implement the solution in the Endpoint and overall IoT architecture. Thus, this model is a dangerous TCB to implement as it is, effectively, a time bomb.

6.1.1.2 Personalized Keys

Regardless of whether a PSK or asymmetric solution is implemented, personalization is imperative for a TCB to work effectively. Personalization negates the ability for an adversary to use a compromised trust anchor to subvert the security of the entire IoT ecosystem. If an adversary is only able to compromise a single Endpoint at a time, and they require physical access to do so, it will be extremely slow, expensive, and complex to implement a wide compromise of the IoT technology. This is a significant win for the business.

Because of the standards in Cellular communications that have evolved over the past several decades, Network Operators have perfected the PSK model for personalization of trust anchors, such as UICC. As a result, a UICC with IoT SAFE can be provisioned to serve as an application trust anchor by the IoT Endpoint, helping to form a cost-effective security solution for IoT applications. In the near future, when eUICC are available, this capability can be enabled even on eUICC already deployed in the field.

Today, personalized key technology is the most effective security solution for a trust anchor. TCBs implemented in IoT today should be based on a personalized TCB solution. IoT Service Providers should have a discussion with their Network Operator to determine if the UICC or SIM can be implemented as an application layer trust anchor using IoT SAFE or other mechanism.

6.1.2 TCB Protocols and Technologies

Along with a trust anchor, the TCB must incorporate protocols, policies, and software libraries to provide security to the overall IoT product or service. One advantage to the utilization of Cellular-backed standard trust anchors is the ability to drop in provisioning and personalization software that already exists for Network Operators. Technologies, protocols, and suites such as the following will assist with the TCB's ability to help authenticate the Endpoint to the network:

- oneM2M SM UICC application as specified in oneM2M TS-0003
- IoT SAFE
- Generic Bootstrapping Architecture (GBA) 3GPP TS 33.220 (See Annex A)

Use of these technologies will help speed up the implementation of provisioning and personalization as the libraries and protocols have been vetted by experienced engineers and security analysts for many years. Yet, these protocols may not fully enable the TCB to validate the Endpoint's application, or ensure that the Endpoint can properly authenticate messages, or authorize actions. The TCB must incorporate other protocols to accomplish these tasks, such as firmware validation, over-the-air update message validation, and more.

In the near future, technology such as eUICC will augment the capabilities from the perspective of the application, and proactive UICC enable dual-use technology that can help bootstrap the Endpoint itself, while managing Network security. This is an important augmentation as Network Operators can remotely and securely manage the eUICC device on behalf of the IoT service provider. In addition, the Confidential Card Content Management functionality specified in GlobalPlatform Card Specification [15] enables several actors in the IoT service ecosystems to manage their own application independently of each other, if allowed by the network operator.

6.1.3 Risk

Choosing to not implement a TCB is a critical point of failure for the entire IoT architecture. Without a well defined TCB, the interplay between the trust anchor and core application will be loosely defined, and may have gaps that can be subverted by adversaries. The TCB ensures that communications between the trust anchor, core application, and network peers are secure, reliable, and up-to-date. Without a TCB, there is no central component to manage the security lifecycle of the Endpoint.

6.2 Utilize a Trust Anchor

In order for an Endpoint to participate in an ecosystem, it must be able to verify the integrity of its own platform, and must be able to authenticate the identity of its peers. To do this, Endpoints require a trust anchor incorporated into a Trusted Computing Base.

A trust anchor is a secure hardware element, either a separate physical chip, or a secure core inside a CPU, that is capable of securely storing and processing cryptographic secrets. A UICC or eUICC with IoT SAFE is an example of a secure technology that can be used as a trust element to store authentication secrets.

Using a trust element effectively involves storing, verifying, updating, and processing data. The data can be either secret or public information that must be cryptographically verified. In either case, the trust anchor must be able to securely determine whether messages and identities can be authenticated, and must be able to securely tell the TCB the result of all authentication or cryptographic operations. This allows the Application, and the TCB, to make valuable decisions that will affect the security of the overall Endpoint. For instance, a trust anchor can help an Endpoint determine whether a network peer is impersonating a critical resource, such as a patch deployment server. If the trust anchor cannot validate a network peer, the TCB and Application on the Endpoint should choose not to interact with such a peer, and should alert the user, where possible, of the fraudulent network resource.

Thanks to the decrease in the cost of components and a sharp increase in demand, trust anchors are becoming more available than ever before. This not only includes the actual trust anchor technology, but libraries and interfaces approved for use with the technology. This allows the engineering team to spin up a trust anchor solution in very little time, and will help to ensure that the longevity of the technology is not weakened by custom software or poorly implemented standards. Where possible, standards should be used to diminish the potential for gaps in security.

Another challenge in implementing a trust anchor in lightweight Endpoints is the size of the component. If an external trust anchor is utilized, it will be necessary to maintain a minimal component profile. Achieving this profile is difficult when the form factor incorporates technology such as a UICC. However, the ETSI TS 102 671 standard solves this problem by introducing a very small form factor of approximately 6 millimetres by 5 millimetres in size. These “MFF1” and “MFF2” augmentations to the UICC smart card form factor enabling full access to technologies supported by the UICC while ensuring the physical requirements are minimal. Extra security is added by utilizing a field-provisioned form-factor that is soldered onto the device, making it more difficult for adversaries to transfer a device’s identity to another device.

Expenses incurred in the development and deployment of a trust anchor can include:

- The cost of the base technology, either embedded within the CPU or a separate chip
- The cost of integrating the technology into the circuit, where required
- The cost of engineering or integrating the driver into the OS and TCB
- The cost of engineering the Application to use the trust anchor
- Maintaining the trust anchor, where required
    - Maintaining security keys, revoking keys, and decommissioning identities
    - Maintaining the infrastructure required to secure and manage the keys and metadata

- Monitoring the trust anchor identity on the Service side
    - Implementing device blacklisting, where required

- Integrating carrier services, where available, to monitor and manage trust anchors such as UICC

6.2.1 Risk

The risks of not utilizing a trust anchor are many, but all stem from the same base issue: the ability for an adversary to steal keys relevant to the entire IoT ecosystem. The result of this is that an adversary can:

- Clone Endpoint identities
- Impersonate IoT services
- Deploy unauthorized patches or updates
- Make unauthorized changes to the Endpoint software

These gaps in security can result in costly issues to the business over time, and can allow not only adversaries, but competitors, to abuse the infrastructure toward their benefit.

6.3 Use a Tamper Resistant Trust Anchor

Some Trust Anchors have extra physical security to guard against certain classes of attacks, such as FIBs, side-channel analysis, and glitching. While some attacks, such as the utilization of a FIB, are almost impossible to guard against from a cost perspective, the trust anchor manufacturing can use modern technologies to make attacks more costly. The more costly an attack is, the less likely it will be used against random Endpoint devices. Instead, attacks will be focused on targets where the expense is worth the reward.

In the near future, some trust anchor manufacturers are planning to roll out variations of their technology that are Federal Information Processing Standards (FIPS) [10], EMVCo [11] and Common Criteria approved. Engineers developing new technology should determine whether their current designs will support a move to compliant modules in the near future.

For more information, review the latest version of each standard to evaluate what level of capability is offered by your manufacturer. Note that some levels of security are purposefully close to impossible for consumer-based devices due to the cost and complexity of implementations.

6.3.1 Risk

The risk of not using a tamper resistant trust anchor is extremely high. For example, if a trust anchor is simply cryptographic keys embedded in NVRAM, any Attacker with the tools and skill to extract those keys can potentially subvert the entire infrastructure. However, if the secrets are stored in a tamper resistant trust anchor, the expense to extract the secrets is high, which will make it far less likely that the secrets will be extracted, devaluing the trust anchor as a potential attack target.

It is notable that if the trust anchor implementation is weak, the extraction of secrets resulting in a compromise can be sufficiently high. Any compromise will invalidate the expense incurred during engineering, architecture, production, and fulfilment. This may result in a significant financial loss. Therefore, ensuring that the organization has architected the correct implementation is imperative.

6.4 Utilise an API for the TCB

Once a root of trust has been established within the TCB, a protocol must be used that incorporates the TCB's capabilities and the root of trust effectively. The API should ensure that:

- All signature verification is performed by the TCB
- No private keys are exposed from the TCB
- Key exchange can be performed by the TCB on behalf of the application
- Decryption can be performed by the TCB
- Encryption can be performed on the TCB
- Message signing can be performed on the TCB
- Secure message padding can be performed on the TCB
- Confidentiality and Integrity between the TCB and the application

This set of capabilities will help guarantee that the TCB never exposes critical security assets to an insecure application or hardware environment. This can be accomplished by using an existing specification that applies these requirements in a uniformed fashion. Consider evaluating:

- SIM Alliance Open Mobile API [12]
- GlobalPlatform Secure Element Access Control [13]
- GlobalPlatform Trusted Execution Environment (TEE) API Specification [14]
- Trusted Computing Group (TCG) [22] [23]
- oneM2M TS-0003 [20]

Many trust anchors will come with software libraries that can be implemented as a TCB. These libraries will have APIs that the engineers can use to interact with the TCB. Libraries provided by the trust anchor are preferred, where available, as they have likely been vetted by experts in the field of trust anchor development. However, the engineering team should evaluate the list of requirements set forth in this recommendation, and should ensure that the library adequately accounts for these concerns.

Furthermore, TCBs should *only* be accessible from privileged applications running on the Endpoint. A TCB interface should never be accessible from an unprivileged or untrusted (third party) application running on the Endpoint. All access must be proxied through a trusted service that evaluates requests and optionally alerts the user when suspicious or privacy-centric requests are being made by untrusted applications.

The challenge in implementing this protocol is guaranteeing that all messages cannot be tampered with between the point of origin for the data and the TCB, and vice-versa. It is most effective if a segment of EEPROM, callable from the application, can perform these functions on behalf of the application. By isolating the brunt of the API code to the internal EEPROM, and using internal RAM to process the messages, less critical data will be exposed to external busses.

6.4.1 Risk

If an Application Protocol Interface isn't well defined, using a TCB may have unintended results or side effects. By defining the protocol ahead of time and vetting it for logic and security issues, the engineering team can more quickly and effectively identify flaws that may result in security issues later. Thus, the definition of the protocol should incorporate the evaluation of existing APIs that incorporate the needs of the IoT Service Provider. If an existing and well-established technology can be identified, this will always be favourable over a custom solution.

6.5 Defining an Organizational Root of Trust

An organizational root of trust is a set of cryptographic policies and procedures that govern how identities, applications, and communications can and should be cryptographically secured. Strong cryptography should be used, either in the form of unique symmetric keys, certificates, or public keys. This depends on the model available for use in the TCB, the capabilities of the trust anchor, and what makes sense for the engineering team.

A root private key, either symmetric or asymmetric, should be used to digitally sign other keys used in the hierarchy. For example, if our example organization, Example IoT Company LLC, wants to create an organizational root of trust, they would generate a root key on a trusted machine. This key will represent the organizational root. They would then generate new keys representing each sub- organization that should have independent security hierarchies. Examples may be:

- Code signing key
- Server communications key
- Peer-to-peer communications key
- Endpoint identity key
- Master revocation key

Each of these keys should be signed by the organizational root key. All of these keys, their corresponding signature, and the root key should be stored in the trust anchor used by the TCB. Then, whenever the application linked to a particular key is used, the application can use the specific keys to validate messages sent over communications channels.

This model helps to ensure that all messages are secured through the cryptographic hierarchy. By separating duties among specific key types, compromised keys can be revoked through the same communications process.

Some existing protocols that assist in deploying this method are:

- Transport Layer Security (TLS); The latest valid specification
- Secure Shell (SSH2)
- Online Certificate Status Protocol (OCSP) IETF RFC 2560
- Generic Bootstrapping Architecture (GBA) (See Annex A) 3GPP TS 33.220

Difficulty arises when services that need the cryptographic keys must be deployed. Instead of placing a security-critical asset, such as the Server communications key, on a web server accessible to the Internet, a separate certificate or key pair should be generated specifically for that server tier. Then, this certificate may be signed by the Server communications key. This way, any Endpoint can verify that the service has been authenticated by the root of trust, but the critical organizational key will not be exposed to adversaries.

If a key is ever compromised, it can be revoked from use by using the Revocation master key to authenticate revocation.

It goes without saying that all core keys in the organizational root of trust are critical to the security of the infrastructure. These keys must be heavily guarded, and only used by trusted internal members of the core team. Utilizing an approved Hardware Security Module (HSM) to store, access, and use the keys is highly recommended.

While an HSM can often be a significant expense at the start of a technology's deployment, the long-term financial effects are highly positive. Rather than incur a significant expense later in forensic analysis and engineering to diagnose and combat a particular risk that could have been solved by a TCB and an HSM, a relatively small up-front expense is incurred.

6.5.1 Risk

The risk of not using an organizational root of trust is that any compromise to a single key can result in compromise of the entire ecosystem. By separating the organization into a hierarchy, and deploying separate keys for the hierarchy, keys can be cycled at regular intervals and according to the priority of the application or sub-organization the key relates to. This creates a separation of duties between facets of the organization, and diminishes the ability for a compromised key to subvert the security of the entire infrastructure.

6.6 Personalize Each Endpoint Device Prior to Fulfilment

Endpoint devices must be enabled with cryptographically unique identities to ensure that adversaries, competitors, and hobbyists can't impersonate other users or devices in production environments. To accomplish this adequately, the personalization process must be performed at fabrication. This can be done either through the manufacturer of the particular TCB solution, or during the Printed Circuit Board Assembly (PCB/A) process.

To solve the personalization process, perform the following:

- Generate a unique cryptographic key
- Sign the key using the organizational Endpoint Signing Key (or a derivative of)
- Store the key in the TCB's trust anchor
- Generate (or use) a unique internal identifier for that specific Endpoint
- Store the unique identifier in the TCB's trust anchor
- Save the unique identifier, the key, and the signature in the IoT Service back-end authentication system

Note that personalization of the Endpoint platform is separate from personalization of the network identity. Utilizing a UICC for network authentication is beneficial for many reasons, and where possible, the UICC could be used as a trust anchor. However, if the network trust anchor can only be used for authentication of the network, personalization of the application trust anchor must be performed separately. Cryptographic uniqueness of the application trust anchor is required to ensure that the application platform is verified prior to execution of the Endpoint application.

