ICT Risk Assessment

Fabrizio Baiardi
f.baiardi@unipi.it

Syllabus

Definition of attack and countermeasures for the internet of things

Internet of things

- *Internet of Things =* Extending the current Internet and providing connection, communication, and inter-networking between devices and physical objects, or "Things"

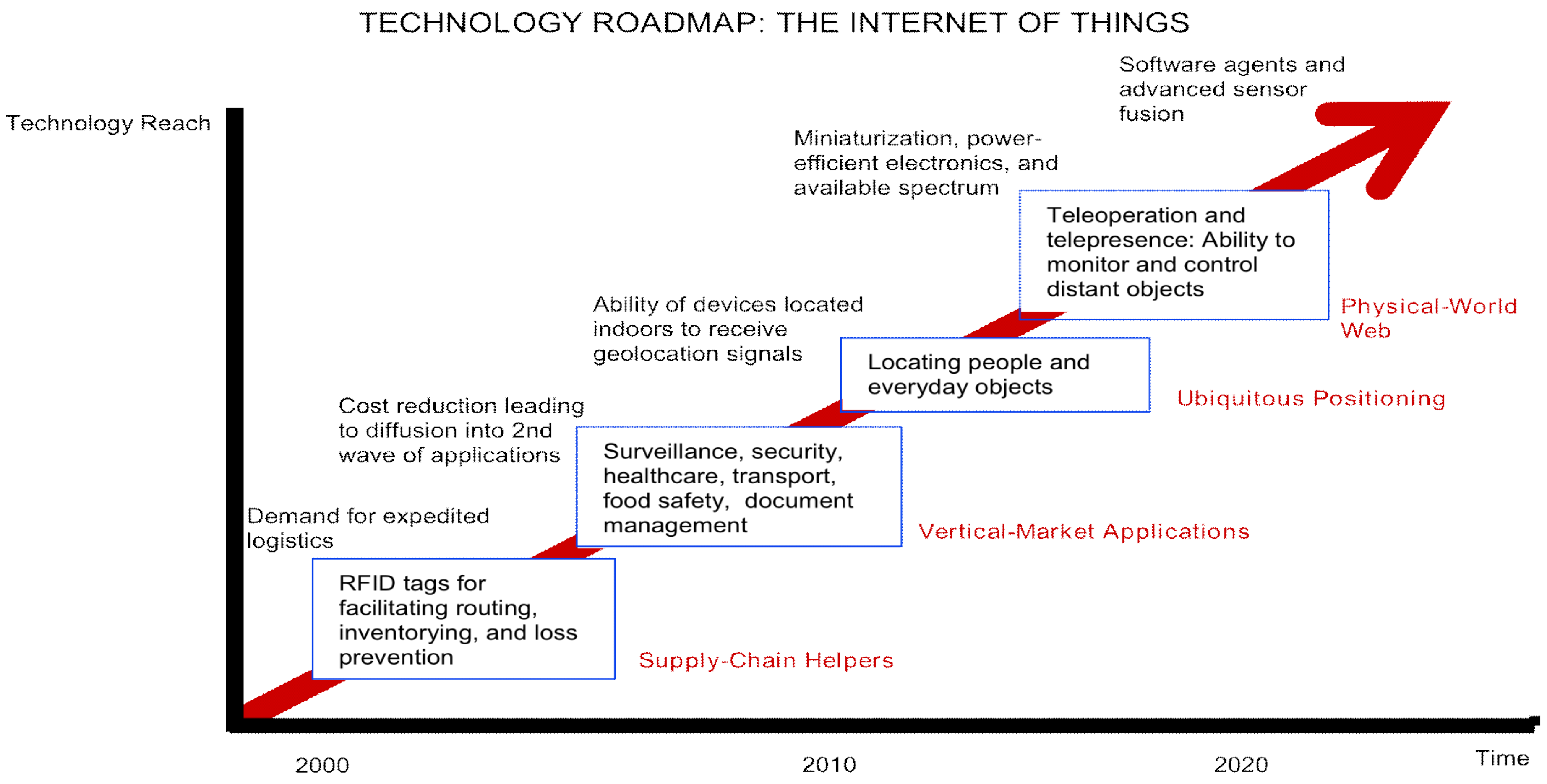
- The technologies and solutions that enable integration of real world data and services into the current information networking technologies are often described under the umbrella term of the Internet of Things (IoT)

- Oxford defines the Internet of Things as: “A proposed development of the Internet in which everyday objects have network connectivity, allowing them to send and receive data.”

Internet of things

- The modern interpretation of IOT is the connection of devices, systems and services that goes beyond the traditional machine to machine (M2M) and covers a variety of protocols, domains and applications.
- The applications range from medical sensors, medication tracking, physical activity level tracking, home heating, lighting and appliance control, intelligent lighting control, the ubiquitous intelligent fridge.
- The use of big data in the form of environmental real time data , traffic information, weather alerts. Businesses will make use of stock and asset tracking, data from remote or mobile equipment.
- 26 billion devices installed by 2020 , and that figure will continue increasing by some 8 billion devices per year, this market has been valued at some £600 billion in 2020.

IOT Diffusion


TECHNOLOGY ROADMAP: THE INTERNET OF THINGS
Technology Reach
Software agents and advanced sensor fusion
Miniaturization, power-efficient electronics, and available spectrum
Teleoperation and telepresence: Ability to monitor and control distant objects
Physical-World Web
Ability of devices located indoors to receive geolocation signals
Locating people and everyday objects
Ubiquitous Positioning
Cost reduction leading to diffusion into 2nd wave of applications
Surveillance, security, healthcare, transport, food safety, document management
Vertical-Market Applications
Demand for expedited logistics
RFID tags for facilitating routing, inventorying, and loss prevention
Supply-Chain Helpers
2000
2010
2020
Time

Source: SRI Consulting Business Intelligence

What may happen …

To explore the issues let's assume you are equipping your house with lots of nice IoT goodies all of which are connected to the internet. Your lounge will have a smart central heating thermostat , smoke detectors in your hall way, smart lighting , door locks on your front door and garage and an IoT fridge in the kitchen should just about cover our needs.

All these devices will be monitoring your home or life , recording temperatures and setting the heating for when you come home and shutting it down when you go to work. They will know the status of your doors , when they were last accessed and when you come home.

In fact they will know more about you than your best friend probably.

The merging of IoT and 5G results in dramatic security problem

What may happen …

If you plug an IoT device into your home network and configure it correctly it will start collecting data and doing its job sending data 'home' to the companies servers.

- Which data is collected?
- Who is responsible for that data ?
- Whilst within your home system it is most certainly you, you will ensure your firewall is correctly configured and that you use the highest level of protection possible.
- However once the data leaves your system you are at the mercy of others who may not be so careful.
- Vulnerabilities will exist when data is enroute. Your data may be stored in a data collation hub waiting to be uploaded to its final destination and all that time it is vulnerable to interception.

What has happened …

In 2014 a fridge has been discovered sending out spam after a web attack managed to compromise smart gadgets. It was one of more than 100,000 devices used to take part in the spam campaign.

The attack is believed to be one of the first to exploit the lax security on devices that are part of the "internet of things".

Between 23 December 2013 and 6 January about 750,000 messages were sent as part of the junk mail campaign. The emails were routed through the compromised gadgets.

About 25% of the messages did not pass through laptops, desktops or smartphones, The malware managed to get itself installed on other smart devices such as kitchen appliances, the home media systems and web-connected televisions.

Many of these gadgets have computer processors onboard and act as a self-contained web server to handle communication and other sophisticated functions.

What may happen …

This all seems pretty innocuous but this data isn’t confined to your home network , its sent to the relevant company servers and stored, processed and data mined to provide you with the ‘Smart’ features you want , like the learning thermostat or the smoke detector that emails you when the battery needs changing.

Each one of these pieces of information, in itself , is of very little importance however its the synergy of all the data from all the different sources and built up over time that builds a bigger more complete picture.

By noting that your smart lights are off, your thermostat has turned down your central heating and that all your doors and windows are locked its pretty obvious you aren’t at home and by looking at the history of your thermostat setting it would be possible to work out when you are likely to return.

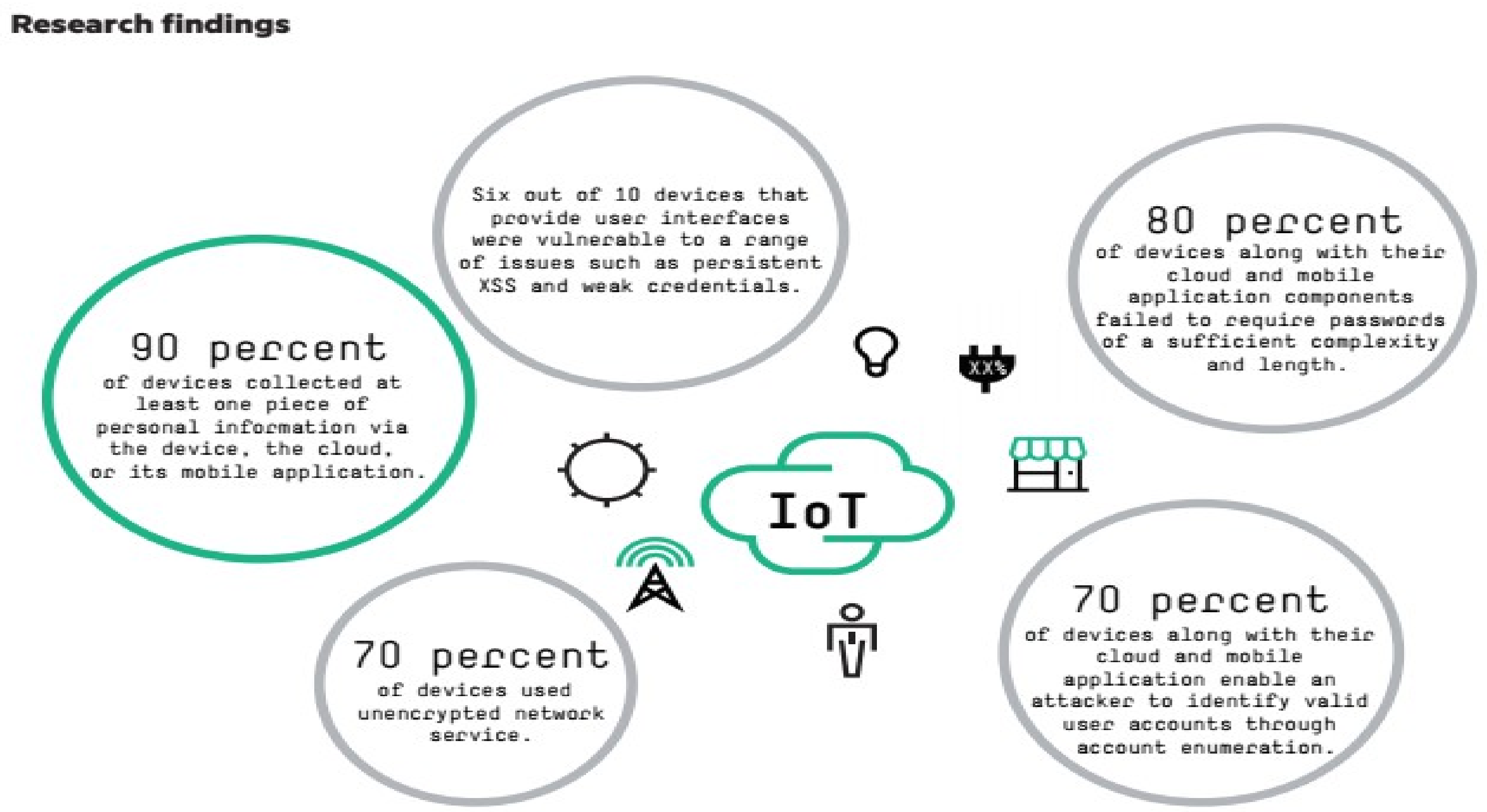
This is akin to hoisting a big neon sign on your front lawn to advertise when you are out.

What has happened …

The Mirai botnet, composed primarily of embedded and IoT devices, took the Internet by storm in late 2016 when it overwhelmed high-profile targets with massive DDoS attacks

- Mirai bots scan the IPv4 address space for devices that run telnet or SSH, and attempt to log in using a hardcoded dictionary of IoT credentials
- Once successful, the bot sends the victim IP address and associated credentials to a report server, which asynchronously triggers a loader to infect the device
- Infected hosts scan for additional victims and accept DDoS commands from a command and control (C2) server
- The released Mirai source release included 46 unique passwords, some of which were traceable to a device vendor and device type. Mirai primarily targeted IP cameras, DVRs, and consumer routers
- Mirai launched 15,194 attacks between September 27, 2016–February 28, 2017. These include Application-layer attacks, Volumetric attacks, and TCP State exhaustion, all of which are equally prevalent
- Only 2.8% of Mirai attack commands relied on bandwidth amplification, despite built-in support in Mirai’s source code

What has happened …


Research findings

Six out of 10 devices that provide user interfaces were vulnerable to a range of issues such as persistent XSS and weak credentials.

80 percent of devices along with their cloud and mobile application components failed to require passwords of a sufficient complexity and length.

90 percent of devices collected at least one piece of personal information via the device, the cloud, or its mobile application.

70 percent of devices used unencrypted network service.

70 percent of devices along with their cloud and mobile application enable an attacker to identify valid user accounts through account enumeration.

Output of an HP research that investigated the security of 10 of the most popular IOT devices

Two prospectives on IOT Security

- Any smart device increases the attack surface of your system and its connection can result in new attacks (attacks to the device, device as an intermediate step)
- Any smart device can store some malware to attack your system (attacks from the device)
- While most of the research has focused on the first issue, the second one is becoming more and more critical
- Internet 4.0 includes a huge number of devices with code that cannot be easily accessed and tested
- Fuzzing and reverse engineering will become more and more important to preserve confidentiality and integrity

Anatomy of an IOT attack

- What does an IoT device look like under the hood?
- What does an IoT malware attack look like?
- What do you do to protect your IoT devices from attack?

Anatomy of an IoT malware attack How to prevent your IoT devices from joining the zombie bot horde, J Steven Perry October 31, 2017,

IBM DeveloperWOrk

IOT Device Hardware

- Data acquisition and control
- Data processing and storage

Essentially, IoT devices contain *sensor s*, *act uat or s*, or both. Sensors acquire data, and actuators control the data or act on the data.

- **Sensors** *monitor* Things and provide data about the Thing, whether it be the temperature, light intensity, or battery level.
- **Actuators** *control* the Thing through hardware in the device, like the controls in a smart thermostat, the dimmer switch in a smart light bulb, or the gear motors in a robotic vacuum cleaner. They represent the physical interface to the Thing that make it "go,"

All IoT devices have a way to process sensor data, store that data locally (if necessary), and provide the computing power that makes the device operate.

If data from multiple sensors needs to be coordinated, or if data needs to be stored in flash memory (for whatever reason), it is the data processing component of the IoT device that does it

IOT Firmware

The onboard software that that runs an IoT device sits between the hardware and the outside world,

Embedded firmware

- IoT devices are resource-constrained, so they often use custom-built, embedded firmware, which is another term for the software that runs on the device. In many cases, the only cost-effective solution for device manufacturers is to write embedded firmware to interact with the hardware.
- Embedded software engineers write the embedded firmware, the software to interact with the hardware, along with the application software to interface with the device's user, such as the interface to configure the device

OS-based firmware

- As IoT devices have grown "smarter" (more complex) the demand for more complex software to manage and exploit the new capabilities has also grown.
- An IoT device now probably runs an operating system (OS) that provides an abstraction layer between the hardware and other software on the device.
- Embedded software engineers (who understand the hardware) can now spend their time writing device drivers, and application programmers (who do not need to understand the hardware intimately) spend their time writing the software that makes the device "smart".
- A popular OS choice for many device manufacturers is Busybox, a stripped down version of the Unix operating system that contains many of the most common utilities, has a very small footprint, and provides many capabilities of Unix in a single executable

IOT Wireless Communication

- IOT devices most often communicate wirelessly, which means they can be anywhere in your home or enterprise. The communication needs of the device change depending on how it is designed to work.

- Some devices are designed to work by making a direct 802.11 Wifi connection to your router. From there, the device can access the internet. A motion-activated security camera is a popular example of this type of device, which uses Wifi because it potentially needs a fair amount of bandwidth.

- Some devices are meant to work as part of a group of IoT devices. For example, a window open/ closed sensor that is connected to a smart home gateway device (sometimes called a hub) uses a wireless protocol like Z-Wave or Zigbee (or any of a half-dozen others) so it can report that the window has been opened.

IOT Device Management

- Provision the device
  - Many IoT devices do not have built-in user interaction hardware and are called . Headless devices may be configured through the Wifi Protected Setup (WPS) through a WPS-enabled device and a WPS-enabled router. You press the WPS buttons on IoT device and on the router, and the two are connected.
  - Some devices create a Wifi access point to connect using your smart phone to access a setup program where you to enter your Wifi network credentials.
  - Other devices scan and add devices they detects are in setup or pairing mode.
- Monitor and control the device
  - When a device is connected you can monitor and control it, through a smartphone, either connected to the gateway directly or through an interface to a cloud service.
  - Some devices like CCTV security cameras connect directly to the internet and have dedicated IP addresses. They devices can be accessed directly over the internet, bypassing the need for a cloud service provider or gateway.
  - Many IoT devices are exposed directly to the internet by enabling port-forwarding. This allows the device to be accessed from anywhere on the internet to monitor and control it.

IOT Attack

- IOT Security is most often an afterthought because it is difficult to create a reliable, resource-constrained device that can connect to a wireless network, use very little power, and is most importantly cheap. Because there is so much to do to just produce a working device, security is the last thing to be considered
- Attack vectors
  - **Weak Password** In order to make the device easy to setup and use, the manufacturer provides some login e.g. a single userid/password combination.
  - **Lack of encryption** Many IoT devices do not support encryption, hence you need to investigate the devices
  - **Backdoor.** Manufacturers put "hidden" access mechanisms to make the device support easier. Once a backdoor becomes known, the manufacturer rather remove it, just made it more difficult to access (or so they think)
  - **Internet Exposure** (they accept internet traffic) unlike a hardened server where you can control the firewall and how the host is accessed, most IoT devices have little or no security and are particularly susceptible to attack.

How to protect your device

- Always change the default password
- Remove devices with telnet backdoors
    - To discover these devices you can use IoT device scanners that check with an IoT search engine Shodan to reveal if your devices are vulnerable based on the IP address of the computer where you originate the scan
- Never expose a device directly to the internet
    - When you are faced with the question of whether or not to expose a device to the internet by opening up your firewall, the right answer is almost always no
    - IoT device scanner can run a "deep scan" to check for any open ports on your publicly exposed IP address assigned by your ISP
- Port Scan all your machines

A more refined framework

- An alternative classification
- A more detailed classification of attack
- Hardware/firmware countermeasures resulting in ARM SecureZone and in TEE environments

A first device classification - I

IoT Nodes may be classified by

- Performance according to the node components
- Functionally according to intelligence distribution

Performance based classification:

1. Ultra-constrained node. An RTOS or bare metal with 16K of RAM. Energy harvesting limits radio transmissions to conserve power.
2. Constrained node. Using an RTOS with 32K-64K RAM. Most likely running on a battery and all the software is optimised for battery life. Again limiting radio transmissions to a minimum.
3. Mainstream node. A feature rich RTOS with 128K RAM. The interaction with the context get more complex since there is room for more local operation, as opposed to sending all data upstream.
4. Gateway node. An advanced OS with 64MB RAM. A sophisticated node that can run advanced software and is running from main power. It has multiple radios to support the local network.

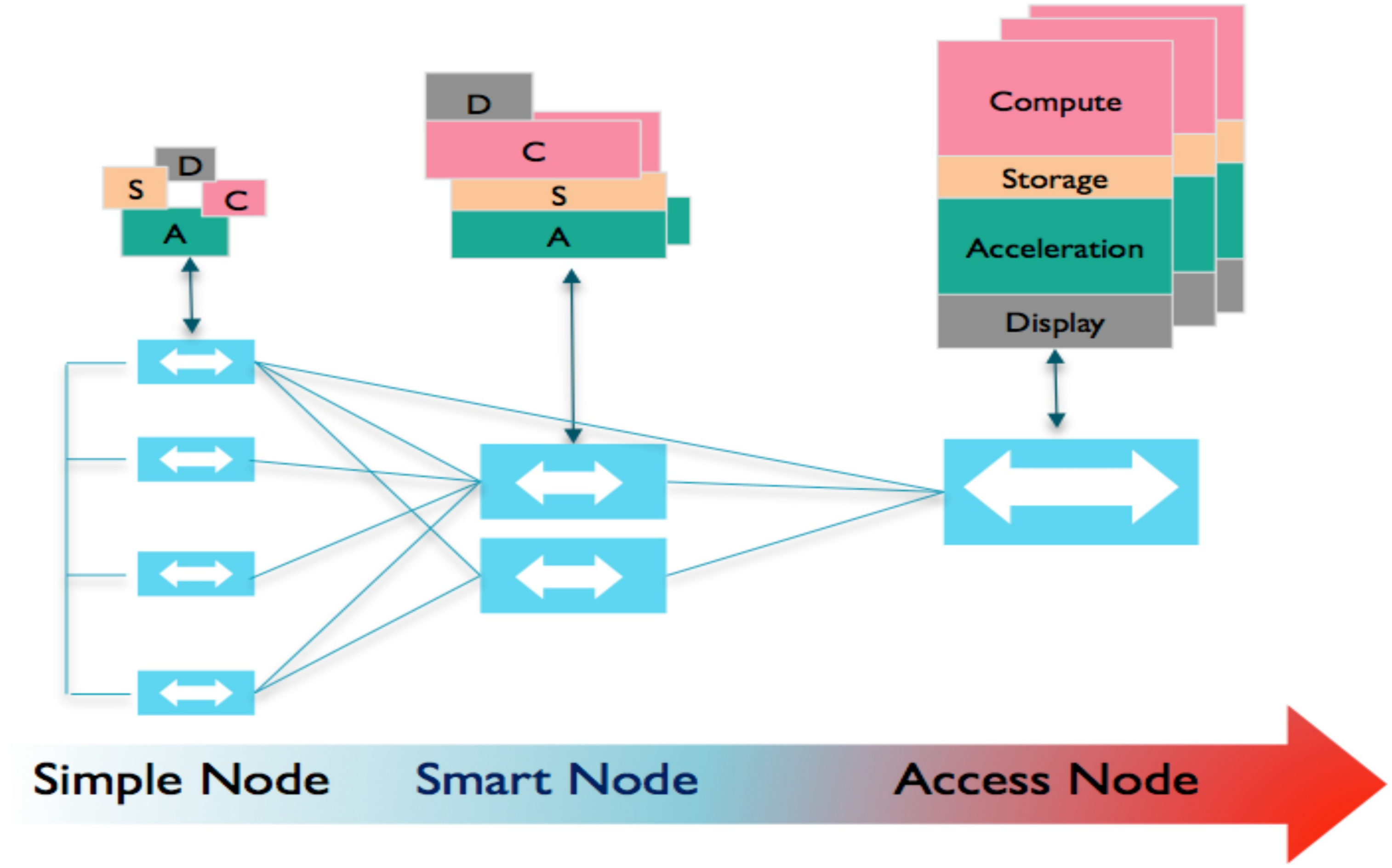
A first device classification - I

Functional based classification:

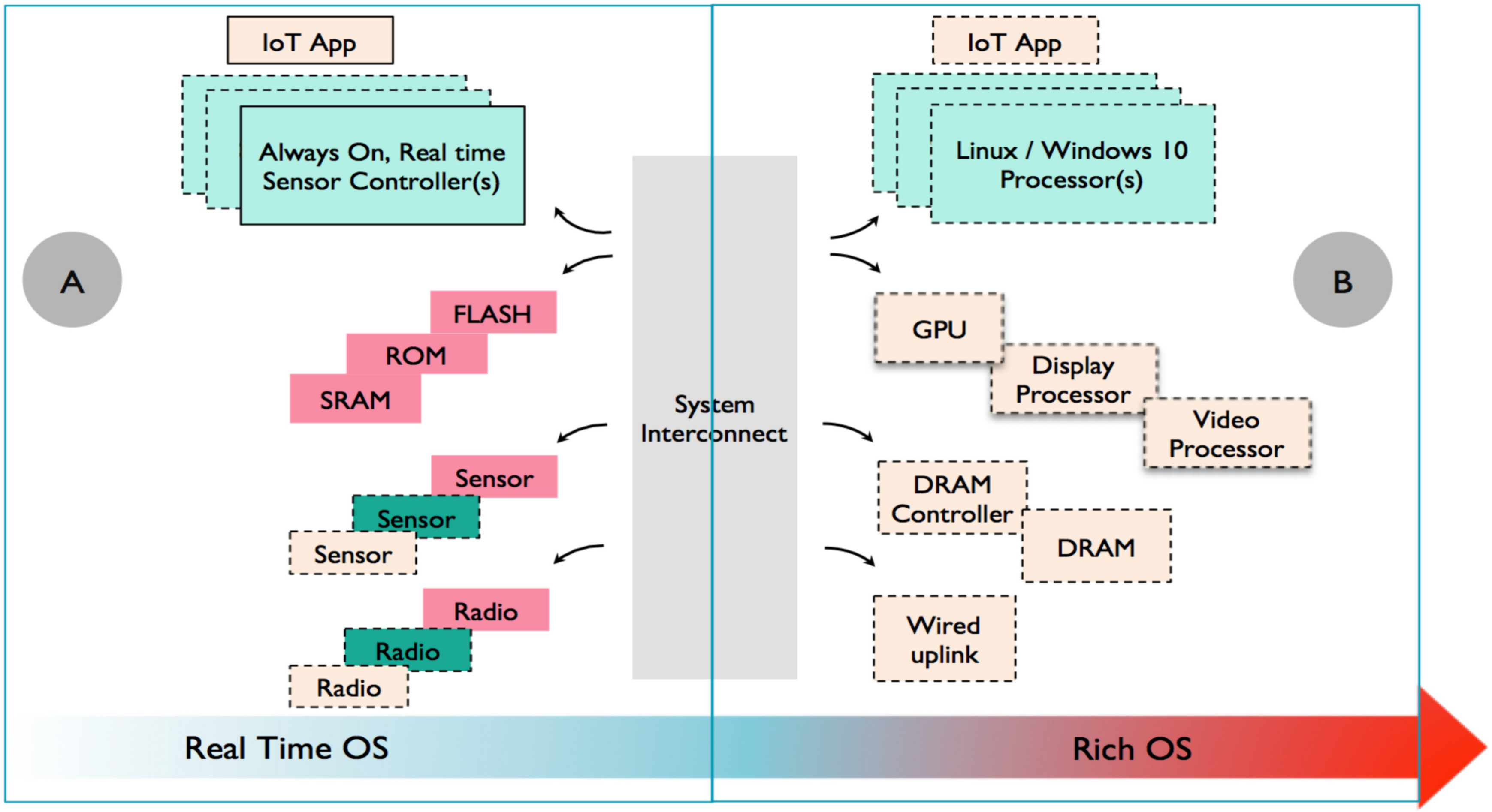
1. Simple node : it is not aware of the rest of the local network. It collects and reports information to the specified destination. Any of 1-3 may a simple node.
2. Smart node : it is fully aware of all other network nodes mainly via software that understands mesh networks, local topologies and authorised interactions between nodes in the same network.
3. Access node. This is the box at the edge to connect the local network to the Internet via whatever broadband link appropriate for the application. It has multiple radios facing the local network.

No restrictions on creating a node that is both a smart node and a gateway.

A first device classification - II



Architecture of an IOT node



ASSET GROUP	ASSETS	DESCRIPTION
IIoT End Devices Level 1	Sensors	These devices detect and/or measure events in their environment and transmit information to other electronic systems to be processed. There are sensors for many purposes, such as to measure temperature, motion, vibration etc.
	Safety Instrumented Systems (SIS)	These systems consist of sensors, logic solvers, and final control elements (actuators) whose objective is to bring the process to a safe state in case of a violation of predetermined conditions.
	Actuators	These devices interact with the environment by moving or controlling a mechanism or system. In order to do so, they convert energy (e.g. electrical, hydraulic or pneumatic) into motion.
ICS Level 2	PLCs	These specialised industrial computers are used to automate control functions within the industrial network. Typically, they are equipped with additional plug-in modules, such as Input / Output modules to connect sensors and actuators.
	RTUs	These devices are used typically in substations or remote locations. Their objective, similar to PLCs, is to monitor field parameters and send data to the central station.
	DCS	These control systems distribute intelligence, i.e. management logic, about the controlled process instead of relying on a single central unit.
	SCADA	These systems are used to collect data from industrial assets and processes, their visualisation, supervision and control. Such workstations usually operate on the Windows operating system.
	Human Machine Interfaces	These control panels and dashboards allow the operators to monitor and control PLCs, RTUs and other electronic devices.
ICS communication networks & components Levels 1 - 3	Routers	These networking devices forward data packets between different networks in industrial environments and IoT ecosystems.
	IIoT Gateways	These network nodes are used to interface with another network from an IoT environment using different protocols. Gateways may provide protocol translators, fault isolators, etc., to provide system interoperability.
	Switches	These network components filter and forward packets within the local area network.
	Wireless Access Points	These components enable wireless devices to connect to a wired network using Wi-Fi, or related standards.
	Firewall	These network security devices or systems control network traffic between networks or between a host and a network based on predetermined rules.
	Networks	They allow the different nodes of an IoT ecosystem to exchange data and information with one another, via a data link. There are different kinds of networks related to their spatial coverage, including e.g. (W)LANs, (W)PANs, PANs and (W)WANs, among others.

Node Architecture

- For software productivity,requires 32 bitreal time embedded processor due to the large need to quickly develop and release a large amount of software to handle connectivity, security and IoT applications while properly managing and controlling one or multiple sensors
- Software productivity is important since one node design may be used to release products fo multiple different markets.
- Multiple processors maybe required, one to handle radio stacks and connectivity, one to manage the sensor/actuator and a third to run the whole system and network membership.
- It is very easy to use multiple processors of the same or different model sinc the same bus connects to the system fabric.
- Multiple processors in many cases reduce power consumption since only the right amount of processing power is activeat any moment to handle the task of the moment. There is no need to wake up the main processor if no trasnmission is required but the radio processor may check for incoming message

Memory Subsystem

- Most real time embedded designs use
  - Flash memory to store the program,
  - SRAM to store code and data
  - ROM to hold the basic system description.
- The size of memory blocks depend on the system configuration and the intended operation and software complexity
- Sometime one design exploits many segments since one memory size fits many applications and some extra space simplifies future expansion.
- Only when an application requires a substantial increase in memory, one creates distinct products since there is no need to burden the smaller one with extra memory that will definitely remain unused.

JTAG

- JTAG is a common hardware interface that provides your computer with a way to communicate directly with the chips on a board.
- It was originally developed by a consortium, the Joint (European) Test Access Group, in the mid-80s to address the increasing difficulty of testing printed circuit boards (PCBs).
- JTAG has been in widespread use ever since it was included in the Intel 80486 processor in 1990 and codified as IEEE 1491 that same year. Today JTAG is used for debugging, programming and testing on virtually ALL embedded devices.
- The JTAG interface gives manufacturers a way to test physical connections between pins on a chip. When using JTAG to debug a chip, we are making sure pin A on chip A is physically connected to pin B on chip B, and that all those pins are functioning correctly. Since JTAG gives you direct hardware access to a device, it is a tool for security research.

JTAG Fundamentals - I

The JTAG standard requires 4 standard pins and defines an optional 5th. The signals, and the small bit of silicon logic that connects and controls them, are referred to as the Test Access Port, or TAP controller.

- TCK, Test Clock: dictates the speed of the TAP controller. Voltage on this pin pulses up and down in a steady beat. On every “beat” of the clock, the TAP controller takes a single action. JTAG does not specify the actual speed. The TAP controller accepts it from the outside device controlling JTAG.
- TMS, Test Mode Select: Voltages on this pin control what action JTAG takes. By manipulating this voltage, you tell JTAG what you want it to do.
- TDI, Test Data-In It feeds data into the chip.
- TDO, Test Data-Out: data coming out of the chip. The JTAG standard does not define protocols for communication over these two pins
- TRST: Test Reset (Optional) This optional signal is used to reset JTAG to a known good state.

JTAG Fundamentals - II

- JTAG IEEE 1149.1 specifies not only the signals but how the "State Machine" in the TAP Controller of a device must behave.
- By manipulating the voltage on the TMS pin of the TAP controller and the TCK pin, you control the way the State Machine changes state. On each beat of the clock, the JTAG TAP controller in a device checks if there is voltage on the TMS pin, and interprets that as either a 1 (voltage) or a 0 (no voltage). This series of 1's and 0's moves JTAG through a relatively simple State Machine.
- The JTAG interfaces of multiple chips on the same board can be daisy-chained, so you don't need a separate set of JTAG pins for every chip on the board. They can all be tested through a single set.
- This is particularly interesting with System-On-Chip (SoCs) where multiple components (ethernet, wifi, bluetooth, etc) are in a single chip. The JTAG chain often passes between these different cores internally in a single chip.

Debug

- A debug circuitry build in a processor can be used to debug embedded software running on the processor.
- In some processors the debug circuitry is integrated with the JTAG circuitry.
- The JTAG circuitry in the processor has extra registers to stop execution, read the status of the processor core or to read and write data in external memory chips.
- These registers are build in the core of the processor. The way these registers are build in the core is not defined in the IEEE 1149.1 standard and is implemented by the chip designers.
- Therefore the debug mode can be different for each chip or may not be implemented at all.
- In debug mode the JTAG test access point is only used as a communication channel between the target board and a PC running the debugger software.

Using JTAG

- JTAG is an interface hence it does not know how a chip will react to command strings. Hacking a device using JTAG alone would technically be possible, but it requires a deep, holistic understanding of all the internal workings and architecture of the chip.
- We need a tool to translate between human-readable code and the low-level instructions coming in and out of the JTAG TAP controller.
- OpenOCD is an Open Source repository (knowledge base) for a variety of chips and interfaces. OpenOCD can (when paired with a JTAG interface device) manipulate the JTAG's TAP controller on a target device to send bits to it (via the state machine above), which the chip will then interpret as valid commands.

Using JTAG and OpenOCD

- Running OpenOCD over the JTAG interface will allow you to pause and step through an operation, inspect memory, write bytes directly into memory, set watch-points and break-points, and even inject code into the process or process memory.
- This technique is usually referred to as "Hardware-based Software Debugging." and it manipulating the hardware to perform traditional software debugging tasks.
- OpenOCD supports familiar software debugging tools by spawning a GDB server, allowing you to debug firmware and software in devices via GDB, IDA Pro, or anything that is capable of acting as a GDB client
- OCD uses special config files to know how to communicate with the device
- The config file teaches OpenOCD the correct way to manipulate the TAP controller for that particular chip's architecture. Config files for many of the most common chip types are included with OpenOCD and most others are easily downloadable online.

JTAG in a few words …

If an attacker has unrestricted physical access to your device, it is no longer your device. JTAG is a technology that makes this happens statement the truism that it is

An attacker with JTAG access can:

- Read and Write from memory
- Pause execution of firmware (set breakpoints and watchpoints)
- Patch instructions or data into memory
- Inject instructions directly into the pipeline of the target chip (without modifying memory)
- Extract Firmware (for reverse engineering/vulnerability research)
- Bypass protection mechanisms (encryption checks, password checks, checksums, you name it)
- Find hidden JTAG functionality that might do far more than we imagine.

JTAG countermeasures

Manufacturers

- are aware of JTAG risk, and will often take steps to prevent access to JTAG interfaces.
- may try to obfuscate JTAG traces on the board, or even cut them entirely. They may deliberately blow fuses in the JTAG wiring as part of the manufacturing process.

These methods are somewhat effective, but a determined attacker who is talented with a soldering iron can almost always repair the damage.

Several different standards have been suggested for adding encryption and cryptographic authentication to the JTAG standard, but they have not been codified, and are rarely implemented in practice.

Attacks against a IOT node – I

Hack attack

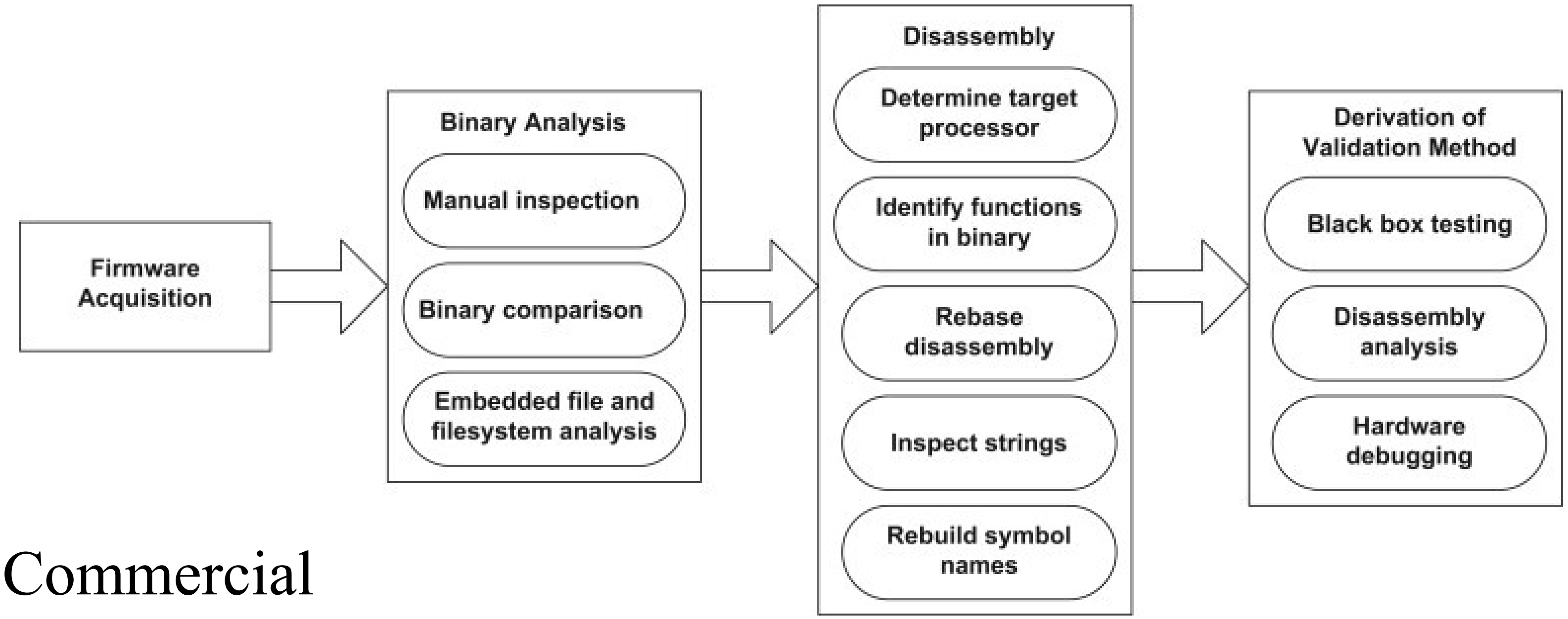
- A software attack. Examples of hack attacks include viruses and malware which are downloaded to the device via a physical or a wireless connection.

Shack attack

- A shack attack is a low-budget hardware attack, using equipment that could be bought on the high street from a store. This implies the attackers has physical access to the device, but not enough equipment or expertise to attack within the integrated circuit packages.
- The attackers can attempt to connect to the device using JTAG debug and built-in self test facilities. It can passively monitor the system using logic probes and network analyzers to snoop bus lines, pins and system signals. It can perform simple active hardware attacks, such as forcing pins and bus lines to be at a high or low voltage, reprogramming memory devices, and replacing hardware components with malicious alternatives.

Shack Attack

Discover and reverse engineering



Commercial
Thesis available using the Ghidra toolset
End of commercial

Attacks against a IOT node – II

Lab attack

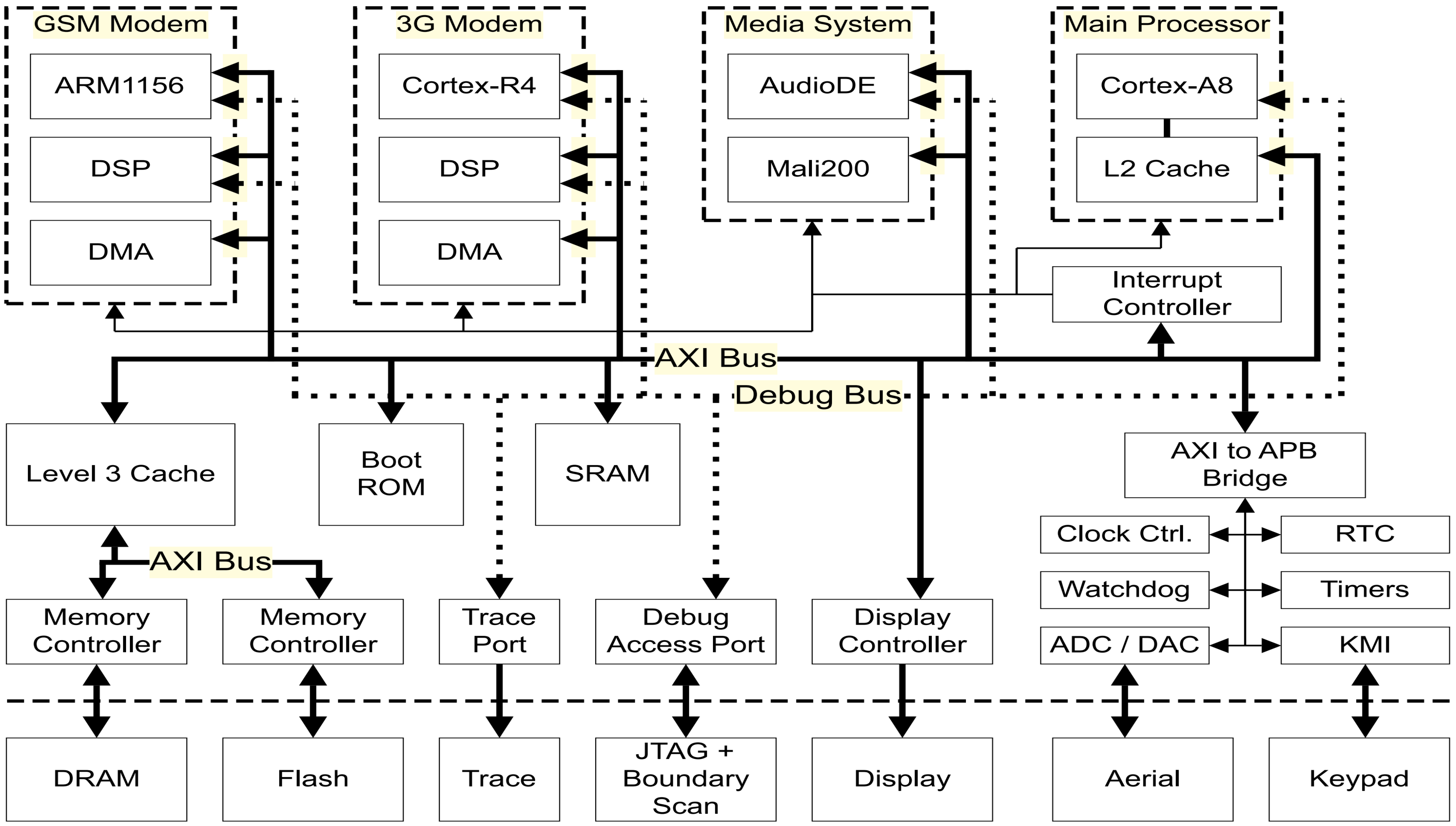
- It must be assumed attackers with access to laboratory equipment, such as electron microscopes, can perform unlimited reverse engineering at transistor-level detail for any sensitive part of the design - including logic and memories.
- Attackers can
  - reverse engineer a design, attach microscopic logic probes to silicon metal layers, and glitch a running circuit using lasers or other techniques
  - monitor analog signals, such as device power usage and electromagnetic emissions to perform attacks such as cryptographic key analysis.

In most cases, a device should not try and defend against lab attack directly, but should take measures which limit the damage when a device is broken and therefore make the lab attack uneconomical. Use of per-device unique secrets is one example where reverse engineering a single device provides the attacker with no useful information; they have no information for the device class

Attacks against a IOT node – Class Attack

- The most sought after attacks is a *class-break*; an easily reproducible attack that can be used to break a whole generation, or class, of devices.
- The most widely published attacks that fall in to this category are those deployed against consumer entertainment devices, such as the attacks that break the software restrictions on games consoles and the content protection schemes on DVD movies.
- In many scenarios of hardware attacks, the first research by the attackers may cost a significant amount to fund access to tools used for silicon-level analysis. The goal is to discover weaknesses to exploit on multiple devices without significant cost.
- Class-breaking attacks shift the balance of the economic argument in the favour of the attacker if the number of attackable devices is sufficiently high.

System on a chip



TrustZone: ARM Solution For SOC

- The TrustZone hardware architecture aims to provide a security framework to enables a device to counter many of the specific threats that it will experience.
- Instead of providing a fixed one-size-fits-all security solution, TrustZone technology provides the infrastructure foundations that allow a SoC designer to choose from a range of components that can fulfil specific functions within the security environment.
- The primary security objective of the architecture is actually rather simple; to enable the construction of a programmable environment that allows the confidentiality and integrity of almost any asset to be protected from specific attacks.
- A platform with these characteristics can be used to build a wide ranging set of security solutions which are not cost-effective with traditional methods

TrustZone: The three features

- All of the hardware and software resources are partitioned so that they exist in one of two worlds
  - Secure world for the security subsystem,
  - Normal world for everything else.

  Hardware logic ensures that Normal world components cannot access Secure world resources, enabling a strong security perimeter between the two.
- By placing sensitive resources in the Secure world, and by robust software on the secure cores, we protect almost any asset against possible attacks
- The second aspect is the extensions in the processor cores. They enable to share a single physical core between the Normal world and the Secure world in a time-sliced fashion. This removes the need for a dedicated security core
- The third is a security-aware debug infrastructure which can enable control over access to Secure world debug, without impairing debug visibility

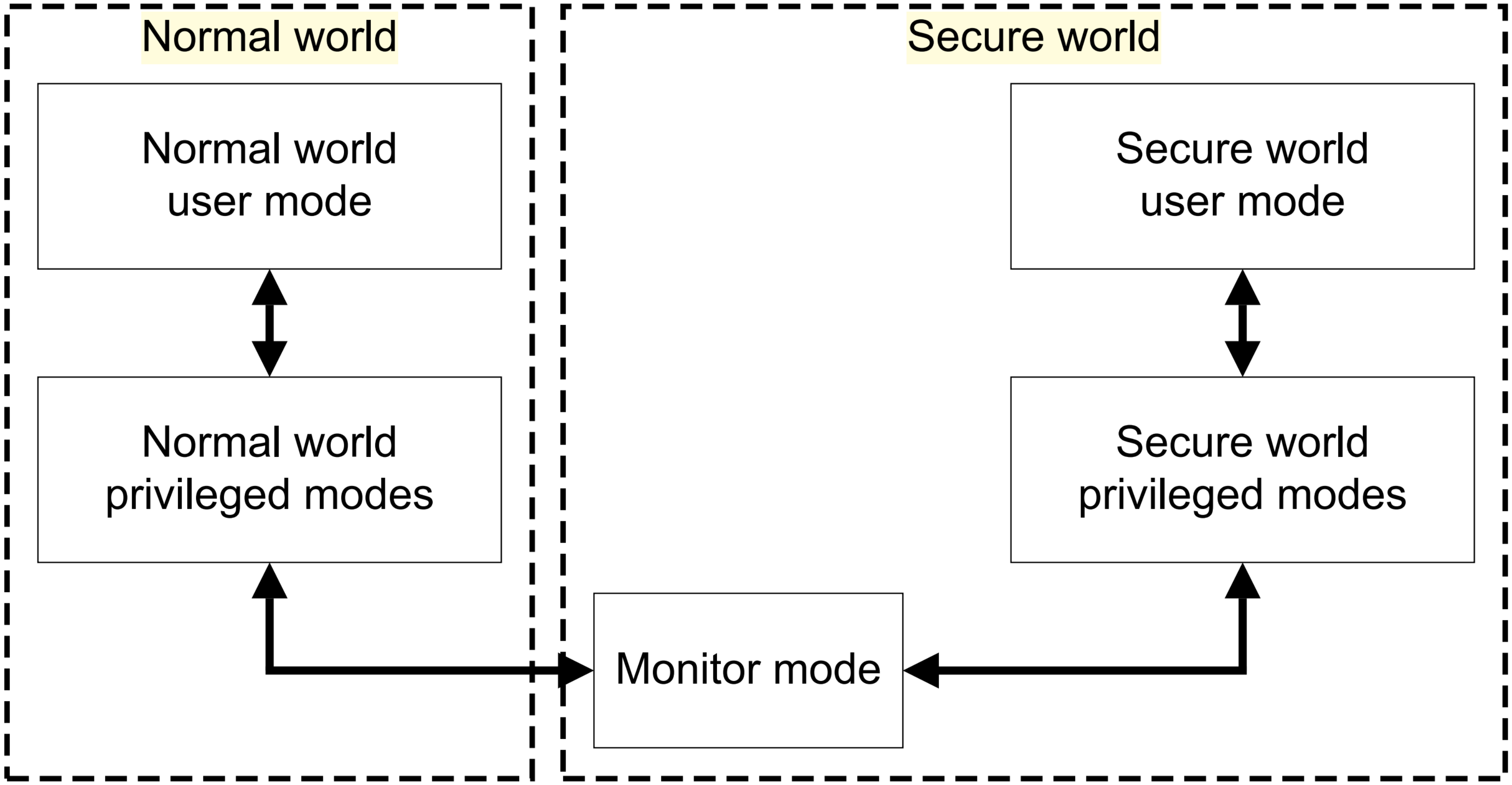
System architecture – system bus

- The most significant feature of the extended bus design is the addition of an extra control signal, the Non-Secure or NS bits for each of the read and write channels on the main system bus.
- All bus masters set these signals when they make a new transaction, and the bus or slave decode logic must interpret them to ensure that the required security separation is not violated.
- All Non-secure masters must have their NS bits set high in the hardware, which makes it impossible for them to access Secure slaves. The address decode for the access will not match any Secure slave and the transaction will fail.
- If a Non-secure master attempts to access a Secure slave it is implementation defined whether the operation fails silently or generates an error. An error may be raised by the slave or the bus, depending on the hardware peripheral design and bus configuration,

System architecture – processor

- Each physical processor cores provides
  - two virtual cores, one considered Non-secure and the other Secure,
  - a mechanism to robustly context switch between them, known as monitor mode.
- The value of the NS bit sent on the main system bus is indirectly derived from the identity of the virtual core that performed the instruction or data access.
- This enables trivial integration of the virtual processors into the system security mechanism;
  - the Non-secure virtual processor can only access Non-secure system resources,
  - the Secure virtual processor can see all resources

System architecture – processor



System architecture – virtual processor switch

- The two virtual processors execute in a time-sliced fashion, context switching through a new core mode called monitor mode when changing the currently running virtual processor.
- The mechanisms the physical processor uses to enter monitor mode from the Normal world are tightly controlled, and are all viewed as exceptions to the monitor mode software.
- The entry to monitor is triggered by a dedicated instruction, the *Secure Monitor Call* (SMC) instruction, or by a subset of the hardware exception mechanisms. Interrupts and exceptions can all be configured to cause the processor to switch into monitor mode.
- The software that executes within monitor mode is implementation defined, but it generally saves the state of the current world and restores the state of the world being switched to. It then performs a return-from-exception to restart processing in the restored world.

System architecture – virtual processor switch

- The world in which the processor is executing is indicated by the NS-bit in the Secure Configuration Register (SCR) in CP15, the system control coprocessor, unless the processor is in monitor mode.
- When in monitor mode, the processor is always executing in the Secure world regardless of the value of the SCR NS-bit, but operations will access Normal world copies if the SCR NS-bit is set to 1.

System architecture – monitor

- The monitor mode software provides a robust gatekeeper which manages the switches between the Secure and Non-secure processor states.
- Its functionality are similar to a traditional OS context switch, ensuring that state of the world that the processor is leaving is safely saved, and the state of the world the processor is switching to is correctly restored.
- Normal world entry to monitor mode is tightly controlled. It is only possible via the following exceptions: an interrupt, an external abort, or an explicit call via an SMC instruction.
- The Secure world entry to the monitor mode is a little more flexible, and can be achieved by directly writing to CPSR, in addition to the exception mechanisms available to the Normal world.
- The monitor is a security critical component, as it provides the interface between the two worlds. For robustness reasons it is suggested that the monitor code executes with interrupts disabled.

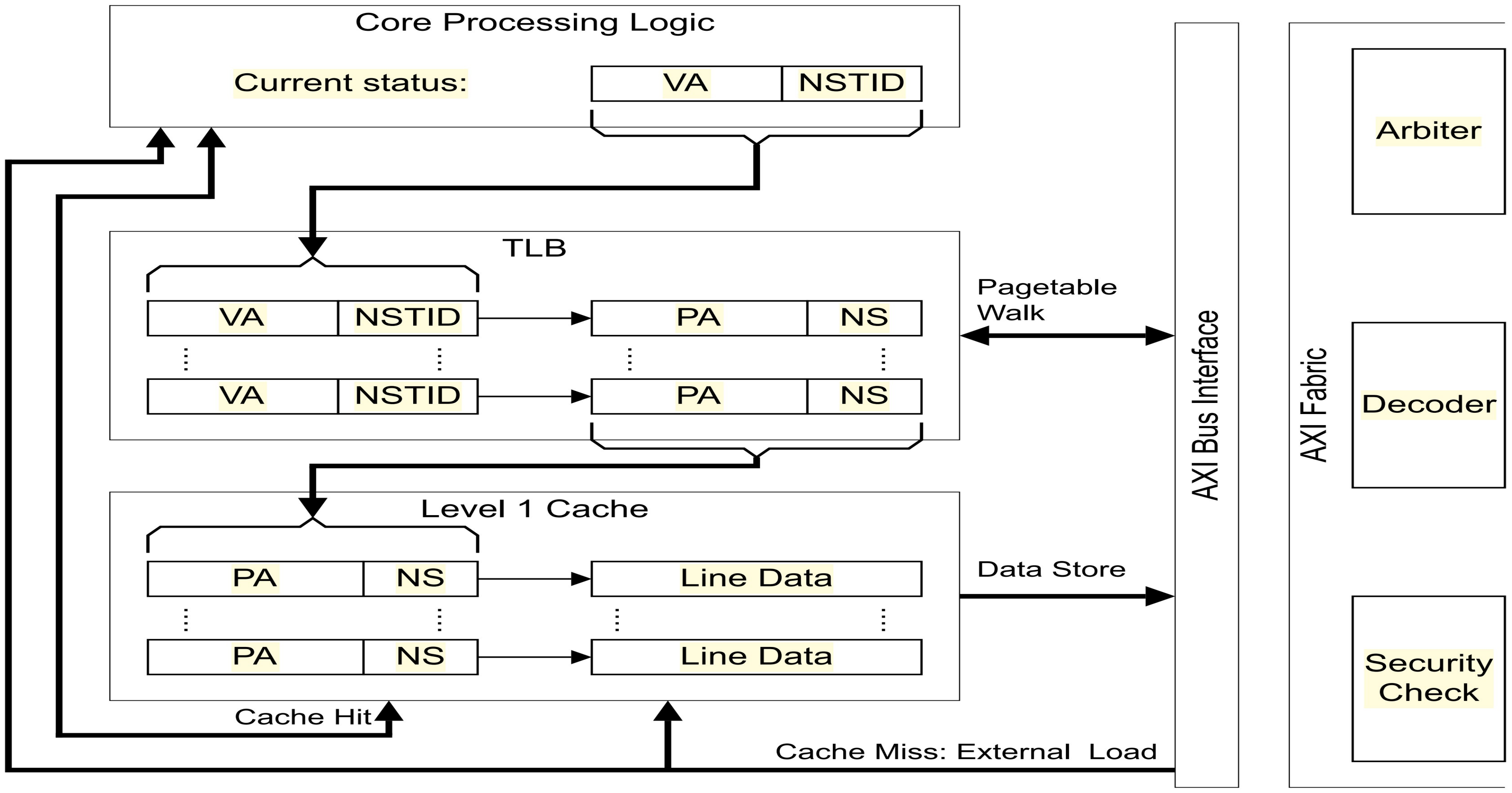
System architecture – memory subsytem

- Two virtual MMUs exist, one for each virtual processor. This enables each world to have a local set of translation tables, giving them independent control over their virtual to physical mappings.
- The L1 translation table descriptor includes an NS field the Secure virtual processor uses to determine the value of the NS-bit to access the physical memory locations associated with that table descriptor.
- The Non-secure virtual processor hardware ignores this field, and NS=1 in any memory access. This enables the Secure virtual processor to access either Secure or Non-secure memory.
- To enable efficient context switching between worlds, entries in the Translation Lookaside Buffers (TLBs) are tagged with the identity of the world that performed the walk. This allows Non-secure and Secure entries to co-exist in the TLBs, enabling faster switching as there is then no need to flush TLB entries.

System architecture – memory subsytem

- Any high performance design should support data of both security states in the caches to remove the need for a cache flush when switching between worlds, and enables high performance software to communicate over the boundary.
- To enable this the L1, and where applicable level two and beyond, processor caches have been extended with an additional tag bit to record the security state of the transaction that accessed the memory.
- The content of the caches, with regard to the security state, is dynamic. Any non-locked down cache line can be evicted to make space for new data, regardless of its security state. A Secure line load may evict a Non-secure line, and a Non-secure load may evict a Secure line.

System architecture – memory subsytem



System architecture – memory subsytem

- Consider a media application where encrypted audio content is loaded in the Normal world media player, and decrypted in the Secure world,
- The Secure world software can map the Non-secure memory containing the data belonging to the media player in the Secure world translation tables.
- In this way, the Secure world can directly access the Non-secure cache lines containing the audio content that needs to be decrypted; this type of memory is known as World-shared memory.
- A Normal world application can therefore pass data to a companion task in the Secure world though any level in the cache hierarchy.
- This enables a high performance system in comparison to solutions that require cached data to be flushed out of the cache and in to external memory.

System architecture – interrupts

- Two interrupt lines exist, IRQ and FIQ, trapped in the monitor, without intervention of code in either world
- Once the execution reaches the monitor, the trusted software routes the interrupt request accordingly. This allows a design to provide secure interrupt sources the Normal world software cannot manipulate.
- The recommended model uses IRQ as a Normal world interrupt source, and FIQ as the Secure world source. IRQ is the most common interrupt source in most operating environments, so the use of FIQ as the secure interrupt should mean the fewest modifications to existing software.
- If the processor is running the correct virtual core when an interrupt occurs there is no switch to the monitor and the interrupt is handled locally in the current world.
- Otherwise the hardware traps to the monitor that causes a context switch and jumps to the restored world, at which point the interrupt is taken.

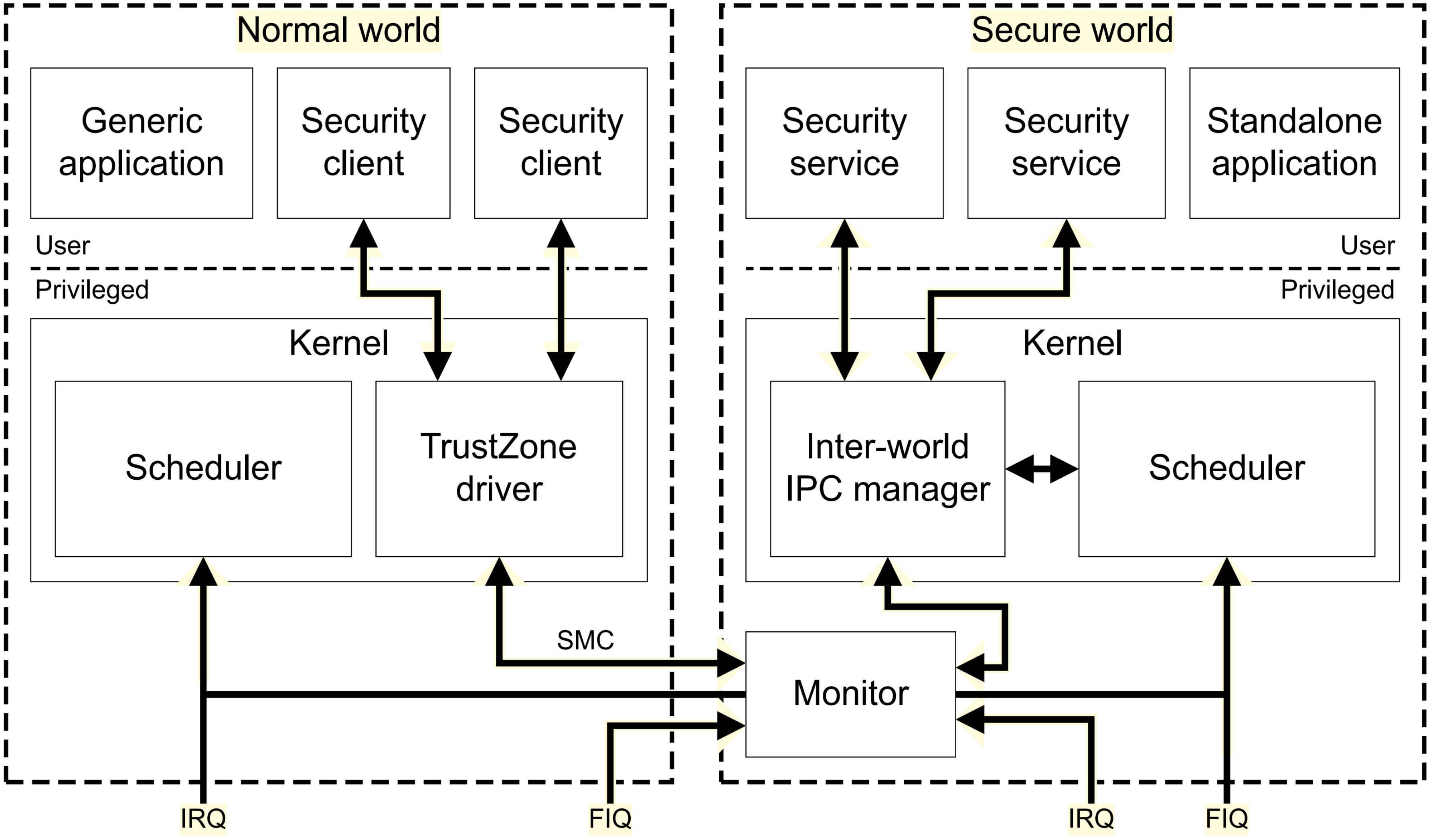
System architecture – debug

- The debug extensions separate the debug access control into independently configurable views of each of the following aspects:
  - Secure privileged invasive (JTAG) debug
  - Secure privileged non-invasive (trace) debug
  - Secure user invasive debug
  - Secure user non-invasive debug
- The Secure privileged debug access is controlled by two input to the core, **SPIDEN** (invasive) and **SPNIDEN** (non-invasive).
- The Secure user mode debug access is controlled by two bits, **SUIDEN** (invasive) and **SUNIDEN** (non-invasive) in a Secure privileged access only CP15 register.
- This enable a processor to give control over the debug visibility once the device is deployed. It is, for example, possible to give full Normal world debug visibility while also preventing all Secure world debug

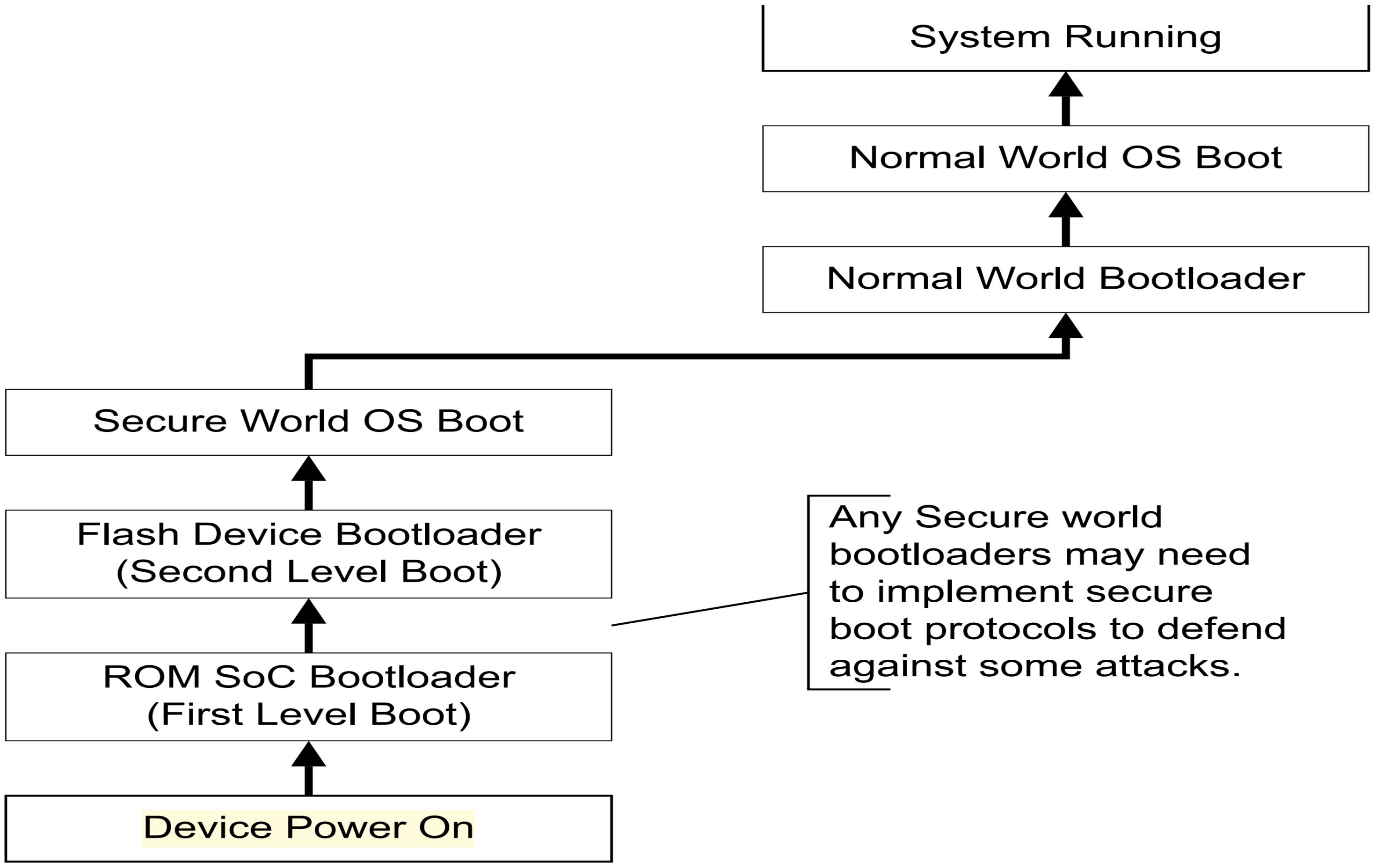
Software Architecture – A secure OS

- A secure OS can simulate concurrent execution of multiple Secure world applications, run-time download of new security applications, and Secure world tasks, completely independent of the Normal world environment.
- An extreme version of these designs closely resembles the software stacks in a SoC with two physical processors in an Asymmetric Multi-Processor.
- The software running on each virtual processor is a standalone operating system, and each world uses hardware interrupts to preempt the currently running world and acquire processor time.
- A tightly integrated design may uses a communications protocol that associates Secure world tasks with the Normal world thread that requested them. This provides many benefits of a Symmetric Multi-Processing (SMP).
- In these designs a Secure world application could, for example, inherit the priority of the Normal world task that it is assisting. This would enable some form of soft real-time response for media applications

Software Architecture – A secure OS



Software Architecture – Boot

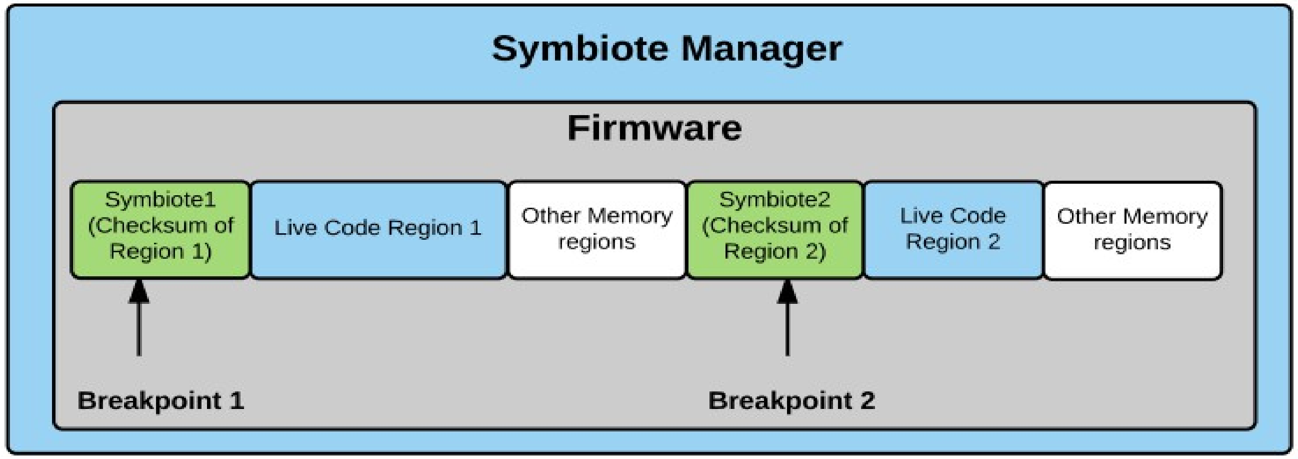

System Running
Normal World OS Boot
Normal World Bootloader
Secure World OS Boot
Flash Device Bootloader (Second Level Boot)
ROM SoC Bootloader (First Level Boot)
Device Power On
Any Secure world bootloaders may need to implement secure boot protocols to defend against some attacks.

Rogue Firmware – I

- After a lab attack, an erroneous update or because of a malicious software developer a device can store some malicious software that attacks the system or offers an hidden backdoor
- Some run time attacks can be discovered by memory introspection or attestation
- Other can be discovered by an intrusion detection system
- Doppelganger is a host-based intrusion detection solution for embedded devices.
- It can detect both kernel- and application-level attacks in embedded devices.

Rogue Firmware – II

- Doppelganger first analyzes the firmware of the embedded device to detect live code regions therein. Live code regions are executable parts of the firmware.
- Once Doppelganger detects the executable area of the memory, it randomly inserts its symbiotes (watchpoints) into the detected live code areas.
- Doppelganger symbiotes contain a CRC32 checksum of the randomly selected live code regions.



Rogue Firmware – III

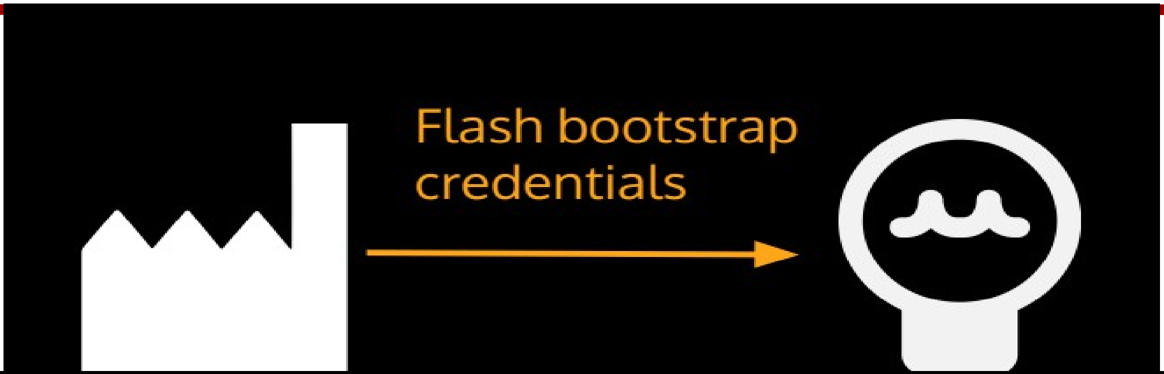
- Doppelganger adds its symbiote manager to the beginning of the firmware.
- The symbiote manager may be seen as a debugger that runs the firmware of the embedded system that causes Doppelganger to run in a different context of the OS to make it resistant to attacks against its runtime.
- During the firmware execution, every time the symbiote manager detects a symbiote in memory,
  - it stops the execution process (treating it as a breakpoint),
  - compares the current CRC32 checksum of the memory area with the symbiote checksum,
  - if the checksum does not match, Doppelganger considers this finding an evidence of a code modification attack does not allow the processor to continue running the code.
- Doppelganger does not defend against attacks that load code in the dynamic memory

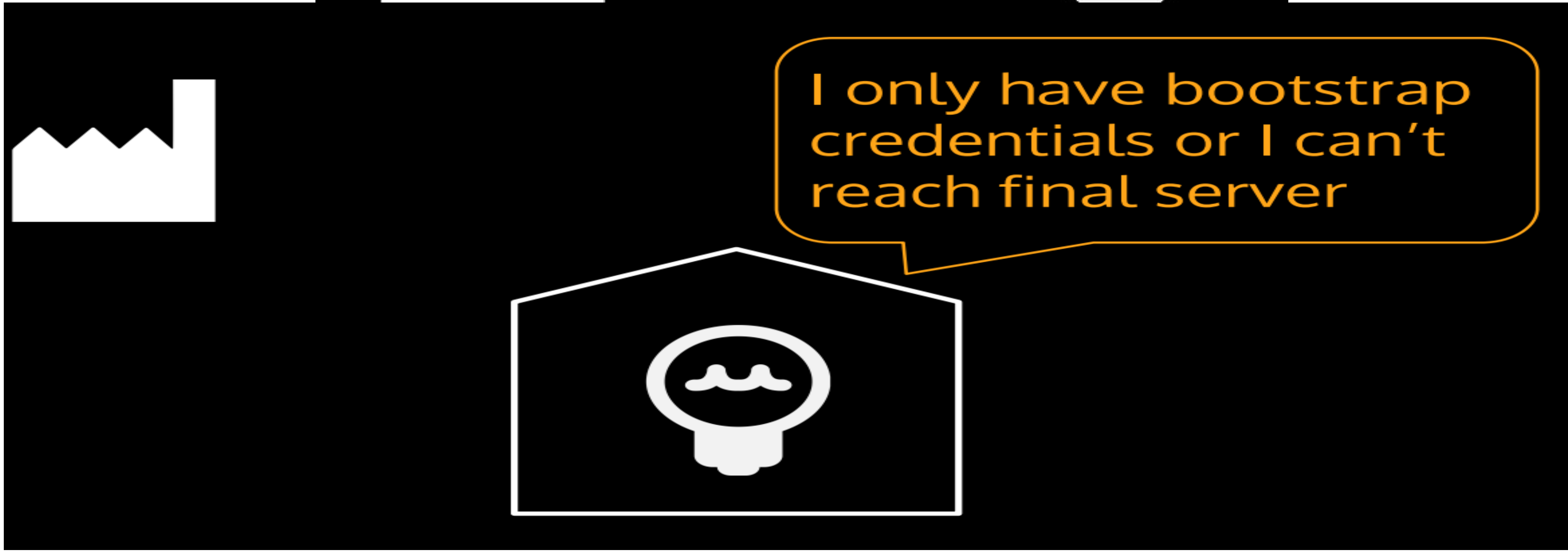
Managing an IOT device

- Several IOT devices are managed by a remote server that can
  - Update
  - Patch

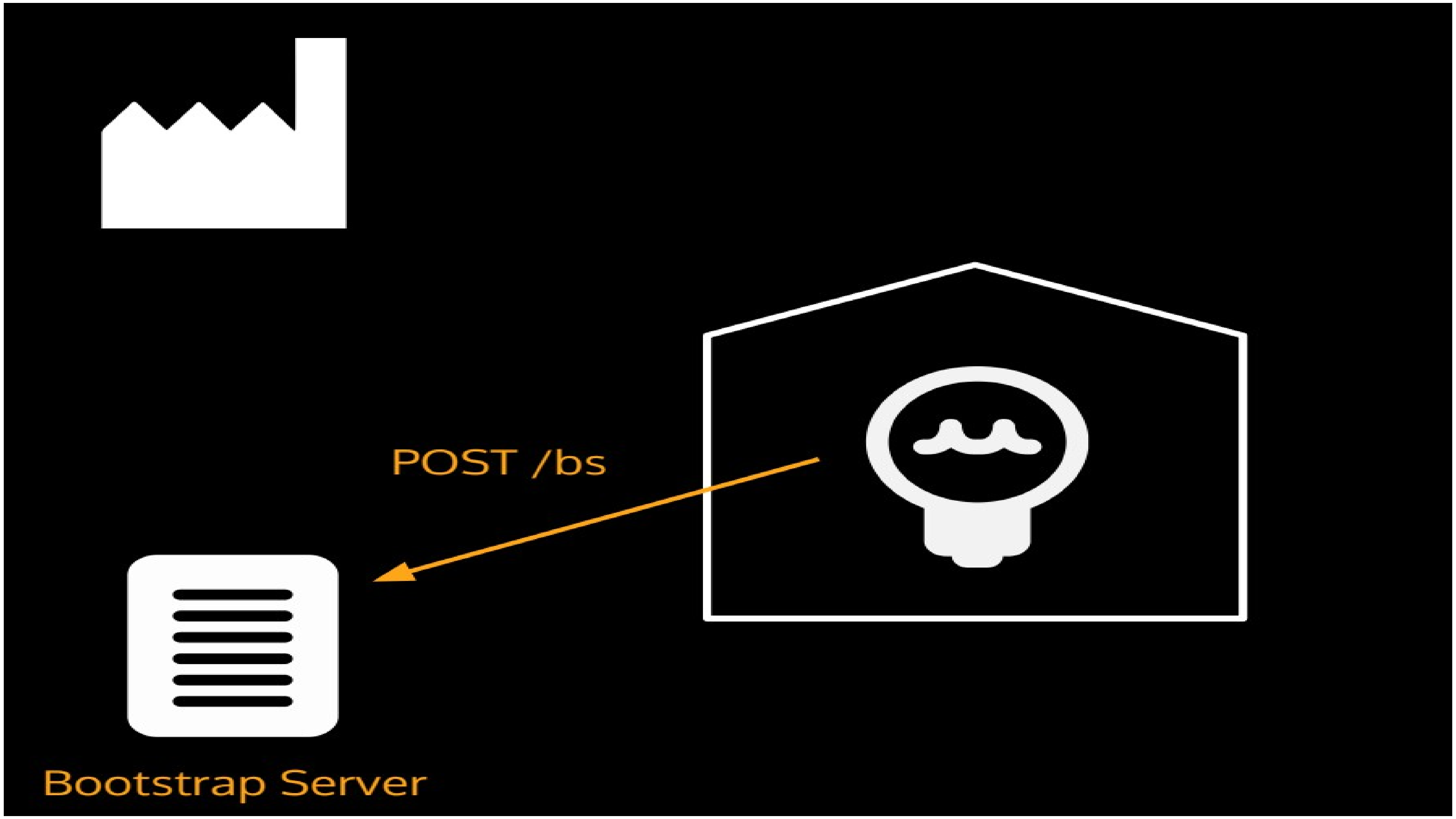
  the firmware in the device
- This poses the problem of how to connect the device to the remote server in a secure way
- Some encryption key has to be stored on the device and a proper protocol has to be adopted for the interactions with the server
- A TrustZone architecture can protect both the encryption key and the code of encryption function but not the transmissions
- However, the use of the encryption keys that are stored on the device poses some problems on the trust on the manufacturer of the device