Using the proper agreement with a network operator or other issuing party, UICCs can sometimes be provisioned before delivery to serve as an application-centric trust anchor using IoT SAFE or other mechanism. In the near future, Endpoint developers should evaluate whether eUICC technology is suitable for use in IoT products and services. These technologies will allow in-the-field provisioning of cryptographic secrets in a fashion similar to an application-centric trust anchor. Since the mobile industry is a leader in the personalization and provisioning process, there may be a significant advantage to using an eUICC as a trust anchor.

Furthermore, these technologies will incorporate remote provisioning capabilities and Secure Channel for secure communication between the application and the eUICC trust anchor. These capabilities will provide in-field personalization, which will reduce the overall cost of Personalization and Provisioning for each individual Endpoint.

A short tutorial on use of UICC cards in an IoT service ecosystem is contained in Annex B.

The challenge comes with managing the Endpoint identities and the signing process. Each identity must be catalogued, along with unique identifiers matching the identity, in a system that cannot be tampered with. While the process is usually performed at the PCB/A facility, a connection from that facility to the business must be set up to securely traffic the identity data.

Rolling out this solution may be straight-forward with some facilities that are more familiar with cryptographic personalization. Other fabrication facilities may not have a process in place to accomplish this. The mobile industry has been able to succeed in this fashion due to their ability to control the fabrication and fulfilment of embedded technologies such as UICC. While the mobile industry has been a leader in this process for some time, the IoT application Endpoint personalization and provisioning process is still in its infancy.

Be prepared to determine whether the identity of the Endpoint should (or could) be managed by a gateway or uplink. Evaluating the architecture of the IoT product or service will help determine whether this attribute of identity management will affect the personalization process. While trust may be distributed to gateways, the organization should determine whether trust can be adequately delegated without diminishing the overall security of the communications and authentication system.

Expenses involved in personalization typically include, but are not limited to:

- Implementation of the personalization process at the chip manufacturer
- Coordination or delivery of the unique personalized values at both the manufacturer and the IoT Service Provider
- Implementation and management of the personalized identities

6.6.1 Risk

If the organization chooses not to implement personalization of the Endpoint device, they risk being unable to differentiate one Endpoint from another. If all keys are conformed across Endpoint systems, it doesn't matter if serial numbers are unique. The reason for this is that if keys are extracted from any single Endpoint, the adversary would be capable of impersonating any Endpoint.

Personalization combats this by forcing the adversary to extract the cryptographic secrets from each Endpoint they want to clone or impersonate. Because the expense of this process can be very high, personalization using a trust anchor is the single strongest method for combatting cloning and impersonation.

6.7 Minimum Viable execution Platform (Application Roll-Back)

A Minimal Viable execution Platform (MVeP) is the minimum amount of work that must be performed in order to create a reliable execution environment to communicate with the trust anchor. Typically, this means:

- Configuring the internal clock or oscillator
- Configuring core peripherals (memory, storage)
- Enabling various hardware bridges or peripheral devices
- Authenticating the next chunk of code to be executed by the CPU
- Executing the next stage of code
- Managing application image roll-back

Once this MVeP has been defined, the minimal bootloader can use the trust anchor to verify a more robust bootloader, or can execute the rest of the bootloader after verifying external applications. This allows a consistent environment to be defined with minimal effort that authenticates subsequent chains of code that will define the applications platform.

Another benefit is that with using the MVeP model, even processors with a minimal amount of internal NVRAM or EEPROM can bootstrap a trusted architecture using an internal or external trust anchor.

Lastly, an MVeP is important for rolling back to stable versions of a particular platform. If an MVeP can be defined that has the minimal functionality required to verify the integrity of application firmware images and configure the execution environment, its functionality can be separated from the core application functionality. Thus, if a firmware update fails for any reason, the MVeP can still be used to reconnect to the back-end network and download another firmware image (either the same image, or an older image). This also enables Endpoints with damaged NVRAM chips to still communicate with the back-end services and submit diagnostic information.

6.7.1 Risk

While it may seem benign, defining a MVeP ensures that the architecture of the overall Endpoint cryptographically verifies each step in the boot process. This step is critical in ensuring that an Endpoint can authenticate itself to the network, and its peers. If the MVeP is poorly architected, it can result in security gaps in the boot process that may be exploitable by adversaries, invalidating the security architecture.

6.8 Uniquely Provision Each Endpoint

While personalization guarantees that each device is unique once it is manufactured, provisioning ensures that a unique device is activated, updated, and associated with a particular customer identity. The provisioning process helps separate devices that have been manufactured from devices that have been purchased and/or deployed in an IoT environment. This helps the IoT Service Provider:

- Distinguish between active and deactivated devices

- Associate Endpoints with networks or other resources linked with a particular customer
- Customize an Endpoint according to the customer's needs
- More easily determine whether a particular customer or Endpoint has been compromised

The provisioning process does not happen during manufacturing, but relies on the personalization process deployed in manufacturing. Provisioning typically occurs in the field, based on the customer that initializes the activation process. Yet, for the process to be secure, provisioning relies on the unique security tokens set during the personalization process to ensure that the unique Endpoint is tied to a unique customer. This way, an adversary cannot arbitrarily register (or unregister) Endpoint devices simply by guessing Endpoint details. They would, instead, require each unique cryptographic token generated and set during the personalization process, which is computationally infeasible.

In this fashion, the IoT Service Provider can mathematically guarantee that it is improbable that adversaries can arbitrarily spoof or register Endpoint devices at will. This leads to a more secure and stable IoT environment, where the relationship between customers and devices can be more trustworthy.

6.8.1 Risk

Not implementing the provisioning process can result in a desynchronization between the organization and its Endpoint nodes. It will be more difficult for the organization to track Endpoints and established which devices have been enabled for use in the ecosystem or decommissioned. Furthermore, it may be difficult to establish which devices are associated with particular customers, which will increase the difficulty of tracking down a problematic or potentially compromised device in the field.

6.9 Endpoint Password Management

Devices that incorporate user interfaces must be capable of managing passwords effectively. This requires several things

- Brute-force attack mitigation
- Disabling the use of default or hardcoded passwords
- Password best-practice enforcement
- Disallowing display of user credentials on login interfaces
- Enforcing thresholds and incremental delays for invalid password attempts

Users will need to be secured from the simplest attack possible, another user attempting to guess their password. This can be alleviated by simply negating the potential for a brute-force attack. This can be done by increasing the time limit between password attempts. With each failed login attempt, there should be an increased delay before the next password is allowed to be entered. A ceiling should be implemented such that no more than N attempts can be tried at once. Otherwise, a reasonable lockout period should be enforced. The user should be alerted to the brute-force attempt once the real credentials are entered.

Hardcoded or default passwords should *never* be used in IoT systems. There should *never* be an administrative "back door password" to enter a system. There should *never* be a privileged account with default credentials. This is essential to guard user devices against unauthorized intrusion by users trolling randomly on the Internet for weak security.

Passwords must meet minimum quality requirements representative of the current information security best practice. This ensures that brute-forcing a password will be difficult, and helps the user guard against theft. Consider reviewing the OWASP [24] or SANS [25] guidelines for password security to ensure the application complies to recent best practices.

Passwords must never be displayed on a user's screen. Always hide the password with the asterisk character, or another benign glyph.

In addition, all interfaces that accept passwords must utilize brute-force mitigation technology. It is also important that the technology that *validates* the password must do the enforcing. For example, JavaScript embedded in a web page rendered on a web browser *should not* implement brute-force mitigation. Any web savvy Attacker can bypass these controls by interacting with the back-end authentication server over the Internet. The mitigation technology must be implemented on the server side in this model. In mobile applications, where a local pin or password is embedded in the application's secure storage region, the mobile device must mitigate brute-force attacks in this interface.

Furthermore, after each invalid password is attempted, the mitigating system should increase the delay that is required between allowed attempts. There must also be a maximum threshold for invalid password attempts. After this threshold is reached, the user should be locked out pending either two-factor authentication or another more invasive model. Difficulty

This process is extremely simple to implement, and takes very little effort on the part of the engineering team.

6.9.1 Risk

The risk of not implementing this recommendation is:

- The ability for stolen devices to be subverted through brute-force password guessing
- "Drive by" Internet attacks can subvert the security of IoT systems by simply using hardcoded passwords
- Users can be compromised through "shoulder surfing" if the user interface displays the actual password being input into the system

6.10 Use a Proven Random Number Generator

Determine whether your TCB is capable of truly random number generation. This is important as without it, the cryptographic verification process can be impaired, making encrypted data more guessable and weakening data integrity.

This is also extremely important for unique cryptographic key generation. Given a set of environmental conditions, an adversary must not be able to influence the environment to cause a TCB to generate predictable numbers during key generation, signing, or cryptographic message padding.

This process is as simple as identifying whether the TCB is capable of FIPS [10], EMVCo [11] or Common Criteria [26] approved random number generation.

6.10.1 Risk

Utilizing cryptography without a strong random number generator is dangerous for many reasons. While the reasons are too many to list here, there are some key weaknesses that should be noted:

- Cryptographic key generation may be compromised, causing weak or predictable keys to be generated
- One Time Passwords/Pads or keys may be weak or predictable
- Message padding used to negate the potential for message replay may be compromised

These issues can result in significant failures in the overall integrity of the cryptographic security of the entire IoT ecosystem. This risk does not only affect the Endpoint device, it affects the entire network.

6.11 Cryptographically Sign Application Images

All applications stored outside of a CPU's core EEPROM must be cryptographically authenticated. To do this, simply follow the procedure:

- Identify the metadata representing the version of the application image
- Generate a cryptographic hash of the application image, including the metadata
- Validate that the application metadata matches the internal metadata
- Validate that the hash value matches the value internal to the trust anchor
- Cryptographically validate the signature with the Application Signing Key
- Cryptographically validate that the Application Signing Key was signed by the Organizational Root

This process is ordered to perform the most volatile activities first, and the operations least likely to fail last. This way, the least amount of work is performed in order to observe the most likely risks.

This process is exceptionally easy to implement, especially when the TCB is capable of performing the brunt of the processing on the application's behalf. The real challenge is more subtle: it is *what application* is performing the operation.

An application that hasn't been cryptographically verified cannot perform the operation, as it has no way of knowing whether its own code has been subverted by an adversary. Altering code in NVRAM is a common way for Attackers to manipulate embedded systems, if the embedded system doesn't verify the application.

An internal EEPROM application must, instead, perform this procedure first, on any application image in external persistent storage. Then, that application may either perform the operation itself, or may request an application encoded into internal EEPROM to perform these types of tests on its behalf.

6.11.1 Risk

If the application image stored in Endpoint firmware (NVRAM) is not cryptographically signed, the system will not be able to differentiate between authorized code and code injected by an adversary. This could allow not only an adversary to abuse executable code to manipulate a physically compromised Endpoint, but could allow a rival business to install their own software on an Endpoint.

6.12 Remote Endpoint Administration

While not all Endpoints require remote administration, the ones that do must be architected in a way that ensures that third parties cannot abuse administrative credentials to compromise some (or all) of the Endpoints in the field. The appropriate solution will depend on the capabilities of the Endpoint. However, the following guidelines should be used

- Do not place private cryptographic components in insecure storage on Endpoints, such as SSH private keys, TLS private keys, or passwords
- Where possible, generate administrative tokens (cryptographic keys or passwords) per each Endpoint
- Where passwords are used, enforce the use of passwords that conform to best practices regarding password complexity and length
- Where possible, enforce two-factor authentication for administrators
- Ensure that the end-user is made aware when an administrator remotely accesses the Endpoint
- Consider restricting administrative access to a Virtual Private Network (VPN)
- Do not embed remote administrative capabilities into a publicly accessible application or API, use a separate and distinct communications channel
- Enforce confidentiality and integrity on the administrative communications channel
- Diminish the potential for replay of administrative commands by ensuring the communications protocol has adequate entropy by using an industry standard communications protocol

6.12.1 Risk

Failure to define, implement, and enforce a policy on remote administration may result in the remote compromise of Endpoints. If there isn’t a rigid security model for super-user access to the Endpoint devices, adversaries may be able to reverse engineer the technology, or extract security keys from the Endpoints that will result in access to every Endpoint in the ecosystem. Administrative access is often one of the first technologies abused by adversaries in embedded systems, as they are often misconfigured or technologically weak.

6.13 Logging and Diagnostics

In order to assess problems with Endpoint devices, the IoT Service Provider should constantly evaluate the behaviour of the Endpoint and determine whether the Endpoint is functioning within the set of approved behaviours. To accomplish this, three strategies should be used

- Anomaly detection
- Endpoint logging
- Endpoint diagnostics

An Endpoint should log its own behaviour and intermittently upload this log to back-end services for processing. This log should be composed of normal activity such as kernel logs, application logs, and other metadata.

Diagnostic information should also be observed at regular intervals and delivered to the back-end service either with or separate from normal logs. Diagnostic messages should include as much environmental data about the Endpoint as possible, including temperature, battery life, memory usage, execution time, process lists (where applicable), and more. This information will help to identify when – and what service(s) – is related to a problematic or anomalous event.

Anomaly detection in the network should assist in catching a problem that can't be revealed through log or diagnostic analysis. It also will help to classify issues that can be observed in the logs or diagnostics, or attribute the issues to a specific component that may be reacting poorly in the physical world. For example, a Cellular module that keeps reconnecting to the network, or a sensor that generates bad data.

Together, this information will not only help identify whether a flaw in the technology is observed in the field. It will also help to identify whether anomalous behaviour is indicative of a security event.

6.13.1 Risk

Failing to implement logging and diagnostics may cause the organization to miss critical information. This information may not simply impact the security of the ecosystem, but may help diagnose critical product engineering flaws.

6.14 Enforce Memory Protection

Embedded systems are often designed with microcontrollers that are not capable of robust technology such as Memory Management Units (MMU) and Memory Protection Units (MPU). However, these technologies *must* be used in any platform that wants to:

- Run unprivileged applications
- Run untrusted (third-party) apps or applications
- Run an emulator or virtual machine in an unprivileged process

Any environment that requires an unprivileged application to run must be able to secure itself from rogue or compromised applications. This ensures that these rogue or compromised applications cannot access areas of memory that control privileged resources such as the TCB, the trust anchor driver, or hardware peripheral registers.

The challenge in this area is often migrating from an eight-bit microcontroller platform to a more robust platform, such as a 32bit microcontroller or a full processor architecture. However, there are many operating systems available either free or with a nominal license fee for embedded systems that correctly implement memory protection with either an MPU or MMU.

6.14.1 Risk

If these technologies are not used, rogue or compromised applications will not be restricted from altering core resources such as drivers, peripheral registers, or even privileged services such as the kernel and other applications. A lack of memory protection allows any application to have full access to the full range of memory present on the microcontroller or processor. Unprivileged applications *must* be restricted from abusing these resources.

6.15 Bootloading Outside of Internal EEPROM

Most bootloader code is embedded within Electrically Erasable Read-Only Memory (EEPROM), internal to the CPU. This is not always the case, however. Determine whether your CPU loads its Bootloader from an external source. If the CPU has no EEPROM allowing it to verify the Bootloader code, it may be manipulated by a local Attacker to configure the CPU in a fashion beneficial to the Attacker.

Depending on the level of protection provided to the chip or region of memory hosting the Bootloader, an adversary may be able to use a local bus (such as Serial Peripheral Interface (SPI)) or a remote API (such as Firmware Over-the-Air) to manipulate the embedded code. This will result in an adversary being able to subvert the computing platform by placing custom code at the most trusted point of execution, the first stage of executable code. Another attack could be an adversary simply swapping one Bootloader chip for their own chip containing custom instructions by desoldering then soldering the new chip. Without a way to verify the integrity of the external code, the user will be unable to distinguish between approved and unapproved software.

In order to customize a boot loader, an Attacker would either need to develop, or outsource, bootloader development. Depending on available resources, and the target processor, the difficulty of this action can range wildly from extremely easy to extremely hard.

Consider using a CPU or MCU/MPU with an internal EEPROM or lock-capable NVRAM to store the bootloader. This will help to ensure that the platform can at least verify the first executable loaded and executed by the architecture, resulting in a more trustworthy device.

6.15.1 Risk

Not evaluating the chain of trust and enforcing a verification of integrity for the initial code loaded by the CPU can result in a full system compromise. This step is critical toward securing the IoT Endpoint device and, thus, the ecosystem.

6.16 Locking Critical Sections of Memory

Critical applications stored in executable regions of memory, such as first-stage bootloaders or Trusted Computing Bases, should be stored read-only. This ensures that the device can be booted into a valid configuration without interjection from an adversary. Without this assurance, executable code loaded after the first stage of execution will not be able to trust that it was booted into a valid configuration or state.

While it is true that adversaries can still subvert the system by replacing these critical sections of memory with their own code, it requires them to build their own custom version of the software, which may be a complex and challenging process. This vastly increases the overall cost of the attack, and the skill required to succeed. Additionally, if personalization and provisioning are used, these steps will force the Attacker to recreate the process for every Endpoint, customizing their solution to the unique cryptographic characteristics of the local system. This makes the overall attack exceptionally costly, and decreases the feasibility.

To remediate this risk, simply identify whether the technology that stores critical sections of memory is capable of being locked. Alternatively, start with a lockable EEPROM technology.

Ensure that if a lock is used, the lock is not set in software. Software-defined locks are enabled only after the software has executed the respective functionality to engage the lock. There will be a few millisecond window in which an adversary can abuse the unlocked state toward their gain. Thus, hardware locks, such as fuses or lock bits, should always be employed where possible.

6.16.1 Risk

Without a lock or read-only state, critical sections of memory can be easily altered by an adversary. This may give them enough privilege to compromise the entire Endpoint platform without further action, subverting all subsequent security controls used in the system.

6.17 Insecure Bootloaders

The job of a bootloader is to not only configure the CPU for execution of a primary application, but to load and transfer executive control to the application. To achieve this, the bootloader typically finds and loads the main application into main CPU memory. The problem arises when default bootloaders are used on certain types of systems.

Many bootloaders used by microcontroller vendors, for example, allow external firmware to be loaded into CPU memory for execution, or allow firmware updates over serial interfaces. Other bootloaders may prompt a user for locations that contain application images, allowing a user to execute any application they choose.

While this functionality is expected in an environment such as a desktop, laptop, or even server, this is unacceptable in embedded systems. This is because if a bootloader loads and executes an unverified and untrusted application, there is no guarantee as to the reliability or security of the executed application, leaving the state of the embedded device in question.

Therefore, to remediate this issue:

- The bootloader must be capable of cryptographically verifying the application image to be executed.
- The default/standard bootloader should *not* be used if it allows alternative images or firmware flashing.
- The bootloader must not allow application images loaded from arbitrary storage locations.
- The first-stage bootloader executable image should be locked in EEPROM and should only be updated through a secure process.

Furthermore, the design of a bootloader should be subject to scrutiny by a third-party security analyst. Compromising a bootloader through manipulation of bugs in the software can lead to execution of custom code, or a bypassing of integrity verification checks. This may lead to *jailbreaking*, which may not be beneficial to the business. Ensure that all bootloaders used in the system are thoroughly audited for software programming flaws that could lead to security risks.

6.17.1 Risk

An insecure bootloader can be as damaging as a poorly architected bootloading process. Securing the bootloader is a critical step toward ensuring the integrity of the IoT Endpoint.

6.18 Perfect Forward Secrecy

Perfect Forward Secrecy (PFS) deals with the disclosure of cryptographic keys exchanged during the setup of communications between two Endpoints. Generally, the Endpoints will have asymmetric certificates used to authenticate their identities. Upon completion of the authentication phase, a symmetric key is generated and mutually agreed upon by using asymmetric encryption to protect key negotiation. Once this key is generated and agreed upon, it will then be used to secure the rest of the session between the two entities. This is used to lower the computational expense involved in asymmetric cryptography. Symmetric cryptography is computationally cheaper, which means both faster and less power-intensive in embedded or low-power technologies.

However, there is a catch. This common key agreement model presumes that the asymmetric keys are *always kept secret*. This may not be the case. In the future, a sufficiently funded entity may be capable of computing the private key for any given public asymmetric key. If the Attacker *saves every communications session* between a target entity and their peers, the entity will then be able to decrypt *every communication message from the past* by generating the private key sometime in the future.

In addition, a server’s cryptographic key may be compromised by anonymous third parties or even business insiders. If this occurs, anyone that has been storing communications messages secured by the stolen asymmetric key can now decrypt those messages.

One solution to this problem is to generate an ephemeral asymmetric key pair during the key negotiation process. Only the public key for this ephemeral key pair is passed to each side of the communication link, it can be used to traffic a symmetric key.

This ephemeral key should be generated with sufficient entropy and a key size large enough to negate the potential for a computational exhaustion attack within a reasonable period of time. This will ensure that the key negotiation process is sustainable and less likely to be subject to attack in the future.

Furthermore, this methodology ensures that peers use their persistent asymmetric key only for authentication, not for confidentiality and integrity. If this asymmetric key is stolen or exposed to the public it will only affect the authentication process, not the confidentiality and integrity of the communications channel.

To make this process even more resilient from attack, the asymmetric key used for authentication must be subject to a secure revocation process that guarantees that an Endpoint will be able to verify if a key has been compromised. The Endpoint should no longer trust that key for authentication if it has been notified that such a compromise has occurred.

6.18.1 Risk

Not implementing PFS can expose all network communications to an adversary if that adversary ever gains access to a private key used to secure the communications channel. At any time in the future, if the adversary captures the private key, all communications captured by the adversary in the past will then be decrypted. This will lead to serious consequences.

6.19 Endpoint Communications Security

While covered in several other recommendations and risks throughout this guide, it is important to succinctly note that Endpoint communications security is the biggest threat to Endpoints in IoT. The ability for an adversary to manipulate the communications channel is the simplest way for an Endpoint to become compromised.

As a result, Endpoint designers must implement communications security from the following perspectives:

- Authentication of network peers
- Confidentiality of data
- Integrity of messages

Although clear-text messages can be sent and received in order to interoperate with Endpoints designed by other organizations, data communicated over *any* channel that incorporates commands, user privacy data, or critical system messages *must* be secured. The first step is to authenticate the peer device to ensure that it is what it claims to be. This is especially important if the peer represents a system service.

Next, data confidentiality is required to ensure that third parties cannot read critical data passed over a communications channel.

Finally, message integrity is required to ensure that secret messages have not been tampered by an adversary.

These three attributes, combined together, will result in a communications model that can survive for years with few engineering changes.

This process is made far simpler through the use of existing and well analysed security protocols, such as, but not limited to:

- The latest approved TLS standard
- The latest approved DTLS standard
- SSH2 for authentication and key exchange
- GBA for key generation and exchange
- OAuth2 for authorization
- BEST, Battery Efficient Security for very low Throughput Machine Type Communication (MTC) devices [21]

While the engineering team can use any suite that adheres to the aforementioned requirements, utilizing a standard communications protocol suite will reduce the number of errors that will be observed in the field. This is because experts in information security and cryptography are involved in the development of standardized protocols.

The security properties of 3GPP based Cellular communications technology, including the standardised LPWA network technologies NB-IoT and LTE-M, can be found in GSMA PRD CLP.14 [4].

6.19.1 Risk

While it should go without saying that communications security is a requirement, it is sometimes confusing as to *why* it is a requirement. Communications security doesn’t just ensure that an adversary can’t read data. It also ensures:

- An Endpoint cannot be impersonated
- A critical service cannot be impersonated
- Abused messages can be detected
- Changes to software or security configurations can be performed safely

Without communications security, there are no guarantees as to the quality, reliability, or privacy of an IoT product or service.

6.20 Authenticating an Endpoint Identity

If each Endpoint carries a cryptographically unique identity, such as a unique serial number, the device must be able to prove that it *truly represents that serial number*. To do this, the TCB must cryptographically sign a message with a key known only to the TCB and to the IoT back-end service, a complexity that can be managed with technologies such as GBA. The message should contain the unique identity (serial number or other token) and metadata respective to the Endpoint.

The message to be signed by the TCB must also contain a challenge issued by the back-end system. This negates the ability for an adversary to *replay* an authentication message already submitted from the TCB to the back-end. If sufficient *entropy* is contained in the challenge, the potential for message-replay is negated.

To challenge an Endpoint’s identity:

- Receive a request from the Endpoint which contains the unique identity token
- Generate a unique challenge and send it to the Endpoint
- Receive the challenge reply from the Endpoint containing the signature and the message
- Verify that the signature is correct using the shared key
- Ensure that the signed message contains the correct identity token and any other relevant metadata
- Acknowledge the verified signature

To process a challenge:

- Connect to the back-end system
- Receive the back-end system’s cryptographic identity
- Cryptographically authenticate the identity of the back-end system using the TCB
- Send a message containing the Endpoint identity and other metadata to the back-end
- Receive a challenge from the back-end
- Generate a message containing the unique identity token, metadata, and the challenge
- Sign the message
- Send the message and its signature to the back-end

- Verify that the back-end system approved the signed message

6.20.1 Risk

The risk of not implementing this recommendation is that Endpoints will be clonable or vulnerable to impersonation attacks. This can open up the organization's infrastructure to attacks from both competitors and adversaries. Competitors can use a lack of Endpoint identity authentication to build a competing platform from the same Bill of Materials, but at a lower cost.

Alternatively, a competitor can use the lack of authentication to sell hardware that piggybacks off of the organization's infrastructure. These issues can result in a loss of revenue for the business, and an increased operational expense as the competitor can benefit from the use of the business's network infrastructure, even though they are not paying to use it. Since network bandwidth has a quantifiable cost, and Cloud servers, CPU usage, disk usage, and other resources have a quantifiable cost, this kind of parasitic business can have a serious impact on a vulnerable organization.

7 High Priority Recommendations

High priority recommendations represent the set of recommendations that should be implemented, but only if the Endpoint architecture requires it. For example, not all Endpoint architectures require tamper resistant product casing. These recommendations should be evaluated to determine if the business case deems them a requirement.

7.1 Use Internal Memory for Secrets

Where possible, processors should use internal CPU memory for the processing of core secrets and cryptographic keys not contained within a trust anchor. This will ensure that if an adversary is monitoring, or capable of manipulating, the memory bus, they will not obtain core secrets, but will only see the effects of the use of these secrets on a running application.

This model will create longevity with regard to the cryptographic secrets, forcing the Attacker away from uncovering those secrets. Instead, the Attacker will need to rely on manipulating bits in the RAM that equate to the *effects* of using said secrets. This will require the Attacker to change bits in memory every time the secrets are used internally, massively increasing the complexity of the attack.

Not all operating systems define models for utilizing internal RAM for the processing of secrets. Therefore, it might be required for the engineering team to implement this themselves. While this process is not difficult, it is not trivial, either. The executable code must ensure that its memory routines all use specific regions guaranteed to represent internal processor memory. This may require extra work, depending on the operating system and compiler toolchain used.

7.1.1 Risk

Most microprocessors and some CPUs have a small amount of internal SRAM dedicated to code running from internal EEPROM or internal NVRAM. This SRAM is typically inaccessible to external peripherals, unless it is purposefully exposed by using technology such as DMA.

If kept private, cryptographic secrets processed by the code have a much smaller likelihood of being exposed to adversaries capable of intercepting RAM communications.

While it is not a high risk, cryptographic secrets should not pass over publicly accessible busses, in order to diminish the potential for attack. Well equipped adversaries capable of intercepting RAM communications at potentially high speeds *can* capture data such as cryptographic secrets. However, it would take a skilled reverse engineer to capture messages across RAM that could be attributed to cryptographic operations.

As a result, while this is an important recommendation, it may not be critical to ensuring physical security. If core cryptographic keys are stored within the trust anchor, and only session keys are processed by the application, processing the keys in external RAM is not likely to result in an immediate compromise. However, this presumes that the cryptographic architecture limits the exposed keys to those that are not critical to core IoT operations, such as key rotation, session key generation, and certificate revocation.

7.2 Anomaly Detection

Modelling Endpoint behaviour is an imperative part of IoT security. This is because a compromised Endpoint can be indistinguishable from an Endpoint behaving normally if only successful interactions with the device are logged and analysed. For a more comprehensive perspective of an IoT environment, the full behavioural fingerprint of a device should be catalogued to identify anomalies that may be indicative of adversarial behaviour.

Anomalous behaviour emanating from an Endpoint may include:

- Erratic reboots or device resets
- Leaving or joining a communications network at erratic intervals
- Connecting to abnormal service Endpoints, or connecting to service Endpoints at inappropriate times
- A significantly different network traffic fingerprint than normal
- Multiple poorly-formed messages sent from the Endpoint to server Endpoints

If the normal behaviour of an Endpoint type is catalogued by the IoT Service Provider, the organization will be able to identify behavioural patterns that should be indicative of anomalous behaviour. By setting a baseline of behaviour, then continually monitoring for potential outliers, the organization can more quickly diagnose both security and performance problems in production environments.