Secure Management of a Device - I

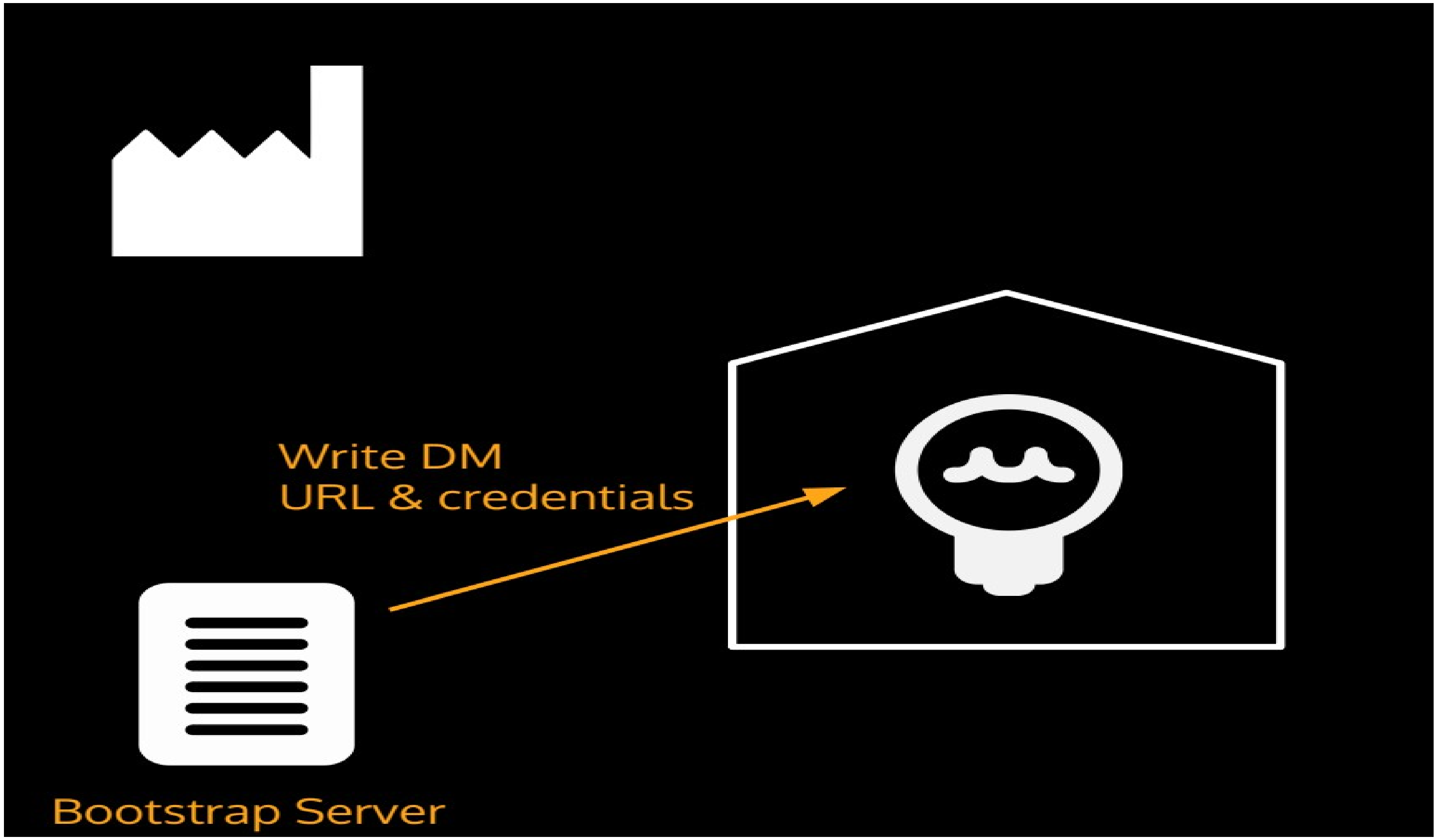



I only have bootstrap credentials or I can't reach final server

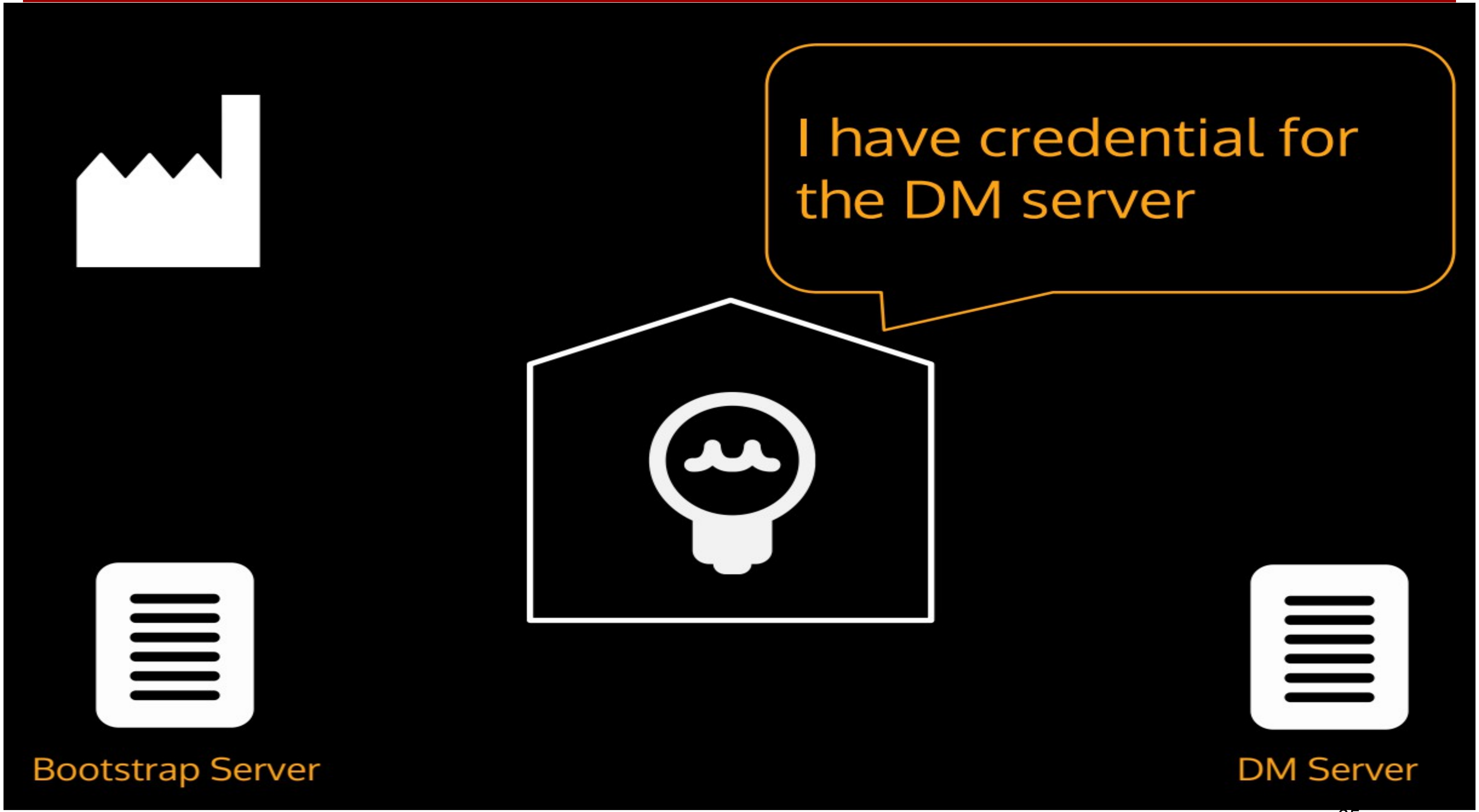
Secure Management of a Device - II



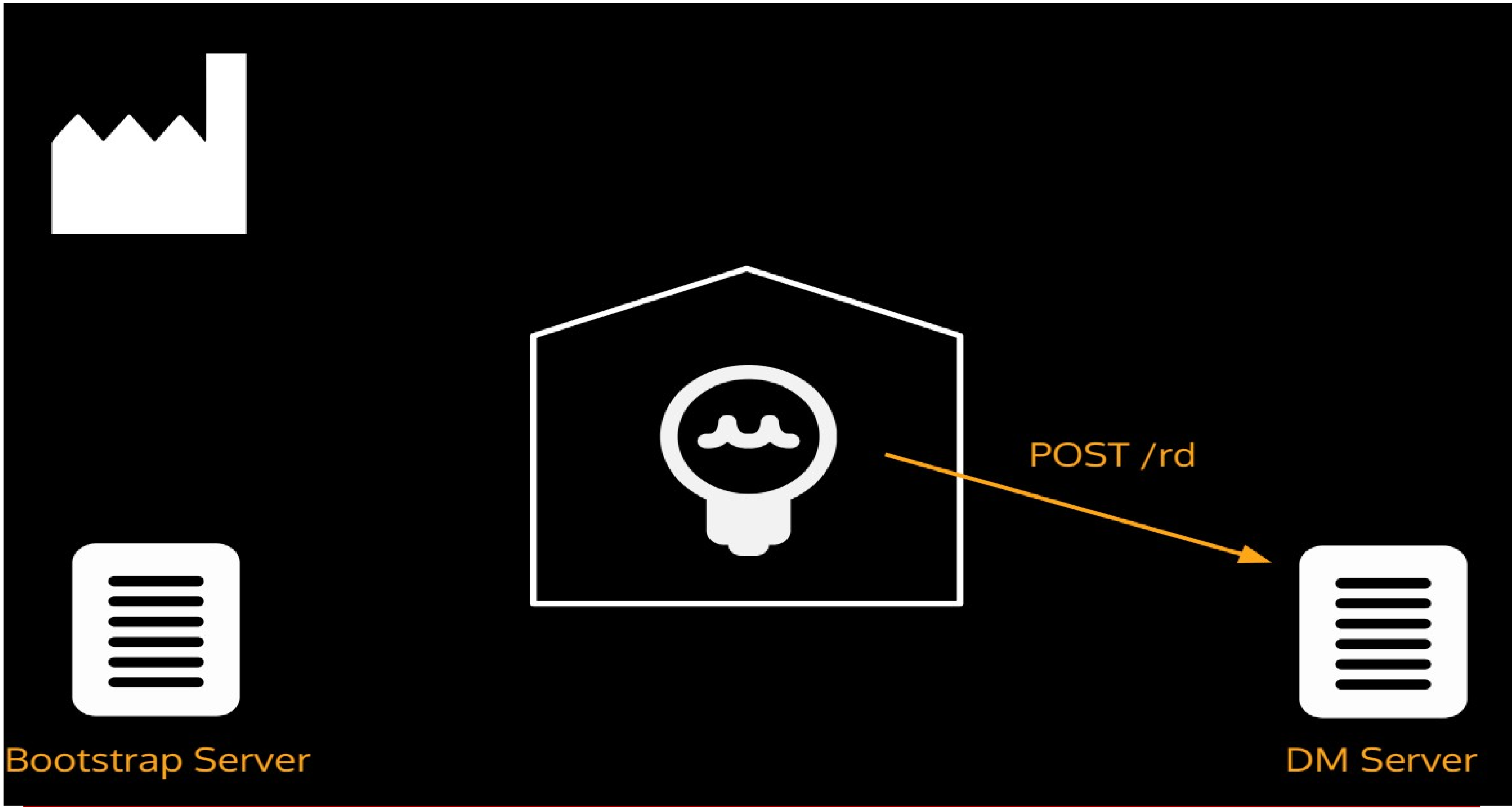
Secure Management of a Device - III



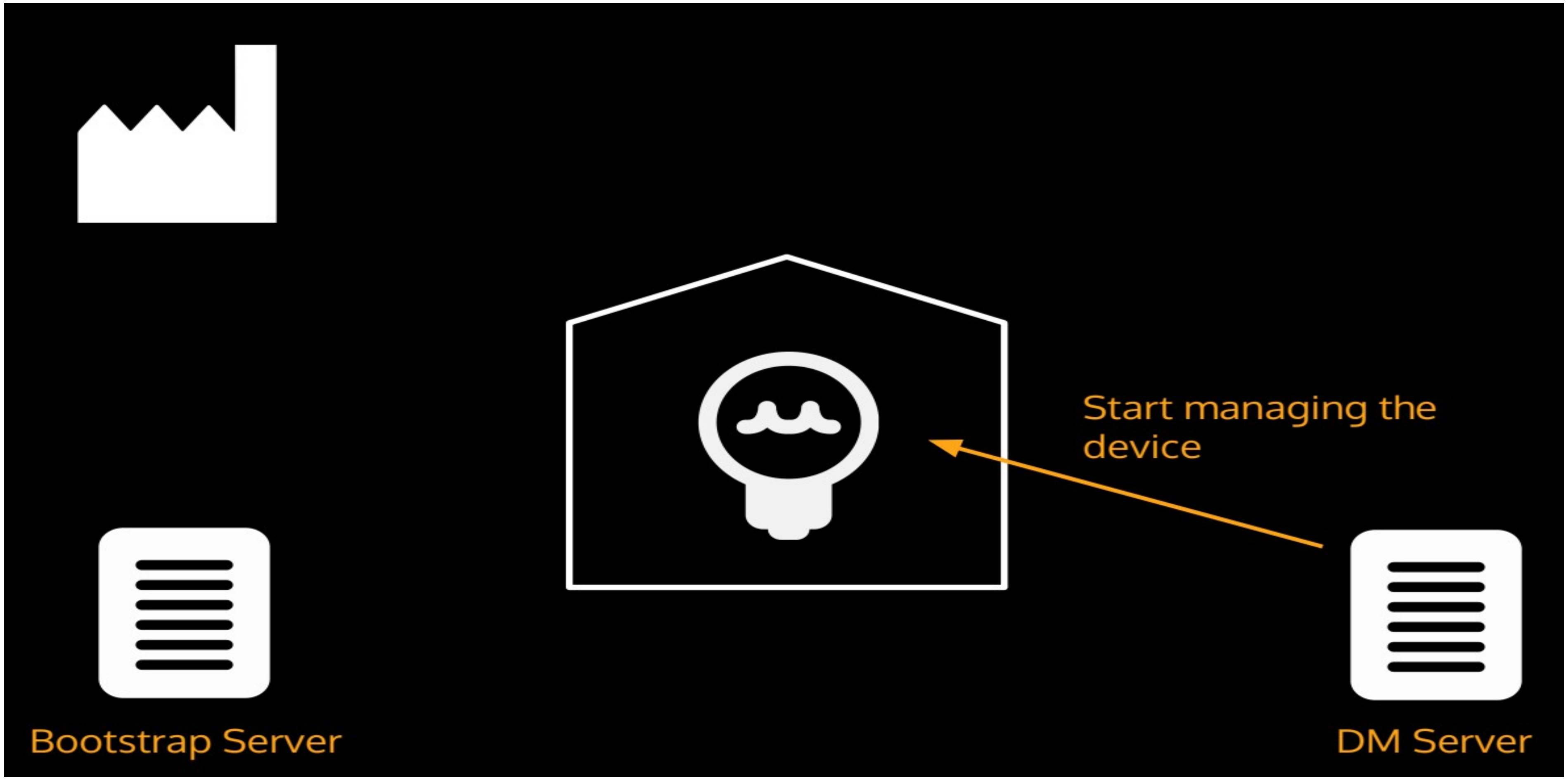
Secure Management of a Device - IV



Secure Management of a Device - V



Secure Management of a Device - VI



SIMON & SPECK encryption for IOT

- Simon and Speck are two families of block ciphers proposed by NSA
- Each of them comes in a variety of widths and key sizes.
- They fill the need for secure, flexible, and analyzable lightweight block ciphers.
- Many lightweight block ciphers exist, but most were designed to perform well on a single platform and were not meant to provide high performance across a range of devices.
- Each family offers excellent performance on hardware and software platforms, is flexible enough to admit a variety of implementations on a given platform, and is amenable to analysis using existing techniques.
- Both perform well across the full spectrum of lightweight applications, but
  - SIMON is tuned for optimal performance in hardware,
  - SPECK for optimal performance in software.

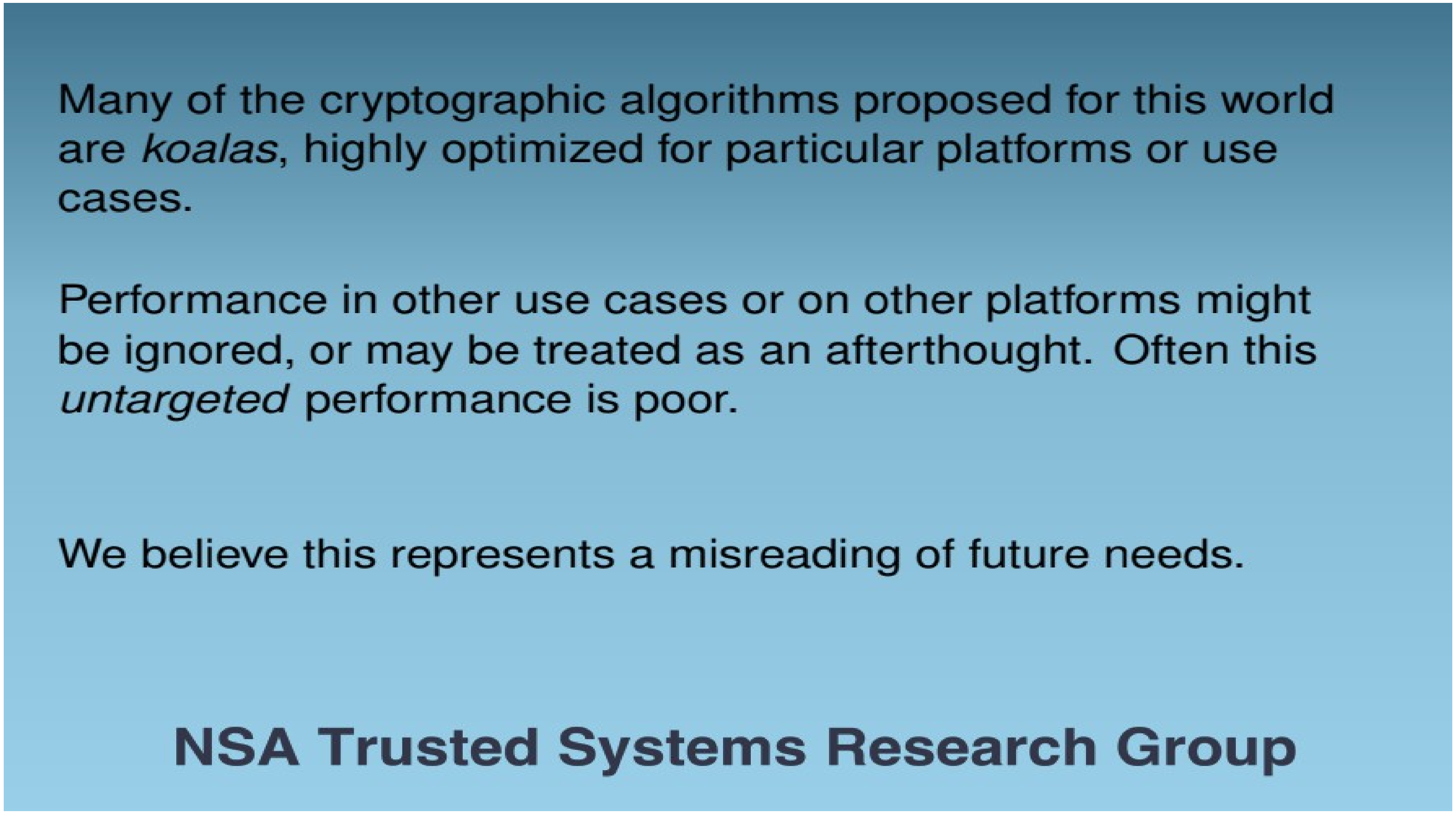
SIMON & SPECK in the NSA words …

The koala — a specialist; diet consists almost entirely of euca-lyptus leaves.

The American crow — a highly adaptable generalist.

NSA Trusted Systems Research Group

SIMON & SPECK in the NSA words …



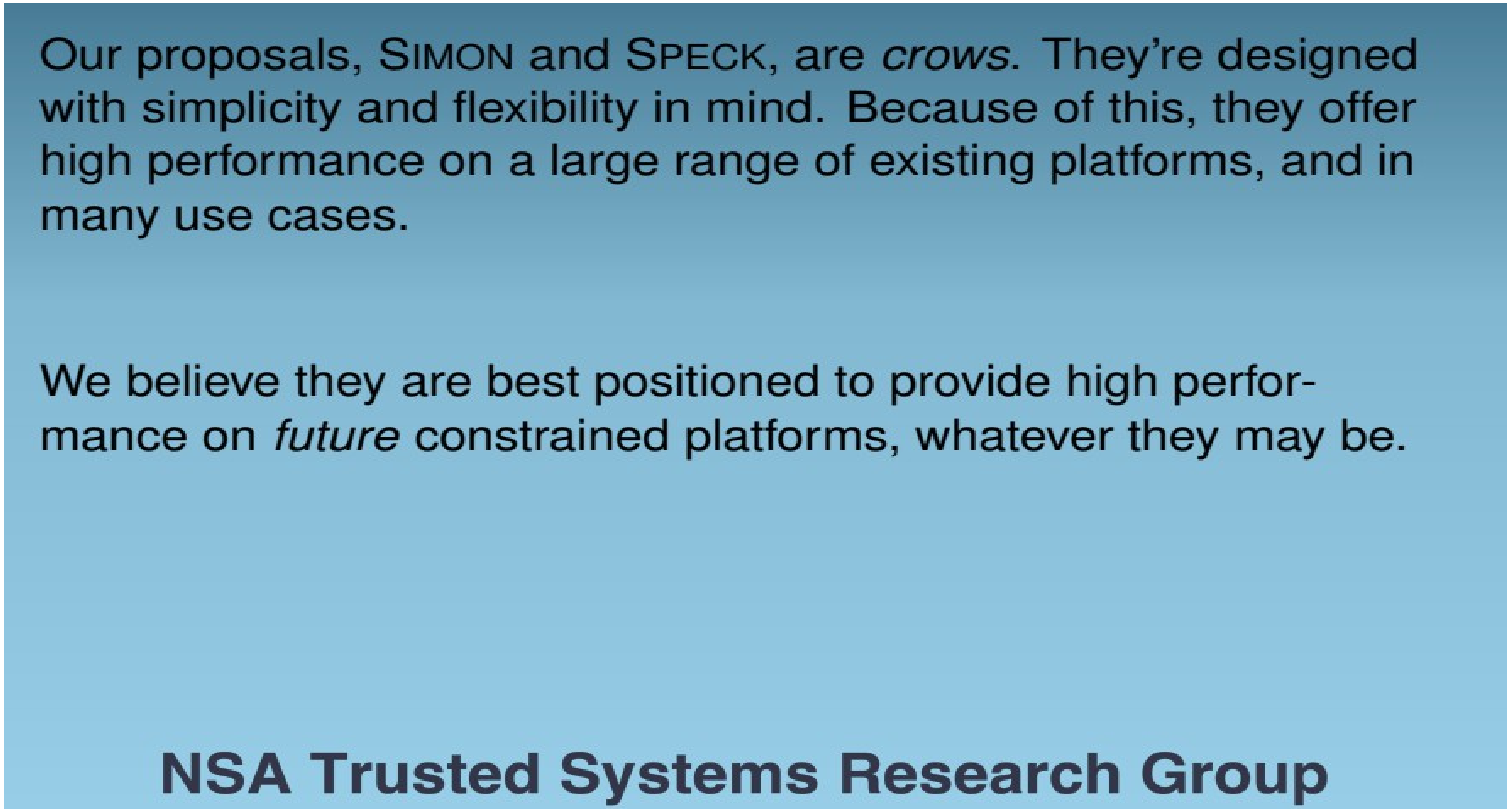
Many of the cryptographic algorithms proposed for this world are *koalas*, highly optimized for particular platforms or use cases.

Performance in other use cases or on other platforms might be ignored, or may be treated as an afterthought. Often this *untargeted* performance is poor.

We believe this represents a misreading of future needs.

NSA Trusted Systems Research Group

SIMON & SPECK in the NSA words …

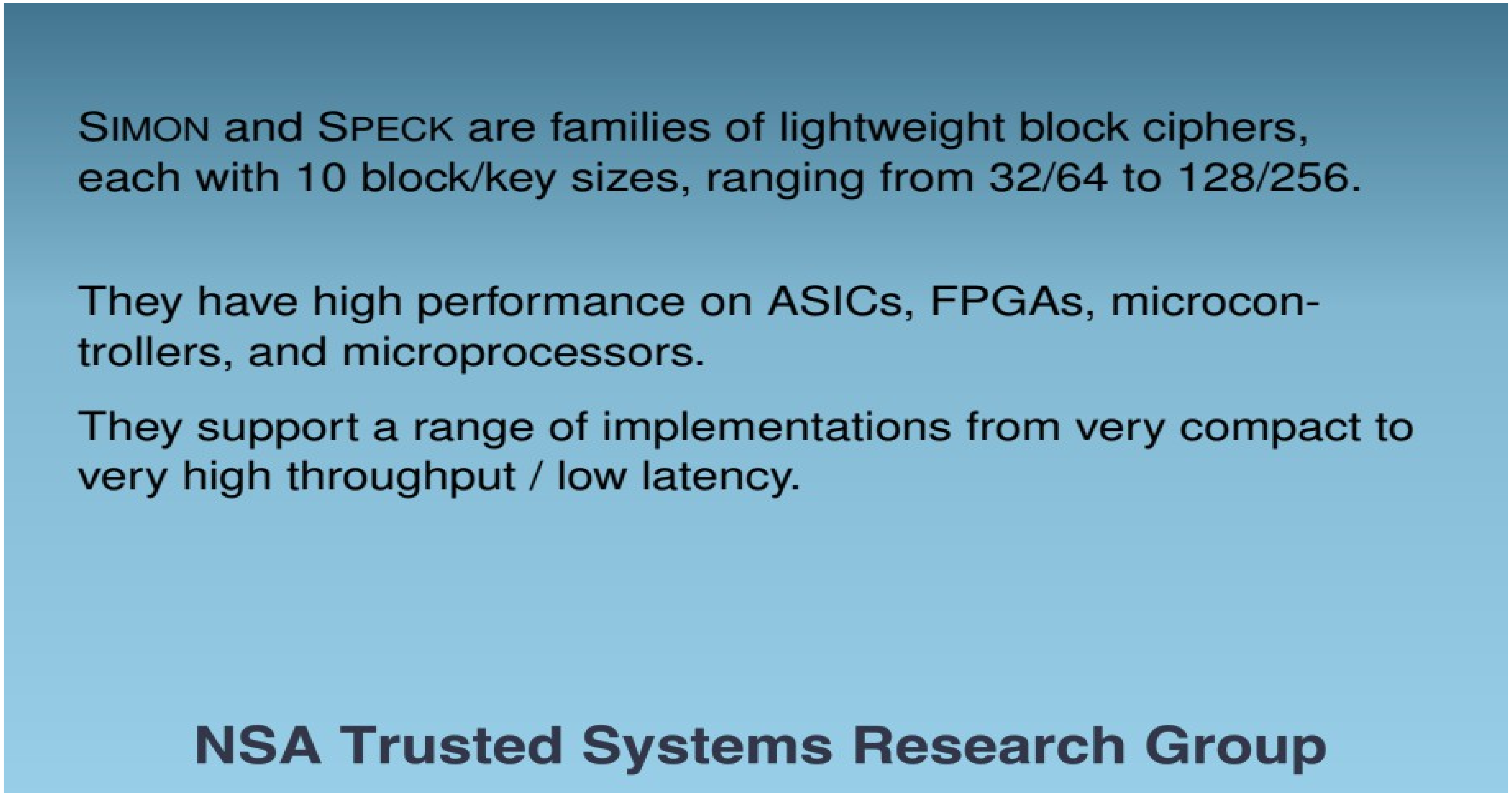


Our proposals, SIMON and SPECK, are *crows*. They're designed with simplicity and flexibility in mind. Because of this, they offer high performance on a large range of existing platforms, and in many use cases.

We believe they are best positioned to provide high performance on *future* constrained platforms, whatever they may be.

NSA Trusted Systems Research Group

SIMON & SPECK in the NSA words …



SIMON and SPECK are families of lightweight block ciphers, each with 10 block/key sizes, ranging from 32/64 to 128/256.

They have high performance on ASICs, FPGAs, microcon-trollers, and microprocessors.

They support a range of implementations from very compact to very high throughput / low latency.

NSA Trusted Systems Research Group

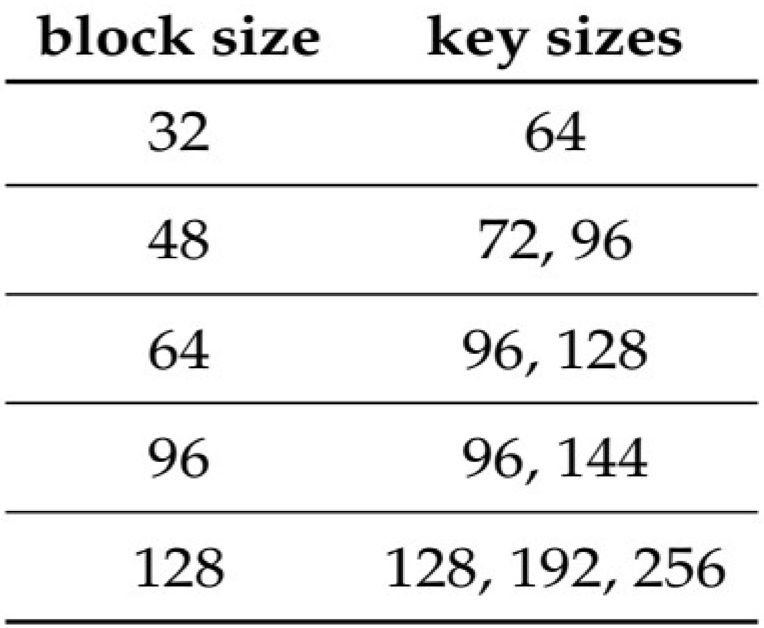
SIMON&SPECK – Design Consideration

- The main aim is to provide algorithms that have both
  - very small hardware implementations,
  - software implementations on small, low-power microcontrollers, with minimal flash and SRAM usage.
- A desire for low-area hardware designs favors simple, low complexity round functions, even if that means many rounds are required.
- While for lightweight applications, throughput is not the top priority some minimal throughput requirement may be adopted, eg at least 12 kilobits per second (kbps) at 100 kHz.
- For constrained hardware, very low-area implementations should be achievable, but it should achieve larger throughput by exploiting a larger-area.
- For software applications, very small flash and SRAM usage should be attainable, but high-throughput, low-energy implementations should be achievable as well.

SIMON&SPECK – Design Consideration

- Simon and Speck are block cipher families,each with ten algorithms
- They supports block sizes of 32, 48, 64, 96, and 128 bits, with up to three key sizes to go along with each block size.
- Block sizes of 64 and 128 bits are prevalent in the world of desktop computing,
- Typical block sizes of 48 or 96 bits are optimal for some electronic product code (EPC) applications.
- Key sizes are related to the desired level of security:
    - a very low-cost device may achieve adequate security using 64 bits key
    - more sensitive applications (running on suitably higher-cost devices) may require as many as 256 bits of key.

SIMON and SPECK parameters



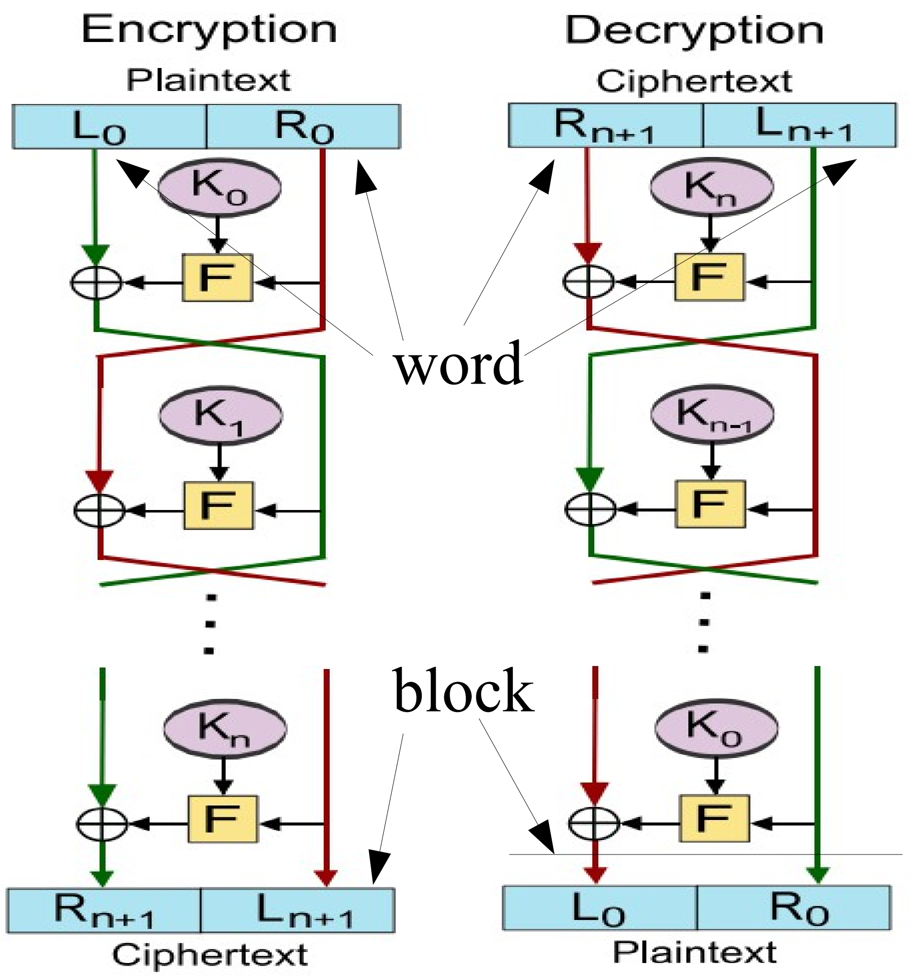
block size	key sizes
32	64
48	72, 96
64	96, 128
96	96, 144
128	128, 192, 256

SIMON/SPECK
- with a block size of *k* bits
- a key size of *r* bits

is denoted SIMON/SPECK *k*/*r*.

For example, SIMON 64/128 refers to the SIMON variant that uses a block size of 64 bits and a key size of 128 bits.

SIMON Algorithm

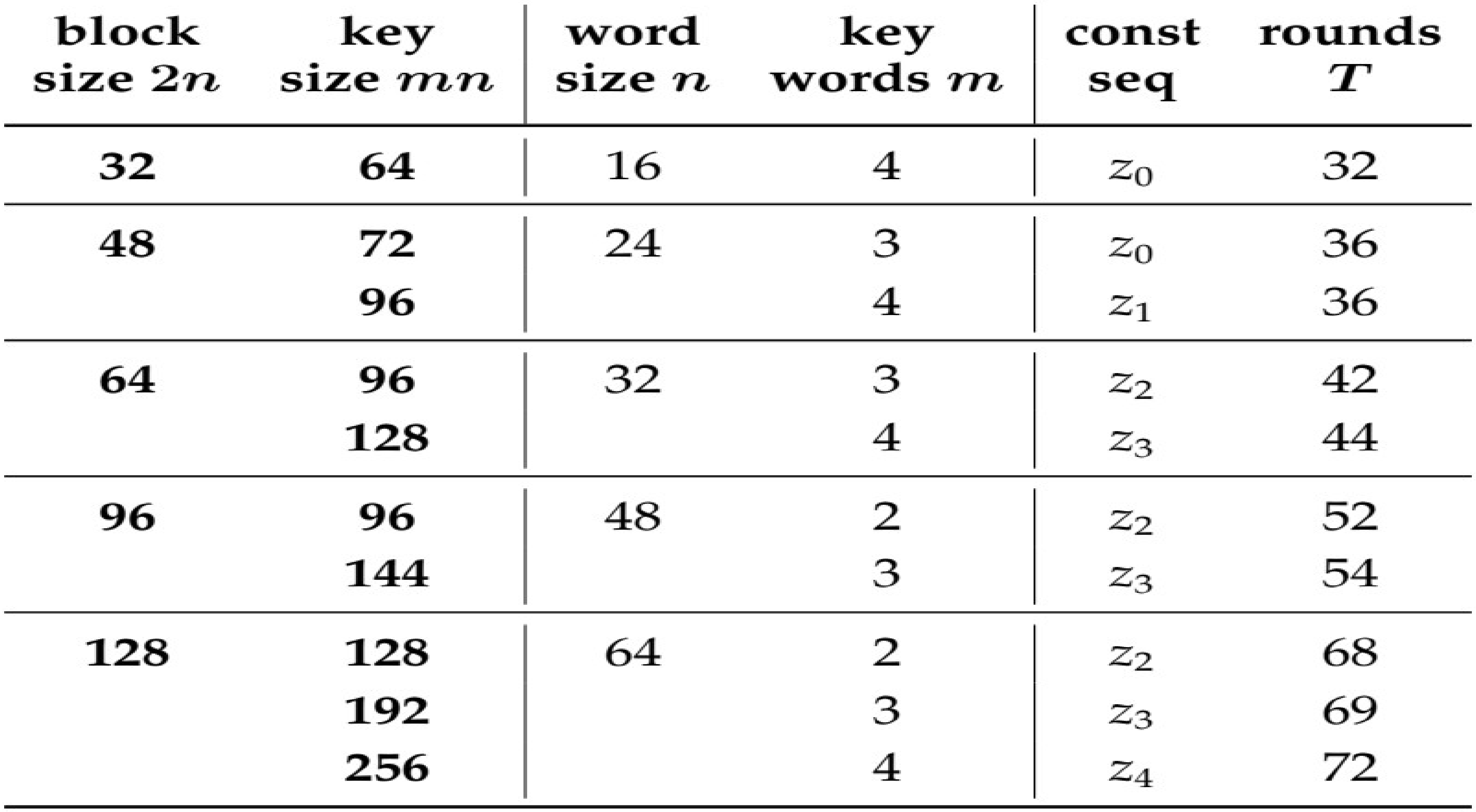


Simon with a block of 2n bits works on two words of nbits e.g. Simon32 works on two words each with 16 bits

This implies that n ∈[16-64] because the block size ∈[32-128]

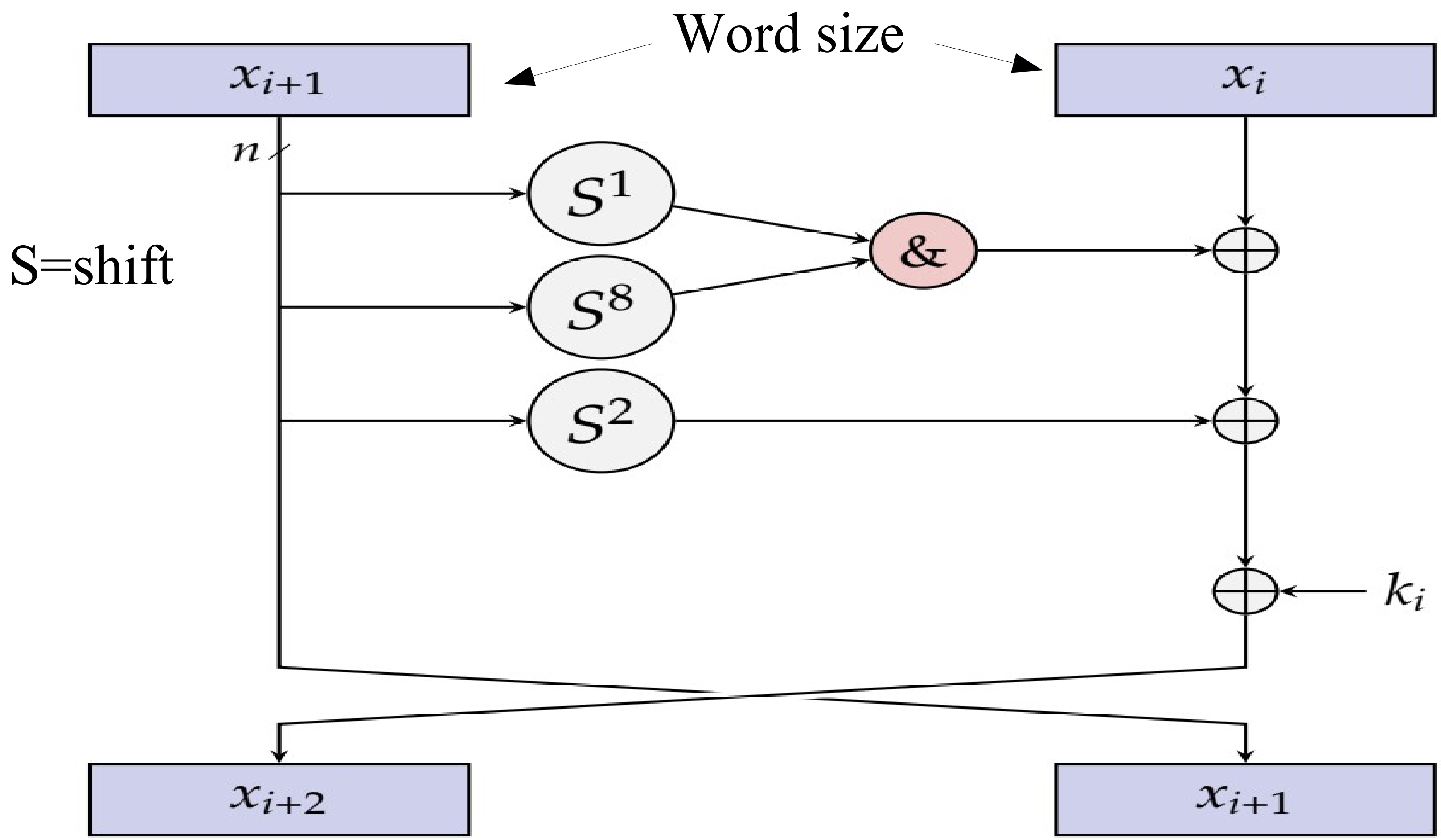
Each algorithm in the family repeatedly applies the Feistel chiper diagram

SIMON Algorithm – The Number of rounds



block size $2n$	key size mn	word size n	key words m	const seq	rounds T
32	64	16	4	z_0	32
48	72	24	3	z_0	36
	96		4	z_1	36
64	96	32	3	z_2	42
	128		4	z_3	44
96	96	48	2	z_2	52
	144		3	z_3	54
128	128	64	2	z_2	68
	192		3	z_3	69
	256		4	z_4	72

SIMON Algorithm – The Basic Step



SIMON Algorithm – The Key Sched - 1

z_0 $u = u_0u_1u_2\ldots = 1111101000100101011000011100110\ldots,$

z_1 $v = v_0v_1v_2\ldots = 1000111011111001001100001011010\ldots,$

$$z_2 = (z_2)_0(z_2)_1(z_2)_2\ldots = 1010111101110000001101001001100$$
$$0101000010001111110010110110011\ldots,$$

$$z_3 = (z_3)_0(z_3)_1(z_3)_2\ldots = 1101101110101100011001011110000$$
$$0010010001010011100110100001111\ldots,$$

$$z_4 = (z_4)_0(z_4)_1(z_4)_2\ldots = 1101000111100110101101100010000$$
$$0010111000011001010010011101111\ldots,$$

SIMON Algorithm – The Key Sched - 1

z_0 $u = u_0u_1u_2\ldots = 1111101000100101011000011100110\ldots,$

z_1 $v = v_0v_1v_2\ldots = 1000111011111001001100001011010\ldots,$

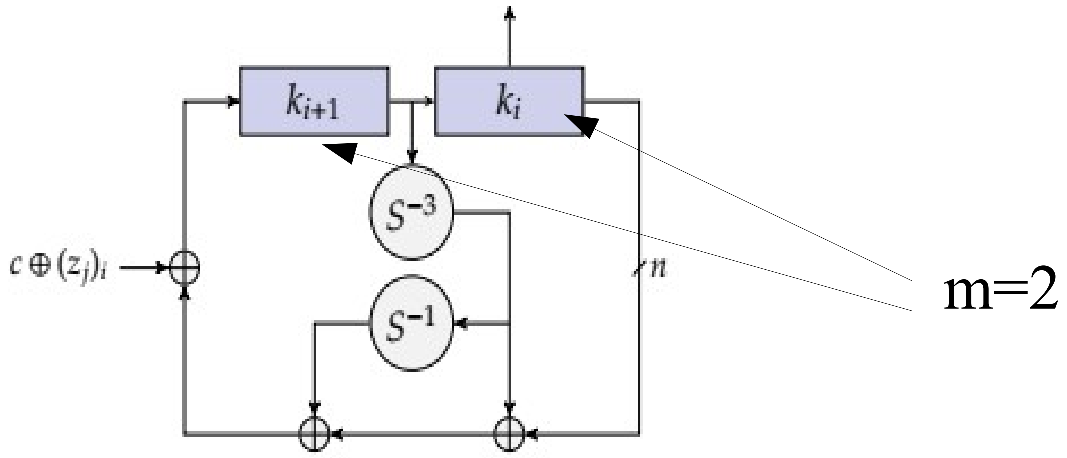
$$z_2 = (z_2)_0(z_2)_1(z_2)_2\ldots = 1010111101110000001101001001100$$
$$0101000010001111110010110110011\ldots,$$

$$z_3 = (z_3)_0(z_3)_1(z_3)_2\ldots = 1101101110101100011001011110000$$
$$0010010001010011100110100001111\ldots,$$