Cataloguing the behavioural fingerprint may also assist the organization in more quickly linking a faulty set of functionality to a particular feature or environmental condition. This may lead to engineering solutions at a more rapid pace than if behavioural data isn’t collected.

7.2.1 Risk

Without anomaly detection, it may take an excessively large amount of time to detect a compromised Endpoint within the IoT ecosystem. If the Endpoint’s anomalous behaviour is only visible outside of normal operations, the administrative team may have no reason to distrust the Endpoint. However, if anomaly detection is implemented throughout the ecosystem, malicious behaviour may be detected – and thus contained – far sooner.

7.3 Use Tamper Resistant Product Casing

The physical device should not only be tamper resistant at the chip level, it should also be tamper resistant at the product level. The case used in the product should provide protection from adversarial or curious users. This can be accomplished in several ways:

- Circuits that invalidate NVRAM when a casing is opened
- Sensors that blow security fuses when light is detected
- Sensors that trigger an alert when a physically static device's location is moved
- Epoxy covering core circuit components
- Alerts raised with either internal or removable components are removed from the device

Using these methodologies can improve the tamper resistance of a physical Endpoint. However, it may be more cost effective to improve the design of the circuit, itself. While these methodologies will go far to diminish the potential for compromise by amateur hobbyists or adversaries, they will not mitigate well equipped and experienced security analysts.

Thus, these methods improve the organisation's ability to ensure that the product itself cannot be tampered with while it is out of the possession of the consumer that owns it. In other words, if a consumer leaves their device at home or in the field, an adversary must not only gain physical access to compromise the device, but they must also defeat the tamper resistant security controls as well in order to alter then replace the device. This negates the ability for devices to be quickly compromised and replaced, which is a valuable improvement to physical device security.

Yet, if the threat model ignores this aspect and focuses on remediating physical attack in general, including advanced and equipped Attackers, it does not fully remediate that threat. In that case, these tamper resistant additives will slow down an adversary, but will not stop an adversary with time and expertise.

Thus, a balance must be met between what is cost effective, and the threat model of the given device. An Automated Teller Machine (ATM) is a sufficient example of such a device. Tamper resistance in the encasing is required for ATM security, to ensure that an adversary cannot open and alter the physical encasing to, say, capture magnetic stripe data and record access numbers. However, savvy adversaries have devised local-alike components, skimmers, to be adapted on top of an existing ATM. Thus, physical tamper-proofing can only achieve part of the desired result. Application and hardware design must go the extra step to diminish physical attacks.

Engineers and business leaders should evaluate the threat model of a given product or service and balance the risk of attack with the tamper resistant measures implemented in the device. Each type of tamper resistance will incur a cost, depending on the process, engineering, and materials involved. And yet, the effort may not result in the level of security desired.

An example of this problem is with coating chips with epoxy. While this process is valuable, there are two things an Attacker can easily do to bypass the use of epoxy:

- Tap circuits emanating from the epoxy covered component

- Remove the epoxy

While epoxy hides the chip component from view, it does not – and cannot – hinder the electrons traveling across circuits that emanate from the epoxy coated chip. Thus, if critical secrets are communicated across the hardware bus, epoxy will not stop the adversary's ability to intercept this data.

Furthermore, the epoxy itself can simply be removed. Home grown hobbyist techniques have surfaced in the past several years that clearly outline a practical method for removing epoxy from a circuit using consumer ready chemicals and processes. While the process can be caustic and potentially dangerous, the procedures outlined by skilled reverse engineers are sound and can be implemented by anyone with a properly vented laboratory or office.

Thus, a risk assessment must be performed that clearly weighs the benefits of the tamper resistant technology with the ease of compromise. If each device is simply to be secured from an adversary wishing to easily manipulate or abuse a random device, tamper resistance should be employed. If the requirement is that advanced Attackers must be mitigated from intercepting messages across hardware busses, a more resilient security architecture for the application and operating system should be considered over tamper resistance.

7.3.1 Risk

As noted in the previous section, the risk of not deploying tamper resistance varies wildly with the requirements for the device. If the requirement is that the device should alert the user if the physical device was opened, broken, or altered, tamper resistance is important. If the requirement is that the device should be protected from analysis by an amateur or skilled security researcher or adversary, architectural security is probably the correct resolution for the risk.

In either case, the risk of not deploying tamper resistance in the case is such that the user will not be able to determine if an adversary tampered with the physical device. While this may not mean much to applications with a robust and hardened hardware and application security architecture, it will mean *a lot* to products that offer critical services to its users, such as medical devices, telematics systems, and home security or automation systems.

7.4 Enforce Confidentiality and Integrity to/from the Trust Anchor

All communications to and from the trust anchor should be authenticated and should enforce confidentiality and integrity. The only exception to this model is if the trust anchor is internal to the core of the processor. Any external trust anchor, such as a UICC, can only be trusted if the messages received and sent can be trusted.

To do this, choose trust anchors that are capable of authentication and encryption and validate that all messages containing answers to challenges are sent confidentially and, where possible, with verifiable integrity.

UICCs that can be managed with Secure Channel are capable of confidentiality and integrity. The IoT Service Provider should discuss with the Network Operator whether the UICC Secure Channel technology can be used to assist with application security utilising a service such as IoT SAFE . In the future, eUICC will be capable of application security.

Secure Channel may then be used to facilitate Endpoint application security from the bootloader stage to the network authentication stage.

While this should seem like a simple exercise, there are subtleties to this process. Testing each aspect of the communications layer is necessary. Some messages from various trust anchors, may not be confidential or enabled with integrity. For example, a message that indicates whether an operation succeeded or failed may seem benign, but must be protected to ensure that an adversary doesn't send a tailored response, tricking the application.

Some trust anchors may not be capable of integrity in the communications channel. Integrity is preferred, and should be employed to guarantee a message hasn't been tampered with. But, doing this requires a base of trust on both the host processor and the trust anchor, which may not be reasonable for the application.

Since all embedded systems are capable of compromise from a sufficiently equipped physical adversary, it may be overkill to require a root of trust on both processors simply for local bus communications. However, in applications where physical security is critical, integrity should be implemented.

7.4.1 Risk

The risk of not enforcing confidentiality and integrity is an interesting one. This risk can range from a complete system compromise to benign information gathering. This is because certain messages *can* be gamed. For example, if a TCB requests that the trust anchor verify a message's integrity, it will pass the message over a hardware bus to the trust anchor.

If the trust anchor is internal to the CPU, it is unlikely that an Attacker can alter this message without sophisticated and expensive equipment. However, if the trust anchor is a separate chip on the circuit board, there may be an opportunity for the adversary to alter the message by splicing the circuit and inserting their own hardware. If, for instance, the trust anchor receives the message and simply responds to the query stating "Yes, this message is valid" without any integrity, the TCB will be unable to verify if the message had been manipulated by an Attacker with physical access to the bus.

Furthermore, even if the response *is* integrity verified, an adversary with physical access to the bus can simply compromise the circuit, absorb the message request from the TCB, emit their own trusted message to the trust anchor, and let the real trust anchor's response pass through to the TCB. If the hardware communications bus isn't properly secured, this attack is possible as well, negating the trust anchor's ability to perform its job.

However, expecting both the CPU and the trust anchor to have individual internal trust anchors creates a paradox. How can a bootable CPU trust itself if the CPU can be changed by an adversary, yet the CPU has to use its own EEPROM to verify the integrity of the trust anchor! This creates a conundrum, but one that can be solved.

One solution is to insert a public key into the CPU's ROM. This key can be used to verify the integrity of messages sent by the trust anchor. If an arbitrary message (to be verified) is transmitted over the hardware bus to the trust anchor, the trust anchor can respond with a signed message *that includes the original message* as a part of the reply. This verifies that the message did in fact originate from the trust anchor, and that the message being processed is indeed the message that was expected to be processed. The only concern left would be to ensure that the nonces used in message padding ensured that the cryptographic messages were not *replayable*.

With the above in mind, it is easy to identify that cryptography can fail due to very subtle issues in not just the cryptography, but the algorithms that support cryptographic communications. This is why implementing confidentiality and integrity (correctly) is so important.

7.5 Over the Air Application Updates

Remotely updating an Endpoint's *application* image can be a simple and straight-forward process. The complexity comes from over-engineering the solution in ways that don't actually address realistic security flaws. From a persistent-storage perspective, the engineering process is very simple:

- Define a location for the active application image
- Define a location for the backup application image (if any)
- Define a location for the emergency application image
- If a backup application image space exists, update this space with the active image
- Cryptographically verify the active image using the signature stored in the TCB
    - This ensures the storage media isn't corrupted as well as an adversary didn't modify bits during the write process
- Download the new image either in whole or in deltas and its metadata and signature
- Patch the active image with the deltas
- Verify the cryptographic signature using the TCB
- Reboot into the new image

If the process fails at any point, the system should either revert to a backup image to ensure the application performs as needed, or the emergency system can be used to *call home* and notify the IoT Service Ecosystem that a fault has occurred.

The difficulty comes from creating a storage model that addresses two issues:

- An Attacker attempting to manipulate the update process
- A hardware anomaly

Without a backup system or emergency partition, the device will have no choice but to fail. Because embedded systems typically don't have robust user interfaces, this may present a significant point of stress between the business and its customers. Failing as eloquently as possible is imperative not only for user confidence, but for system reliability as well.

It is notable that some Attackers may want to corrupt the update process in purpose, to force a system into a persistently vulnerable state. For example, if an exploitable vulnerability is found in the active version of the application, but a patch is available in the newest version of the application.

The benefit of this model is that even if the Attacker corrupts the network negotiation process, the back-end system has the opportunity to take note of this event. If the back-end network identifies that a node is communicating normally *except* for updates, an alert should be raised for administration to determine whether that Endpoint node is being abused.

7.5.1 Risk

If the OTA application update process is not properly architected, it can result in adversaries remotely injecting executable code into Endpoints. If the adversary has a privileged position on the network, they could potentially affect thousands of Endpoints at once. The result of the attack could range from simple code execution, to denial of service (bricking the Endpoints), or completely altering the purpose of the Endpoint device.

7.6 Improperly Engineered or Unimplemented Mutual Authentication

In communications environments, peers speak to each other through the protocol's semblance of *identity*. This means different things in different contexts, but in every environment an *address* of some kind identifies the destination of a message. Any communications module that implements a given protocol is capable of stating that it is the owner of *a particular address*. Even if a particular *implementation* of a protocol is designed, or forced, to use the hardware address of a local radio module, there is no rule that states that a user can physically alter the EEPROM of that module and change the hardware address. Even if the implementation refuses to allow a user to change the hardware address dynamically, it can still be manipulated into changing the address. The result of this functionality is, essentially, spoofing: or the act of taking on another computer's identity for the purposes of intercepting messages destined for that computer.

7.6.1 Client Authentication

All environments are vulnerable to spoofing. For example, any Cellular radio can signal that it is the owner of any given International Mobile Subscriber Identity (IMSI), whether it is true or not. Any laptop can change its Ethernet address, impersonating other computers on the Local Area Network (LAN). Regardless of whether the topology traverses a physical or an airwave space, a communication Endpoint's *identity* can be impersonated.

The protection against this is authentication. For example, in the Cellular network, anyone with the right equipment can claim to own any IMSI they choose. However, Cellular carriers enforce *authentication* by encoding a cryptographic key into the Subscriber Identity Module (SIM) that is unique per that subscriber (IMSI). When a Cellular device communicates with a base station stating that it is representing a particular IMSI, the base station will issue a cryptographic challenge that can only be solved by someone with the unique cryptographic key stored in the SIM card provisioned for that particular identity. If the Attacker cannot solve the cryptographic challenge, the base station can verify that the Attacker does *not* represent the IMSI in question, and can disallow that user from associating with the network.

The model described above depicts *client-based authentication*. This is the model where the server subsystem (including base stations) allow clients (Endpoints) to join and leave the network as long as the clients can cryptographically authenticate their identity. However, there is an inverse problem that exposes clients to manipulation: *server authentication*.

7.6.2 Server Authentication

In the 3GPP model, only Endpoints (called User Equipment in 3GPP) are authenticated. Endpoints do not authenticate the base stations they connect to. Thus, any base station can claim to serve on behalf of any Cellular carrier. Individuals capable of manipulating or building a Cellular base station may then impersonate any Cellular carrier of their choosing. A custom Cellular base station currently costs under 1,000 USD to build, but the resultant power only allows the interception of messages in the local area. Once the fake tower is built, the base station can impersonate a local Cellular carrier, and intercept phone calls, text messages, and even data, from Endpoints in the local area.

Newer 3GPP network protocols, such as UMTS and LTE, enforce mutual authentication of both entities. This allows Endpoints to cryptographically verify that the base station is serving on behalf of the Cellular carrier it claims to serve. An adversary must now break the Cellular carrier's cryptography to impersonate a base station, significantly increasing the complexity, difficulty, and cost of an attack.

7.6.3 Cellular Interrogators or Fake Base Stations

There are exceptions to this rule, however, such as Cellular interrogators. These devices, typically used by government contractors, governments, and intelligence services, are encoded with cryptographic keys provided to these entities by certain Cellular carriers, for purposes of national security. These systems use these keys to either passively intercept bi-directional communications, or to actively perform man-in-the-middle attacks against specific targets.

In the modern communications threat model, however, access to this technology is not limited to actors in the government and intelligence areas. Today, these systems can be built from parts that are only several hundred US dollars, resulting in a cost-effective fake base station capable of intercepting or impersonating Cellular communications.

7.6.4 Communications Security is Gate-To-Gate Security

Bringing up Cellular interrogators helps summarize this section quite adequately by touching on the idea that communications security is not absolute. It only protects the communication channel between two entities. These entities, however, act as *gates* allowing data to pass in and out of the ecosystems these entities are connected to.

For example, a particular SIM card may be provisioned for use in an industrial control system such as an oil well monitoring device. A SIM card, by design, is a removable component. Anyone with physical access to the oil well monitoring device can extract the SIM card and place it in a laptop. If the laptop has software on it that can simulate the functionality of the oil device, the back-end server will be unable to differentiate between the actual oil device and the laptop. Yet, the laptop will be authenticated to the Cellular network because of the SIM card! Thus, the Cellular network has authenticated the SIM card, but not the laptop.

7.6.5 Solving For Mutual Authentication

Each peer in an IoT ecosystem must authenticate all other peers that participate in that ecosystem. To accomplish this, a TCB must be used to ensure that proper cryptographic architecture is driving the communications technology. Mutual authentication can't occur if keys are easily exposed to adversaries. Review the TCB section of this document for more information.

Once authenticated, each peer must encrypt and sign messages sent to other peers in the network. Each peer that receives a message must cryptographically validate the data prior to acting on it. Since not all communications protocols are capable of mutual authentication, or have strong cryptography, it is imperative that the application engineer design a sufficient protocol that enforces confidentiality and integrity, rather than relying on the communications protocol.

Even more robust protocols that incorporate mutual authentication, such as LTE, do not address the security of the infrastructure beyond the Cellular communications network. Only higher layer protocol security can address the risk of weaknesses in infrastructure beyond the control of the Cellular carrier.

7.6.6 Risk

The risk of not adhering to strong application security is that the Endpoint must trust the security of the communications layer. As depicted in this recommendation, it may not be adequate to solely trust the network to resolve the security issues in the application. Even if the MNO can be trusted, the messages may pass through multiple pieces of networking infrastructure not owned or controlled by the MNO before the data reaches servers owned by the IoT Service Provider. Therefore, the IoT Service Provider risks anyone with control of those systems intercepting, altering, or manufacturing messages to or from Endpoint systems.

7.7 Privacy Management

An imperative aspect of IoT technology is their ability to connect the physical world to the digital world. The result of this is a gap in privacy, as the user’s physical environment is directly associated with the things they like and view online. This may cause undesirable effects over time.

As a result, it is important that IoT Service Providers consider the privacy of their consumers and develop Privacy Management interfaces that are integrated into both the Endpoint, where possible, and the product or service’s web interface.

This technology should allow the user to determine what attributes of their privacy are being utilized by the system, what the Terms of Service are, and the ability to turn off the exposure of this information to the business or its partners. This granularity and opt-out system will help to ensure that users have the right and the ability to control the information that they share about themselves and their physical world.

Note: Detailed recommendations on how to approach data privacy is contained in Annex A of the GSMA IoT Security Guidelines Overview Document [1].

7.7.1 Risk

The potential risks of not protecting consumer privacy are many. Issues from stalking, harassment, profiling, threats, and more, are realistic and practical results of not protecting user’s data.

7.8 Privacy and Unique Endpoint Identities

Each Endpoint is known digitally by a fingerprint. This fingerprint is composed of *addresses*, *serial numbers*, and *cryptographic identities* that are unique to the specific Endpoint. However, these tokens may also directly associate a device to a particular customer, location, or service. In many situations, this is undesirable. For example, smart phones can be tracked because the phone's built-in Wi-Fi address was used when actively scanning for 802.11 access points. These addresses could then be tracked as they travelled from location to location. This would allow anyone able to associate a particular Wi-Fi address with a particular user and watch their movements around the world. To combat this, smart phone software manufacturers generated random Wi-Fi client addresses when scanning for access points, making it nearly impossible for phones to be tracked in this fashion.

IoT Endpoints can be tracked similarly through Bluetooth Low Energy (BLE) addresses, 802.15.4 addresses, Wi-Fi, or even Cellular IMSI. Where possible, the IoT Service Provider should develop their Endpoint technology in such a way that a random radio address is used to connect to new environments, allowing the user's privacy to stay intact.

This is also true of cryptographic keys, such as SSH public keys. While users typically want their public keys to be known to the public, cryptographic public keys on Endpoints will expose the user's identity of a particular Endpoint, which is *not* desirable. Instead, the user should be able to select whether they want their identity known when they are connecting to a new environment.

Note: Detailed recommendations on how to approach data privacy is contained in Annex A of the GSMA IoT Security Guidelines Overview Document [1].

7.8.1 Risk

Not adequately mitigating this risk will allow users with mobile Endpoints to be tracked as their devices leave and join networks. This opens up significant gaps in privacy that legal teams, legislators, and even insurance companies are currently analysing. Not adequately implementing privacy to diminish the potential for tracking may open up a new IoT Service Provider to legal consequences in the near future.Run Applications with Appropriate Privilege Levels

Applications running on an Endpoint typically do not require super-user privileges. Most often, applications require access to device drivers or a network port. While some of these devices, ports, or other objects may require super-user privileges to initially access them, the super-user privileges are not required to perform subsequent operations. Thus, it is best practice to only use super-user privileges at the start of the application to gain access to these resources. Then, super-user privileges should be dropped.

Dropping super-user privileges is a common process that is well documented, and has been implemented exceptionally well in applications such as the Secure Shell (SSH), apache2, and other well engineered servers. The process usually encompasses:

- Starting the application with elevated privileges
- Accessing all resources that require elevated privileges
- Identifying a user identity (for example, UNIX User ID and Group ID) that the application should run as

- Fully changing the process's identity to the target user/group ID, thus removing super-user privileges from the running application

A more complex model can be seen in the SSH implementation of *privsep*, which runs a privileged service whose sole purpose is to bootstrap the main application under a target user/group identity. This way, if the service exits it can be restarted easily without the compromise of privileged resources.

For more information see: SSH Privilege Separation:
http://www.citi.umich.edu/u/provos/ssh/privsep.html

7.8.2 Risk

Running applications with elevated privilege levels can result in a full system compromise if a single application is compromised. Since super-user privileges grant an application full access to the entire running system, there is no way to contain an adversary once they compromise such an application. Dropping privileges helps contain the adversary, and limits their ability to increase their privilege within the embedded system. This may be the difference between a full system compromise and a minor annoyance.

7.9 Enforce a Separation of Duties in the Application Architecture

Applications running on an Endpoint should have different user identities associated with each unique process. This ensures that if one application is compromised, a separate application on the same Endpoint cannot be compromised without a successful second attack. This extra step required on behalf of an Attacker is often a critical hindrance to the overall exploit development process and increases the cost and complexity of an attack against an Endpoint.

For example, a network service that allows a user to retrieve information about the state of the Endpoint must not also be able to manipulate the TCB over the same process. That capability would be *out of scope* relative to the purpose of the service. These two distinct operations should be handled in separate applications, and ran under separate user IDs on the local Operating System, helping to separate the duties of the application, and reduce the risk of abuse if one component were compromised.

To implement this correctly, memory protection must be enabled in the underlying hardware architecture, and the operating system must have a concept of privilege levels. Unprivileged software must be restricted from accessing privileged resources, such as drivers, configuration files, or other objects.

Services should make requests to access privileged resources, but through a constrained API such as a system call, to ensure that all messages are well formed and fit the requirements of the security architecture.

The concept of multi-tiers of privilege is a half a century old concept. However, in embedded systems, it is often overlooked as users are not allowed to log into the console and run their own applications. As a result, all services are often deployed as a privileged user. However, this is flawed.

Each application or service must be implemented using a custom privilege. In most environments, this is a separate *user identity*. This separation of duties by enforcing different user identities ensures that if one service is compromised it cannot not directly affect the resources used by another service on the same system. To compromise other services and users, secondary exploits must be found in the local operating system to elevate privileges.

This requires planning and a sound application architecture that utilizes privilege separation correctly.

7.9.1 Risk

If a separation of duties is not enforced, any compromise to a single service on the Endpoint will result in a compromise of the entire device, because each service or application running on the device will share the same user and/or group identity. If the recommendation *is* implemented, a low privileged service that is compromised over the network *will not* immediately result in compromise of the entire system.

Because this recommendation is simple to implement, it is critical to the security of IoT Endpoints. It should be noted that it often takes a large amount of expertise to remotely compromise a network service. If the adversary is also required to elevate privileges by implementing a kernel level exploit, or another secondary exploit, to gain control of the full system, the adversary may not have the time, skills, or equipment to execute the attack.

Increasing the difficulty of an attack with a simple configuration change such as this will go a long way toward ensuring the longevity of the device.

In addition, as compromised services can be detected through process monitoring and other analytics, any service compromise can alert the Service Ecosystem that a device compromise has been detected. This allows administrators to act to secure the system before a full system compromise has been achieved. This also allows administrators to diagnose and patch the vulnerable software prior to rampant abuse of the particular vulnerability. This gives the business a significant edge against even skilled Attackers.

7.10 Enforce Language Security

Programming languages have varying degrees of security, depending on the purpose of the language and how high level it is. Some languages provide constructs for limiting access to raw memory, and enforce constraints around how memory is used. The engineering team should identify a language that is capable of providing security to the application run-time or resultant binary.

The compiler or run-time should be security hardened, where possible, to restrict the potential for a vulnerability to be abused by an adversary. In a well defined run-time environment, even an easy-to-trigger programming flaw can be extremely difficult to fully exploit. This presumes that security enhancements are used to protect the way the application executes, accesses memory, and is supported by the operating system’s security enhancements.

7.10.1 Risk

The risk of not hardening the programming language and resultant application is an easy-to-exploit application. Some programming systems such as PHP are notoriously buggy and should never be used by a professional engineering team. Other languages, such as Python, are suitable for production environments, but have subtle security risks that must be evaluated. Thus, the volatility of the resultant risk can be anywhere from a critical level to a benign level. The engineering team *must* use the risk assessment and threat modelling process to sufficiently evaluate what language is best for their production environment.

7.11 Implement Persistent Pentesting

Performing a security audit only at deployment time is not enough for most IoT deployments where new endpoints can be released in field and configured anytime. It is recommended to use a persistent pentesting approach in order to achieve early detection of vulnerable endpoint software and unsecure configurations.

Implementing a persistent pentesting strategy can provide a quick detection and early management of the threats identified, increasing the mitigation speed and reducing the threat exposure period.

A complete persistent pentesting strategy should provide an automatic and scheduled way to perform: asset discovery to create an inventory of accessible assets, asset identification and analysis, known vulnerability verification and explotation, insecure configurations check and appropriate reporting and alerts that should help with remediation.

7.11.1 Risk

The risk of not implementing a persistent pentesting strategy is that security audits may be run only once at deployment time but new enpoints and configurations are never assessed. This situation can lead to a set of vulnerable endpoints that are never identified as exposed until they are compromised by an attacker.

8 Medium Priority Recommendations

The medium-priority set of recommendations encompasses the set of recommendations that are relevant depending on the design choices of the Endpoint technology. For example, enforcing Operating System Level Security Enhancements is only valid if there is an Operating System running on the Endpoint. If the Endpoint is composed of a monolithic kernel application, or an embedded Real-Time Operating System (RTOS) with a single embedded application, the recommendation may not apply. Where recommendations *do* apply to the Endpoint design, they should be implemented.

8.1 Enforce Operating System Level Security Enhancements

Applications running on an Operating System should be designed to use (either transparently, or intentionally) the security enhancements of the underlying Operating System and Kernel. This includes technologies such as:

- ASLR
- Non-Executable Memory (Stack, Heap, BSS, ROData, etc.)
- User-Pointer Dereference Protection (UDEREF)
- Structure Leakage (information disclosure) Protection

Each operating system used in an embedded system will provide different variations and combinations of these technologies, sometimes under different names. Determine what the operating system and kernel are capable of providing, and enable these technologies, where possible, to enhance the security of applications.

The challenge comes from identifying what each Operating System is capable of. For example, applications running on platforms that have no Memory Management Unit (MMU) may not be capable of ASLR. However, the equivalent of UDEREF *can* be enforced even in environments with only a Memory Protection Unit (MPU). Evaluate which technology is being used and its capabilities, and determine what level of security can be achieved through the combination of architecture, kernel, operating system, and application protections.

8.1.1 Risk

Not enforcing this recommendation will result in an application run-time environment that is substantially easier to exploit. These enhancements will significantly constrain the number of adversaries that are capable (if at all) of developing a reliable exploit for a vulnerable service.

Thus, if an application developed by the organization has a security flaw that could be abused to gain remote code execution capabilities, the potential for abuse can be negated by enforcing ASLR, NX, UDEREF, and other technologies. This will limit the ability for an Attacker to develop an exploit in a reasonable amount of time, as the exploit developer will be required to use advanced and challenging techniques that must be customized against each individual target. This increases not only the difficulty, but the time and expense required to achieve a fully working exploit.

Without these enhancements, a fully working exploit can be developed using off-the-shelf and freely available software within hours.

8.2 Disable Debugging and Testing Technologies

When a product is being developed it is often enabled with debugging and testing technologies to facilitate the engineering process. This is entirely normal. However, when a device is ready for production deployment, these technologies should be stripped from the production environment prior to the definition of the Approved Configuration.

The Approved Configuration that a product is deployed with should never contain debugging, diagnostic, or testing interfaces that could be abused by an adversary. Such interfaces are:

- Command-line console interfaces
- Consoles with verbose debugging, diagnostic, or error messages
- Hardware debugging ports such as JTAG or SWD
- Network services used for debugging, diagnostics, or testing
- Administrative interfaces, such as SSH or Telnet

All such technologies should be disabled in the Approved Configuration.

Serial ports that can be removed by the system should also be physically removed from the circuit board. However, many times serial ports such as UART/USART are enabled over hardware pins on the microcontroller or processor. If these pins are still enabled as a console, an adversary can simply tap the pins to interact with the console. Removing the physical serial port itself, such as a DB9 interface, does not disable the console.

Furthermore, debugging ports such as JTAG and SWD should not simply be disabled via software. These devices should be disabled by altering security fuses or locks. Disabling these technologies from software offers a window of opportunity for an adversary to connect to JTAG, SWD, or a similar hardware debugging interface prior to the time at which the software disables the interface. This window of opportunity is often sufficient enough for an adversary to succeed.

8.2.1 Risk

Without implementing this recommendation, organizations open themselves up to extraction of critical secrets from the central processing unit. This may allow adversaries to load their own firmware into NVRAM or EEPROM, and allow them to extract or alter critical secrets that further compromise the IoT network or device.

Disabling debugging ports is a critical step to ensuring the integrity of the IoT product or service. However, it is important that the organization evaluate the risk of disabling these technologies and weigh them against the benefit of being able to diagnose and debug problems identified in the field. It may be significantly more challenging to remediate production-level flaws in the product if there is no way to debug a running system.