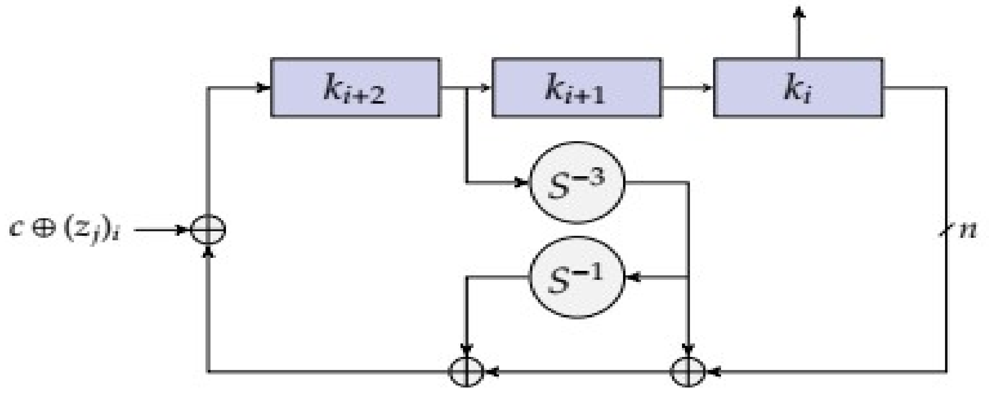
$$z_4 = (z_4)_0(z_4)_1(z_4)_2\ldots = 1101000111100110101101100010000$$
$$0010111000011001010010011101111\ldots,$$

SIMON Algorithm – The Key Sched - 2

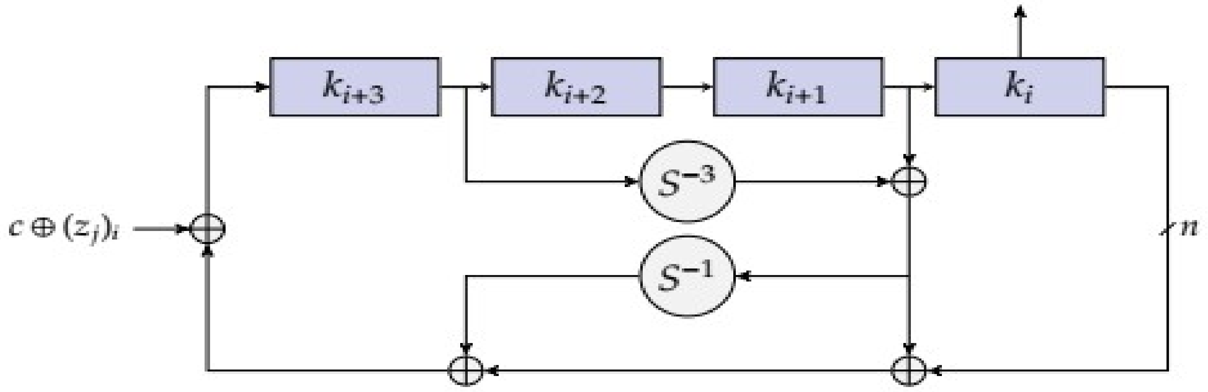
$$k_{i+m} = \begin{cases} c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+1}, & \text{if } m = 2, \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+2}, & \text{if } m = 3, \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})(S^{-3}k_{i+3} \oplus k_{i+1}), & \text{if } m = 4, \end{cases}$$



SIMON Algorithm – The Key Sched - 3

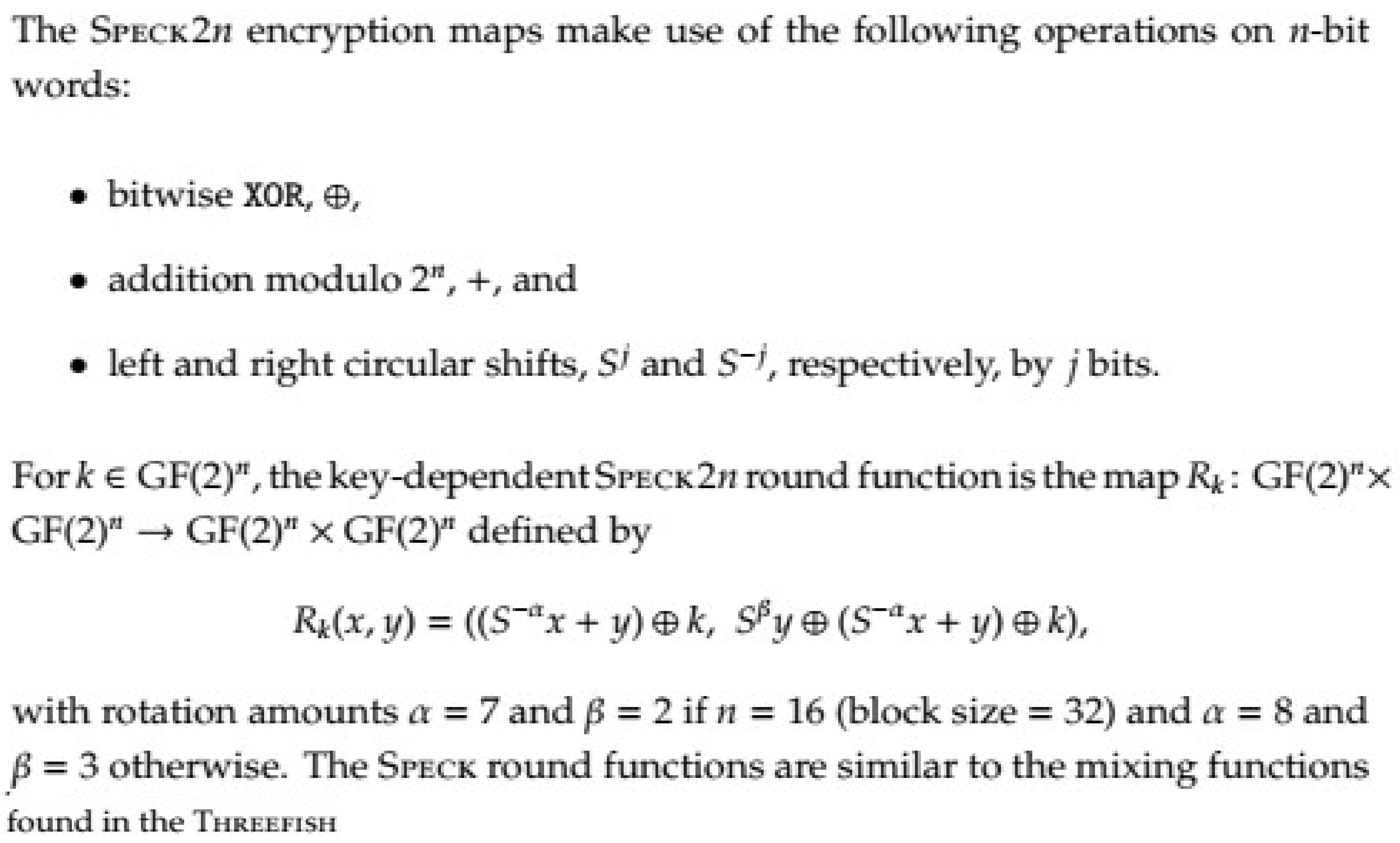


m=3



m=4

SPECK Family - Encryption



The Speck2n encryption maps make use of the following operations on n-bit words:

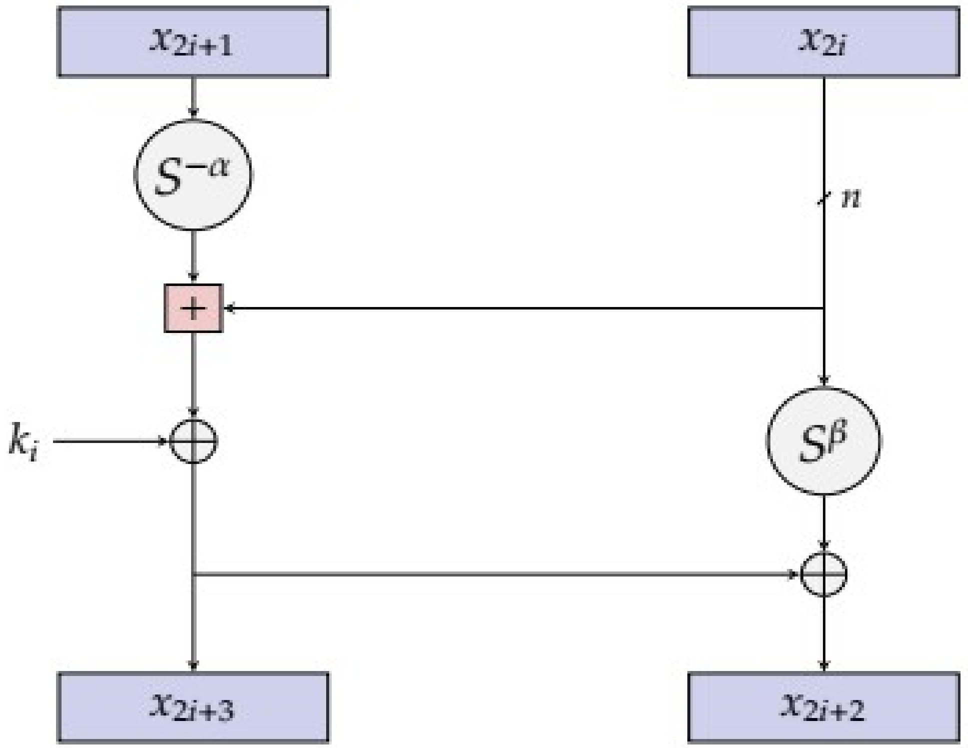
- bitwise XOR, $\oplus$,
- addition modulo 2^n, $+$, and
- left and right circular shifts, S^j and S^{-j}, respectively, by j bits.

For $k \in \mathrm{GF}(2)^n$, the key-dependent Speck2n round function is the map $R_k : \mathrm{GF}(2)^n \times \mathrm{GF}(2)^n \rightarrow \mathrm{GF}(2)^n \times \mathrm{GF}(2)^n$ defined by

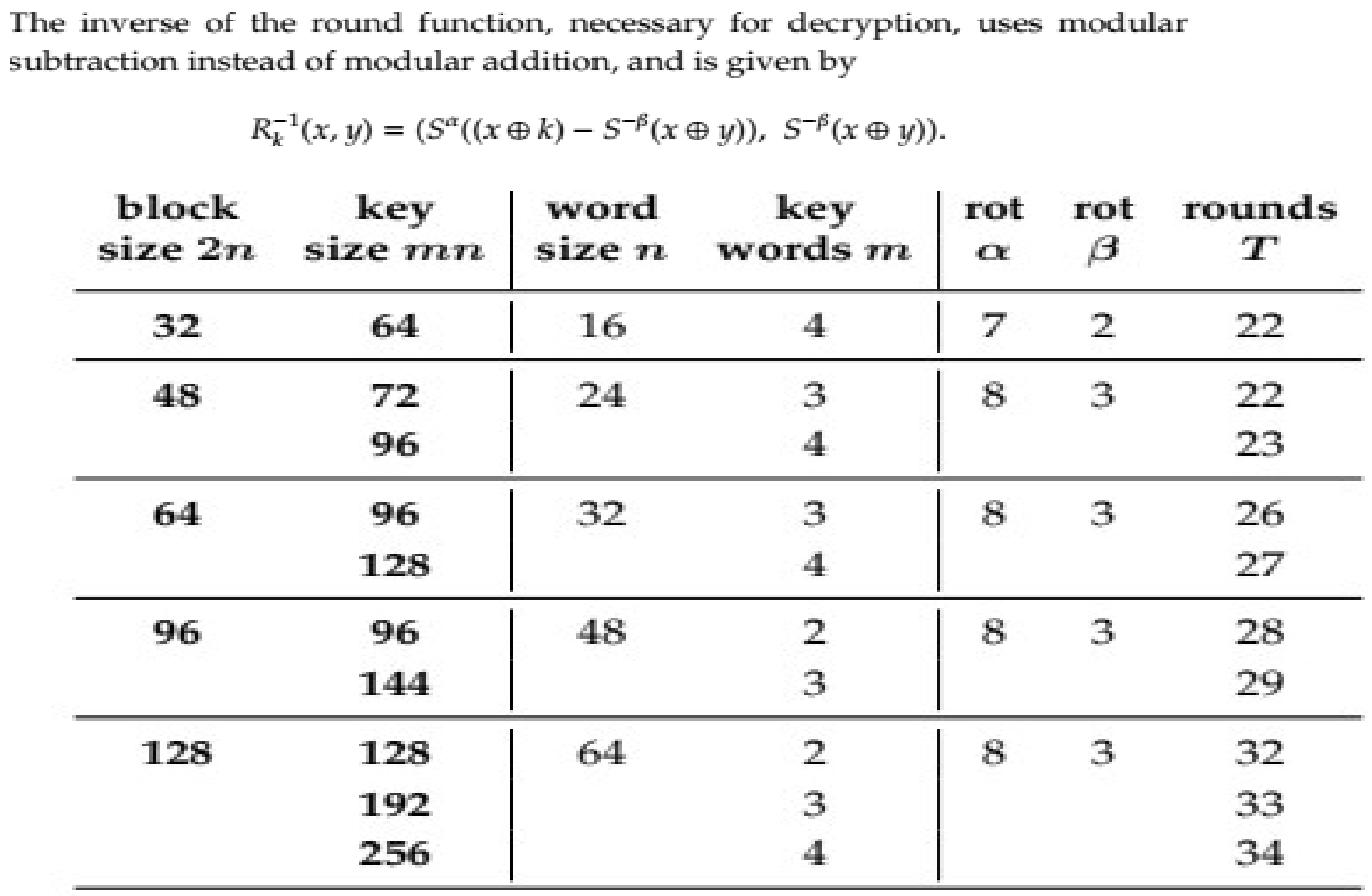
$$R_k(x, y) = ((S^{-\alpha}x + y) \oplus k,\ \ S^{\beta}y \oplus (S^{-\alpha}x + y) \oplus k),$$

with rotation amounts $\alpha = 7$ and $\beta = 2$ if $n = 16$ (block size = 32) and $\alpha = 8$ and $\beta = 3$ otherwise. The Speck round functions are similar to the mixing functions found in the Threefish

SPECK Family - Encryption



SPECK Family – Decryption - Parameters

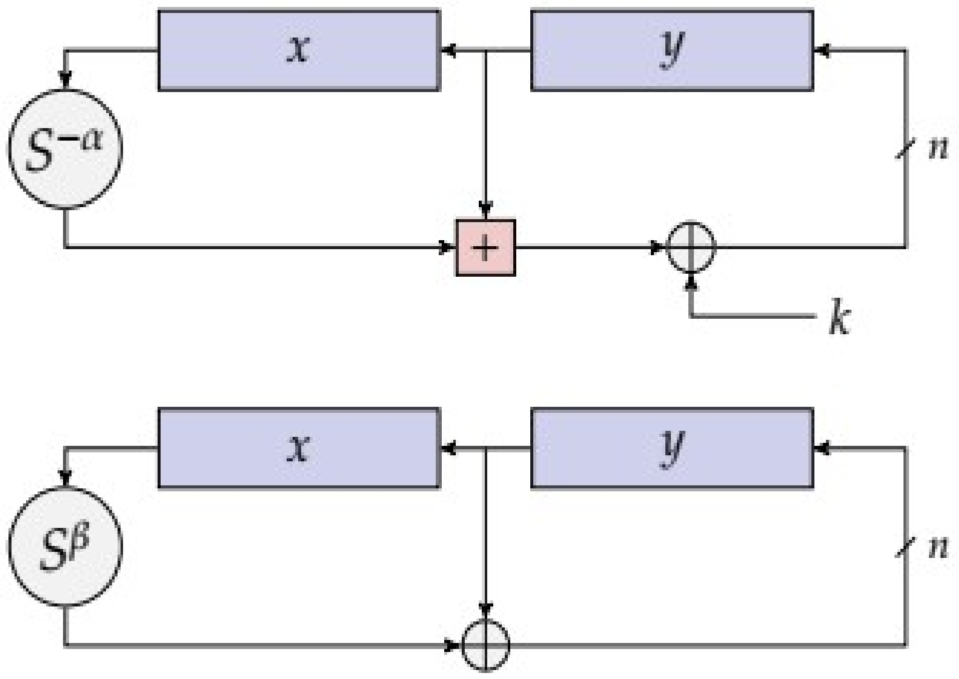


The inverse of the round function, necessary for decryption, uses modular subtraction instead of modular addition, and is given by

$$R_k^{-1}(x, y) = (S^{\alpha}((x \oplus k) - S^{-\beta}(x \oplus y)),\ S^{-\beta}(x \oplus y)).$$

block size $2n$	key size mn	word size n	key words m	rot α	rot β	rounds T
32	64	16	4	7	2	22
48	72	24	3	8	3	22
	96		4			23
64	96	32	3	8	3	26
	128		4			27
96	96	48	2	8	3	28
	144		3			29
128	128	64	2	8	3	32
	192		3			33
	256		4			34

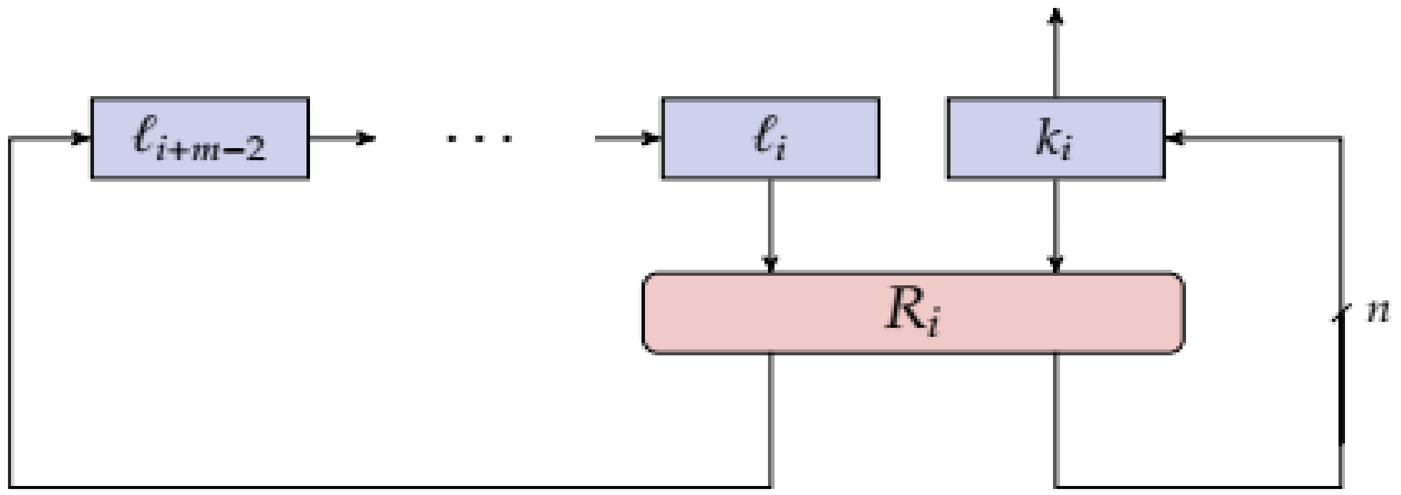
SPECK Round – Composition of two Feistel



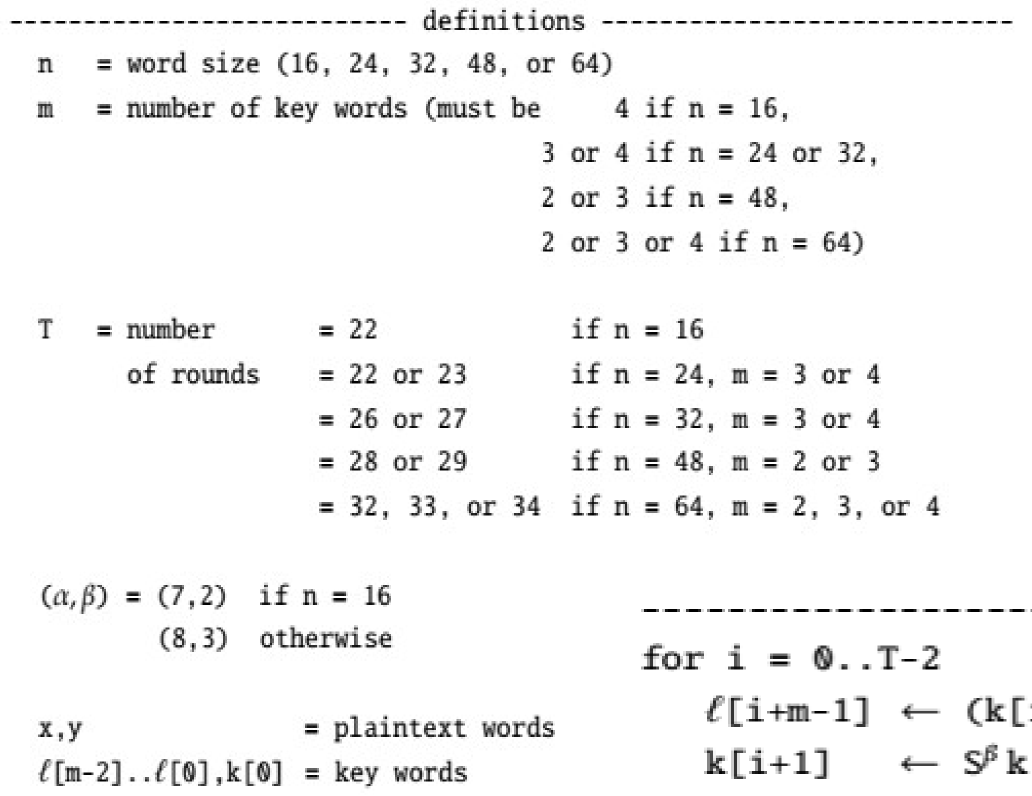
SPECK Key Scheduling

The SPECK key schedules use the round function to generate round keys k_i. Let K be a key for a SPECK2n block cipher. We can write $K = (\ell_{m-2}, \ldots, \ell_0, k_0)$, where $\ell_i, k_0 \in GF(2)^n$, for a value of m in $\{2,3,4\}$. Sequences k_i and ℓ_i are defined by

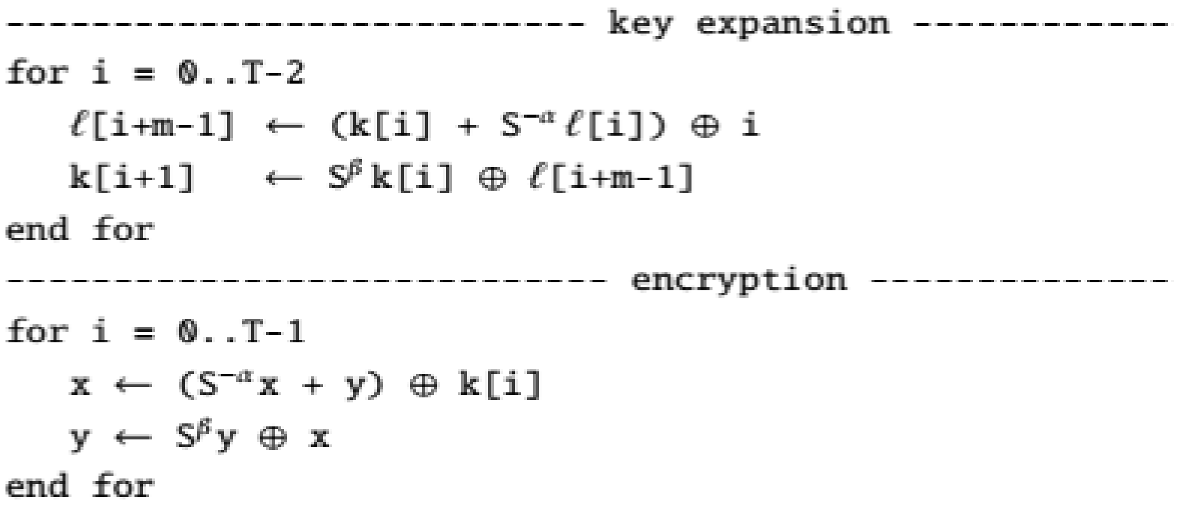
$$\ell_{i+m-1} = (k_i + S^{-\alpha}\ell_i) \oplus i \quad \text{and}$$
$$k_{i+1} = S^{\beta}k_i \oplus \ell_{i+m-1}.$$



SPECK Pseudo Code

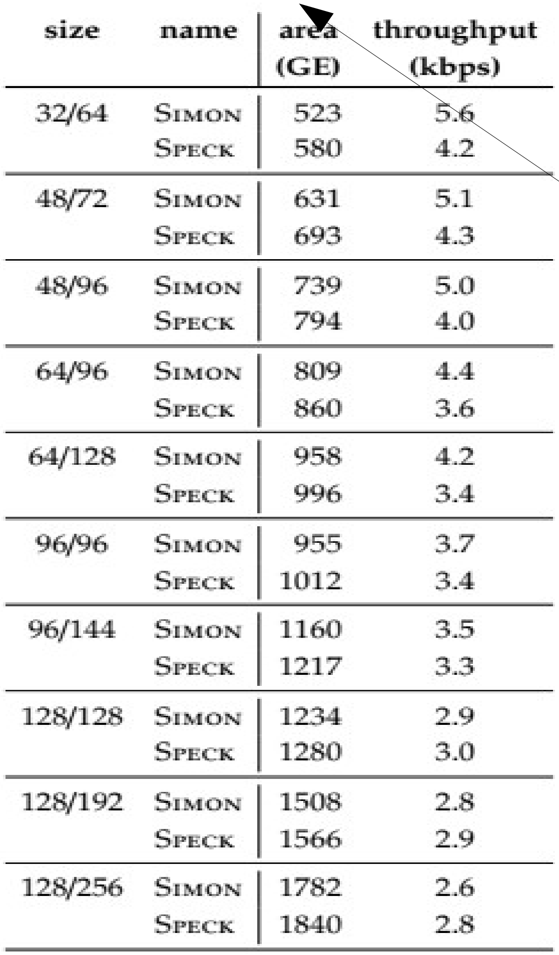


```
--------------------------- definitions ----------------------------
  n   = word size (16, 24, 32, 48, or 64)
  m   = number of key words (must be     4 if n = 16,
                                    3 or 4 if n = 24 or 32,
                                    2 or 3 if n = 48,
                                    2 or 3 or 4 if n = 64)

  T   = number        = 22             if n = 16
        of rounds     = 22 or 23       if n = 24, m = 3 or 4
                      = 26 or 27       if n = 32, m = 3 or 4
                      = 28 or 29       if n = 48, m = 2 or 3
                      = 32, 33, or 34  if n = 64, m = 2, 3, or 4

  $(\alpha,\beta)$ = (7,2)  if n = 16
          (8,3)  otherwise

  x,y                  = plaintext words
  $\ell$[m-2]..$\ell$[0],k[0]  = key words
```



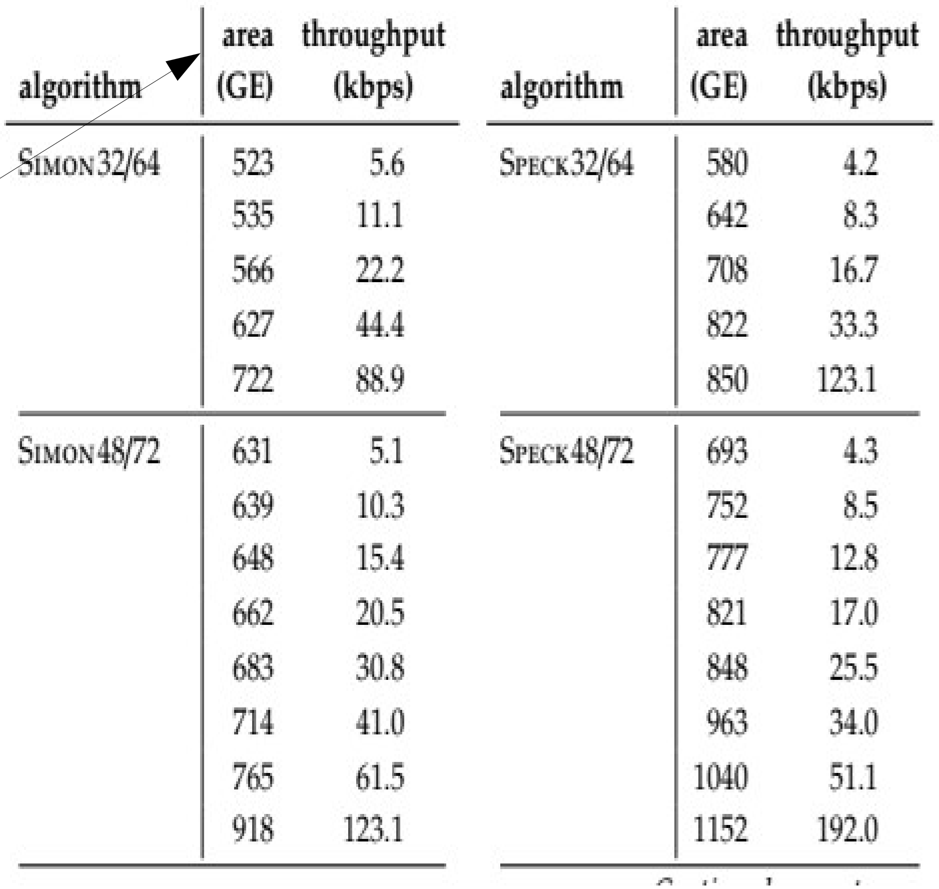
```
---------------------------- key expansion ------------
for i = 0..T-2
    $\ell[i+m-1] \leftarrow (k[i] + S^{-\alpha}\ell[i]) \oplus i$
    $k[i+1] \leftarrow S^{\beta}k[i] \oplus \ell[i+m-1]$
end for
----------------------------- encryption --------------
for i = 0..T-1
    $x \leftarrow (S^{-\alpha}x + y) \oplus k[i]$
    $y \leftarrow S^{\beta}y \oplus x$
end for
```

SIMON/SPECK Hardware (ASIC) Performance



size	name	area (GE)	throughput (kbps)
32/64	Simon	523	5.6
	Speck	580	4.2
48/72	Simon	631	5.1
	Speck	693	4.3
48/96	Simon	739	5.0
	Speck	794	4.0
64/96	Simon	809	4.4
	Speck	860	3.6
64/128	Simon	958	4.2
	Speck	996	3.4
96/96	Simon	955	3.7
	Speck	1012	3.4
96/144	Simon	1160	3.5
	Speck	1217	3.3
128/128	Simon	1234	2.9
	Speck	1280	3.0
128/192	Simon	1508	2.8
	Speck	1566	2.9
128/256	Simon	1782	2.6
	Speck	1840	2.8

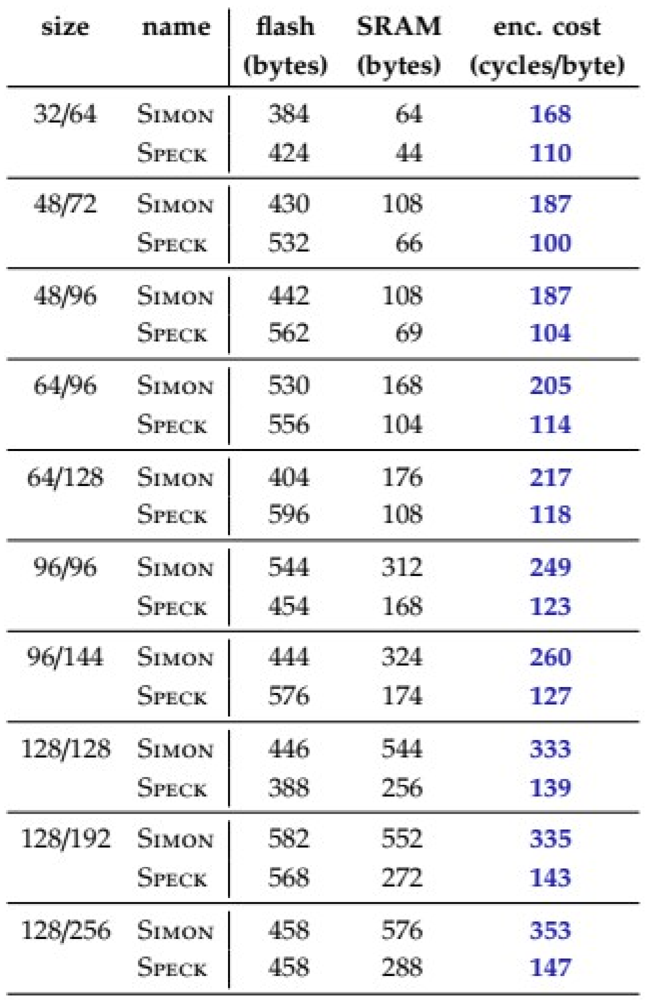
gates



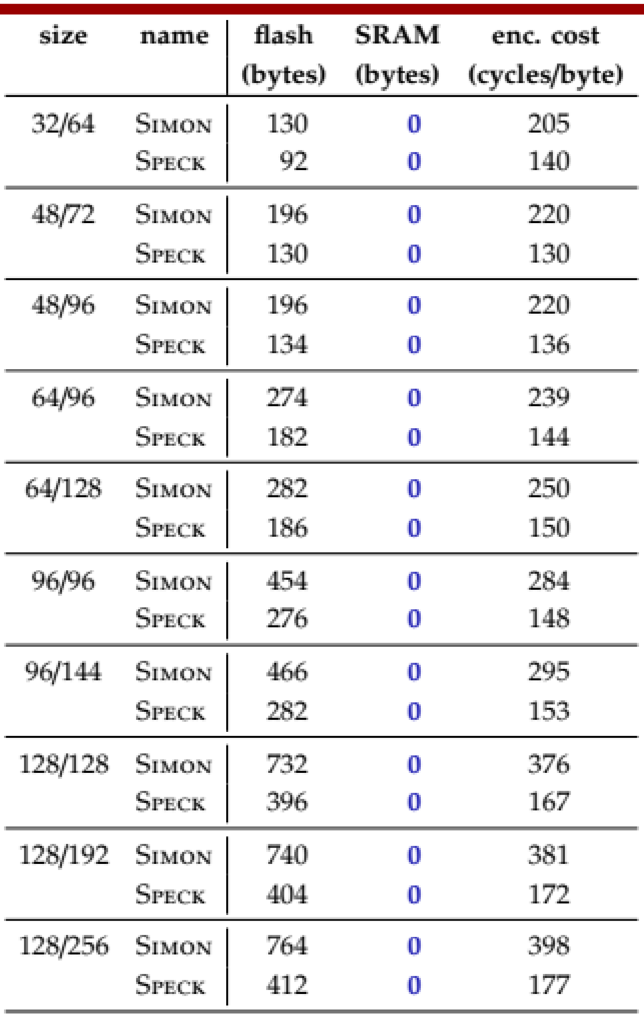
algorithm	area (GE)	throughput (kbps)	algorithm	area (GE)	throughput (kbps)
Simon32/64	523	5.6	Speck32/64	580	4.2
	535	11.1		642	8.3
	566	22.2		708	16.7
	627	44.4		822	33.3
	722	88.9		850	123.1
Simon48/72	631	5.1	Speck48/72	693	4.3
	639	10.3		752	8.5
	648	15.4		777	12.8
	662	20.5		821	17.0
	683	30.8		848	25.5
	714	41.0		963	34.0
	765	61.5		1040	51.1
	918	123.1		1152	192.0

Space/throughput trade off

SIMON/SPECK Software (8 bits devices)



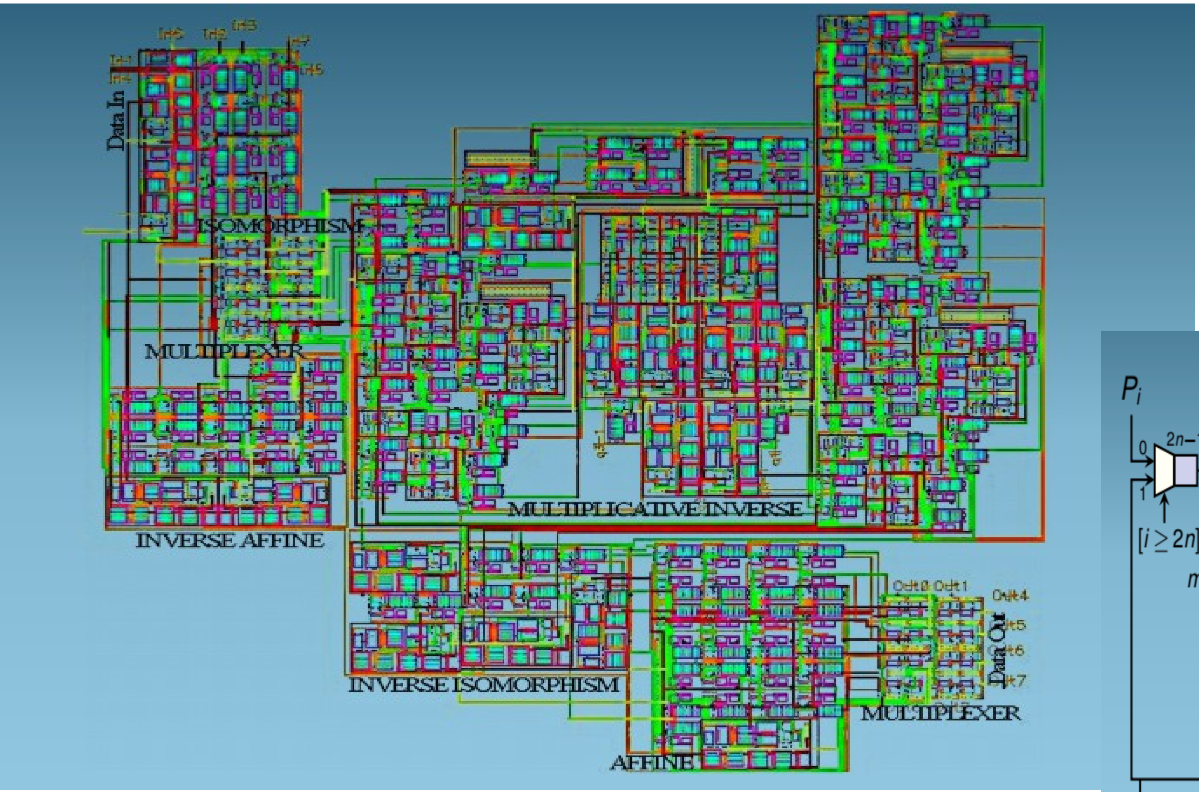
size	name	flash (bytes)	SRAM (bytes)	enc. cost (cycles/byte)
32/64	Simon	384	64	**168**
	Speck	424	44	**110**
48/72	Simon	430	108	**187**
	Speck	532	66	**100**
48/96	Simon	442	108	**187**
	Speck	562	69	**104**
64/96	Simon	530	168	**205**
	Speck	556	104	**114**
64/128	Simon	404	176	**217**
	Speck	596	108	**118**
96/96	Simon	544	312	**249**
	Speck	454	168	**123**
96/144	Simon	444	324	**260**
	Speck	576	174	**127**
128/128	Simon	446	544	**333**
	Speck	388	256	**139**
128/192	Simon	582	552	**335**
	Speck	568	272	**143**
128/256	Simon	458	576	**353**
	Speck	458	288	**147**



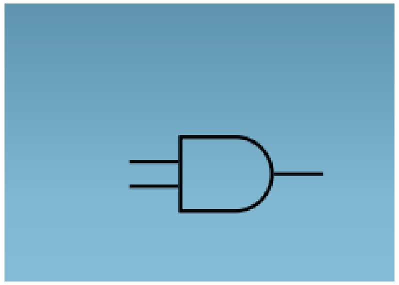
size	name	flash (bytes)	SRAM (bytes)	enc. cost (cycles/byte)
32/64	Simon	130	**0**	205
	Speck	92	**0**	140
48/72	Simon	196	**0**	220
	Speck	130	**0**	130
48/96	Simon	196	**0**	220
	Speck	134	**0**	136
64/96	Simon	274	**0**	239
	Speck	182	**0**	144
64/128	Simon	282	**0**	250
	Speck	186	**0**	150
96/96	Simon	454	**0**	284
	Speck	276	**0**	148
96/144	Simon	466	**0**	295
	Speck	282	**0**	153
128/128	Simon	732	**0**	376
	Speck	396	**0**	167
128/192	Simon	740	**0**	381
	Speck	404	**0**	172
128/256	Simon	764	**0**	398
	Speck	412	**0**	177

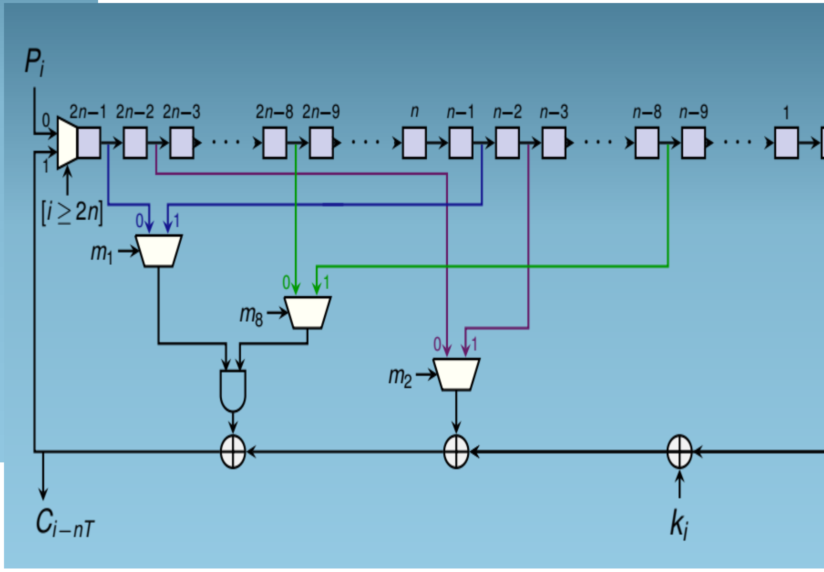
SIMON/SPECK Comparison

AES



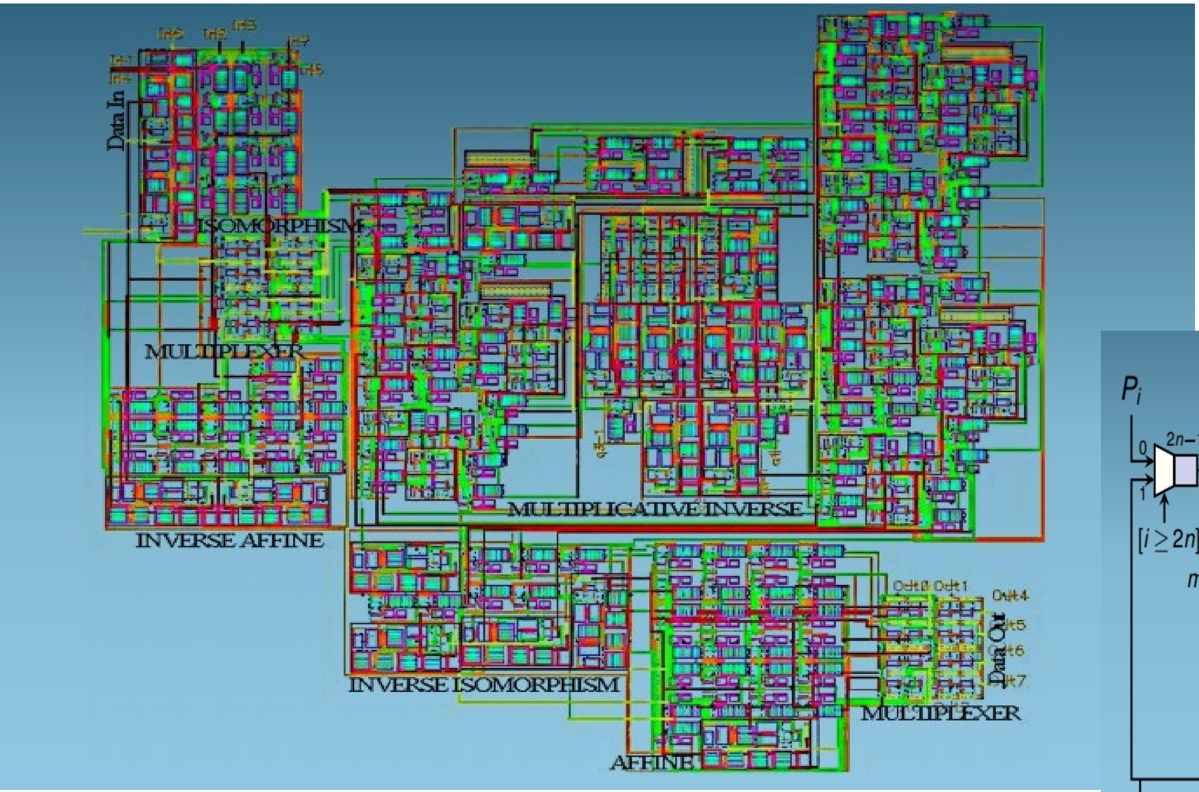
Simon



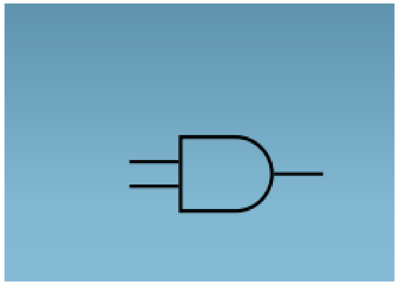


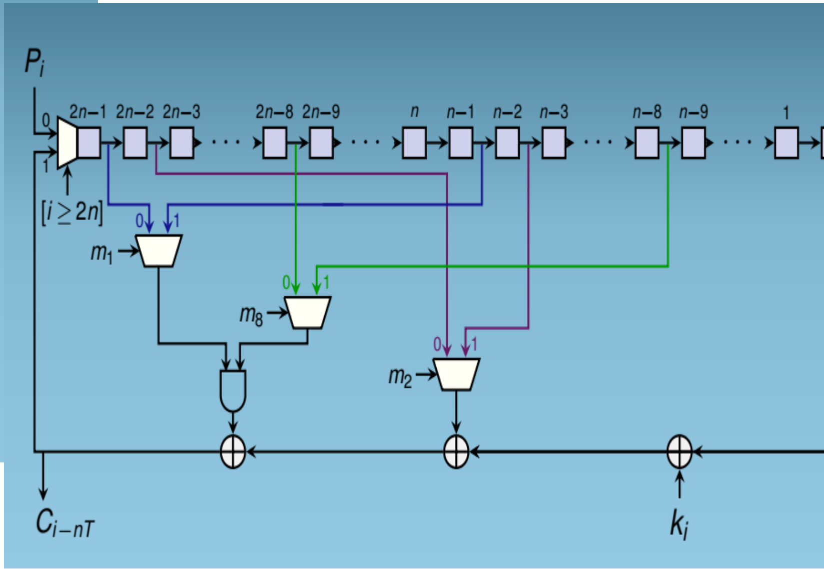
SIMON/SPECK Comparison

AES

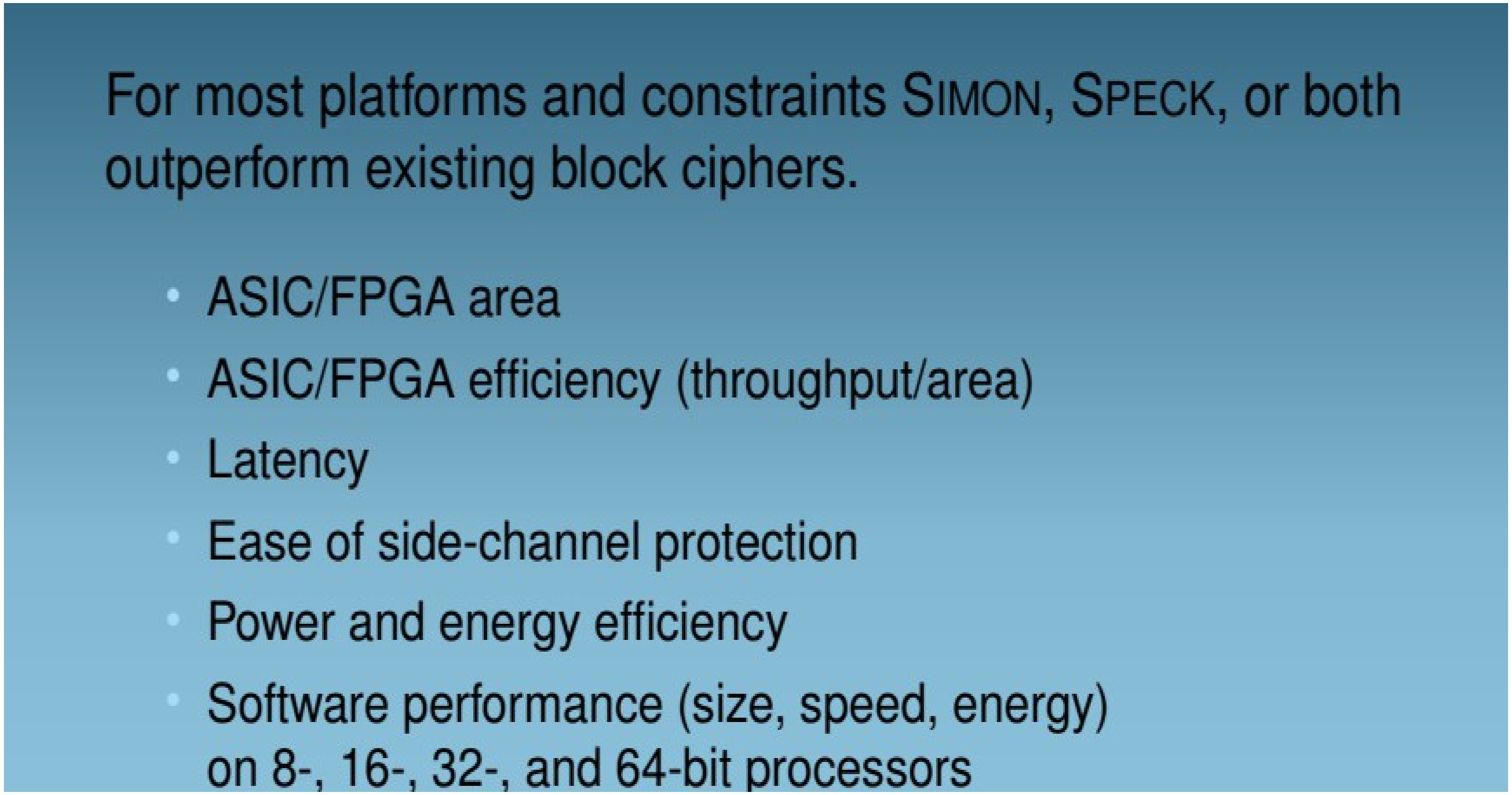


Simon




P_i
$2n-1$ $2n-2$ $2n-3$ $2n-8$ $2n-9$ n $n-1$ $n-2$ $n-3$ $n-8$ $n-9$ 1
$[i \geq 2n]$
m_1
m_8
m_2
C_{i-nT}
k_i

SIMON/SPECK Comparison



For most platforms and constraints SIMON, SPECK, or both outperform existing block ciphers.

- ASIC/FPGA area
- ASIC/FPGA efficiency (throughput/area)
- Latency
- Ease of side-channel protection
- Power and energy efficiency
- Software performance (size, speed, energy) on 8-, 16-, 32-, and 64-bit processors

SIMON/SPECK – The end (???) of the story

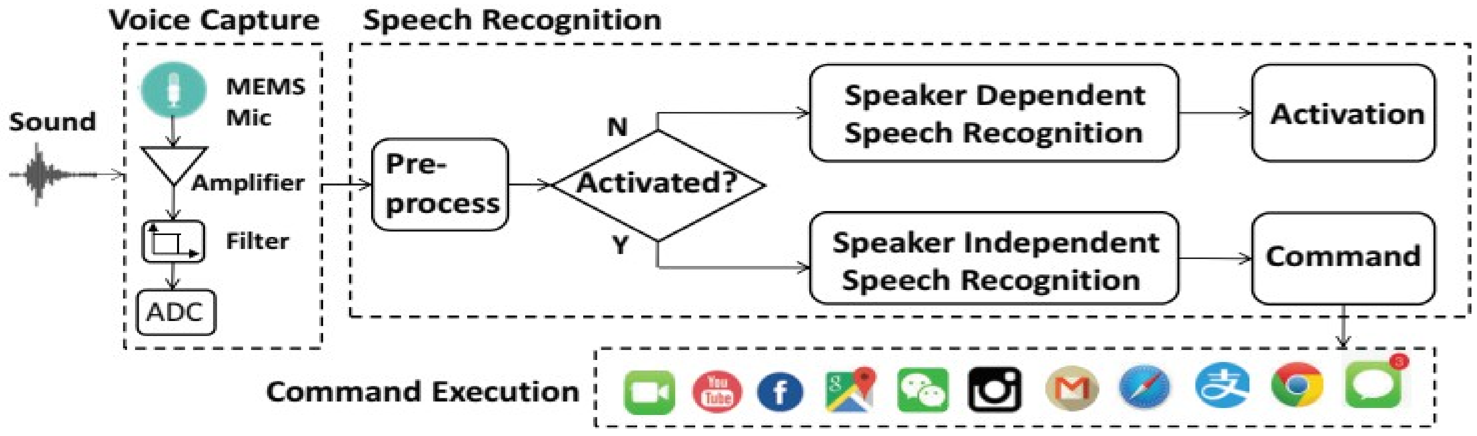
- On April 24, ISO delegates met behind closed doors in Wuhan, China, and voted to end a program to adopt two forms of encryption championed by the NSA. The plan had already been reduced in 2017 due to delegates' suspicions towards the agency.
- The NSA has a track record of trying to install vulnerabilities, or backdoors, into security tools, including forms of encryption. This dispute over the Simon and Speck algorithms – to be included in household objects such as smart speakers, fridges, lighting and heating systems – showed NSA still lacks the trust of many countries, including U.S. allies.
- In response to inquiries, NSA Capabilities Technical Director Neal Ziring said: "Both Simon and Speck were subjected to several years of detailed cryptanalytic analysis within NSA, and have been subject to academic analysis by researchers worldwide since 2014. They are good block ciphers with solid security and excellent power and space characteristics."

IoT : A new perspective on attacks

- The interaction of an IoT device with the physical world makes it possible to exploit some physical properties to attack an ICT system in new and unexpected ways
- An interesting example are attacks against a voice interface using frequency that the interface can hear but the human user cannot hear
- Hidden voice commands result in the design of a completely inaudible attack, the DolphinAttack, that achieves inaudibility by modulating voice commands on ultrasonic carriers > 20 kHz
- The attack is effective on popular speech recognition systems, e.g. Siri, Google, Samsung, Huawei, Cortana and Alexa. Proof-of-concept attacks include activating Siri to initiate a FaceTime call, activating Google Now to switch the phone to the airplane mode, and manipulating the navigation system in an Audi automobile

IoT : A new perspective on attacks

Architecture of a Voice Capture System



- a speaker-dependent SR is typically performed locally
- a speaker-independent SR is performed via a cloud service [28].

To use the cloud service, the processed signals are sent to the servers, which will extract features

IoT : A new perspective – Threat Model

- **No Target Device Access.** : The attacker is fully aware of the characteristics of the target devices.
- **No Owner Interaction.** We assume that the target devices may be in the owner's vicinity, but may not be in use and draw no attention. The device may be unattended, may be stolen, and an adversary may try every possible method to unlock the screen. Nevertheless, the adversaries cannot ask owners to perform any operation, such as pressing a button or unlocking the screen.
- **Inaudible.** Since the goal of an adversary is to inject voice commands without being detected, she will use the sounds inaudible to human, i.e., ultrasounds ($f > 20$ kHz). Attack cannot use frequencies in the range 18 kHz .. 20 kHz that are still audible to kids
- **Attacking Equipment.** Adversaries can acquire both the speakers designed for transmitting ultrasound and commodity devices for playing audible sounds. An attacking speaker is in the vicinity of the target devices. For instance, she may secretly leave a remote controllable speaker around the victim's desk or home. Alternatively, she may be carrying a portable speaker while walking by the victim

IoT : A new perspective – Threat Device

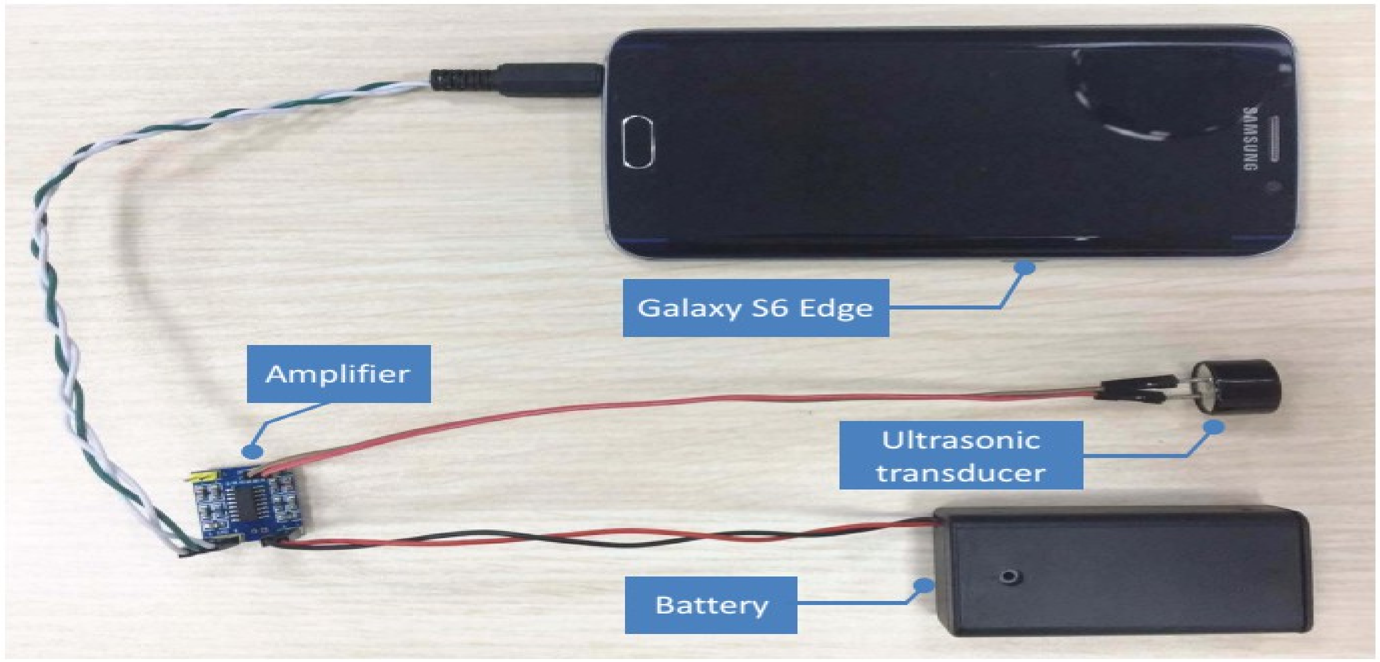
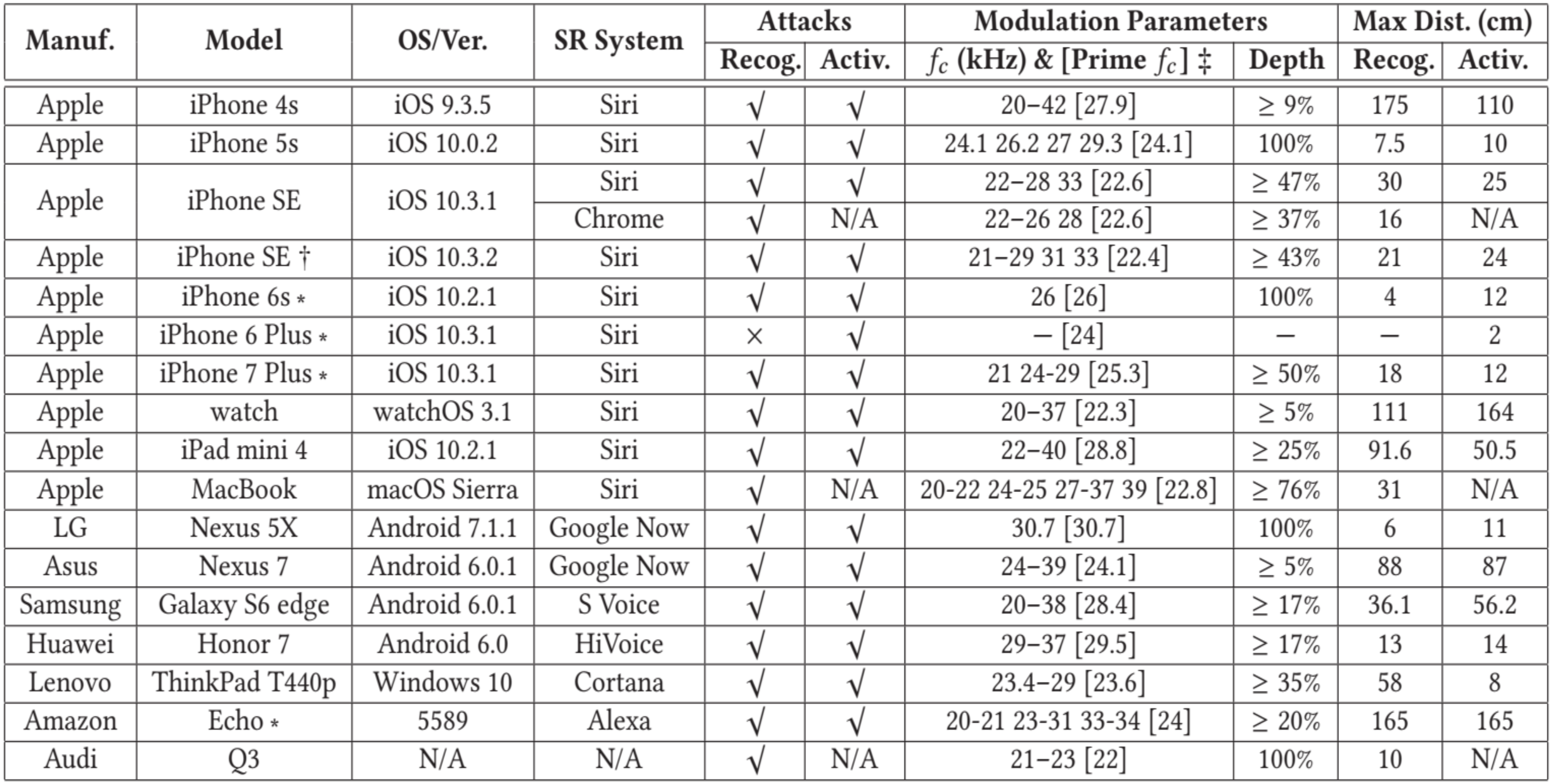


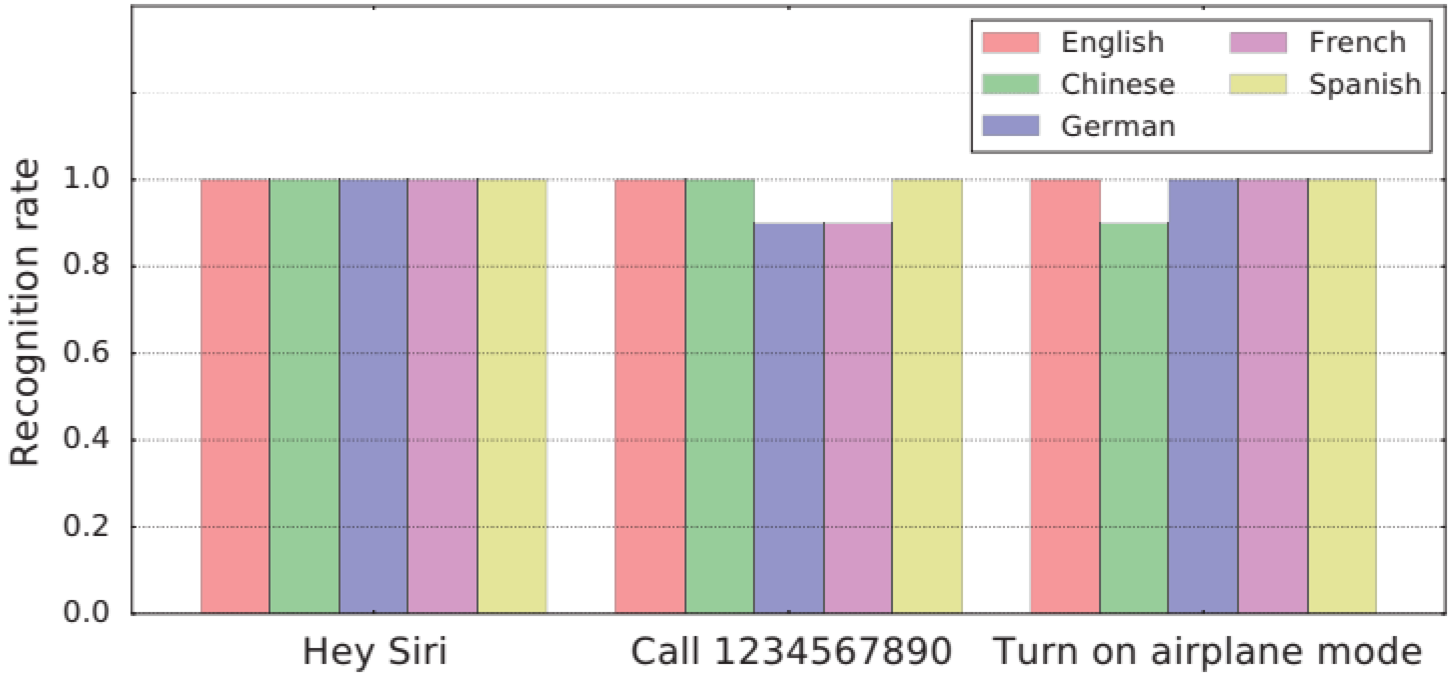
Figure 11: Portable attack implementation with a Samsung Galaxy S6 Edge smartphone, an ultrasonic transducer and a low-cost amplifier. The total price for the amplifier, the ultrasonic transducer plus the battery is less than $3.

IoT : A new perspective – Results



Manuf.	Model	OS/Ver.	SR System	Attacks		Modulation Parameters		Max Dist. (cm)	
				Recog.	Activ.	f_c (kHz) & [Prime f_c] ‡	Depth	Recog.	Activ.
Apple	iPhone 4s	iOS 9.3.5	Siri	√	√	20–42 [27.9]	≥ 9%	175	110
Apple	iPhone 5s	iOS 10.0.2	Siri	√	√	24.1 26.2 27 29.3 [24.1]	100%	7.5	10
Apple	iPhone SE	iOS 10.3.1	Siri	√	√	22–28 33 [22.6]	≥ 47%	30	25
			Chrome	√	N/A	22–26 28 [22.6]	≥ 37%	16	N/A
Apple	iPhone SE †	iOS 10.3.2	Siri	√	√	21–29 31 33 [22.4]	≥ 43%	21	24
Apple	iPhone 6s ⋆	iOS 10.2.1	Siri	√	√	26 [26]	100%	4	12
Apple	iPhone 6 Plus ⋆	iOS 10.3.1	Siri	×	√	— [24]	—	—	2
Apple	iPhone 7 Plus ⋆	iOS 10.3.1	Siri	√	√	21 24-29 [25.3]	≥ 50%	18	12
Apple	watch	watchOS 3.1	Siri	√	√	20–37 [22.3]	≥ 5%	111	164
Apple	iPad mini 4	iOS 10.2.1	Siri	√	√	22–40 [28.8]	≥ 25%	91.6	50.5
Apple	MacBook	macOS Sierra	Siri	√	N/A	20-22 24-25 27-37 39 [22.8]	≥ 76%	31	N/A
LG	Nexus 5X	Android 7.1.1	Google Now	√	√	30.7 [30.7]	100%	6	11
Asus	Nexus 7	Android 6.0.1	Google Now	√	√	24–39 [24.1]	≥ 5%	88	87
Samsung	Galaxy S6 edge	Android 6.0.1	S Voice	√	√	20–38 [28.4]	≥ 17%	36.1	56.2
Huawei	Honor 7	Android 6.0	HiVoice	√	√	29–37 [29.5]	≥ 17%	13	14
Lenovo	ThinkPad T440p	Windows 10	Cortana	√	√	23.4–29 [23.6]	≥ 35%	58	8
Amazon	Echo ⋆	5589	Alexa	√	√	20-21 23-31 33-34 [24]	≥ 20%	165	165
Audi	Q3	N/A	N/A	√	N/A	21–23 [22]	100%	10	N/A

IoT : A new perspective – Results - II



IoT : A new perspective – Countermeasures

- Hardware Based
  - Microphone Enhancements (only frequencies lower than 20Mhz)
    - Iphone 6 plus
  - Inaudible Voice Command Cancellation

- Software Based
  - Classification of the signal to discover modulated voice. Implemented through a classification system
    - considers 15 features
    - no false positive
    - no false negative

A general definition

- A new taxonomy on ttacks on IoT devices based on how the attacker deviates feature from their “official” functionality.
- Almost all the attack ideas published so far can be clustered into four broad types of attacking behavior:
  - Ignoring the functionality
  - Reducing the functionality
  - Misusing the functionality
  - Extending the functionality

Ignoring the functionality

- The attacker ignores the intended physical functionality of the IoT device, and considers it only as a standard computing device which is connected to the LAN and/or to the internet.
- For example, the attacker combine many compromised IoT devices into a botnet which can be used to send spam or to mine bitcoins.
- Alternatively, it may penetrate the victim's home network and infect his computers by exploiting the IoT devices
- IoT devices are the best attack vectors since there are going to be many cheap devices made by a variety of small companies with minimal security protections, and it will probably be impossible to upgrade or patch their discovered vulnerabilities.
- These attacks are a serious security threat but they are the least interesting ones because they are applicable to essentially any networked computing device and are not unique to IoT devices.

Reduce the functionality

- The attacker tries to kill or limit the designed functionality of the IoT device:
  - the TV will stop working,
  - the refrigerator will not cool its contents,
  - the lights will not turn on, etc.
- This can be done in order to annoy an individual or organization, to inflict financial loss, or to create large scale chaos and panic.
- In some cases the consequences of lost functionality can be more serious: For example, in internet-connected medical devices such attacks can be fatal.
- The broader scope of IoT devices opens up interesting new opportunities. In particular, the attacker can use ransomware to temporarily lock an expensive physical device and demand a large payment to restore its functionality.

Misusing the functionality

- This kind of attack uses rather than destroys the designed functionality of the physical device, but does it in an incorrect or an unauthorized way.
  - a climate control device is supposed to cool the house in the summer and to heat it in the winter, but the attacker reverses this behaviour and cause discomfort.
  - the attacker can turn on all the lights and open all the faucets as soon as the user leaves home for a long vacation.
- However, most of these attacks are likely to be irritating pranks rather than serious problems.

Extending the functionality

- The attacker extends the designed functionality of the IoT device, and uses it in order to achieve a completely different and unexpected physical effect.
- This requires more imagination and sophistication, since it is not clear how a smart air conditioner can start a fire, or how an internet-connected Roomba can unlock the front door. Such unexpected effects are not easy to achieve.

- We will explore some unexpected applications of connected LEDs, and use them to demonstrate such functionality extending attacks.

A covert channel - I

- To protect sensitive data, an organization separates its internal network from the Internet.
- This might be implemented a security gateway or even by employing a complete air gaped separation in top secret networks.
- Sophisticated attackers can try to infect those networks with malware using one time access (either physical or virtual). However, it is much harder to find a reliable and continuous exfiltration channel to leak out sensitive data.
- Let us assume that such an organization chooses to implement a smart connected light solution, and connects it to its internal sensitive network. We describe how an attacker can use the connected LEDs to create a covert channel.

A covert channel - II

- The basic idea is to misuse the LED's API to switch back and forth between two light intensities, under the following seemingly contradictory conditions:
    - The two light intensities should bead close enough to make the transition imperceptible to the human eye, but robustly distinguishable by a light sensor.
    - This is made possible by the fact that most LED lights adjust their luminosity by rapidly switching between on and off states and adjusting the duty cycle.
    - Since the sensor can easily distinguish between such extreme states and accurately measure the duty cycle, it can robustly measure small changes in the light intensity even in the presence of other lights, air turbulence, and other sources of noise

A covert channel – The receiver - I

- To detect slight changes in light intensity at interval as small as 10 microseconds requires a sampling rate of around 100 Khz i.
- A smartphone's light sensor is ruled out as the sensors available on standard smartphones could only measure in the millisecond range.
- A custom device to implement the attack is required
- Most common light sensors are photo-resistors or LDRs (light-dependent resistor). The light is measured by reading the voltage divided between the LDR and a serially connected resistor. However this type of sensors has a latency of up to 10 milliseconds which is not fast.
- The attack requires a color light-to-frequency converter. This sensor converts light intensity to digital frequency output, that for high light intensity can go up to 500 KHz. In addition, It has a very low latency (around 100 nanoseconds).

A covert channel – The receiver - II

- To decode out leak channel we require the ability to measure and collect the light sensor's frequency output at a very high rate and for long periods of time.
- As this option is only available in high end and expensive scopes, the low level drivers of a 16 MHz CPU Arduino board can sample the light sensor at a high rate.
- The Arduino comes with a hardware counter to count the number of rising or falling edges of the sensor's output.
- This hardware counter is sampled at 10 micro second intervals (100Khz sampling rate), and the output is sent it to a laptop (using 160 CPU cycles to measure and send).
- The post-processing of the data is done on the laptop.

A covert channel – The receiver - III

- To decode the leaked data from a long distance a 12in Meade LX200 amateur's telescope is used with a dual purpose:
  - Reducing outside light noise by focusing only on the LED.
  - Focusing the flickering light on the small sensor to increase the light intensity measured.
- The telescope is directed towards the LED
- A preliminary test of the leak channel considers a long corridor from a distance of over 50 meters, with lots of other light sources along the corridor.
- In later measurements conducted outside the building ranges of over 100 meters have been achieved

A covert channel – The attack - I

- Two connected lighting systems have been tested:
  - Philips Lux - a high end white lighting solution by Philips' lighting department. This light system and the color version called Hue are the most prominent players in the market. A starter kit containing 9 Watt 750 Lumens light bulbs (model LWB004 software version 66012040) and a controller (model PHDLOO software version 1.0) was used.
  - LimitlessLED - A cheap lighting solution that started as a Kickstarter project. It sells the same products under different brand names (we used the one called MiLight). 6 Watt 400 Lumens light bulb with version 3 controller have been used.

A covert channel – The attack -II

- Most of today's systems have a dedicated controller that acts as the gateway between the Internet or LAN and the lights. The controller uses an RF transmitter to send commands to the lights (and sometimes a receiver to receive the light's status).
- It also has a control interface that is connected to the LAN, and sometimes to the Internet or to a cloud service. Most types of controller can control a number of different lights, and different groups of light, allowing the user to use one controller to separately control the lights in different rooms in the house.
- The controller usually enforces restrictions on the rate of commands sent in the system. This might be due to bottleneck issues or as a type of safety method to prevent intended or unintended misuse of the light system.

A covert channel – The attack -III

- The human ability to perceive changes in brightness or color is very complex. Vast empirical research had been conducted in this field. In recent years this research was used to design flicker free LED lighting with smooth colors.
- One might assume humans can't detect flickers at a rate above the 24 Hz used in movies and televisions. However, depending on the intensity and color, people can in fact detect flickers at 60 Hz [10], and in some cases of fast movement, a phenomena called phantom array may be perceived even at 200 Hz [11].
- To be sure that the covert channel will not be detected it flicked at over 60 Hz (over this frequency, the flickering will be fused and seen as reduced brightness).
- The other option is to flicker between two close levels of brightness, at the top of the brightness range.

A covert channel – The attack -IV

- LimitlessLED has only 27 brightness levels (which in fact reduce to 24 as the 3 top levels were the same). Unfortunately the changes between adjacent brightness levels were in fact visible to a human observer.
- Philips Lux has 256 different brightness levels. To provide smooth light Lux works at a very high frequency of 20 KHz or 50 micro seconds. Assuming linear changes in duty cycle we will need to measure differences of around 200 nanoseconds in a period of time between adjacent brightness level. To achieve that requires a very responsive light sensors and high end measuring equipment.

A covert channel – Hacking the controller

- To setup the controller as part of the WIFI network, the user should send the controller his WIFI password. This is done as follows
  - Upon first power on or after factory reset, the controller boots up as a WIFI hotspot of an un-encrypted network.
  - The user is requested to join this network with his smartphone.
  - Using the provided Android or iOS application, the user choose the WIFI network the controller should join, and sends the required password unencrypted (!) over the controller's WIFI.
- After rebooting, the controller joins the user's chosen encrypted WIFI network using the password it received.
- An adversary that can sniff the WIFI communication during the setup process will acquire the user's secret WIFI password and gain access.

A covert channel – Summing up

- The covert channels are created with the LimitlessLED and the Philips Lux connected LEDs.
- The portable experimental set up can accurately detect two different symbols from the Philips Lux light from a range of over 100 meters.
- This building block can be used to covertly leak several bits per second, while using the optical receiver at a safe distance from the target's apartment or office.
- The covert channel can leak data even from airgapped and Tempest protected networks, with no wireless connections (some connected products can be controlled by wired connections).
- As lighting in offices is turned on most hours of the day, the slow channel can be used to leak more then 10KB per day. That is enough bandwidth to leak private encryption keys and passwords.

Countermeasure for data injection attacks

- Some countermeasures have been devised to detect attacks that inject some fake data into an IoT systems
- These countermeasure exploit the existence of a physical process of interest outside the IoT and uses the rules of this process to
- The countermeasures may be classified into
  - Anomaly detection
  - Trust management
- While these countermeasures enable the discovery of data injection they are not effective against confidentiality attacks that aim to leak to a third party the information the sensors have collected because this attack is unrelated to the physical process of interest

Anomaly detection - I

- If a sensor injects some fake data the original measurements substituted with fabricated ones cannot be observed directly
- Anomaly detection requires they are to be characterised indirectly with related information.
- The relationship between two pieces of information is a *correlation*,
- It can be calculated online, with historical data, or modelled a-priori.
- Here *correlation* is intended iin a broad sense, meaning that there is some kind of continuous dependency, as opposed to Pearson's correlation coefficient $PXY = E[(X - \mu X)(Y - \mu Y)]/\sigma X \sigma Y$

  were E, μ and σ are the expected value, the mean and the standard deviation respectively,
- This coefficient measures only linear dependency

Anomaly detection - II

We consider correlations across three different domains:

- **Temporal correlation** is the dependency of a sensor's reading on its previous readings. It models the coherence in time of the sensed physical process.
- **Spatial correlation** is the dependency in readings from different sensors at the same time. It models the similarities in how the sensed phenomenon is perceived by different sensors.
- **Attribute correlation** is the dependency in readings that are related to different physical processes. It models physical dependencies among heterogeneous physical quantities such as temperature and relative humidity.

- Usually a combination of these different kinds of correlation is used.

Anomaly Detection: Temporal Correlation

- Variations in time of the sensed data can be modelled as a random process where the random variables at different time are correlated
- The variation of a sensor's measurements in time depends on both the variations introduced by the physical attribute and the measurements' error.
- The variation of the physical attribute in time is subject to constraints, since the phenomenon observed usually follows the laws of physics.
- If the measurements are gathered with sufficiently high frequency, consecutive measurements would be subject to similar constraints.
- This justifies a procedure that identifies errors (including malicious injections) when temporal variations do not respect these constraints
- Two main difficulties in applying this observation to assess deviations:
  - the time evolution of the process is subject to uncertainty factors
  - the measurements are subject to noise.

Anomaly Detection: Spatial Correlation -I

- Sudden events can rapidly change the dynamics of a physical process
- Often detecting such events, such as a forest fire, a volcanic eruption, a cardiac attack is the very purpose of the Iot.
- These events may disrupt temporal correlations, giving rise to false anomalies.
- False anomalies may be detected because distinct different sensor nodes are affected by the event and they produce measurements that are spatially correlated to the event source
- This is known as spatial correlation: the measurements of different sensors are correlated during the manifestation of the event
- The most widespread and simplest spatial correlation model is the *spatially homogeneous:* all sensors would produce the same measurements in the absence of errors and noise i.e., they measure the same value

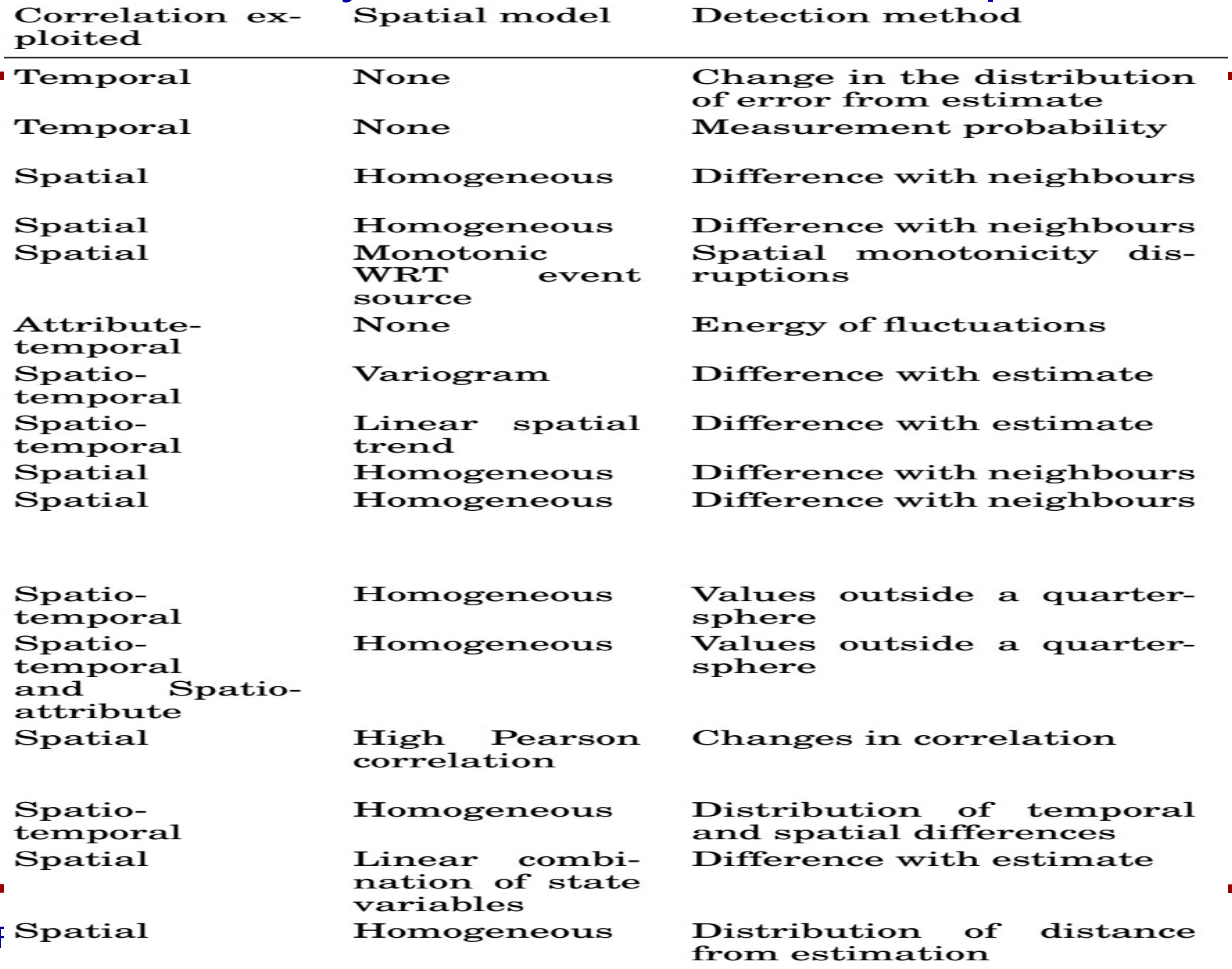
Anomaly Detection: Spatial Correlation -II

- The homogeneous model is suitable only for small space regions free of obstacles.
- When the deployment topology and characteristics of the physical phenomena violate the homogeneity assumption, a monotonic spatial correlation may hold
- The values of the physical attribute at a point in space, should either increase or decrease as the distance from that point increases.
- For example, when monitoring for forest fires the temperature decreases monotonically as the distance from the fire increases
- Other solutions
  - the correlation depends upon the relative distances among the sensors
  - a function that computes the expected output of a sensor in terms of the outputs of other sensors is applied
  - Integrate time and space correlation

Anomaly Detection: Attribute Correlation

- Sensors coexist to observe different physical attributes such as light, vibrations, temperature etc. Some of these attributes may be correlated because of the physical relationship between them e.g., temperature and humidity.
- Commonly, at every deployment location, different sensors measuring different physical processes are connected to a single sensor node. We expect attribute correlations to be observable in the measurements the sensor nodes report.
- However, attribute correlations between sensors belonging to the same node are not informative as an attacker who has compromised a node may tamper with all the measurements collected on that node.
- Attribute based expectations are very useful in conjunction with spatial correlations, when spatial redundancy is limited. For example, health care body networks have limited redundancy since it is impractical to cover the patient with several sensors. We can still exploit correlation among different physiological values (the attributes) measured by different sensor nodes.