8.3 Tainted Memory via Peripheral-Based Attacks

Processing systems rely on consistency to ensure that the output of algorithms is predictable with respect to a set of given inputs. Processing systems also expect components to act reliably, and that for every bit written, that bit is stable and unaltered until it is changed by the processor. Within closed systems, this theory is applicable. When anomalies to this model occur, they can compromise, or simply damage, a processing environment.

Information security presents the class of anomalies purposefully induced in order to gain access to objects that otherwise would be inaccessible. An abusable window for the induction of anomalous behaviour beneficial to an adversary is Direct Memory Access (DMA). Put simply, DMA is a technology that processors can use to allow external components (peripherals) to gain access to main processor memory without interference by the CPU. In other words, the CPU can grant a peripheral direct access to a region of memory. This peripheral may then read or write to that region of memory.

If the processor does not properly restrict the region of memory usable by the peripheral, the peripheral may have access to more of main memory than is required for the intended functionality. In other words, if the peripheral (say, an Ethernet controller) is allotted a DMA region intended for use as a circular buffer for received Ethernet frames, and the DMA region allotted comprises the entire expanse of main memory, the firmware on the Ethernet controller may now arbitrarily read and write to all system memory. The CPU will have no way to block the Ethernet controller firmware from writing to memory.

The result of this attack is two-fold. Data can be leaked from main memory and encoded into network packets or application information for covert or immediate exfiltration. Alternatively, an Attacker could covertly insert a backdoor (malware) into main memory by overwriting an application’s executable code.

From the processor’s perspective, there is little it can do to identify whether an overly permissive window of memory has been abused by a malicious peripheral device. To combat this attack, identify whether the processor used in the Endpoint system is capable of restricting DMA to small predictable regions of memory. If so, ensure that each region of memory is defined per each peripheral device that requires it. Do not enable arbitrary windows memory, where possible, to peripherals.

Some processors may not allow granular restriction on the size or location in linear or virtual memory of a DMA window. As DMA attacks should be considered a realistic threat to IoT Endpoints for critical applications, evaluate whether it makes sense to consider an alternative processor with more granular features.

For platforms that expose ports such as IEEE1394, Thunderbolt, Express Card, or other ports that allow direct access to Peripheral Component Interconnect (PCI) DMA, canned and cost effective attacks are already available.

For platforms where a DMA based attack requires abuse of a local hardware component, the difficulty will certainly increase, but it is not out of the scope of an offense-based security engagement to reflash a peripheral's firmware in order to subvert DMA for compromise of a local Endpoint. Cost, time, and expertise will however be a factor, making the actor in this case likely a sponsored (paid) adversary.

8.3.1 Risk

Choosing not to restrict the ability for DMA to be abused by external components may subject the platform to a full compromise, or, at least, the extraction of key secrets, privacy-centric data, or intellectual property from the Endpoint.

8.4 User Interface Security

IoT Endpoints that have user interfaces such as touch screens, rich displays, or alternative interface technologies, must be able to render information to the user and take information from a user in a secure manner.

While attributes of the user interface, such as passwords, have already been covered in this document, there are some more subtle issues that must be discussed:

- Alerting systems
- Action confirmation

When an anomaly has occurred, such as physical tampering or an application behaving in an unintended fashion, the user should receive a visible alert. Alternatively, the user should be able to review alerts from the system from within the User Interface.

Furthermore, all actions performed by the device that are driven by encodings or seamless transitions from one interface to another should be confirmed by the user. An example of this is if the device camera reads a QR code, or a NFC or RFID interaction requests that the device connect to an URL. In these cases, the user should be prompted to confirm the action and validate that the action performed is desirable. The user should be given the option to cancel the action. The user should be able to view all details about the given action, including the full URL that will be connected to.

8.4.1 Risk

If this recommendation is not implemented, users will be vulnerable to attacks that cannot be detected. While some system designers appreciate the seamlessness of transitioning from an RFID chip to, say, the corresponding product website, there may be undesirable effects of this behaviour. Users could be forced to view undesirable materials without their consent, or users could be tricked into visiting websites or performing actions that weakens their security posture or privacy.

Also, users that have a difficult time reviewing their Alerts may not understand the risks of using a potentially tampered device. This may decrease the user's physical security and could put them at risk.

8.5 Third Party Code Auditing

Any time a section of code, such as a bootloader, is a critical component in constructing a secure run-time platform, it must be audited for risks. If a bootloader can be manipulated by an adversary into executing untrusted code, or into bypassing the authentication sequence, it is rendered useless. This would negate the finances, time, and experience utilized by the organization in the deployment of this technology, nullifying the engineering expense.

A gap in security in this area may also result in a competitor's advantage against the business through spoofing, API abuses, data interception, device cloning, and even device rebranding. Thus, it is imperative that critical sections of code be audited by an approved third party, to ensure that technology is not at risk of abuse. Therefore, to find an information security team adequate to perform the audit, evaluate what types of code will be audited. Typically, in this model, that means: C, Assembly, and possibly C++ or Java.

Identify a team that is well versed in these languages, as well as the underlying architecture. While many information security teams perform source code auditing, not many of them may perform auditing on the particular platform used by the IoT business. Each platform has subtle differences, and it is best to use a team with familiarity with the platform being used.

8.5.1 Risk

While hiring third party consultants to evaluate internally developed technology can be a challenge, it is a requirement for security. This is because engineers developing technology must be able to show that their architecture is provable. This is difficult to do if the engineers developing the architecture are the only ones reviewing it. Engineers tend to visualize their code base from the architecture that they have *attempted* to design and implement, not from the *actual implementation*. Thus, third party eyes are often needed to find subtleties in the architecture and implementation that could cause gaps in security.

8.6 Utilize a Private APN

In 3GPP Cellular networks, an Access Point Name (APN) acts as a private network configured specifically for a set of authenticated devices. Typically, a private APN (also called "secure APN") is a private network only accessible to authenticated devices associated with a specific business. By utilizing an APN, businesses can restrict what Endpoints are allowed to connect to their service infrastructure over the Cellular network. This helps to reduce the amount of users that have direct access to IoT services in the back-end infrastructure.

Other attributes of a private APN can help diminish the potential for rogue Endpoints to abuse the IoT ecosystem. Firewalls can limit what services or computers can be connected to from the APN. A well configured APN will disallow Endpoints from making direct connections to each other, which disallows a compromised Endpoint from migrating horizontally through the network infrastructure to other Endpoints.

Engage with the Cellular carrier or Mobile Virtual Network Operator (MVNO) that the organization is working with to determine what technologies are available within the secure APN. Other services such as monitoring, blacklisting anomalous devices, and tying user identities to actions, may be available.

8.6.1 Risk

Utilizing a private APN can alleviate many types of attacks. For example, private APNs allow the business to reduce the amount of connections that can be made from the Endpoint directly to the Internet. Endpoints should never be allowed to directly connect to untrusted Internet resources. Only partner organizations should be trusted, and those services should be authenticated.

Without the use of a private APN, compromised Endpoints can communicate to any Internet service or protocol without restriction. This may allow an adversary to abuse the Endpoint in order to launch a secondary attack on separate infrastructure. This could involve a denial of service (DoS) attack, or could help facilitate a more dangerous attack against another business, government, or civilian.

It is notable, however, that a private APN does not mitigate the risk of an adversary compromising the communication link between the Endpoint and the private APN. In addition, the private APN only acts as a gateway to the back-end services, and doesn't enforce any security between the APN and the back-end services on the IoT Service Provider's private network. These potential gaps in security must be addressed separately, regardless of the improvements that are granted through the use of a private APN.

8.7 Implement Environmental Lock-Out Thresholds

Components within an embedded system are designed to be used within certain environmental thresholds. This includes voltage levels, current draw, ambient or operating temperature, and humidity. Each component is typically rated for certain windows of approved levels. If the device is subjected to states above or below a given window, the component may act erratically, or behave in a fashion that is useful to an adversary.

Therefore, it is important to detect changes to these environmental levels to determine whether the device should continue running, or if it should power off. It should be noted, however, that powering off may be a desired effect, and that the adversary may abuse this engineering decision to leverage a denial of service. The engineering team should evaluate this model to determine if it is more beneficial to shut down or more beneficial to attempt to stay online.

Regardless, the model usually incorporates

- Brown-out and Black-out detection, when voltage drops too low
- Voltage ceiling circuitry protection to ensure voltage levels don't exceed a threshold

- Current limiting circuits to ensure current draw cannot drop or exceed certain levels
- Internal temperature monitoring for CPUs, MCUs, and other components that monitor internal levels
- Optionally, humidity levels can be assessed to determine if the environment is becoming too humid or too arid

Temperature is extremely important as high temperatures can indicate a circuit problem triggered by the user, the environment, or even a hardware or software issue. Monitoring the temperature will allow the operating system or application to shut down resources (or the entire device) to ensure a fire or another issue isn't caused by the Endpoint.

Low temperature levels also change the behaviour of a device. This may slow a circuit down, or cause its components to react in unexpected ways. This may be useful to an Attacker if temperature can cause a predictable anomaly that affects the application or circuitry in a beneficial manner.

Difficulty in lock-out thresholds manifests when analysing temperature and humidity. Voltage and current levels should be mitigated by Brown-out and Black-out circuitry either on the circuit board or in the processor. Since engineers will be able to look up the numbers related to a chip's voltage and current thresholds, they can easily implement protections for these issues.

For temperature and humidity, the decision to act is a bit more challenging as these levels can be manufactured by an adversary without touching the physical device. In the case of temperature, levels that may be indicative of a pending safety event must cause the device to take adequate measures to lower the temperature. However, in critical environments such as Industrial Control Systems or Medical Devices, the device should attempt to continue performing critical operations, where possible. If levels exceed a certain defined point that engineers and business leaders agree upon, only then should the device shut down.

8.7.1 Risk

For voltage and current draw, the risk of abuse is related to glitching and other side-channel attacks that can benefit from changes in these levels. If Brown and Black-out detection is implemented on the processor, the risk of abuse is lowered. Otherwise, the risk is related to spikes in voltage or current that could cause safety issues with the physical device, or allow an Attacker to instrument glitching (and similar) attacks to subvert the security of the components.

These issues should be remediated through the use of circuitry on the PCB that protects the components against anomalous spikes or drops in voltage or current.

For environmental level changes that are dramatic, the risk is related to the safety of the user. High temperatures caused by excessive CPU usage or other anomalies can cause burns, chemical burns, or even fires.

8.8 Enforce Power Warning Thresholds

Endpoints that provide critical services to the user must be enabled with a warning threshold that indicates power-related events. These events may include:

- Low battery state
- Critically low battery state
- Black-out events
- Brown-out events
- Switch to battery back-up events

The user must be warned in a sufficient amount of time to allow them to compensate for the loss of power. This could be accomplished by enabling an LED that denotes a particular power state such as green for OK, orange for Low, and red for Critical.

Systems that are connected to alternating current mains power should be configured to warn the user when black-out or brown-out events have occurred. Also, the Endpoint should log these events in persistent memory to ensure that the user and administration can retrieve the information at a later time. The information should be time stamped.

The challenge in this process is identifying at what rate the battery's power is being depleted and the extra energy required to notify the user of a change in power state. This can all be achieved through electrical engineering, and should not be too challenging of a process for experienced engineering firms.

8.8.1 Risk

Without a well defined power warning system, the users will be unable to adequately prepare for potentially critical power events. While this may be benign in the case of simple devices such as pace counters, timers, and other wearable devices, more critical devices such as personal trackers, telematics systems, and home security systems can be seriously impacted by the loss of power.

8.9 Environments Without Back-End Connectivity

8.9.1 Method

Endpoints, especially Gateways, or Endpoints acting as Gateways, must be capable of enforcing communications security even in environments where connectivity to the back-end network is unavailable. Regardless of whether this lack of connectivity is temporary or not, the Gateway or Endpoint must be capable of enforcing security as if the back-end system were available.

To achieve this, the TCB must be used to authenticate all peers that the Endpoint must communicate privacy-centric, configuration, or command data to. The TCB can be used to ensure that messages sent and received from peers are being sent and received from an entity that has been provisioned by the same organization. This reduces the likelihood that an adversarial device is being communicated with.

Interoperability can still be achieved by communicating with other devices that cannot be authenticated. However, the type of information that is communicated to these devices should be restricted to inter-operational and non-sensitive classes of data.

The challenge comes from deciding what Endpoints to authenticate and what Endpoints to communicate with in clear-text. The organization must decide what types of data are classified and should be kept from unauthenticated peers. Once this classification of data is achieved, the organization will be able to determine what peers are reasonably trustworthy even without the assistance of the core IoT services.

8.9.2 Risk

The risk of deploying solutions to communication-less environments is that it opens up an opportunity for competition to abuse the infrastructure. Competitors can undercut the business by offering interoperability and using connection-less sites as a proving ground.

Instead, the organization can choose to allow interoperability, but to a point. Certain core intellectual property and services can then be reserved for only authenticated peers that are validated through the use of a TCB. This helps to reduce the exposure of the business to intellectual property problems and adversarial competitors.

8.10 Device Decommissioning and Sunsetting

All Endpoint devices have a lifecycle, as discussed elsewhere in this document. Some devices must be decommissioned due to a user cancelling their subscription, while other devices must be decommissioned due to anomalous or adversarial behaviour. Regardless of the reason, the business must be prepared to decommission the device securely using their TCB and communications model.

Sunsetting, as discussed elsewhere in this document, is the process of decommissioning an entire network of devices and the services supporting those devices. A product or service that has been deprecated by a business, or a business deciding to shut down, must sunset their devices and network in order to diminish the risk of abuse by adversaries taking over the sunsetted network.

To accomplish this, the TCB and supporting protocols should be used. Generally, the process is to:

- Create a Decommission message from the Service Ecosystem
- Tailor the message to the unique Endpoint receiving the message
- Sign the message using the Decommissioning PSK or asymmetric key
- Push the message down to the Endpoint
- Receive a message from the Endpoint cryptographically acknowledging Decommission
- Invalidate the Endpoint in the Authenticated Device list
- Disallow further communication from this Endpoint

On the device side, the application running on the software should

- Connect to critical back-end services over Service Ecosystem
- Query the service for critical messages
- Receive the Decommission message
- Verify the message signature using the TCB and the trust anchor
- Generate the Acknowledgement message and cryptographically sign it using the Personalized PSK or asymmetric key
- Perform the Decommission operation
- Send the message back to the critical service

It is important that the message be signed and prepared for transmission prior to decommissioning, as the decommissioning process includes the invalidation and removal of security keys from the trust anchor. Because of this process, the keys used to sign the decommission message will not be available. The service requires the reception of a message that is integrity verifiable to ensure that the Endpoint did indeed receive and process the message.