Anomaly Detection. Some examples



Correlation exploited	Spatial model	Detection method
Temporal	None	Change in the distribution of error from estimate
Temporal	None	Measurement probability
Spatial	Homogeneous	Difference with neighbours
Spatial	Homogeneous	Difference with neighbours
Spatial	Monotonic WRT event source	Spatial monotonicity disruptions
Attribute-temporal	None	Energy of fluctuations
Spatio-temporal	Variogram	Difference with estimate
Spatio-temporal	Linear spatial trend	Difference with estimate
Spatial	Homogeneous	Difference with neighbours
Spatial	Homogeneous	Difference with neighbours
Spatio-temporal	Homogeneous	Values outside a quarter-sphere
Spatio-temporal and Spatio-attribute	Homogeneous	Values outside a quarter-sphere
Spatial	High Pearson correlation	Changes in correlation
Spatio-temporal	Homogeneous	Distribution of temporal and spatial differences
Spatial	Linear combination of state variables	Difference with estimate
Spatial	Homogeneous	Distribution of distance from estimation

Anomaly Detection: Correlation

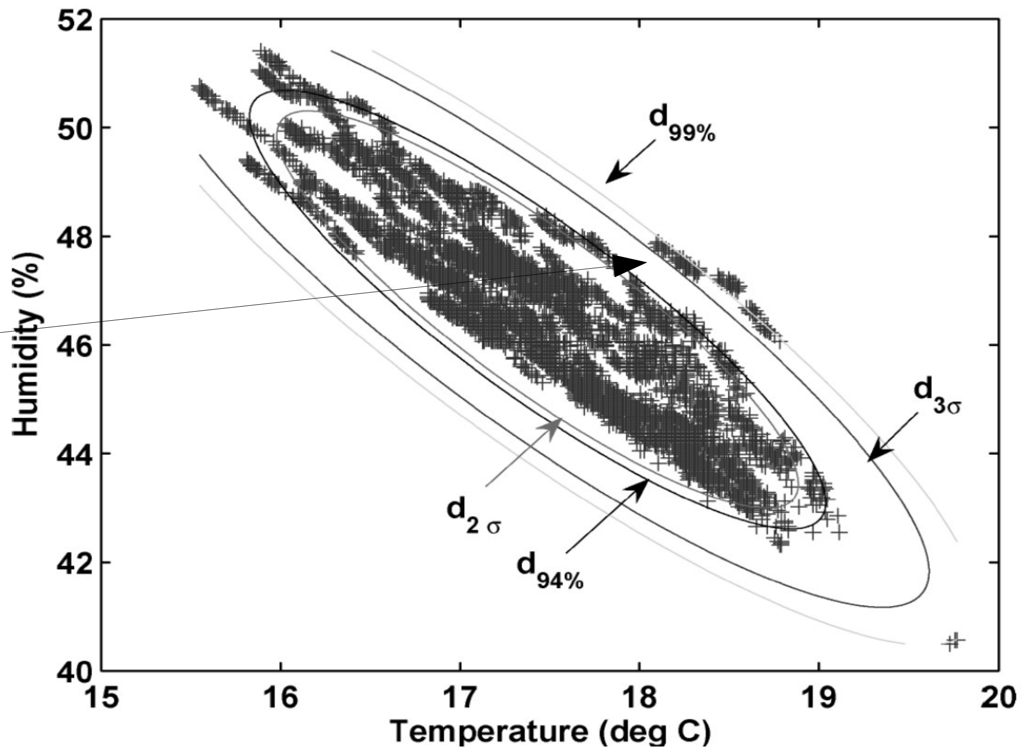
Correlation Type	Information Captured	Variations
Temporal	corr($\varphi^a(s,t_1),\varphi^a(s,t_2)$)	— Time-series evolution model — Time memory (the maximum value of W for which the correlation is modelled)
Spatial	corr($\varphi^a(s_1,t),\varphi^a(s_2,t)$)	— Spatial model, e.g. homogeneous, monotonic, variogram, linear dependency — Correlation variational model, e.g. distance-dependent, sensors-dependent, fixed — Neighbourhood selection criterion
Attribute	corr($\varphi^{a_1}(s,t),\varphi^{a_2}(s,t)$)	— Correlation extraction process, e.g. from physical laws, temporal/spatial analysis etc.

Anomaly Detection: Outlier detection

- Outlier detection methods consider as anomalous data that lies outside of the space where most data samples lie. This technique can identify malicious data injections reasonably effectively as long as maliciously injected values are a minority in the dataset and deviate significantly from the other data.
- Outlier detection has been proposed sometimes with opposing goals: in some cases the techniques aim to filter out outliers, in others the outliers represent the main interest.
- Possible methods include
  - Nearest neighbours: how many neighbour a value has
  - Clustering
  - Principal Component Analysis
  - Statistical based

Anomaly Detection: Outlier detection

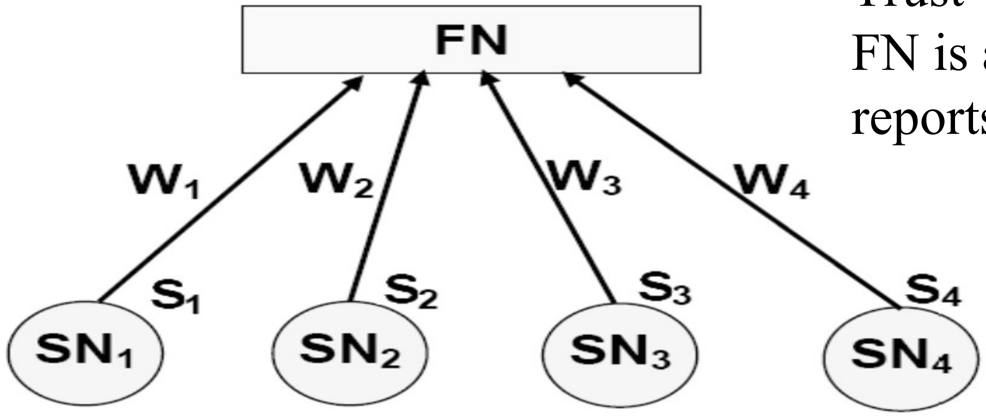
Attribute based Outlier



Trust Based

- Trust-management considers the trustworthiness between two classes of entities: a trustor and a trustee. The trustor assigns each trustee a trustworthiness value, based on how much the trustee's behaviour matches an expectation.
- Trustworthiness values are usually in the range [0; 1], decreasing when the trustee exhibits deviations from the expected behaviour and increasing when the a behaviour matches the expectation.
- Trust-management can reduce the influence of compromised nodes that inject malicious data because if the expected behaviour accurately characterises genuine nodes, compromised nodes would be assigned a low trustworthiness when deviating from it.
- Trust values are a continuous metric defined inside an interval so there is no direct classification of compromised and genuine nodes. Instead, the trust values are used to apply a penalisation proportional to the confidence that the sensor is compromised.

Trust Based



Trust-weighted aggregation
FN is a forwarding node, which collects reports from the sensor nodes SN,

$$W_n = \begin{cases} W_n - \theta r_n, & \text{if } S_n(t) \neq E \\ W_n, & \text{otherwise} \end{cases}$$

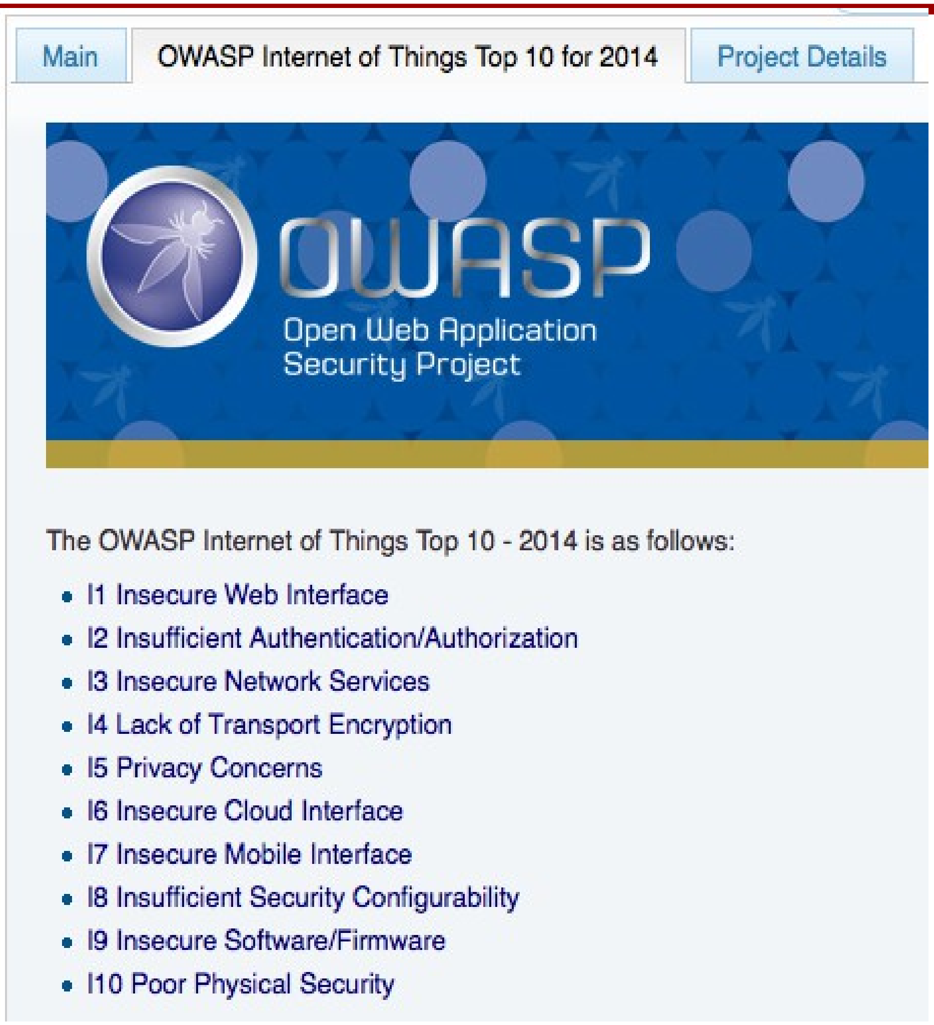
Trust Based

- r_n is the ratio of sensors giving different output over the total number of sensors
- θ is a penalty weight that determines a trade-off between the detection time and accuracy.
- the trustworthiness values, e.g. the weights, are calculated based on the measurements consistency with the aggregated value.
- The aggregate value is considered more reliable than the single readings, since sensors which exhibited inconsistent (e.g. malicious) readings in the past contribute less to the aggregation process.
- malicious nodes are detected by comparing the weights to a threshold
- the algorithm is vulnerable to the *on-off attack*: a node that performs well for a time period, acquires high trustworthiness, then suddenly starts malfunctioning

Trust Based

Correlation exploited	Spatial model	Detection method	Classes considered
Spatio-temporal	Homogeneous	Distance from mean of top-trust sensors	Malicious
Spatial	Homogeneous	Trust under a threshold	Malicious
Spatial	Homogeneous	Trust under a threshold	Faulty, Malicious
Spatial	Homogeneous	Difference with aggregated value	Faulty, Event
Spatio-Temporal	Heterogeneous	Difference with learnt pattern	Malicious
Spatial	Homogeneous	Trust under a threshold	Malicious, Event
Spatial	Homogeneous	Trust under a threshold	Malicious, Permanent Fault, Transient Fault, Event
Spatial	Homogeneous	Trust under a threshold	Malicious, Permanent Fault, Transient Fault, Event

OWASP - Internet of Things Top Ten Project


Main
OWASP Internet of Things Top 10 for 2014
Project Details
OWASP
Open Web Application Security Project

The OWASP Internet of Things Top 10 - 2014 is as follows:

- I1 Insecure Web Interface
- I2 Insufficient Authentication/Authorization
- I3 Insecure Network Services
- I4 Lack of Transport Encryption
- I5 Privacy Concerns
- I6 Insecure Cloud Interface
- I7 Insecure Mobile Interface
- I8 Insufficient Security Configurability
- I9 Insecure Software/Firmware
- I10 Poor Physical Security

Review all aspects of Internet of Things

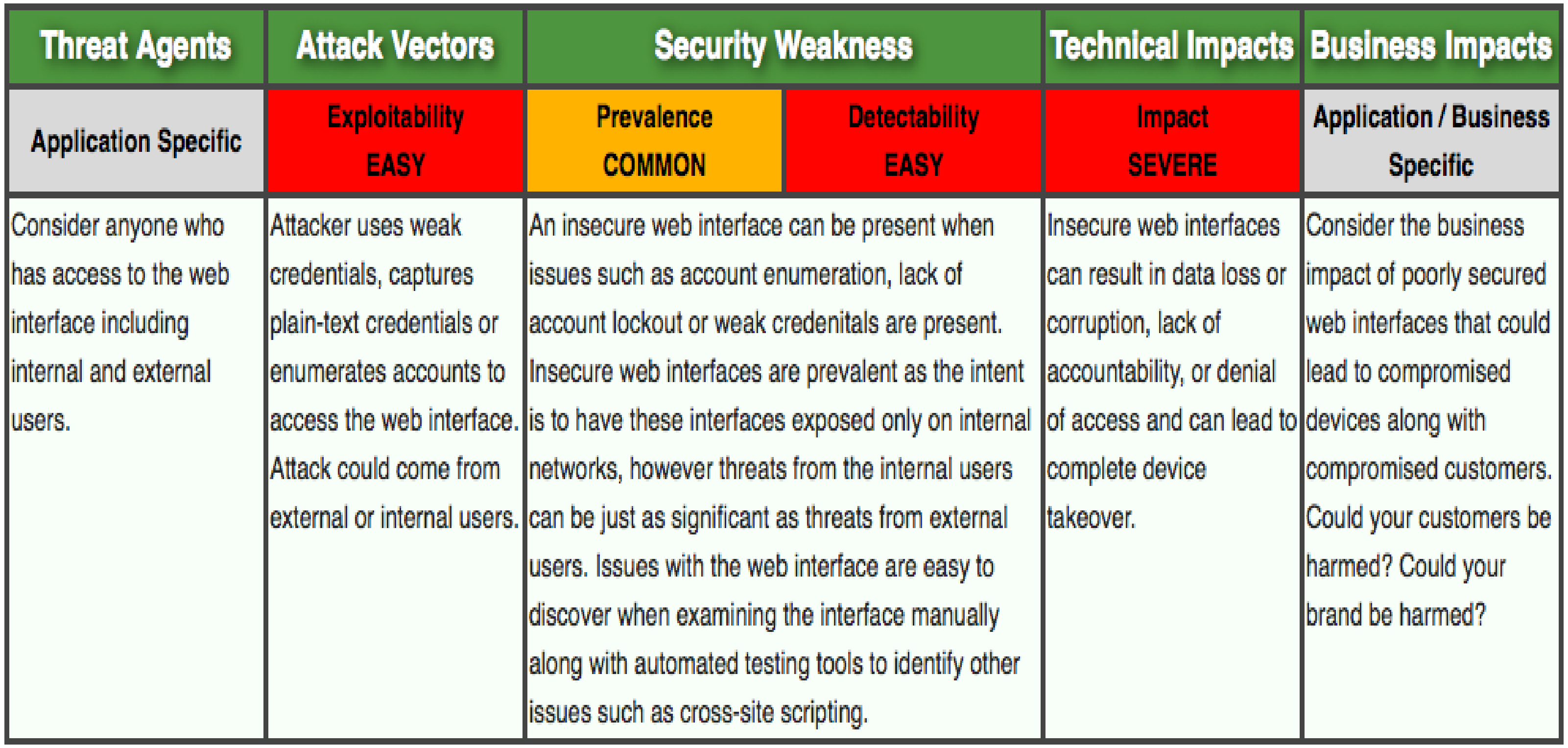
Top Ten Categories

Covers the entire device

Without comprehensive coverage like this it would be like getting your physical but only checking one arm

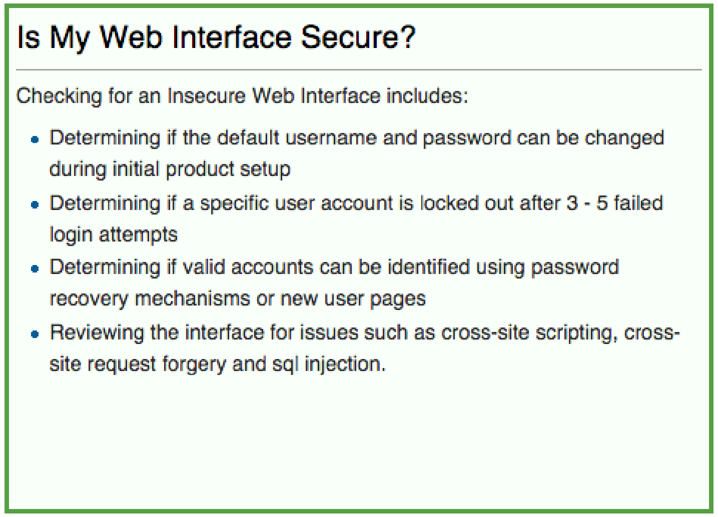
We must cover all surface area to get a good assessment of overall security

I1 | Insecure Web Interface



Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability EASY	Impact SEVERE	Application / Business Specific
Consider anyone who has access to the web interface including internal and external users.	Attacker uses weak credentials, captures plain-text credentials or enumerates accounts to access the web interface. Attack could come from external or internal users.	An insecure web interface can be present when issues such as account enumeration, lack of account lockout or weak credenitals are present. Insecure web interfaces are prevalent as the intent is to have these interfaces exposed only on internal networks, however threats from the internal users can be just as significant as threats from external users. Issues with the web interface are easy to discover when examining the interface manually along with automated testing tools to identify other issues such as cross-site scripting.		Insecure web interfaces can result in data loss or corruption, lack of accountability, or denial of access and can lead to complete device takeover.	Consider the business impact of poorly secured web interfaces that could lead to compromised devices along with compromised customers. Could your customers be harmed? Could your brand be harmed?

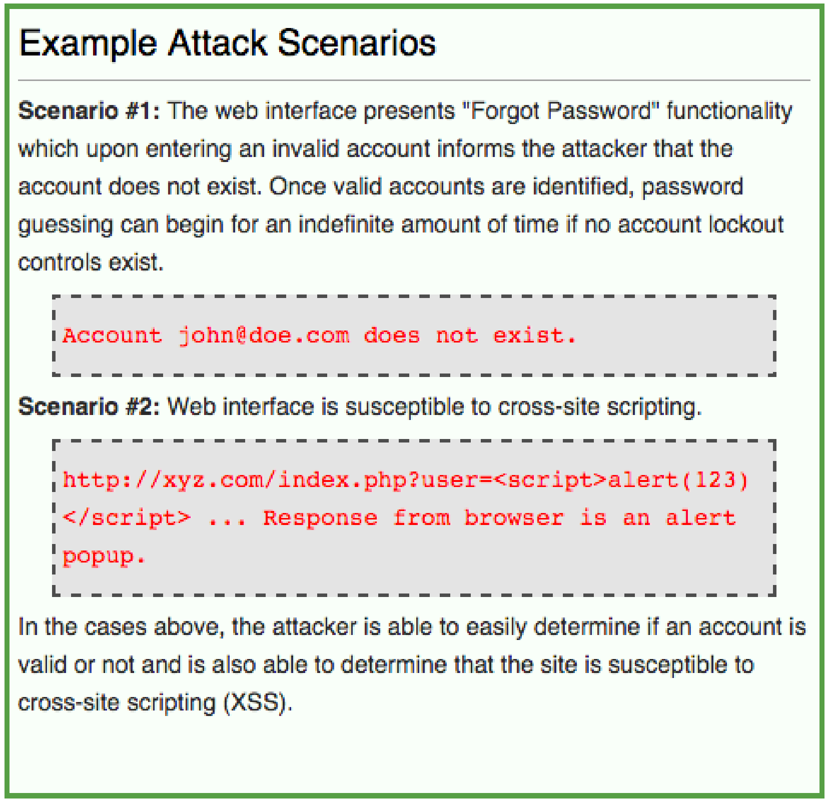
I1 | Insecure Web Interface | Testing



Is My Web Interface Secure?

Checking for an Insecure Web Interface includes:

- Determining if the default username and password can be changed during initial product setup
- Determining if a specific user account is locked out after 3 - 5 failed login attempts
- Determining if valid accounts can be identified using password recovery mechanisms or new user pages
- Reviewing the interface for issues such as cross-site scripting, cross-site request forgery and sql injection.



Example Attack Scenarios

Scenario #1: The web interface presents "Forgot Password" functionality which upon entering an invalid account informs the attacker that the account does not exist. Once valid accounts are identified, password guessing can begin for an indefinite amount of time if no account lockout controls exist.

```
Account john@doe.com does not exist.
```

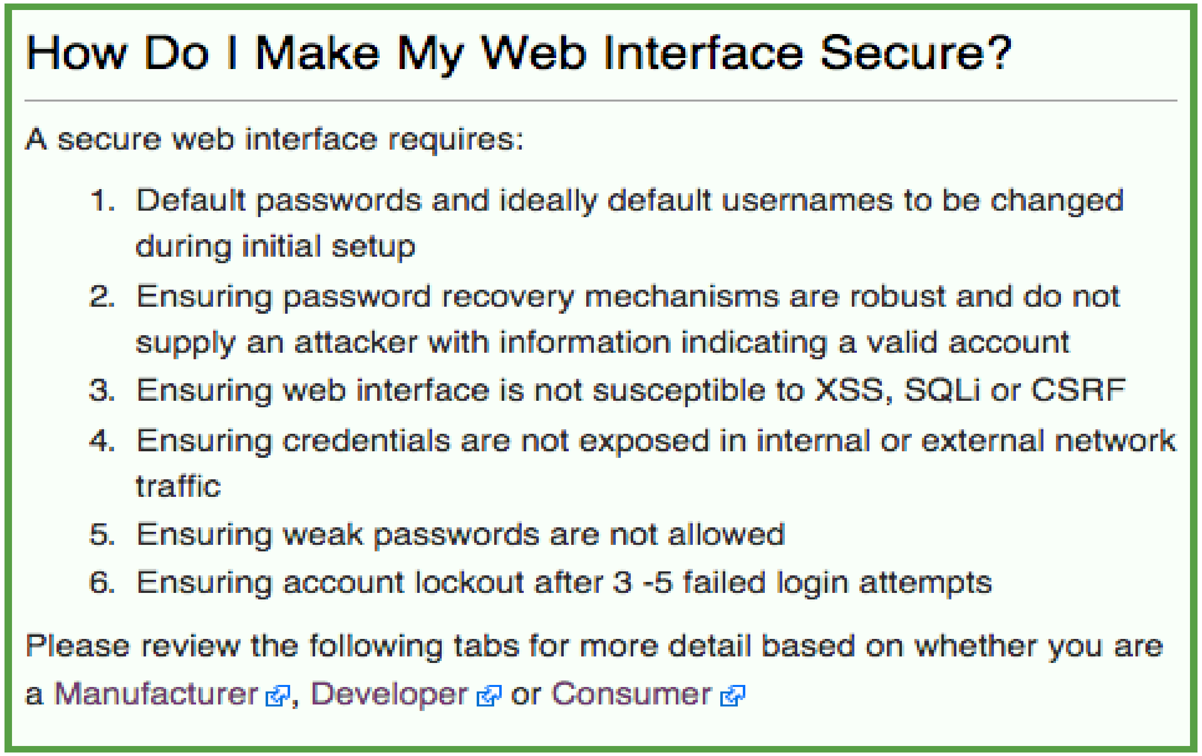
Scenario #2: Web interface is susceptible to cross-site scripting.

```
http://xyz.com/index.php?user=<script>alert(123)
</script> ... Response from browser is an alert
popup.
```

In the cases above, the attacker is able to easily determine if an account is valid or not and is also able to determine that the site is susceptible to cross-site scripting (XSS).

- Account Enumeration
- Weak Default Credentials
- Credentials Exposed in Network Traffic
- Cross-site Scripting (XSS)
- SQL-Injection
- Session Management
- Account Lockout

I1 | Insecure Web Interface | Make It Secure



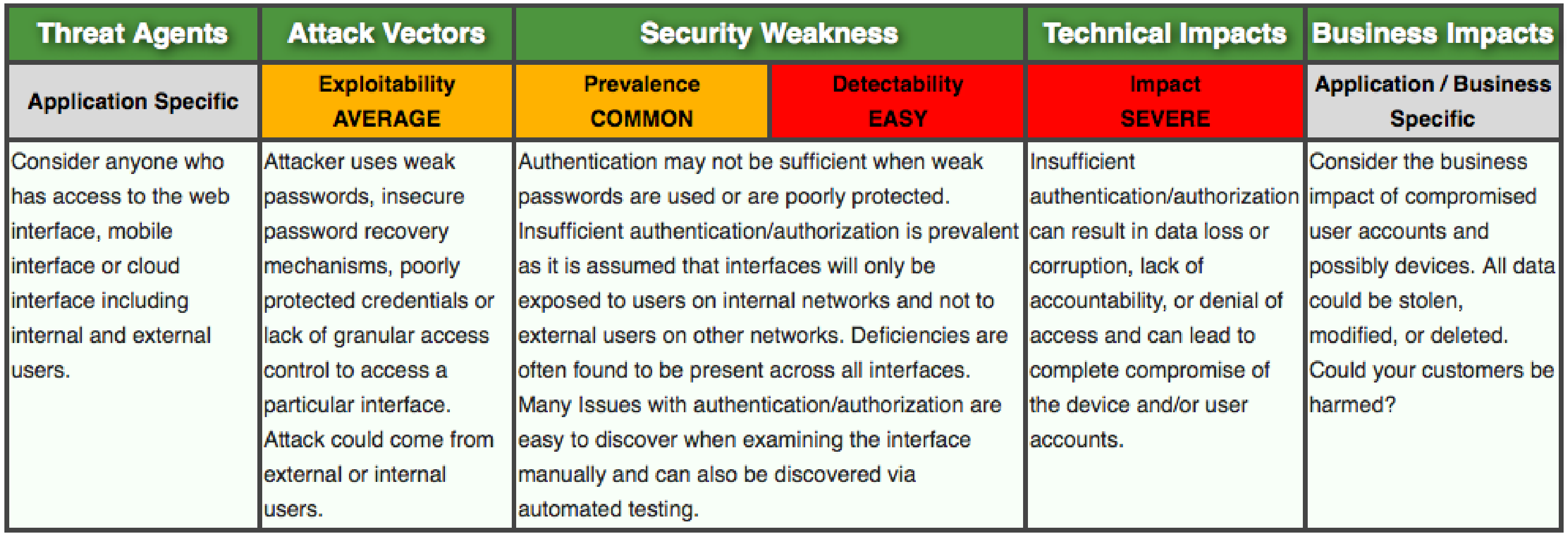
How Do I Make My Web Interface Secure?

A secure web interface requires:

1. Default passwords and ideally default usernames to be changed during initial setup
2. Ensuring password recovery mechanisms are robust and do not supply an attacker with information indicating a valid account
3. Ensuring web interface is not susceptible to XSS, SQLi or CSRF
4. Ensuring credentials are not exposed in internal or external network traffic
5. Ensuring weak passwords are not allowed
6. Ensuring account lockout after 3 -5 failed login attempts

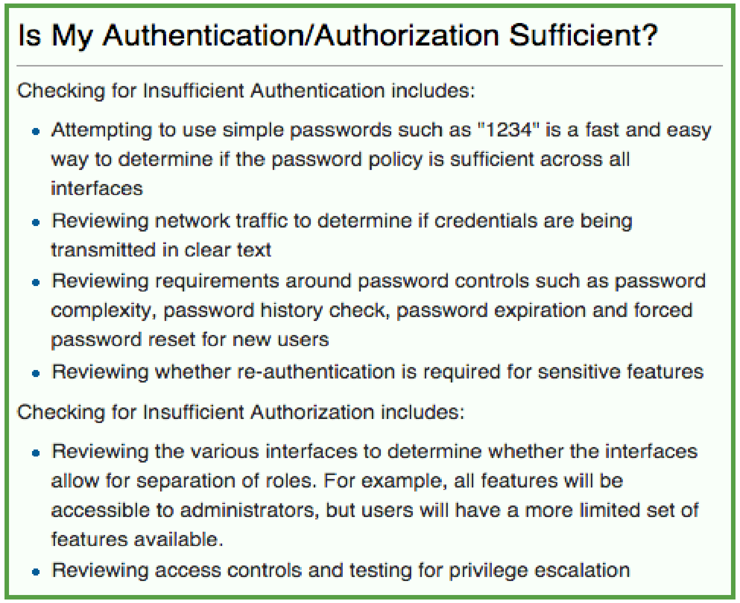
Please review the following tabs for more detail based on whether you are a Manufacturer, Developer or Consumer

I2 | Insufficient Authentication/ Authorization



Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence COMMON	Detectability EASY	Impact SEVERE	Application / Business Specific
Consider anyone who has access to the web interface, mobile interface or cloud interface including internal and external users.	Attacker uses weak passwords, insecure password recovery mechanisms, poorly protected credentials or lack of granular access control to access a particular interface. Attack could come from external or internal users.	Authentication may not be sufficient when weak passwords are used or are poorly protected. Insufficient authentication/authorization is prevalent as it is assumed that interfaces will only be exposed to users on internal networks and not to external users on other networks. Deficiencies are often found to be present across all interfaces. Many Issues with authentication/authorization are easy to discover when examining the interface manually and can also be discovered via automated testing.		Insufficient authentication/authorization can result in data loss or corruption, lack of accountability, or denial of access and can lead to complete compromise of the device and/or user accounts.	Consider the business impact of compromised user accounts and possibly devices. All data could be stolen, modified, or deleted. Could your customers be harmed?

I2 | Insufficient Authentication/Authorization | Testing



Is My Authentication/Authorization Sufficient?

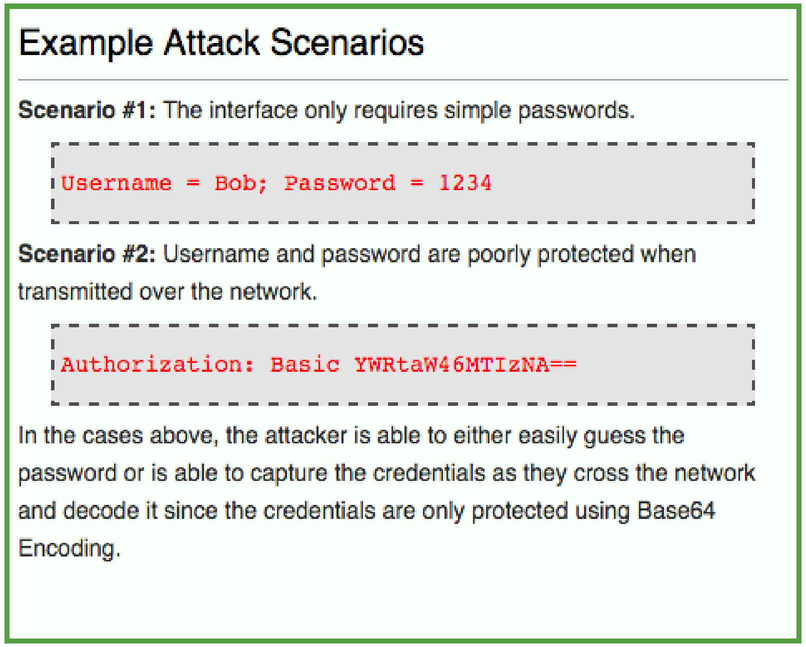
Checking for Insufficient Authentication includes:

- Attempting to use simple passwords such as "1234" is a fast and easy way to determine if the password policy is sufficient across all interfaces
- Reviewing network traffic to determine if credentials are being transmitted in clear text
- Reviewing requirements around password controls such as password complexity, password history check, password expiration and forced password reset for new users
- Reviewing whether re-authentication is required for sensitive features

Checking for Insufficient Authorization includes:

- Reviewing the various interfaces to determine whether the interfaces allow for separation of roles. For example, all features will be accessible to administrators, but users will have a more limited set of features available.
- Reviewing access controls and testing for privilege escalation

- Lack of Password Complexity
- Poorly Protected Credentials
- Lack of Two Factor Authentication
- Insecure Password Recovery
- Privilege Escalation
- Lack of Role Based Access Control



Example Attack Scenarios

Scenario #1: The interface only requires simple passwords.

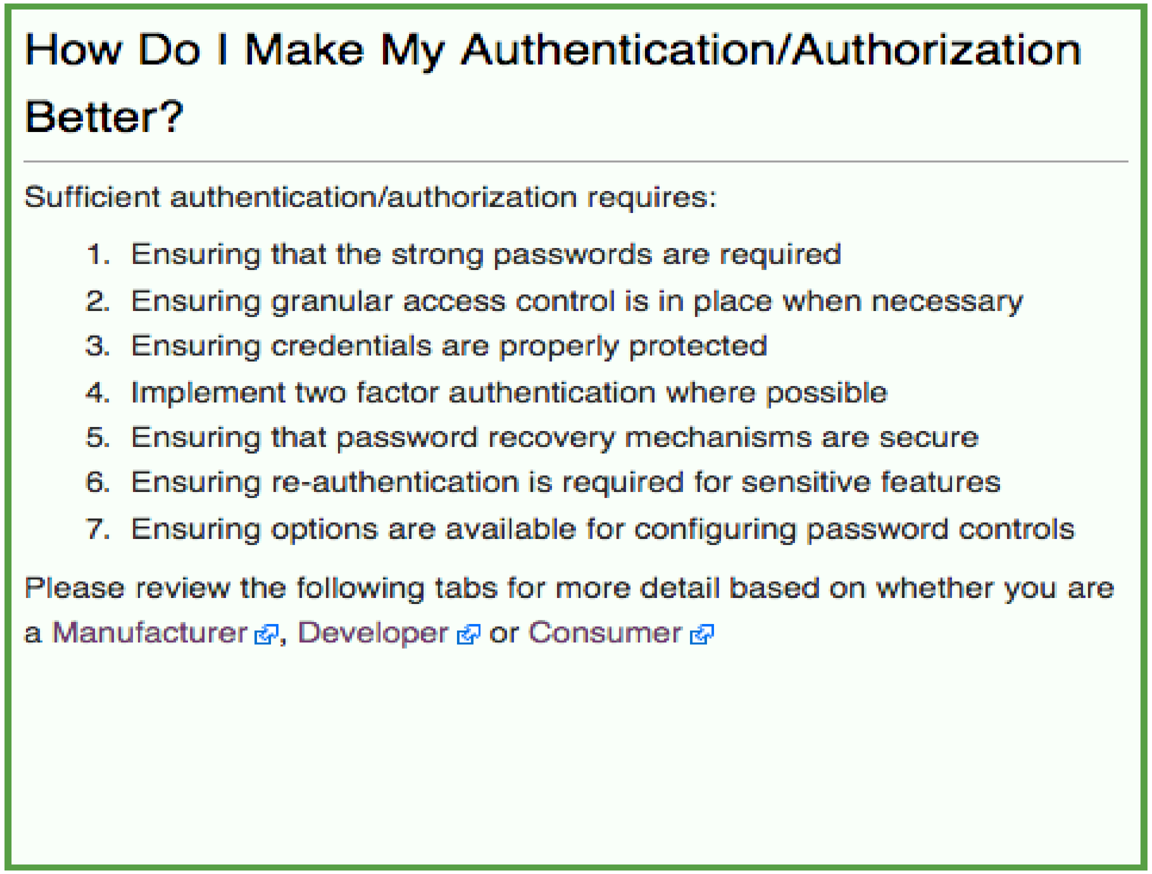
```
Username = Bob; Password = 1234
```

Scenario #2: Username and password are poorly protected when transmitted over the network.

```
Authorization: Basic YWRtaW46MTIzNA==
```

In the cases above, the attacker is able to either easily guess the password or is able to capture the credentials as they cross the network and decode it since the credentials are only protected using Base64 Encoding.

I2 | Insufficient Authentication/Authorization | Make It Secure



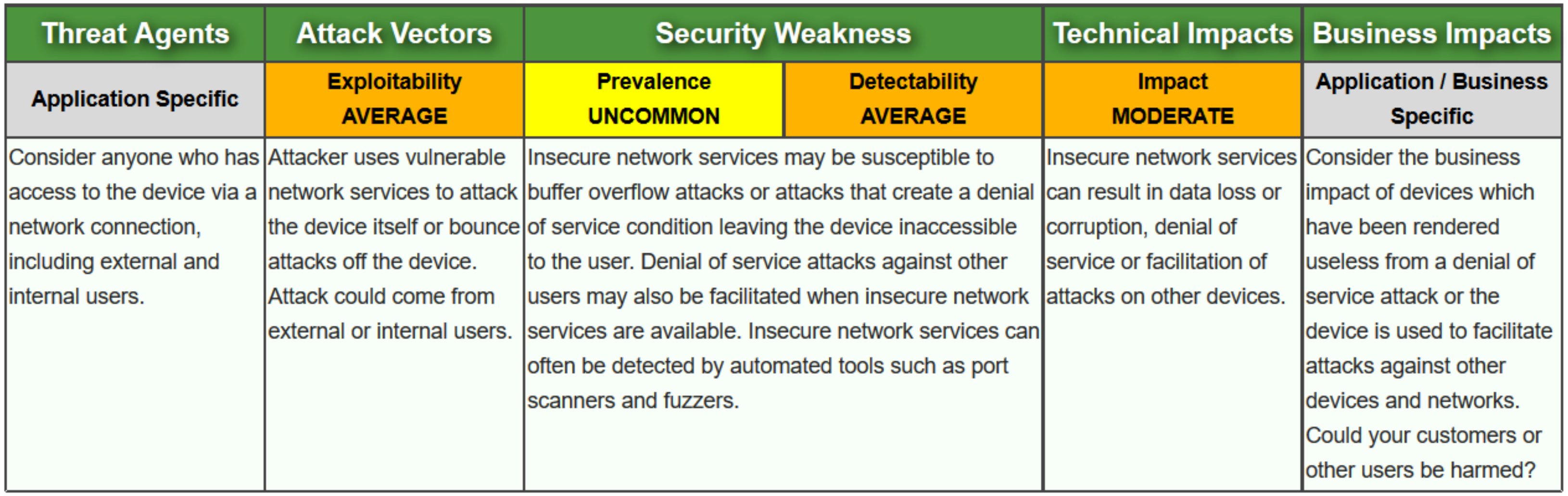
How Do I Make My Authentication/Authorization Better?

Sufficient authentication/authorization requires:

1. Ensuring that the strong passwords are required
2. Ensuring granular access control is in place when necessary
3. Ensuring credentials are properly protected
4. Implement two factor authentication where possible
5. Ensuring that password recovery mechanisms are secure
6. Ensuring re-authentication is required for sensitive features
7. Ensuring options are available for configuring password controls

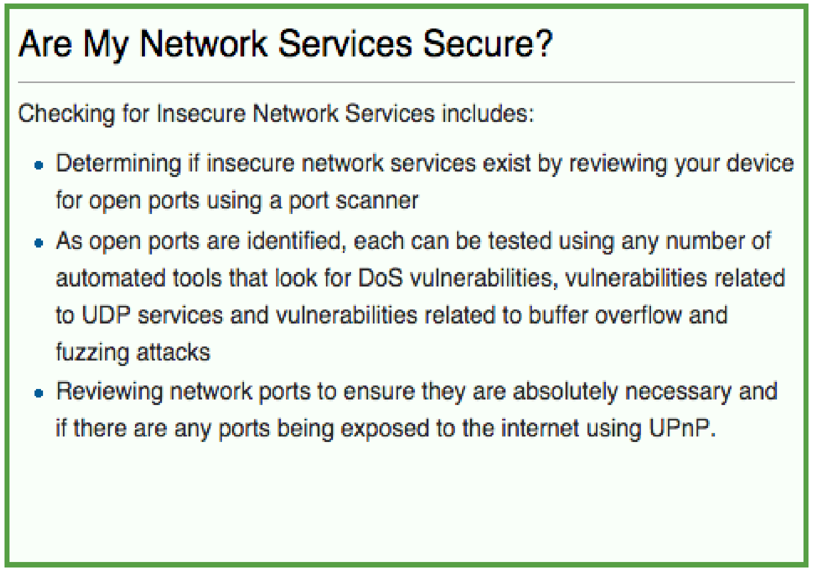
Please review the following tabs for more detail based on whether you are a Manufacturer, Developer or Consumer

I3 | Insecure Network Services



Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	**Exploitability AVERAGE**	**Prevalence UNCOMMON**	**Detectability AVERAGE**	**Impact MODERATE**	**Application / Business Specific**
Consider anyone who has access to the device via a network connection, including external and internal users.	Attacker uses vulnerable network services to attack the device itself or bounce attacks off the device. Attack could come from external or internal users.	Insecure network services may be susceptible to buffer overflow attacks or attacks that create a denial of service condition leaving the device inaccessible to the user. Denial of service attacks against other users may also be facilitated when insecure network services are available. Insecure network services can often be detected by automated tools such as port scanners and fuzzers.		Insecure network services can result in data loss or corruption, denial of service or facilitation of attacks on other devices.	Consider the business impact of devices which have been rendered useless from a denial of service attack or the device is used to facilitate attacks against other devices and networks. Could your customers or other users be harmed?

I3 | Insecure Network Services | Testing

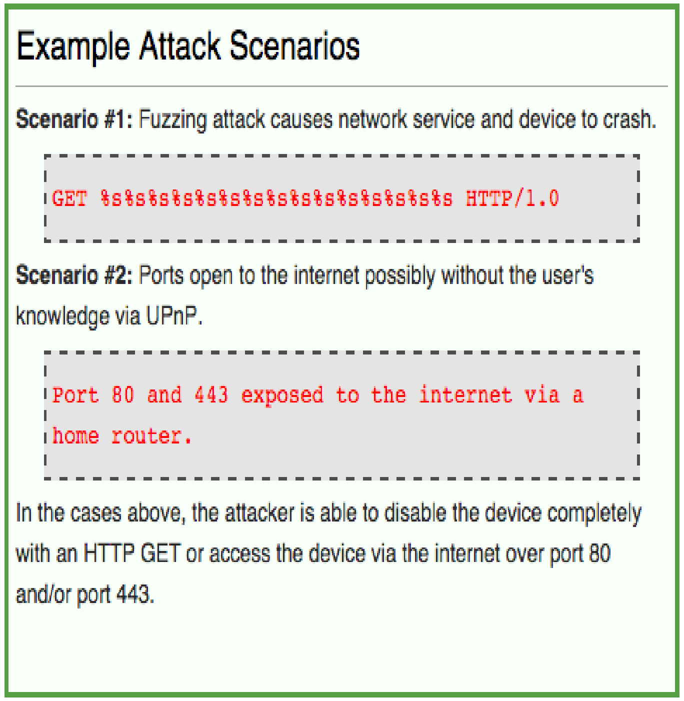


Are My Network Services Secure?

Checking for Insecure Network Services includes:

- Determining if insecure network services exist by reviewing your device for open ports using a port scanner
- As open ports are identified, each can be tested using any number of automated tools that look for DoS vulnerabilities, vulnerabilities related to UDP services and vulnerabilities related to buffer overflow and fuzzing attacks
- Reviewing network ports to ensure they are absolutely necessary and if there are any ports being exposed to the internet using UPnP.

- Vulnerable Services
- Buffer Overflow
- Open Ports via UPnP
- Exploitable UDP Services
- Denial-of-Service
- DoS via Network Device Fuzzing



Example Attack Scenarios

Scenario #1: Fuzzing attack causes network service and device to crash.

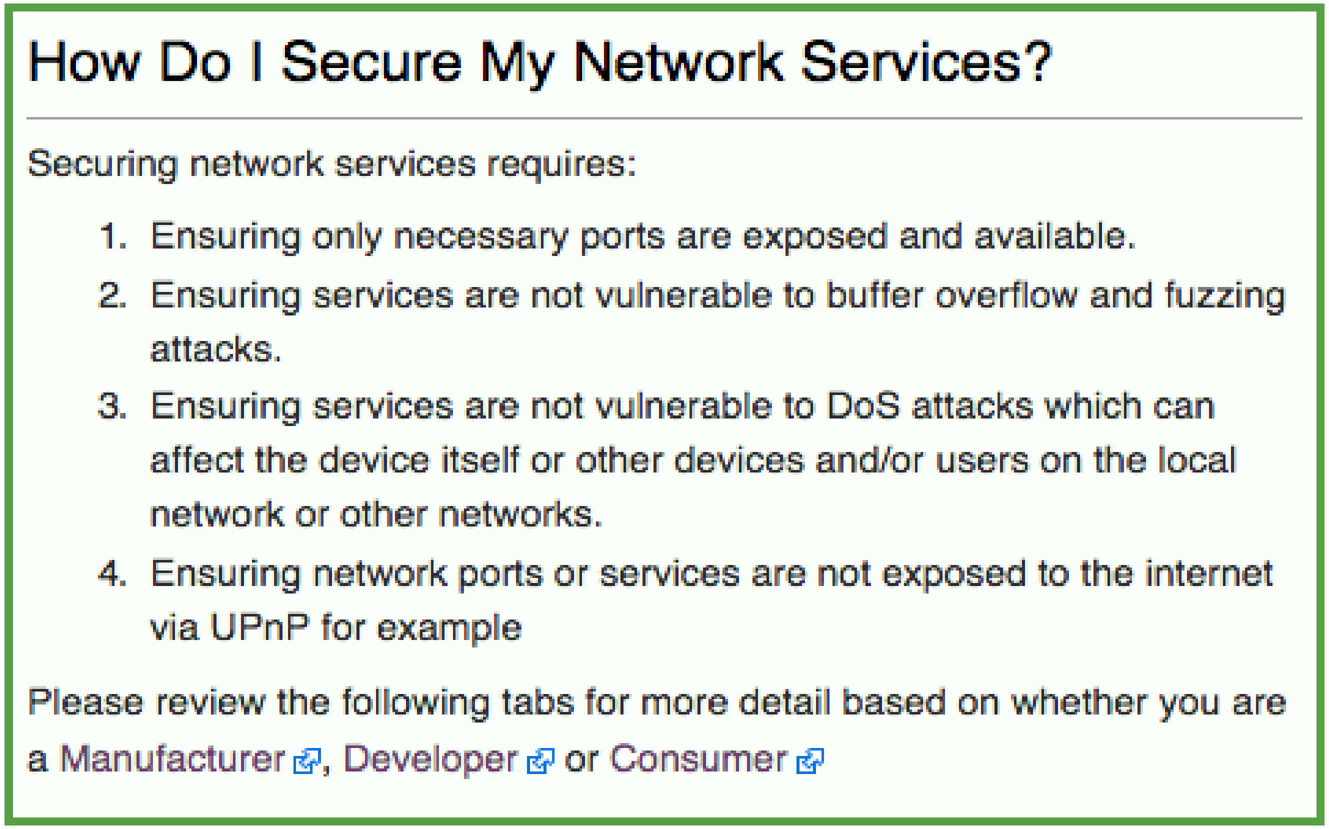
```
GET %s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s HTTP/1.0
```

Scenario #2: Ports open to the internet possibly without the user's knowledge via UPnP.

```
Port 80 and 443 exposed to the internet via a
home router.
```

In the cases above, the attacker is able to disable the device completely with an HTTP GET or access the device via the internet over port 80 and/or port 443.

I3 | Insecure Network Services | Make It Secure



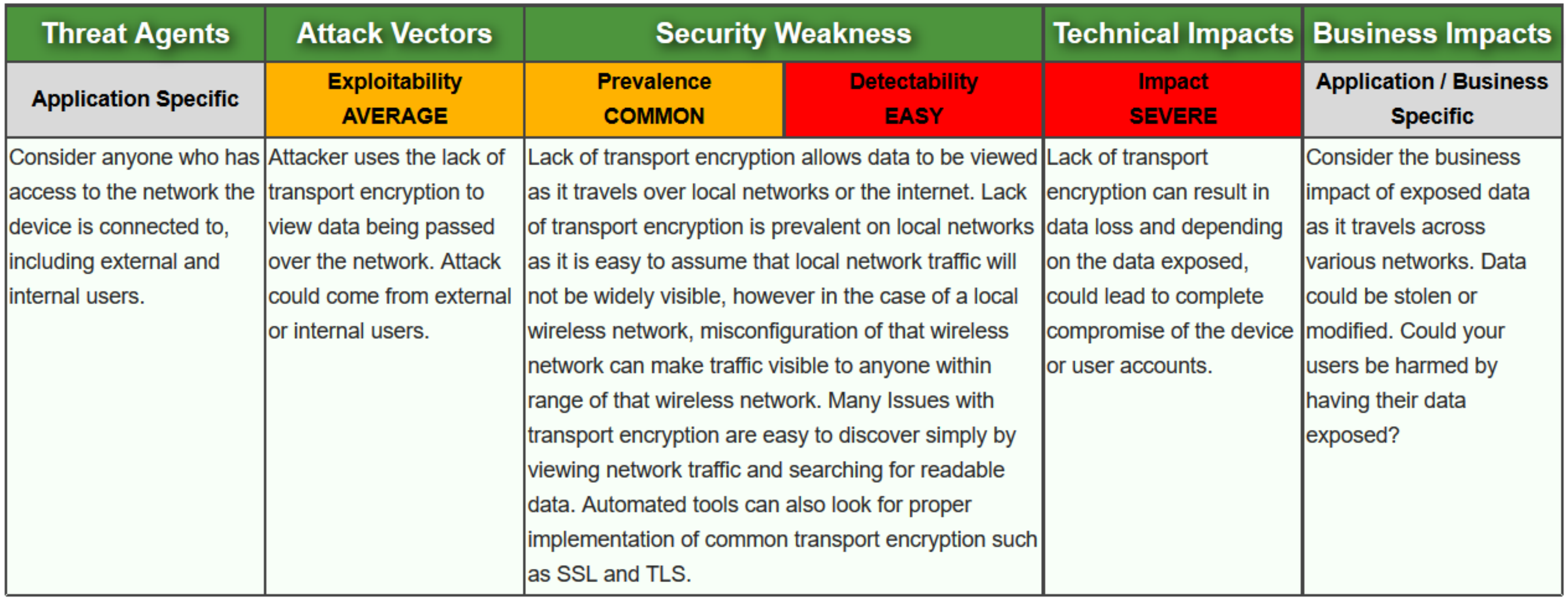
How Do I Secure My Network Services?

Securing network services requires:

1. Ensuring only necessary ports are exposed and available.
2. Ensuring services are not vulnerable to buffer overflow and fuzzing attacks.
3. Ensuring services are not vulnerable to DoS attacks which can affect the device itself or other devices and/or users on the local network or other networks.
4. Ensuring network ports or services are not exposed to the internet via UPnP for example

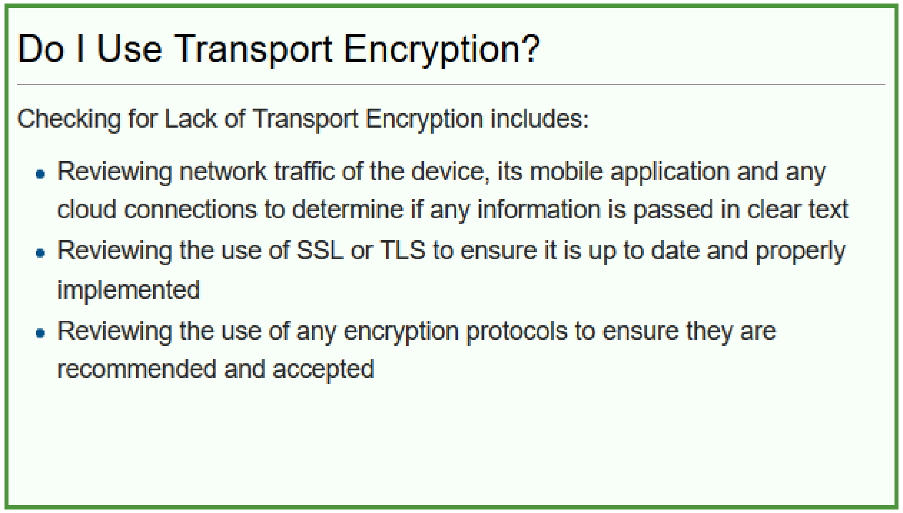
Please review the following tabs for more detail based on whether you are a Manufacturer, Developer or Consumer

I4 | Lack of Transport Encryption



Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence COMMON	Detectability EASY	Impact SEVERE	Application / Business Specific
Consider anyone who has access to the network the device is connected to, including external and internal users.	Attacker uses the lack of transport encryption to view data being passed over the network. Attack could come from external or internal users.	Lack of transport encryption allows data to be viewed as it travels over local networks or the internet. Lack of transport encryption is prevalent on local networks as it is easy to assume that local network traffic will not be widely visible, however in the case of a local wireless network, misconfiguration of that wireless network can make traffic visible to anyone within range of that wireless network. Many Issues with transport encryption are easy to discover simply by viewing network traffic and searching for readable data. Automated tools can also look for proper implementation of common transport encryption such as SSL and TLS.		Lack of transport encryption can result in data loss and depending on the data exposed, could lead to complete compromise of the device or user accounts.	Consider the business impact of exposed data as it travels across various networks. Data could be stolen or modified. Could your users be harmed by having their data exposed?

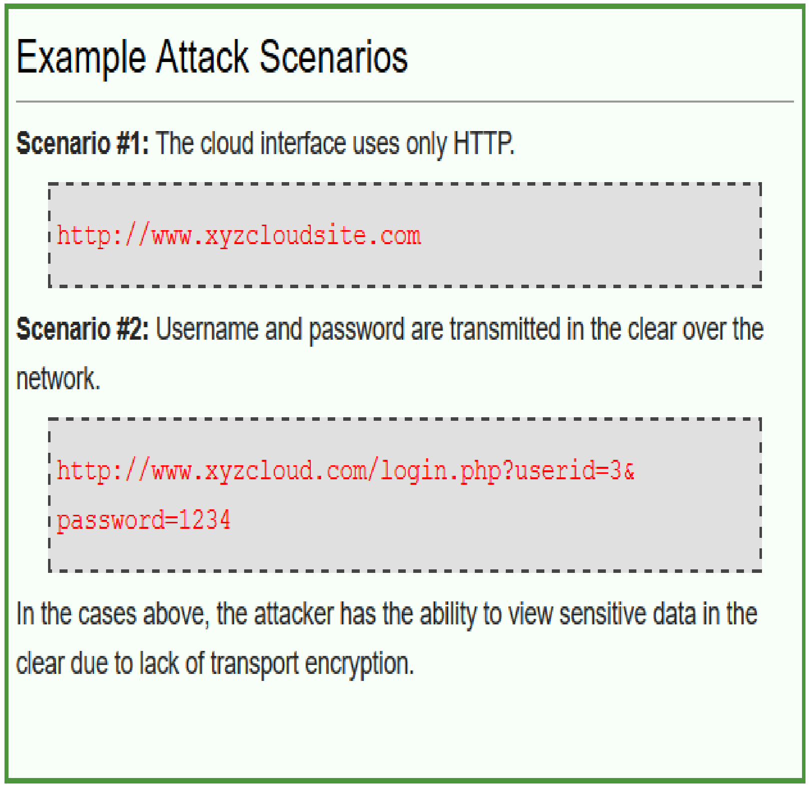
I4 | Lack of Transport Encryption | Testing



Do I Use Transport Encryption?

Checking for Lack of Transport Encryption includes:

- Reviewing network traffic of the device, its mobile application and any cloud connections to determine if any information is passed in clear text
- Reviewing the use of SSL or TLS to ensure it is up to date and properly implemented
- Reviewing the use of any encryption protocols to ensure they are recommended and accepted



Example Attack Scenarios

Scenario #1: The cloud interface uses only HTTP.

```
http://www.xyzcloudsite.com
```

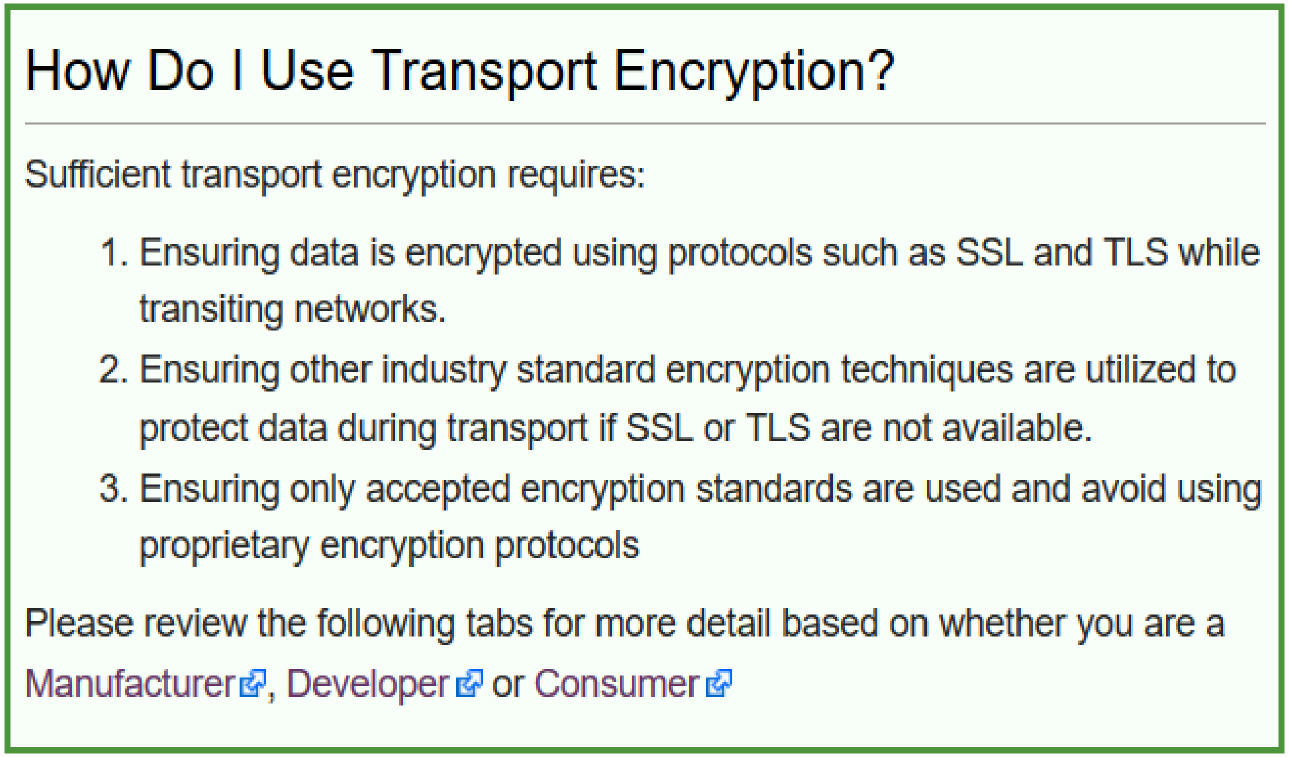
Scenario #2: Username and password are transmitted in the clear over the network.

```
http://www.xyzcloud.com/login.php?userid=3&
password=1234
```

In the cases above, the attacker has the ability to view sensitive data in the clear due to lack of transport encryption.

- Unencrypted Services via the Internet
- Unencrypted Services via the Local Network
- Poorly Implemented SSL/TLS
- Misconfigured SSL/TLS

I4 | Lack of Transport Encryption | Make It Secure



How Do I Use Transport Encryption?

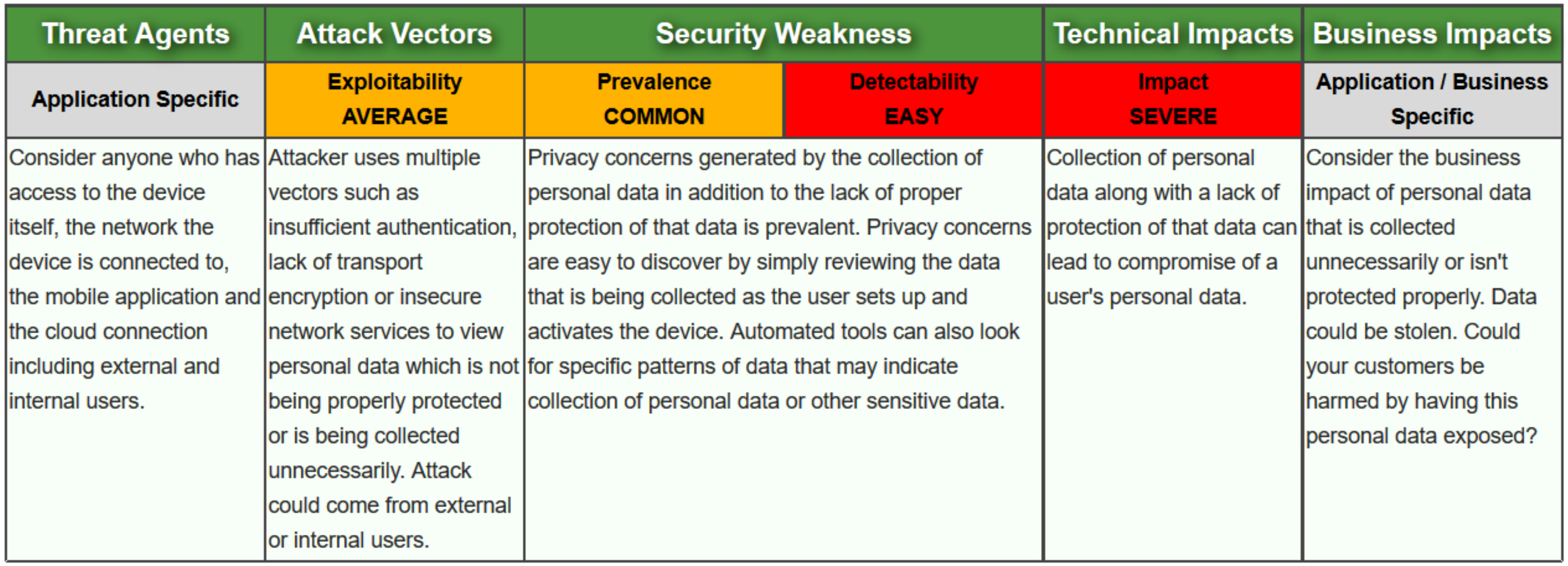
Sufficient transport encryption requires:

1. Ensuring data is encrypted using protocols such as SSL and TLS while transiting networks.
2. Ensuring other industry standard encryption techniques are utilized to protect data during transport if SSL or TLS are not available.
3. Ensuring only accepted encryption standards are used and avoid using proprietary encryption protocols

Please review the following tabs for more detail based on whether you are a Manufacturer, Developer or Consumer

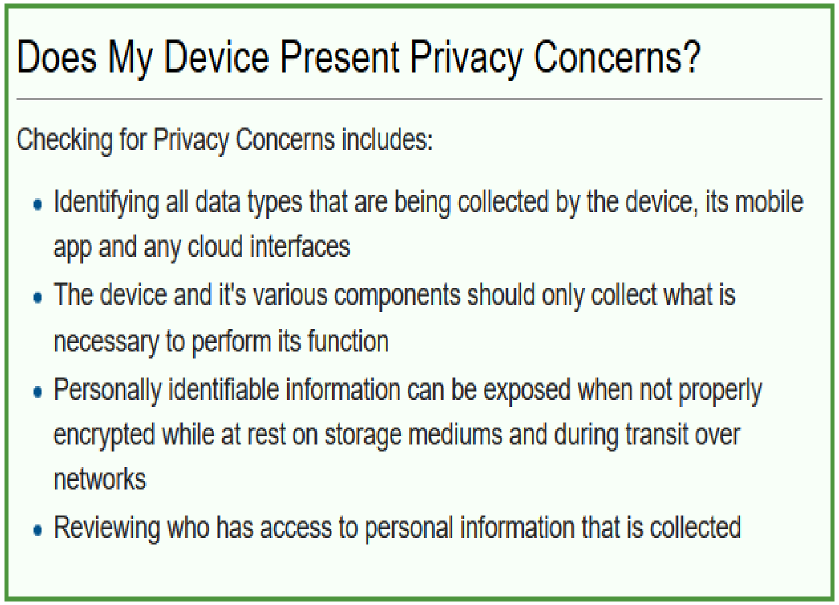
Recall the procedure we have previously outlined to establish a secure connection between a device and a server

I5 | Privacy Concerns



Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	**Exploitability AVERAGE**	**Prevalence COMMON**	**Detectability EASY**	**Impact SEVERE**	**Application / Business Specific**
Consider anyone who has access to the device itself, the network the device is connected to, the mobile application and the cloud connection including external and internal users.	Attacker uses multiple vectors such as insufficient authentication, lack of transport encryption or insecure network services to view personal data which is not being properly protected or is being collected unnecessarily. Attack could come from external or internal users.	Privacy concerns generated by the collection of personal data in addition to the lack of proper protection of that data is prevalent. Privacy concerns are easy to discover by simply reviewing the data that is being collected as the user sets up and activates the device. Automated tools can also look for specific patterns of data that may indicate collection of personal data or other sensitive data.		Collection of personal data along with a lack of protection of that data can lead to compromise of a user's personal data.	Consider the business impact of personal data that is collected unnecessarily or isn't protected properly. Data could be stolen. Could your customers be harmed by having this personal data exposed?

I5 | Privacy Concerns | Testing

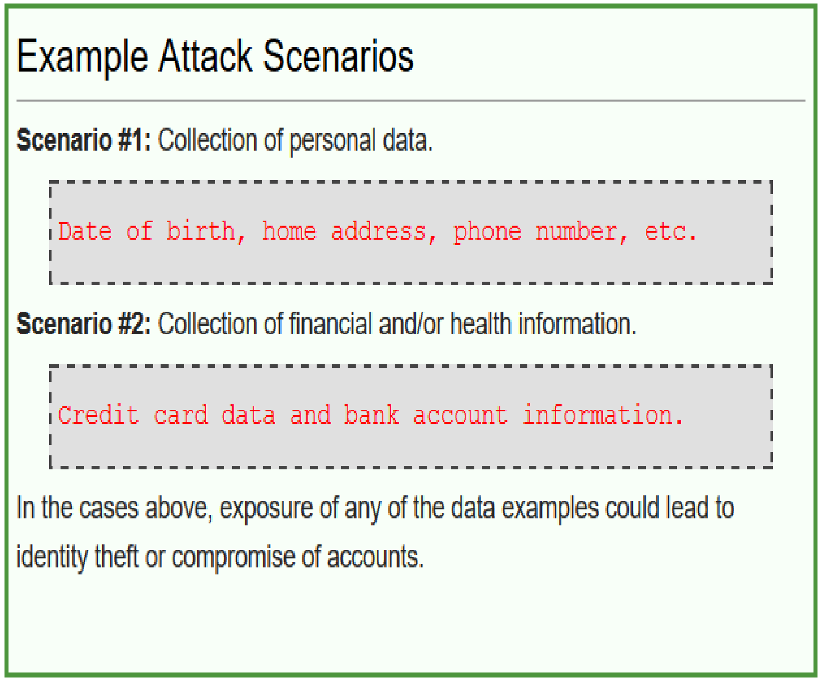


Does My Device Present Privacy Concerns?

Checking for Privacy Concerns includes:

- Identifying all data types that are being collected by the device, its mobile app and any cloud interfaces
- The device and it's various components should only collect what is necessary to perform its function
- Personally identifiable information can be exposed when not properly encrypted while at rest on storage mediums and during transit over networks
- Reviewing who has access to personal information that is collected

- Collection of Unnecessary Personal Information



Example Attack Scenarios

Scenario #1: Collection of personal data.

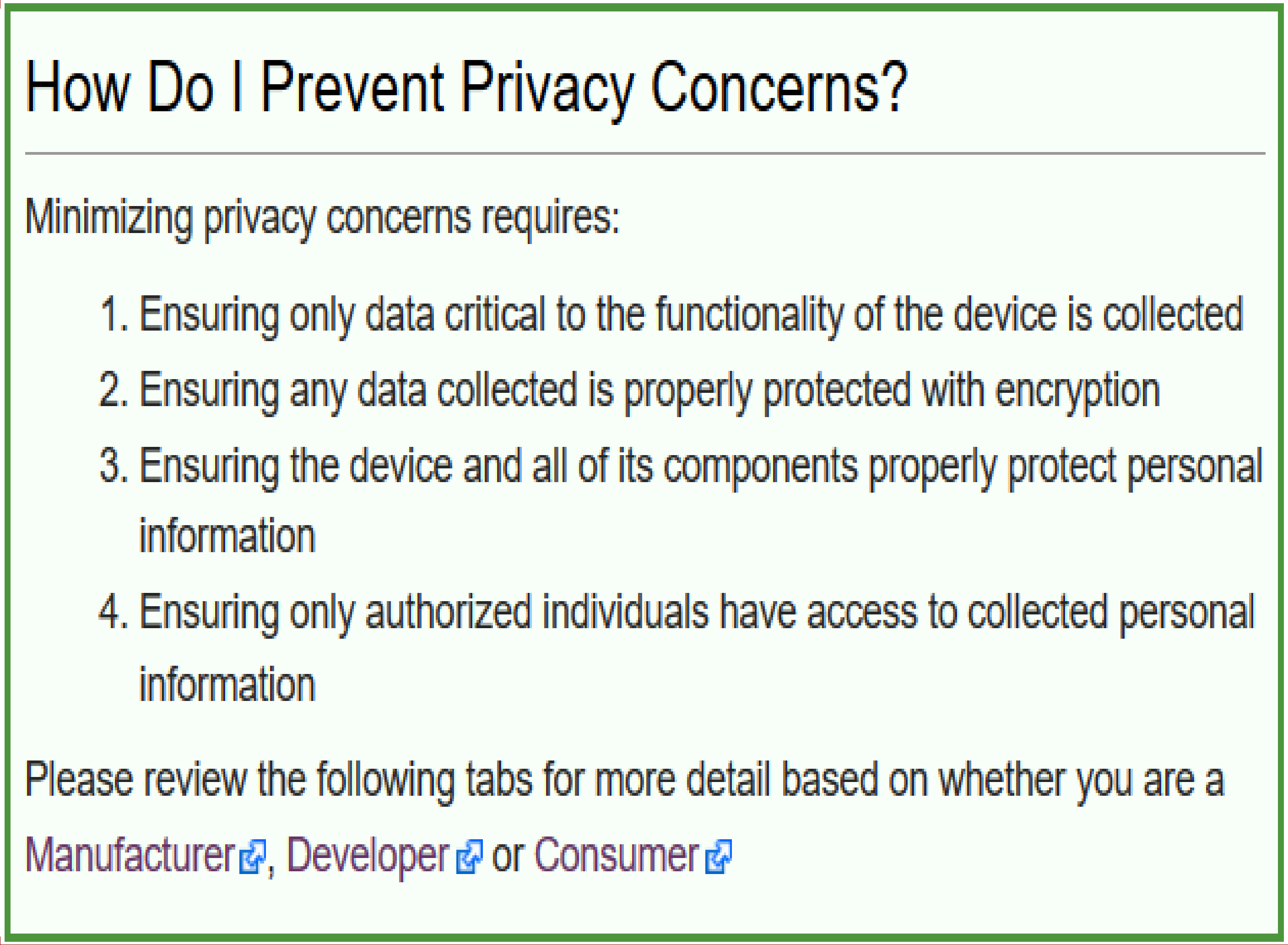
```
Date of birth, home address, phone number, etc.
```

Scenario #2: Collection of financial and/or health information.

```
Credit card data and bank account information.
```

In the cases above, exposure of any of the data examples could lead to identity theft or compromise of accounts.

I5 | Privacy Concerns | Make It Secure



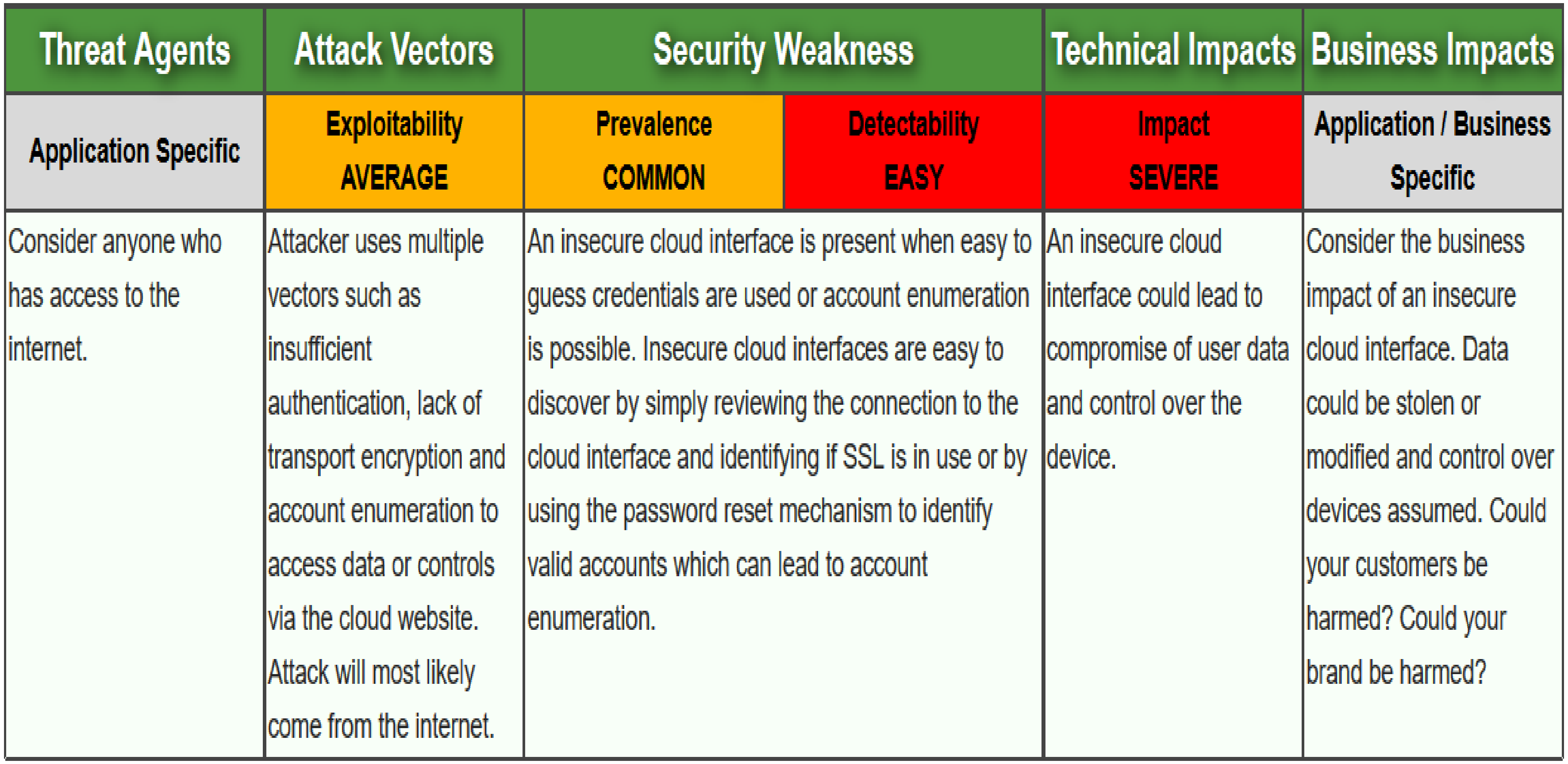
How Do I Prevent Privacy Concerns?

Minimizing privacy concerns requires:

1. Ensuring only data critical to the functionality of the device is collected
2. Ensuring any data collected is properly protected with encryption
3. Ensuring the device and all of its components properly protect personal information
4. Ensuring only authorized individuals have access to collected personal information

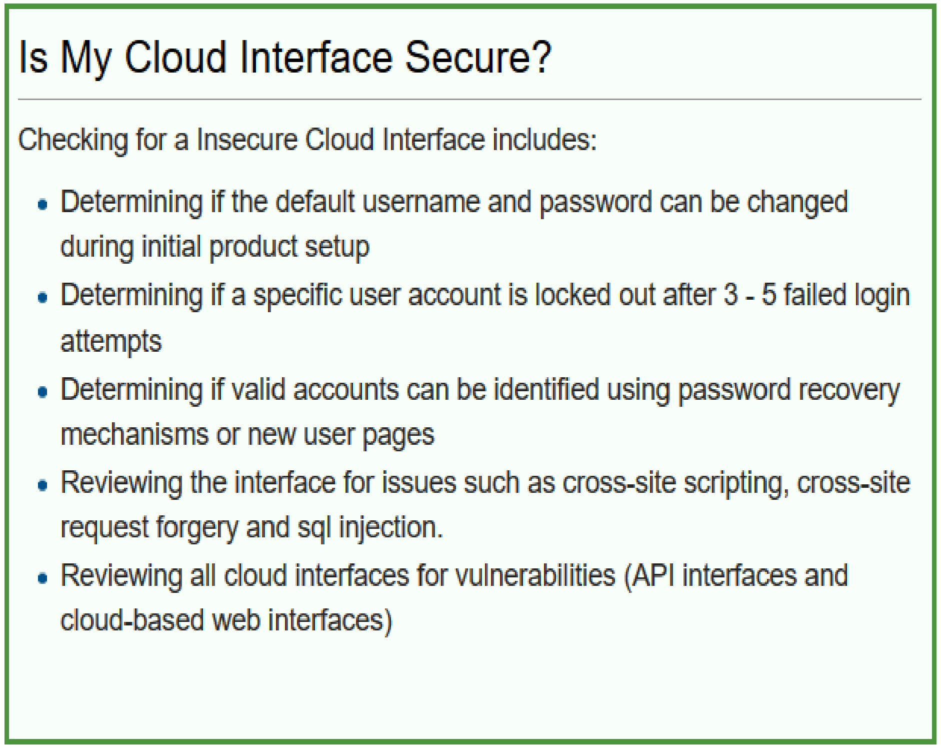
Please review the following tabs for more detail based on whether you are a Manufacturer, Developer or Consumer

I6 | Insecure Cloud Interface



Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence COMMON	Detectability EASY	Impact SEVERE	Application / Business Specific
Consider anyone who has access to the internet.	Attacker uses multiple vectors such as insufficient authentication, lack of transport encryption and account enumeration to access data or controls via the cloud website. Attack will most likely come from the internet.	An insecure cloud interface is present when easy to guess credentials are used or account enumeration is possible. Insecure cloud interfaces are easy to discover by simply reviewing the connection to the cloud interface and identifying if SSL is in use or by using the password reset mechanism to identify valid accounts which can lead to account enumeration.		An insecure cloud interface could lead to compromise of user data and control over the device.	Consider the business impact of an insecure cloud interface. Data could be stolen or modified and control over devices assumed. Could your customers be harmed? Could your brand be harmed?

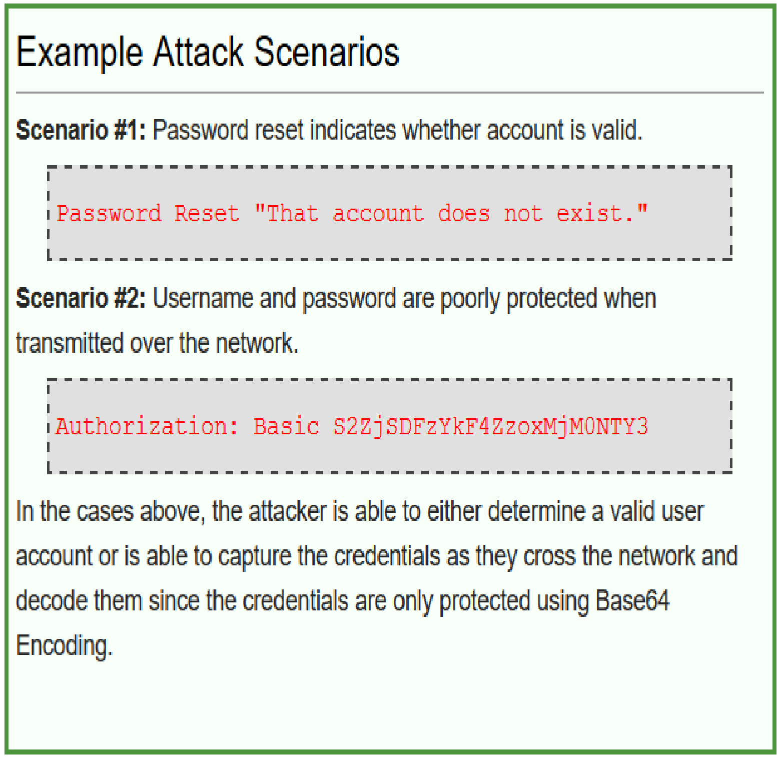
I6 | Insecure Cloud Interface | Testing



Is My Cloud Interface Secure?