The difficulty with this process is primarily in that decommissioning a potentially compromised device presumes that the device has not been compromised to the point where it will reject the decommission command. If it has been sufficiently compromised, it may not honour the decommission command.

As a result, it is imperative that the backend system running in the Service Ecosystem invalidate the Endpoint from being able to communicate with critical services. If the device does attempt to interact with networked peers or critical services, the backend system should raise an alert and let the administration know that the anomalous event has occurred.

8.10.1 Risk

The risks of not implementing decommissioning and sunsetting are many, from complete takeover of an entire network by adversaries to allowing compromised devices to continue using networked services. The most common risk is associated with users that have ended their subscription with an IoT Service Provider. If these users are not decommissioned from the network, they may be able to continue communicating with other peers in the IoT Endpoint network, or may be able to access services that should no longer be accessible to the user. This incurs a cost on behalf of the IoT Service Provider, who must pay for the bandwidth, CPU time, and storage in the Service Ecosystem.

8.11 Unauthorized Metadata Harvesting

Modern IoT is designed to bridge the physical world with the digital world. In this modern model, the effects of technology are potentially far more invasive than in the past. Using metadata, companies or private individuals can intentionally track and monitor the behaviour of random or specific consumers.

Metadata analysis is used when communication between two network entities is encrypted, but protocol structures that identify the type of message or identity of the sender and/or receiver are exposed. This metadata can be used to derive intent.

Consider the scenario where automobiles emanate messages containing metadata that is attributable to a specific consumer. Anyone with the ability to track (either locally or remotely) these pieces of metadata may be able to monitor the consumer’s movements, and then derive behaviour or intent from these movements. If there are security flaws that are exploitable in the vehicle’s telematics system, it may be possible to track and target a specific consumer’s telematics system, putting them at risk of physical harm.

Legal organizations and insurance companies are concerned about how these risks will impact the future of automotive finance, and are beginning to get involved in legislation and standards that will determine how engineers must design telematics equipment. This change will eventually trickle down to the less active IoT verticals, as more technology is developed.

To combat metadata harvesting, encrypt as much data as possible and use unique binary identifiers for communication modules. Enforce a policy that disallows external users from being able to use the IoT system's API to derive hardware serial numbers and other trackable identities from user profiles. Where possible, disallow the structure of a message from being exposed to third parties. Do not allow actions, activities, or behaviours to be exposed to third parties. Enforce confidentiality and integrity on all data that relates to user privacy.

Note: Detailed recommendations on how to approach data privacy is contained in Annex A of the GSMA IoT Security Guidelines Overview Document [1].

8.11.1 Risk

Using weak communications security may enable data or metadata harvesting that endangers an end-user or exposes end-user privacy. As insurance agencies are building a case for enforcing end-user privacy requirements in technology, the business may put itself at risk if it doesn't take responsibility for the data their devices generate.

9 Low Priority Recommendations

Low priority recommendations encompass the set of recommendations that apply to risks that are extremely costly to combat, or are unlikely to affect the Endpoint design. While these recommendations are valuable, and the information detailed within the recommendations is important, the mitigation or remediation strategies discussed may be out of scope with respect to the business. Evaluate each recommendation and determine whether the risks described are relevant or important to the business and its customers. If the customers require these risks to be addressed, apply the recommendations.

9.1 Intentional and Unintentional Denial of Service

For radio communications, there is a constant threat of *jamming*, or the intentional broadcasting of noise or patterns that can be used to scramble legitimate signals. As radio signals are simply composed of electrons flying through space in a specific pattern, it is fairly easy to concoct a series of signals that interrupt or mangle the pattern that forms communications data.

Typically, the goal of such an attack is simple disruption, to disallow or deny service to legitimate users. In other cases, the abuse may be more purposeful. For instance, communications protocols that have no authentication mechanism can be *spoofed*. To achieve this, the actual signal must be *jammed* so that the adversary's spoofed signal is more likely to reach the intended target.

An example of this is Global Positioning Systems (GPS) spoofing. Civilian GPS signals lack encryption and authentication as it is, essentially, a plaintext broadcast signal that anyone can receive. It is also a relatively weak radio signal and is easily attenuated by environmental anomalies such as Ultra High Frequency (UHF) pre-amplifiers for television receivers and microwaves.

For devices that require location information to function properly, a jammed GPS signal can result in a reliability risk that can cascade into an information security risk, especially if spoofing is subsequently employed.

To combat jamming and other forms of intentional denial of service (DoS) attacks, develop a robust communications protocol that focuses on methods to devalue breaks in service. The network should detect whether devices have suddenly or abnormally left the network. Each Endpoint should “say goodbye” when it intends to leave the network. If it doesn’t, the anomaly should be noted for statistical analysis.

In addition, communication security keys should be renegotiated every time a device re-joins the network. The same communications security key should not be used. It should be *bootstrapped* by the same asymmetric cryptographic key, but any symmetric key derived from key negotiation should be novel per each communications session.

Unintentional jamming can occur on a radio for many reasons: environmental conditions that disallow signal propagation, malfunctioning equipment, or even adjacent equipment operating at the same frequency. Regardless of the underlying reason, engineers that rely on radio communications expect that there will be temporal conditions that cause signal degradation or loss. These losses must be compensated for through the design of the application *and* network communications protocol.

Developers are recommended to read the GSMA’s Connection Efficiency Guidelines [9] which contains advice on how to protect against unintentional Denial of Service attacks and provide guidance regarding Device Host Identity Reporting (DHIR).

9.1.1 Risk

Failing to combat the risk of intentional DoS will result in abnormal or insecure Endpoint behaviour. If the Endpoint always uses the same session key, this may be a way adversaries could abuse network architecture to gather information about the symmetric key used to secure the communications. Properly building a secure session after each disconnected session is imperative toward the security of the Endpoint communications.

9.2 Safety Critical Analysis

Most Internet of Things products will incorporate some aspect of the physical world with digital technology. As a result, it is likely that a human will make a decision in the physical world based on information provided from an IoT Endpoint. Alternatively, an IoT Endpoint may make a decision that affects the physical world with information obtained through the digital world.

Therefore, it is imperative that IoT Service Providers evaluate their product from a safety perspective to determine if, how, and when human life may be affected by the technology. If adequate safeguards are not put into place to ensure the technology cannot be abused in order to cause physical harm, their customers may be put at risk.

To help resolve the issue of safety, have a discussion with the IoT Service Provider’s executive, legal, and insurance teams. Ensure that these teams understand the capabilities and limitations of the technology used in the product or service. Determine whether these technologies can meet the needs of the business and offer the customers the level of safety necessary for the intended application.

9.2.1 Risk

Clearly, the result of not taking the time to evaluate the impact of the product or service on the safety of the customers could result in the loss of revenue, unexpected accidents, or even loss of life.

9.3 Defeating Shadowed Components and Untrusted Bridges

Components on the physical circuit typically do not use any semblance of confidentiality and integrity when communicating with each other or the central processing unit. As a result, any adversary can read or write data transmitted on these buses. The effect of this gap in communications security is the ability for an adversary to impersonate legitimate devices on the physical circuit. If the adversary chooses, they can impersonate a critical component such as NVRAM, RAM, or even a trust anchor.

The goal of this attack would be to bypass the security employed between two components on the bus. A typical example of this scenario is utilizing this weakness to bypass the integrity validation process of analysing an application image stored in NVRAM. When the CPU retrieves memory stored in NVRAM, the Attacker can use a pass-through system to provide the real memory contents to the CPU. When the application running on the CPU has verified the integrity of the application image, the Attacker may then instrument the communications on the physical bus to selectively swap out NVRAM contents that are beneficial to the Attacker. In other words, the CPU verifies one application image (the original image) but then loads the Attacker's image into RAM and executes it.

One way to safeguard against this attack is:

- Load NVRAM contents into RAM
- Validate the application image loaded into RAM
- Execute the code directly in RAM or cache the contents in RAM

It should also be noted at this point that an Attacker could subvert RAM as well, weakening this process. However, performing a man-in-the-middle attack against RAM is far more complex and costly than an attack against NVRAM because the speed of the bus and access patters are far faster and more erratic than with NVRAM, which is primarily accessed in blocks.

Alternatively, the Attacker can create checksums for smaller regions of validated NVRAM contents and periodically check the signatures from NVRAM. If the checksums differ, then the content is being manipulated. This may succeed, but has a lower success potential because the adversary may only manipulate a small amount of data that isn't randomly checked by the running application.

It should be noted that while the best way to guard against this attack is to validate the contents of NVRAM then load it into executable RAM, there is no full solution for this problem. The cost of securing physical components is so high that it is not practical to resolve this attack in a more full way, unless the customer requires such security assurances.

This attack is even more simplistic when a more basic physical communications protocol, such as I2C, is used. Buses such as I2C are essentially physical broadcast systems. Thus, any component sitting on the I2C bus can pretend to be any other component. This will allow an adversary to impersonate other devices on the bus that don't enforce confidentiality and integrity on the communications channel. Where this is a concern, enforce confidentiality and integrity in the application protocol used on top of the physical bus protocol.

9.3.1 Risk

The risk of not implementing a solution at all will result in the ability for an adversary to bypass integrity checks in the application. This will allow the Attacker to compromise the application being executed by more privileged code, such as bootloaders or TCBs.

It should be noted, however, that this attack is far less likely than simpler attacks against the bootloader. Performing a man-in-the-middle hardware attack against components like NVRAM, or high speed components like RAM, is challenging, complex, and currently expensive. While it will always be possible for an adversary to subvert an embedded system in this fashion, it may be too cost-prohibitive to do so.

Thus, loading code into RAM and verifying the integrity may be a reasonable solution that will bypass the majority of attacks, if any.

Also, for the reasons described above and more, cryptographic keys should not be kept in insecure privileges such as these. They should be stored in a trust anchor and used by the TCB, not stored in media such as NVRAM that could be impersonated or compromised.

9.4 Defeating a Cold Boot Attack

A cold boot attack [REFERENCE] is a physical attack strategy against computer systems that extracts secrets from a running computer by removing the physical memory from the computer, and placing the memory in a secondary system controlled by the adversary. The benefit of this attack is that the Attacker can run a custom operating system that dumps the contents of RAM to permanent storage. This will allow the Attacker to comb through the retrieved data and determine if there are security related tokens that can be used. This may include:

- Cryptographic secrets or private keys
- Login credentials (user names and passwords)
- Personally Identifiable Information (PII)
- Access tokens for web services

The goal of the attack is to compromise secrets that allow the Attacker to gain long-term access to a resource that would otherwise be out of their reach. For example, breaking the cryptographic algorithms used in the most recent standard of TLS would be impossible for the average Attacker. However, compromising the private client certificate used in a mutual-authentication TLS service would allow the Attacker to simulate the client from a more convenient system.

To succeed in this attack, the Attacker must be able to remove the RAM from the target computer system without the bits stored in the chip changing. As detailed in the research paper, this can be accomplished by cooling the memory chips. However, the RAM must be easily removable. If the RAM is soldered to the circuit board, this would vastly complicate the attack and require the Attacker to use a soldering gun to extract memory, potentially damaging the contents.

It is important to note that scrubbing memory at shutdown is always useful, and is advised, to enhance the privacy of an Endpoint. Yet, a cold boot attack can occur at any time, even while the system is running. Therefore, scrubbing memory may be useful, but may not succeed in defeating a real-world attack.

A more effective mitigation for this attack is to process security-centric actions using RAM that is internal to the CPU. Many CPUs, MCUs, and MPUs have a small amount of internal SRAM that can be used by a running application. If the application limits the use of critical security tokens (such as private keys) to this internal RAM, the contents of removable (or external) RAM will have less value to an Attacker.

9.4.1 Risk

Not consider the risk of a cold boot attack may cause critical security keys to be extracted using a simple attack model. If the security keys are universal to all Endpoints in the IoT Service Provider's ecosystem, a large compromise may be possible.

For more information see - https://citp.princeton.edu/research/memory/

9.5 Non-Obvious Security Risks (Seeing Through Walls)

Despite enabling and enforcing mutual authentication, confidentiality, and integrity in the communications network, traffic patterns can directly correlate to events. When data is trafficked in response to certain physical events, a correlation can eventually be made between physical events and data. This may allow an adversary to monitor for signal patterns, then derive meaning from the patterns whether or not the adversary has direct access to the plaintext data.

An example of this is home automation technology that reacts based on a user's physical presence in a particular room. An adversary capable of remotely monitoring the communications system may be able to observe how many users are in a particular home, where the users are in the home, and who the user is, solely by watching patterns of communication between IoT Endpoints, gateways, and back-end systems.

The adversary may be able to easily differentiate between a highly populated home, and homes where only one individual is home alone, and where that individual is within the home. Insurance companies and legal entities will need to understand how this potentially increases risk to homeowners and other tenants in the living space.

Combatting this risk can be difficult. The most common and simplest model for doing so is to send samples at a pre-defined rate, regardless of whether there is a user present to take samples from. If confidentiality and integrity is enforced, disallowing remote adversaries from evaluating the data's plaintext, an observer will be unable to differentiate between a sample containing user activity and an empty sample.

There are concerns with this model, however, such as increased spectrum saturation, increased power consumption for low-power or battery-enabled technology, and the increased processing level required to decrypt, verify, and interpret the empty sample packets.

An alternative is sending samples at random intervals, with varying bursts. This type of pattern is less costly, less power hungry, and requires less processing power. Yet, it still may be possible to observe subtle changes that indicate user presence. For example, any truly entropic system is fully random and unpredictable. User behaviour, however, is entirely predictable. If a user enters a room and the sensors in that room react and begin to send data to peer IoT Endpoints in the network, the *introduction* of consistent behaviour may indicate the presence of a user.

Any team developing technology that is subject to this type of risk should investigate the potential effects of the exposure of privacy, and consult with the legal team to determine whether the technology would have an effect on the business's legal stance or insurance model.

9.5.1 Risk

If the IoT Service Provider does not evaluate their technology from the perspective of potential privacy exposures and security risks, the architecture may need to be substantially overhauled in order to compensate for the risks that *must* be addressed. Instead of trying to make costly adjustments to the architecture at a later time, engineer these solutions into the product at the start of the engineering phase, or, as early as possible.

9.6 Combating Focused Ion Beams and X-Rays

A Focused Ion Beam (FIB) is a manufacturing instrument commonly used in semiconductor evaluation. The technology is capable of inspecting and altering circuits at the nanometre level, which allows analysts to identify faults in manufacturing, and to test circuit patches before altering the fabrication process.

In information security, a FIB can be used to tap internal busses for the purpose of intercepting data trafficked over internal components. In addition, a FIB can be used to alter internal circuitry, which changes how the internal component will operate, allowing an adversary to bypass a security restriction.

Almost all devices are subject to attack by a FIB. Yet, only certain devices will be ran through a FIB process. This is because a FIB itself is an extremely costly technology, at approximately 1,000,000 USD per unit. Because of the high cost of the technology, few organizations have such a device in their toolkit. Furthermore, the device isn't automated. It requires a high degree of skill to manipulate, as well as a very high degree of expertise in semiconductor analysis, to be usable. Thus, the realistic cost of a FIB is far beyond a simple million-dollar figure, and extends into the multi-millions of dollars for the utility itself, and the education, and salary and expertise of the user.

Organizations are available for outsourcing, however. As reverse engineering is largely legal, organizations will provide semiconductor attack services for customers that are interested in reverse engineering a device. These engagements cost anywhere from 10,000 USD to 1,000,000 USD depending on the level of customization and expertise required to attack a particular component. For example, an outsourcing company would have a *playbook* to bypass protections on a common chip. But, a custom FPGA solution with novel security locking technology would cost far more as no existing *playbook* would be defined. A new process would be required to use the FIB successfully, costing substantial time and money.

Some new technologies, such as modern trust anchor variants, claim resistance from FIB probes. While there is some validity to these claims, any hardware protection that isn't dynamic (and most aren't) will result in a *playbook* after a sufficient amount of time has been put into analysing bypass techniques. Therefore, these new claims may be valid, but may only be valid *for a window of time*.

Therefore, to compensate for invasive but almost-always-successful attack technologies such as these, it is imperative for the engineering organization to design a security strategy that doesn't pin its success on the trust anchor alone. Instead, a sufficient protocol must be designed that uses the technology as a base trust anchor, but personalizes each Endpoint's cryptographic keys such that no compromise of a single device results in a compromise of the entire network of Endpoints.

Consider the scenario where an adversary must use a FIB to extract cryptographic from *every Endpoint they wanted to target*. This would quickly become an extremely costly proposal, and would be out of scope with regard to almost any adversary's budget. Since these attack methodologies cannot be *sufficiently mitigated*, they must be *devalued*, to diminish risk through architecture, not through obscurity.

9.6.1 Risk

The risk of a FIB is that cryptographic secrets and other intellectual property can be extracted from a component, even a security-hardened component. Since defeating a FIB in a cost-effective manner for consumer IoT is impractical, the organization must alter their strategy for protecting Endpoint systems or risk a complete compromise of the Endpoint ecosystem.

9.7 Consider Supply Chain Security

The security of any computing system starts with the raw components that the circuit board are composed of. The silicon, cryptographic tokens, read-only-memory (ROM), firmware, and other core attributes of an embedded system all contribute to the security of such a system. If any one of these components are tampered with, the entire system could be subject to a security compromise.

As a result, IoT Service Providers who are conscious about security must take into account the source of their components, their assembly, and the fulfilment process used to ship the assembled technology. If the process used to generate the technology isn't carefully planned, a single point of failure in the process could result in a critical security failure.

Consider the following issues:

- Where and by whom is the silicon manufactured?
- Has the silicon design been analysed by a credible third party information security team?
- Will the silicon be fabricated in a secure facility?
- How will the EEPROM or NVRAM be populated with an executable image, such as a bootloader?
- Is the process for flashing the executable image secure?
- How will the executable image be delivered to the manufacturer?
- Is the executable image verified once it has been flashed onto EEPROM or NVRAM?

- How are cryptographic secrets provisioned on the chip(s)?
- If secrets are generated at the manufacturer, are they using a proven RNG to generate the keys?
- Are all of the security keys unique per the TCB recommendations?
- How are the cryptographic secrets shared with the IoT Service Provider? Securely?
- How are the unique chip identifiers (serial number, etc.) correlated with the cryptographic secrets, and shared with the IoT Service Provider?

While choosing a more security facility to build and assemble a product may incur a greater cost, it may be an imperative step for the organization. This depends on the product use case, the intended deployment environment, the intended customer, and other factors such as human safety, military applications, and critical infrastructure deployments. Where human life can be impacted by the resultant technology, the supply chain should be assessed for gaps in security.

9.7.1 Risk

Without supply chain security the organization is subject to many risks, some of which may be entirely unexpected, and yet critical to the business:

- Endpoint cloning (illegal manufacture)
- Theft of technology (competitors stealing from and undercutting the Service Provider)
- Credential theft (data interception or impersonation attacks)
- Injection of implants (malicious “back doors” that may be activated at a later time)

9.8 Lawful Interception

Lawful interception is the act of legally intercepting or manipulating communications between a customer and a service provider. This can work in one of two ways. First, the most typical scenario is that a law enforcement agency will submit a legal request to a carrier and ask for access to metadata or actual data from communications made by a specific subscriber. Second, the law enforcement agency will ask the IoT Service Provider for access to a specific subscriber’s data and/or meta-data. In the scenario where the agency requests access via the carrier, the IoT Service Provider may never be notified that there is a problem, depending on the scope of the legal request. Thus, the service provider must be ready to either implement, or comply, with a legal request made by such an agency.

Therefore, the provider should identify what privacy issues may result from a law enforcement request, and should be ready to provide information relevant to the organization’s legal model and privacy policy, within their respective legal capabilities.

In the recent past, businesses such as Google, Apple, and other large entities have adopted *warrant canaries* to legally let users know when a secret request has been made to the company on behalf of an agency. The business may remove a phrase, image, or other artefact that is representative of *not* being in contact with lawful intercept agents. The removal of this object is indicative, of course, of a request being made.

9.8.1 Risk

Not readying the business for a lawful intercept request puts the business at a disadvantage if such a requirement is placed on the business. The business may need to comply to the request, but may not have the legal infrastructure or privacy policies ready, potentially putting them at risk.

Not readying the Endpoint protocol and IoT platform for adequate confidentiality and integrity will enable the communications to be intercepted at the network side without the business's knowledge. This can put the business at risk of the user data being leaked, or associated with an event such as the Snowden NSA leaks, substantially decreasing the public's trust in the organization's ability to secure user data.

10 Summary

In summary, almost every security risk in an IoT product or service can be combatted by a well defined architecture, intelligence to identify risks before and during security related events, and policies and procedures to handle such events. By analysing which high-level security concepts are important to the IoT Service Provider, frequently asked security questions can be reviewed. This should guide the engineering team toward which recommendations are most relevant to resolve gaps in their security architecture.

As the team progresses in its architectural definition, it can review standalone recommendations as their security questions and concerns become more unique to their own implementation.

Overall, every engineering team will face very similar risks. It is imperative that the organization choose to share their concerns with their peers to build common a knowledgebase for both risks and remediation strategies. Together, our organizations can build both technology and knowledge to assist each other in building security into the future of IoT.

Annex A Example Using Generic Bootstrap Architecture

The overall security level of a multi hop network is defined by the weakest link in the chain. Thus the local link between IoT Endpoint and an Gateway needs to be secured with a comparable level of security as the wide area network to keep the same overall level of security.

One candidate technology for achieving this is Generic Bootstrap Architecture (GBA) [17] which can be used for both authentication as well as data integrity. This is based on pre-shared keys which are then used to generate time-limited keys (tokens) as a basis of both authentication and encryption.

Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be. In the IoT space, where billions of Endpoints will be active, determining what communication behaviour is genuine and trustful is paramount. The mechanism established to create this trust relationship needs to satisfy the requirement of being scalable and maintainable. Furthermore, the variety of IoT Services imposes the requirement that the authentication mechanism can be adapted to accommodate those different services and still maintain a common infra-structure. A mechanism that has proven itself over time is network authentication based on the SIM. This authentication infrastructure has the virtue of providing not only authentication, but also encryption capabilities based on pre-shared secrets. The explosion in the number of Endpoints and the global reach of IoT makes the use of SIM limited because of network roaming and the security weakness of being able to physically remove a SIM from an un-attended Endpoint. The arrival of technologies like the Embedded SIM provides a practical infrastructure for authentication based on pre-shared secrets, extending the current SIM based network authentication. Also, IoT growth is most likely to happen in the form of capillary networks (the PAN as shown in the example configurations 2, 3 and 4 in the previous section of this document). These capillary networks are swarms of Endpoint devices connected to a Gateway. Most of these Endpoint devices will be lightweight Endpoint devices (i.e. they do not contain a SIM nor Cellular connectivity). These lightweight Endpoint devices will nonetheless require authentication and encryption capabilities. In capillary networks the principal responsibility of authentication lies on the Gateway, reducing the number of complex SIM based Endpoint devices on the overall network. This authentication and security should be extended from the Gateway to the Endpoint device, creating then a secure channel from the given Endpoint device to the IoT Service Platform.

SIM based authentication is meant to serve a single application, i.e. the authentication of a unique Endpoint device for network attachment. Endpoint devices will have a multitude of services, each with different and exclusive need for authentication. A framework that extends the network authentication to multiple services is required. One framework that was designed for this purpose is GBA, Generic Bootstrapping Architecture. GBA leverages on the SIM based infrastructure to generate time-base share keys between devices and Network Application Functions (NAFs). GBA is an authentication method standardized by the 3GPP in the 3GPP specification TS 33.220 [17]. The method enables the authentication of a device with a 3GPP subscription to a service. The credentials of the subscription are in the device, usually stored on a SIM, such as a Universal Integrated Circuit Card (UICC), or as remotely managed credentials, stored and managed on an embedded SIM (eUICC) for example, such as the GSMA specified Embedded SIM (eUICC) [5].

The advantages of this framework are:

- Mutual authentication based on either PSK uniquely between a device and a Network Application Function or shared key-based UE authentication with certificate-based NAF authentication (TS 33.222) [18].
- Credentials can be secured in a Trusted Environment
- If eUICC is used, the credentials can be changed OTA.
- Scalability. The complexity and economic cost of maintenance increases linearly with the number of devices, since authentication is “built inside” the framework.
- Data integrity. The time-base generated keys during authentication can be used for establishing TLS-PSK tunnels, making this connection will provide very strong data integrity and confidentiality.

Annex B Tutorial on use of UICC cards in an IoT Service

The UICC as standardized in ETSI TS 102 221 is a smart card platform (a programmable tamper-resistant secure element) providing an interoperable secure file system interface and secure application framework to UICC hosting devices. ETSI TS 102 221 provides a framework for a UICC hosting device to discover relevant applications on a UICC, and each UICC application corresponds to a known set of provisioning and configuration information as well as operational procedures (such as authentication or key derivation) that can be supported by the hosting device according to its needs.

In IoT context, UICC can be available in multiple Form Factors and environmental operating ranges as specified in ETSI TS 102 671. In its simplest embodiment, the UICC is typically owned by a network operator and only hosts one Network Access Application (SIM application as per 3GPP TS 51.011, USIM as per 3GPP TS 31.102, CDMA CSIM as specified by 3GPP2, WiMAX SIM, etc.). In this case, the UICC provides a standardized holder to host security provisioning and configuration information as well as cryptographic procedures on a mobile device to enable network access, with additional mechanisms to remotely manage the content of the UICC, using ETSI TS 102 225 / TS 102 226. The mobile network ecosystem has procedures in place to ensure secure personalization and deployment of UICCs under control of the network operator, resulting in the establishment of individual shared symmetric keys between UICC hosting devices and the infrastructure.

One important feature of UICC platform is the support of isolated Security Domains that enables multiple stakeholders in a complex ecosystem to each be assigned their own area on a UICC and manage its content in confidentiality from other stakeholders. This functionality is inherited through ETSI TS 102 226 from GlobalPlatform Card Specification [15] Amendment A. Therefore, in an IoT context, a single UICC enables multiple stakeholders to store and administer their own credentials independently from one another.

In general, a UICC can hold several Network Access applications (with only one being active at any given time), and potentially other applications securing access to more elaborated services, such as ISIM applications for IMS access (as specified in 3GPP TS 31.103) or, in the case of IoT Services, 1M2M SM applications specified in Annex D of oneM2M TS-0003. A 1M2MSM application can support direct provisioning of dedicated IoT Service/ application credentials, as well as derivation from pre-existing network access credentials on the UICC using the GBA mechanism specified by 3GPP. It further enables an IoT Service Provider to customize the cryptographic procedures according to its specific needs, e.g. to support specific service authentication mechanisms.

A single UICC can also hold multiple 1M2MSM applications, enabling the confidential deployment of symmetric keys dedicated to each IoT Service Provider. A UICC owner (typically a network operator or OEM manufacturer in IoT context) may share space on its UICC with IoT Service Providers that request it, so that the accredited UICC personalization chain and infrastructure that enables secure deployment of network access credentials can also be leveraged by IoT Service Providers to deploy their own credentials.

Where IoT application security rely on asymmetric cryptography, UICC applications such as IoT SAFE can be used to facilitate the deployment of public/private key pairs, as needed for a specific IoT Service. Such UICC applications need to be specified and supported on hosting devices on an IoT application specific basis.

Annex C Document Management

C.1 Document History

Version	Date	Brief Description of Change	Approval Authority	Editor / Company
1.0	08-Feb-2016	New PRD CLP.13	PSMC	Ian Smith GSMA & Don A. Bailey Lab Mouse Security
1.1	07-Nov-2016	References to GSMA IoT Security Assessment scheme added. Minor editorial clarifications.	PSMC	Ian Smith GSMA
2.0	29-Sep-2017	References added to GSMA LPWA network resources plus additional minor updates.	IoT Security Group	Rob Childs GSMA
2.1	30-Mar-2019	References updated	IoT Security Group	Ian Smith GSMA
2.2	29-Feb-2020	References to IoT GSMA IoT.04 added	IoT Security Group	Ian Smith GSMA

C.2 Other Information

Type	Description
Document Owner	GSMA IoT Programme
Contact	Ian Smith – GSMA

It is our intention to provide a quality product for your use. If you find any errors or omissions, please contact us with your comments. You may notify us at prd@gsma.com

Your comments or suggestions & questions are always welcome.