Checking for a Insecure Cloud Interface includes:

- Determining if the default username and password can be changed during initial product setup
- Determining if a specific user account is locked out after 3 - 5 failed login attempts
- Determining if valid accounts can be identified using password recovery mechanisms or new user pages
- Reviewing the interface for issues such as cross-site scripting, cross-site request forgery and sql injection.
- Reviewing all cloud interfaces for vulnerabilities (API interfaces and cloud-based web interfaces)



Example Attack Scenarios

Scenario #1: Password reset indicates whether account is valid.

```
Password Reset "That account does not exist."
```

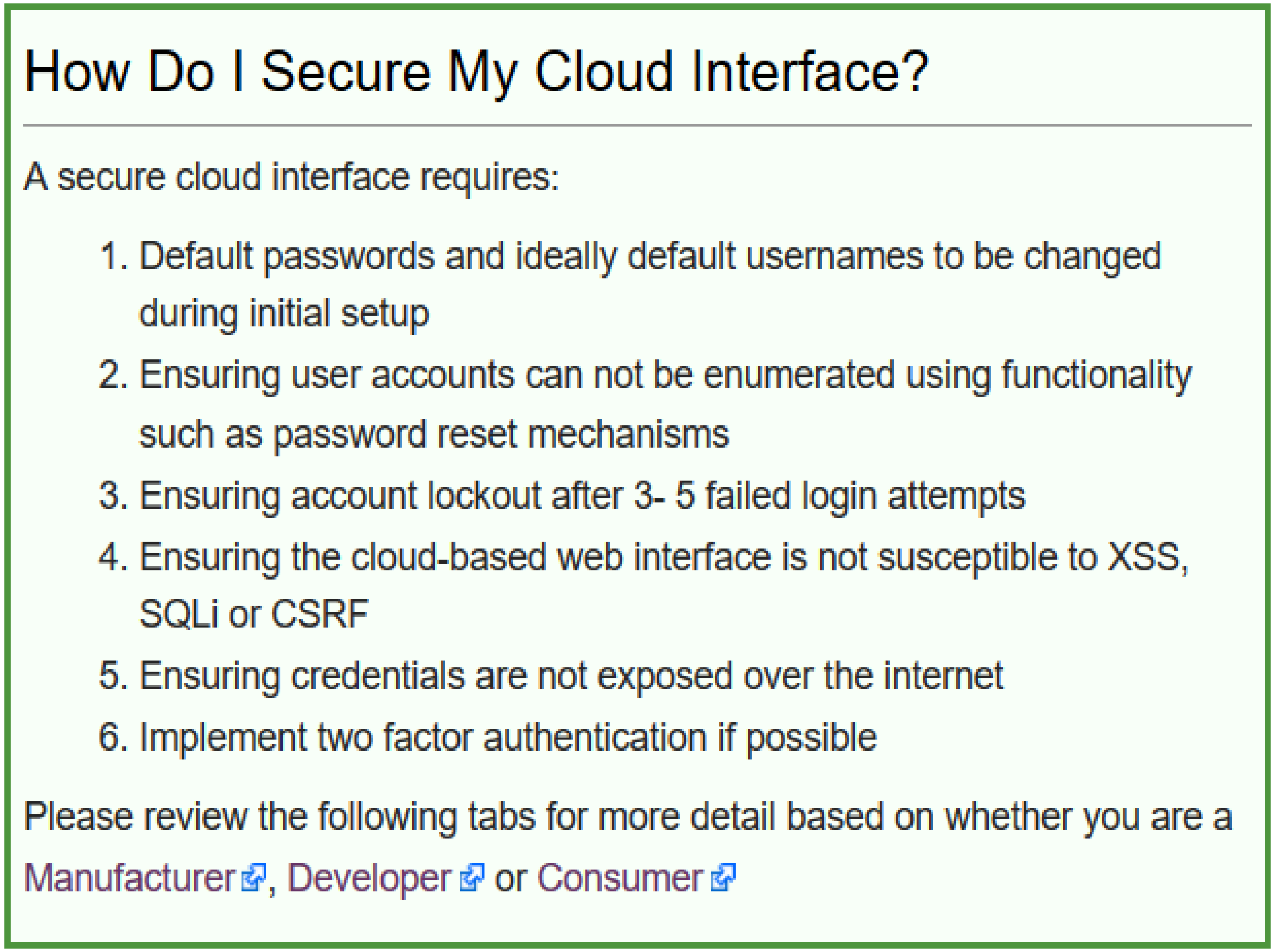
Scenario #2: Username and password are poorly protected when transmitted over the network.

```
Authorization: Basic S2ZjSDFzYkF4ZzoxMjM0NTY3
```

In the cases above, the attacker is able to either determine a valid user account or is able to capture the credentials as they cross the network and decode them since the credentials are only protected using Base64 Encoding.

- Account Enumeration
- No Account Lockout
- Credentials Exposed in Network Traffic

I6 | Insecure Cloud Interface | Make It Secure



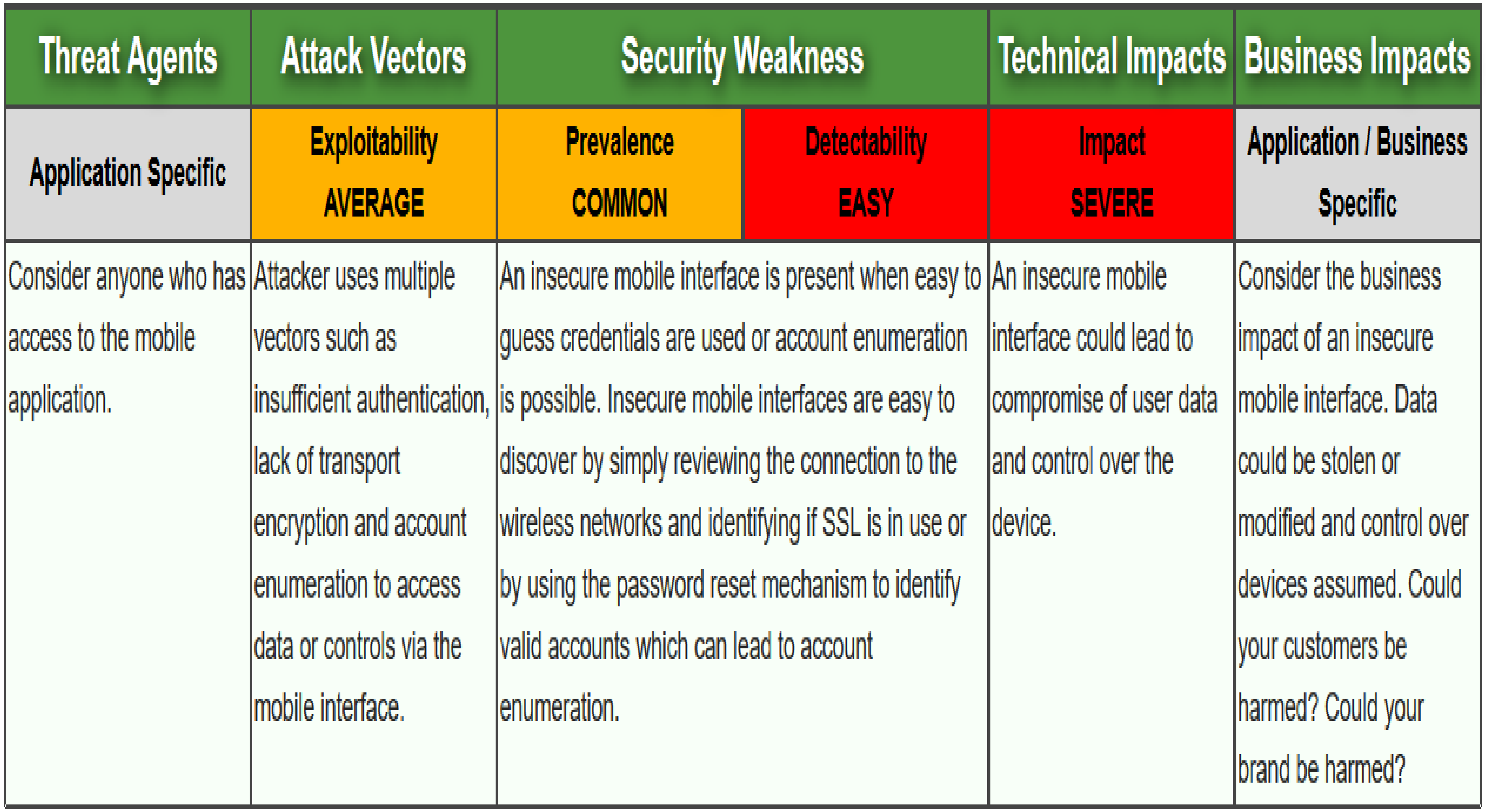
How Do I Secure My Cloud Interface?

A secure cloud interface requires:

1. Default passwords and ideally default usernames to be changed during initial setup
2. Ensuring user accounts can not be enumerated using functionality such as password reset mechanisms
3. Ensuring account lockout after 3- 5 failed login attempts
4. Ensuring the cloud-based web interface is not susceptible to XSS, SQLi or CSRF
5. Ensuring credentials are not exposed over the internet
6. Implement two factor authentication if possible

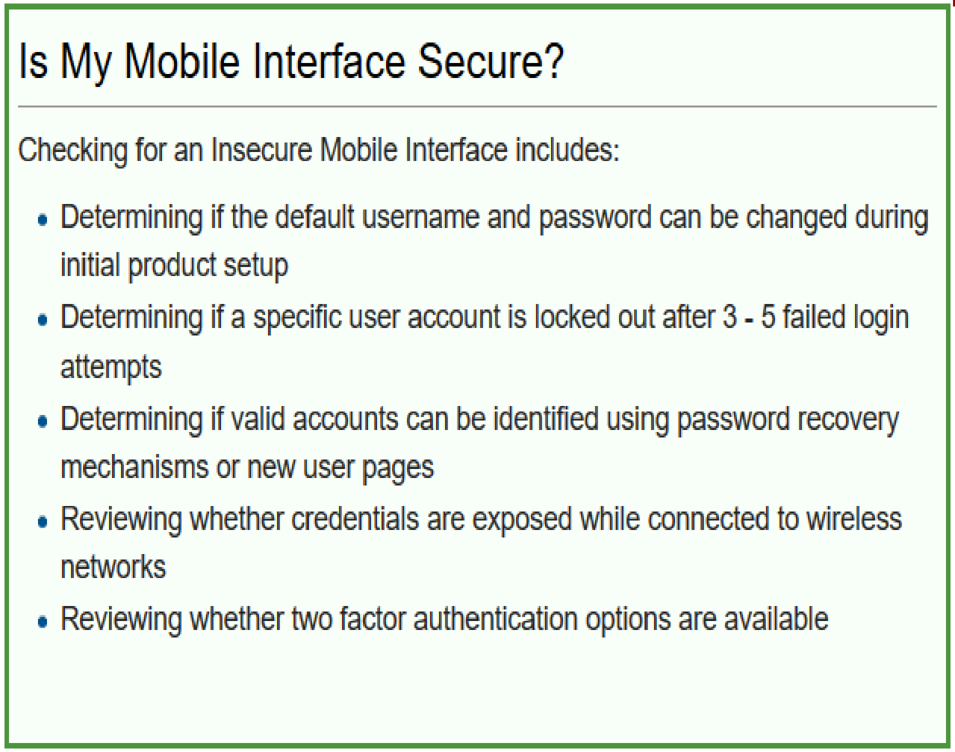
Please review the following tabs for more detail based on whether you are a Manufacturer, Developer or Consumer

I7 | Insecure Mobile Interface



Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence COMMON	Detectability EASY	Impact SEVERE	Application / Business Specific
Consider anyone who has access to the mobile application.	Attacker uses multiple vectors such as insufficient authentication, lack of transport encryption and account enumeration to access data or controls via the mobile interface.	An insecure mobile interface is present when easy to guess credentials are used or account enumeration is possible. Insecure mobile interfaces are easy to discover by simply reviewing the connection to the wireless networks and identifying if SSL is in use or by using the password reset mechanism to identify valid accounts which can lead to account enumeration.		An insecure mobile interface could lead to compromise of user data and control over the device.	Consider the business impact of an insecure mobile interface. Data could be stolen or modified and control over devices assumed. Could your customers be harmed? Could your brand be harmed?

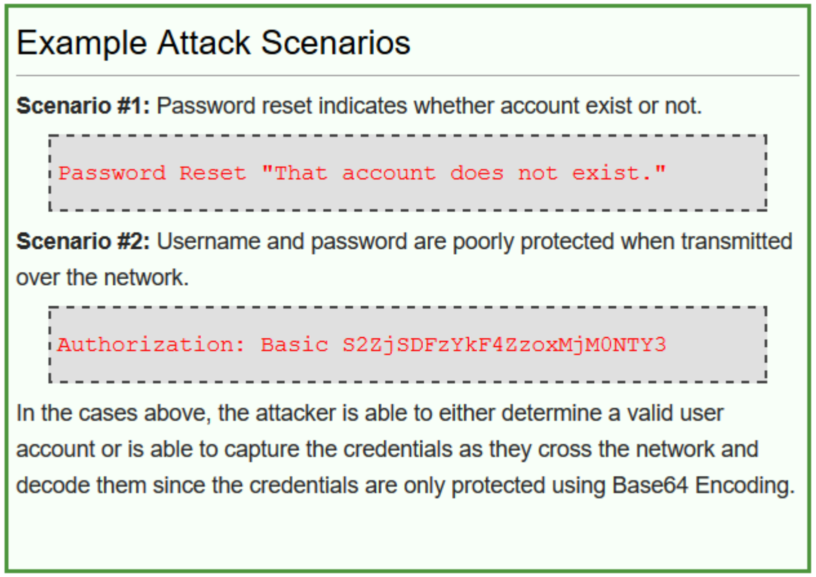
I7 | Insecure Mobile Interface | Testing



Is My Mobile Interface Secure?

Checking for an Insecure Mobile Interface includes:

- Determining if the default username and password can be changed during initial product setup
- Determining if a specific user account is locked out after 3 - 5 failed login attempts
- Determining if valid accounts can be identified using password recovery mechanisms or new user pages
- Reviewing whether credentials are exposed while connected to wireless networks
- Reviewing whether two factor authentication options are available



Example Attack Scenarios

Scenario #1: Password reset indicates whether account exist or not.

```
Password Reset "That account does not exist."
```

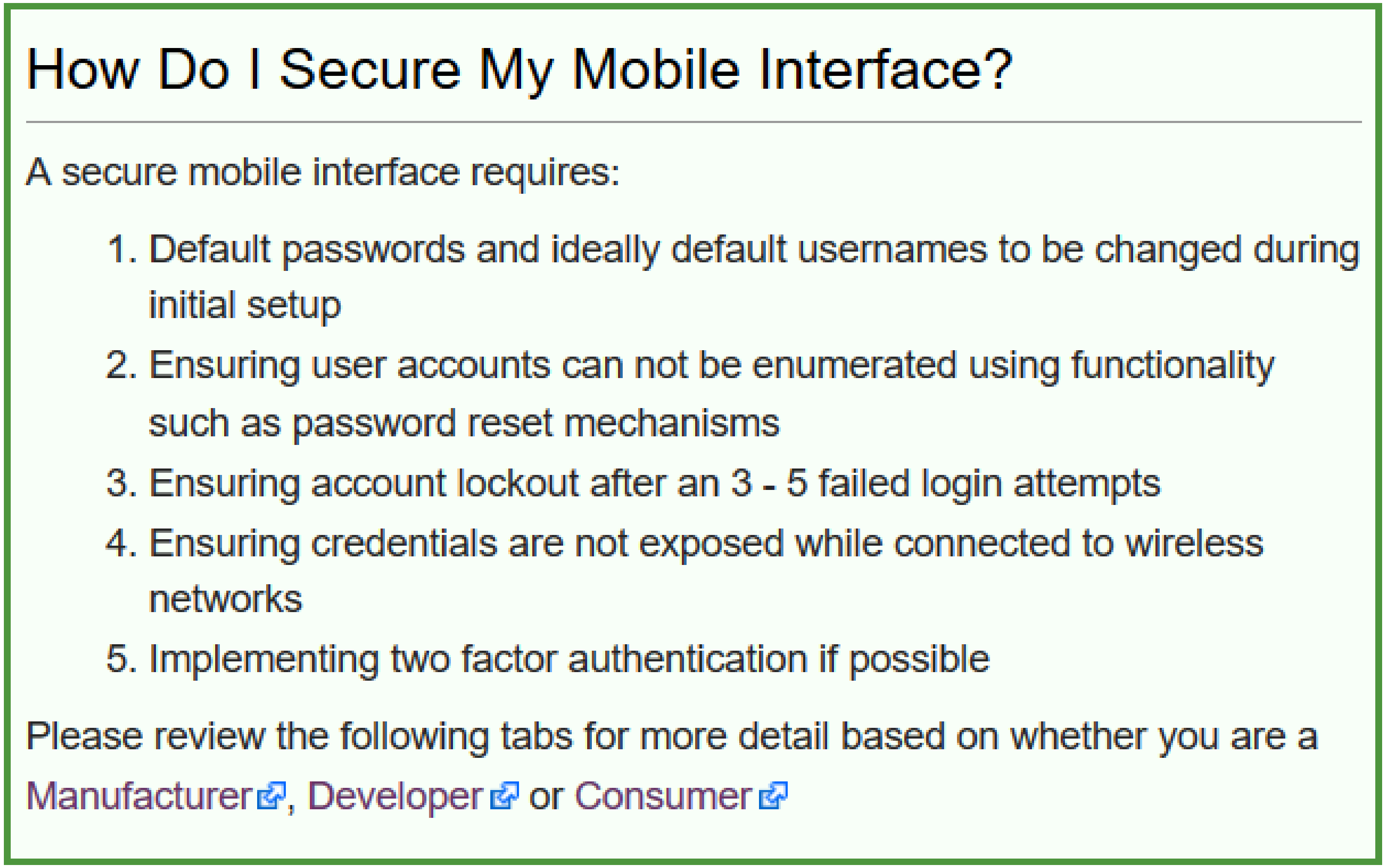
Scenario #2: Username and password are poorly protected when transmitted over the network.

```
Authorization: Basic S2ZjSDFzYkF4ZzoxMjM0NTY3
```

In the cases above, the attacker is able to either determine a valid user account or is able to capture the credentials as they cross the network and decode them since the credentials are only protected using Base64 Encoding.

- Account Enumeration
- No Account Lockout
- Credentials Exposed in Network Traffic

I7 | Insecure Mobile Interface | Make It Secure



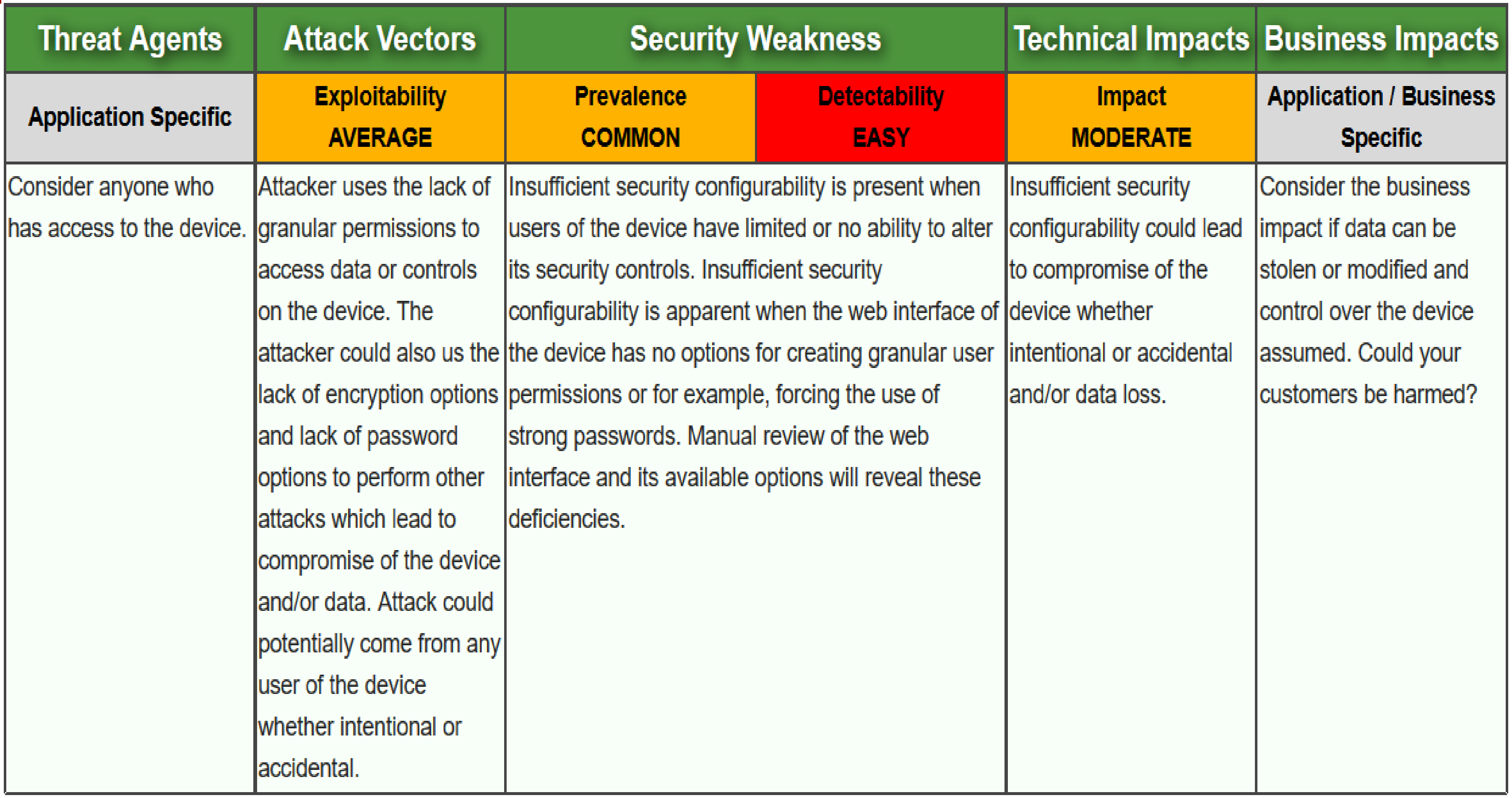
How Do I Secure My Mobile Interface?

A secure mobile interface requires:

1. Default passwords and ideally default usernames to be changed during initial setup
2. Ensuring user accounts can not be enumerated using functionality such as password reset mechanisms
3. Ensuring account lockout after an 3 - 5 failed login attempts
4. Ensuring credentials are not exposed while connected to wireless networks
5. Implementing two factor authentication if possible

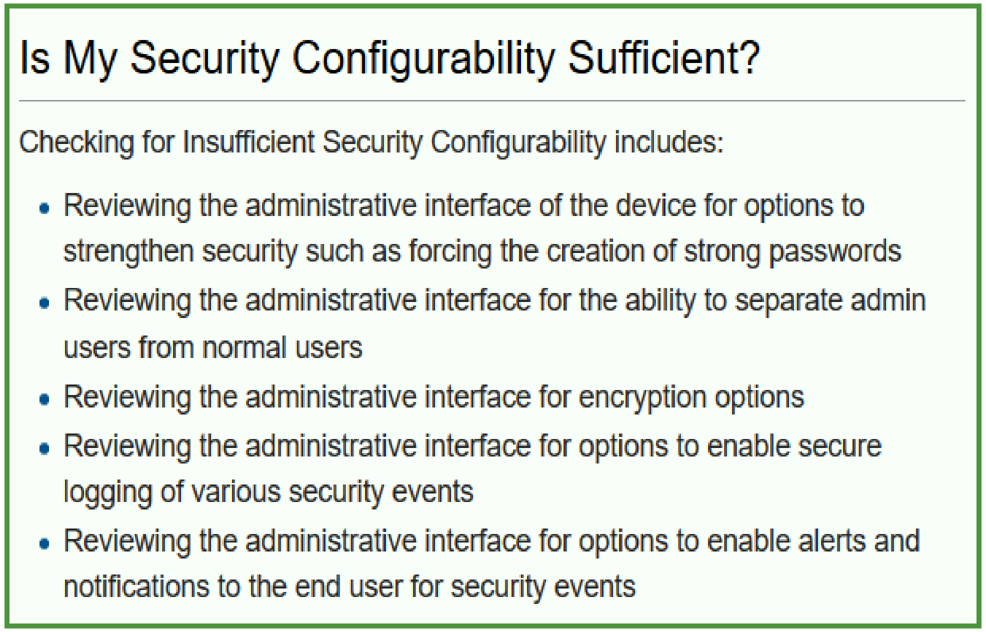
Please review the following tabs for more detail based on whether you are a Manufacturer, Developer or Consumer

I8 | Insufficient Security Configurability



Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence COMMON	Detectability EASY	Impact MODERATE	Application / Business Specific
Consider anyone who has access to the device.	Attacker uses the lack of granular permissions to access data or controls on the device. The attacker could also us the lack of encryption options and lack of password options to perform other attacks which lead to compromise of the device and/or data. Attack could potentially come from any user of the device whether intentional or accidental.	Insufficient security configurability is present when users of the device have limited or no ability to alter its security controls. Insufficient security configurability is apparent when the web interface of the device has no options for creating granular user permissions or for example, forcing the use of strong passwords. Manual review of the web interface and its available options will reveal these deficiencies.		Insufficient security configurability could lead to compromise of the device whether intentional or accidental and/or data loss.	Consider the business impact if data can be stolen or modified and control over the device assumed. Could your customers be harmed?

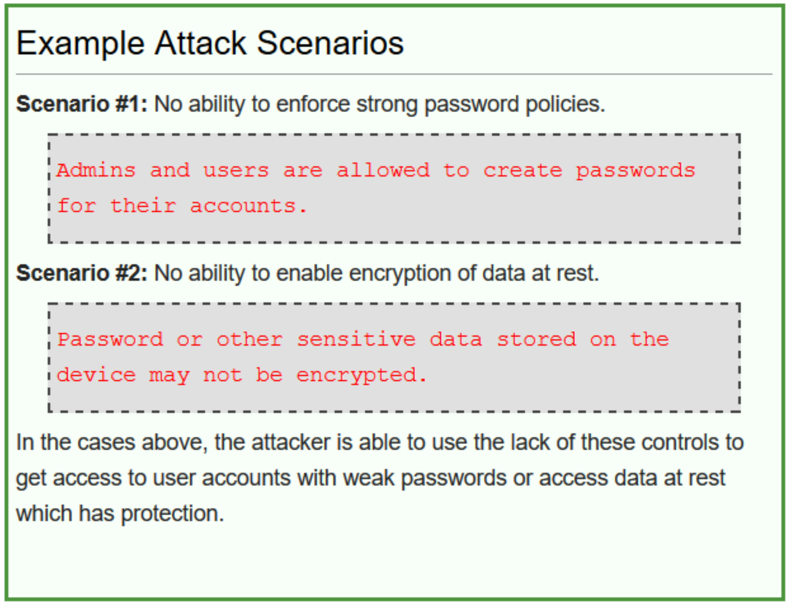
I8 | Insufficient Security Configurability | Testing



Is My Security Configurability Sufficient?

Checking for Insufficient Security Configurability includes:

- Reviewing the administrative interface of the device for options to strengthen security such as forcing the creation of strong passwords
- Reviewing the administrative interface for the ability to separate admin users from normal users
- Reviewing the administrative interface for encryption options
- Reviewing the administrative interface for options to enable secure logging of various security events
- Reviewing the administrative interface for options to enable alerts and notifications to the end user for security events



Example Attack Scenarios

Scenario #1: No ability to enforce strong password policies.

```
Admins and users are allowed to create passwords
for their accounts.
```

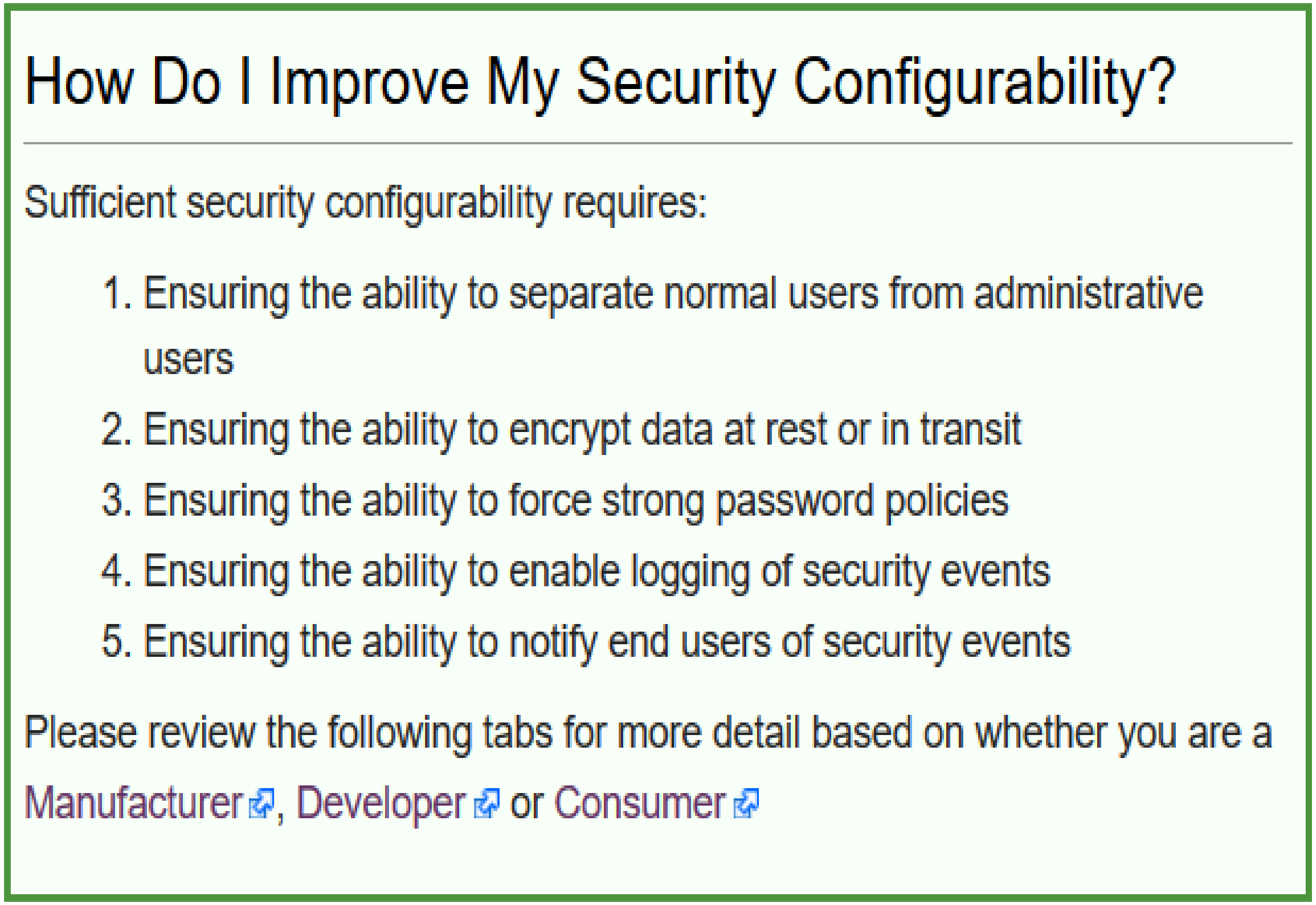
Scenario #2: No ability to enable encryption of data at rest.

```
Password or other sensitive data stored on the
device may not be encrypted.
```

In the cases above, the attacker is able to use the lack of these controls to get access to user accounts with weak passwords or access data at rest which has protection.

- Lack of Granular Permission Model
- Lack of Password Security Options
- No Security Monitoring
- No Security Logging

I8 | Insufficient Security Configurability | Make It Secure



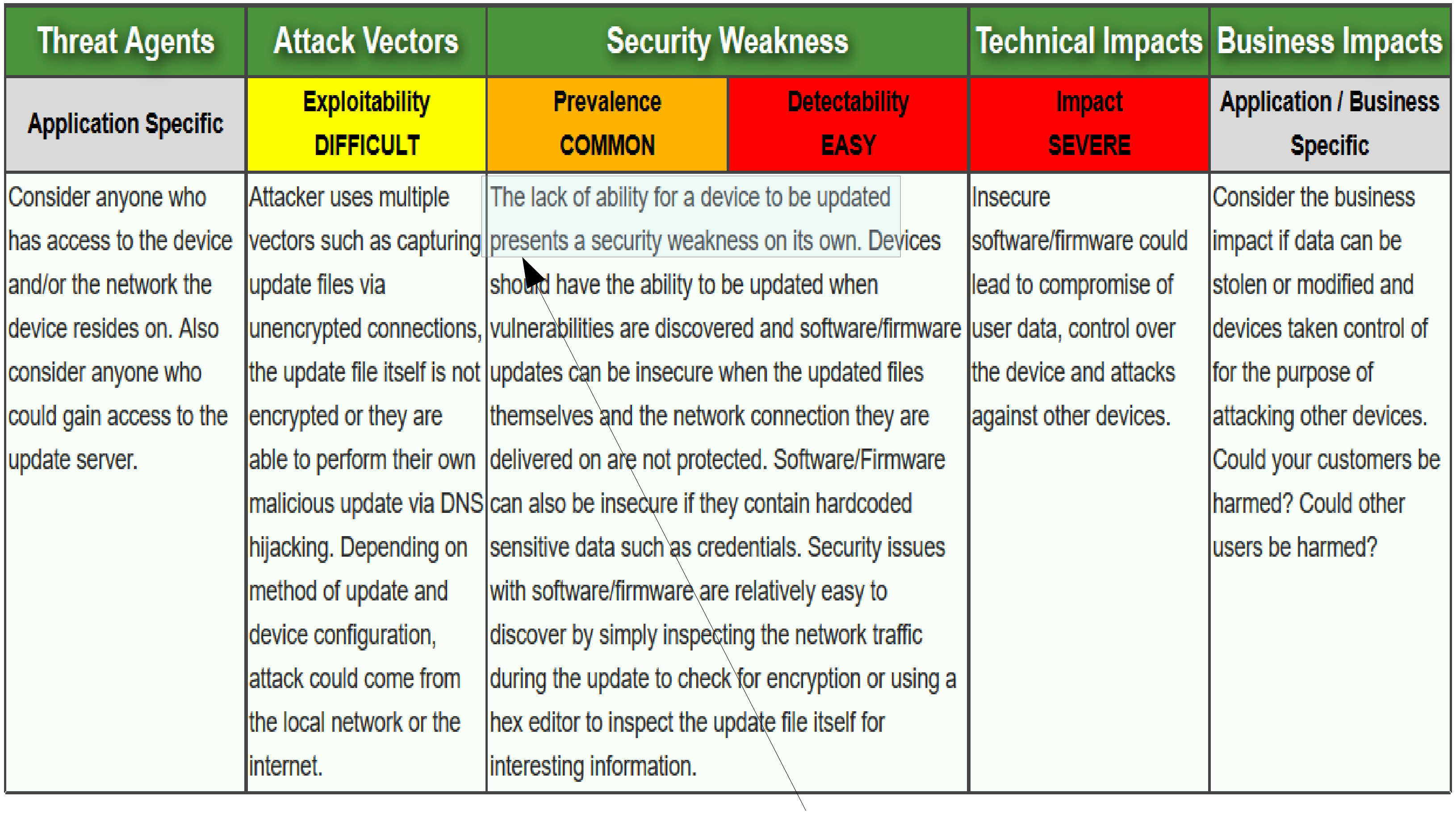
How Do I Improve My Security Configurability?

Sufficient security configurability requires:

1. Ensuring the ability to separate normal users from administrative users
2. Ensuring the ability to encrypt data at rest or in transit
3. Ensuring the ability to force strong password policies
4. Ensuring the ability to enable logging of security events
5. Ensuring the ability to notify end users of security events

Please review the following tabs for more detail based on whether you are a Manufacturer, Developer or Consumer

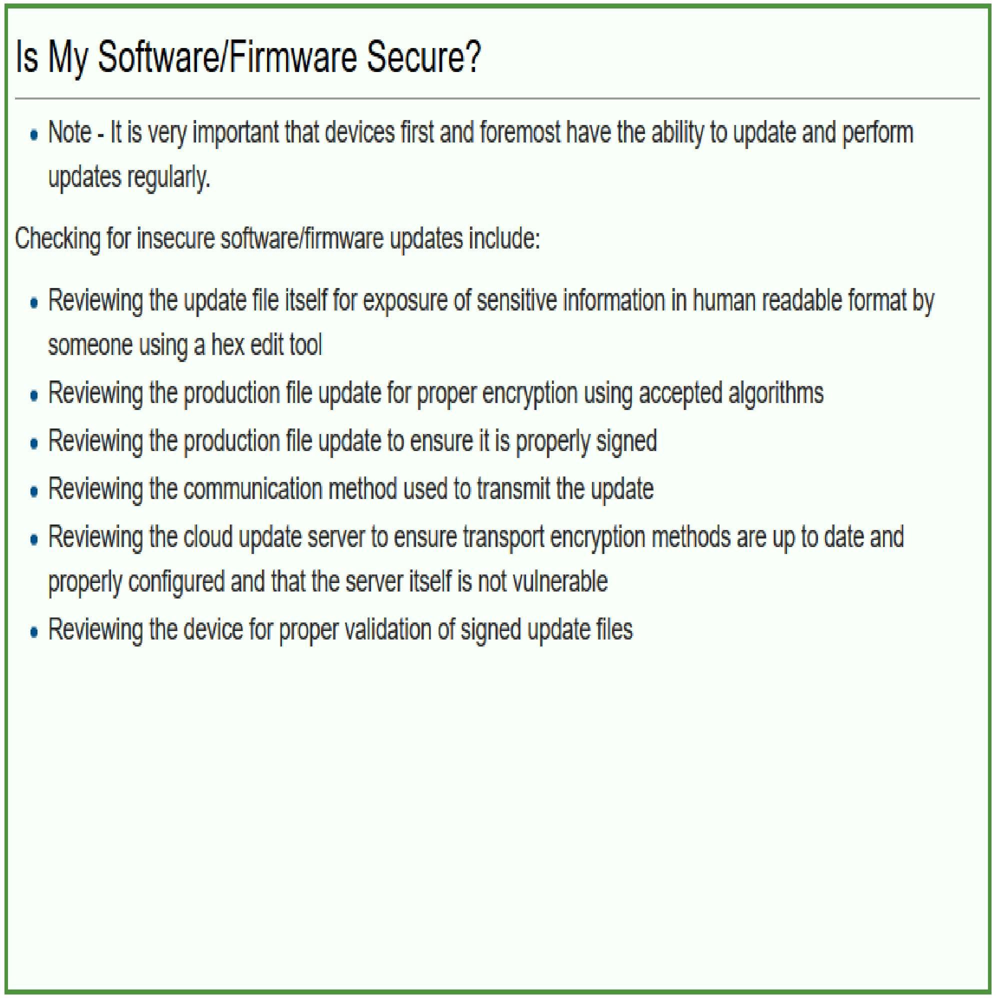
I9 | Insecure Software/Firmware



Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability DIFFICULT	Prevalence COMMON	Detectability EASY	Impact SEVERE	Application / Business Specific
Consider anyone who has access to the device and/or the network the device resides on. Also consider anyone who could gain access to the update server.	Attacker uses multiple vectors such as capturing update files via unencrypted connections, the update file itself is not encrypted or they are able to perform their own malicious update via DNS hijacking. Depending on method of update and device configuration, attack could come from the local network or the internet.	The lack of ability for a device to be updated presents a security weakness on its own. Devices should have the ability to be updated when vulnerabilities are discovered and software/firmware updates can be insecure when the updated files themselves and the network connection they are delivered on are not protected. Software/Firmware can also be insecure if they contain hardcoded sensitive data such as credentials. Security issues with software/firmware are relatively easy to discover by simply inspecting the network traffic during the update to check for encryption or using a hex editor to inspect the update file itself for interesting information.		Insecure software/firmware could lead to compromise of user data, control over the device and attacks against other devices.	Consider the business impact if data can be stolen or modified and devices taken control of for the purpose of attacking other devices. Could your customers be harmed? Could other users be harmed?

Interaction with a remote server, JTAG interface ...

I9 | Insecure Software/Firmware | Testing

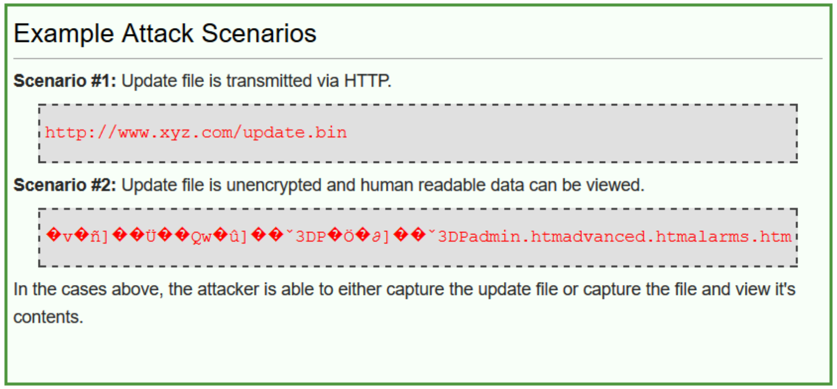


Is My Software/Firmware Secure?

- Note - It is very important that devices first and foremost have the ability to update and perform updates regularly.

Checking for insecure software/firmware updates include:

- Reviewing the update file itself for exposure of sensitive information in human readable format by someone using a hex edit tool
- Reviewing the production file update for proper encryption using accepted algorithms
- Reviewing the production file update to ensure it is properly signed
- Reviewing the communication method used to transmit the update
- Reviewing the cloud update server to ensure transport encryption methods are up to date and properly configured and that the server itself is not vulnerable
- Reviewing the device for proper validation of signed update files



Example Attack Scenarios

Scenario #1: Update file is transmitted via HTTP.

```
http://www.xyz.com/update.bin
```

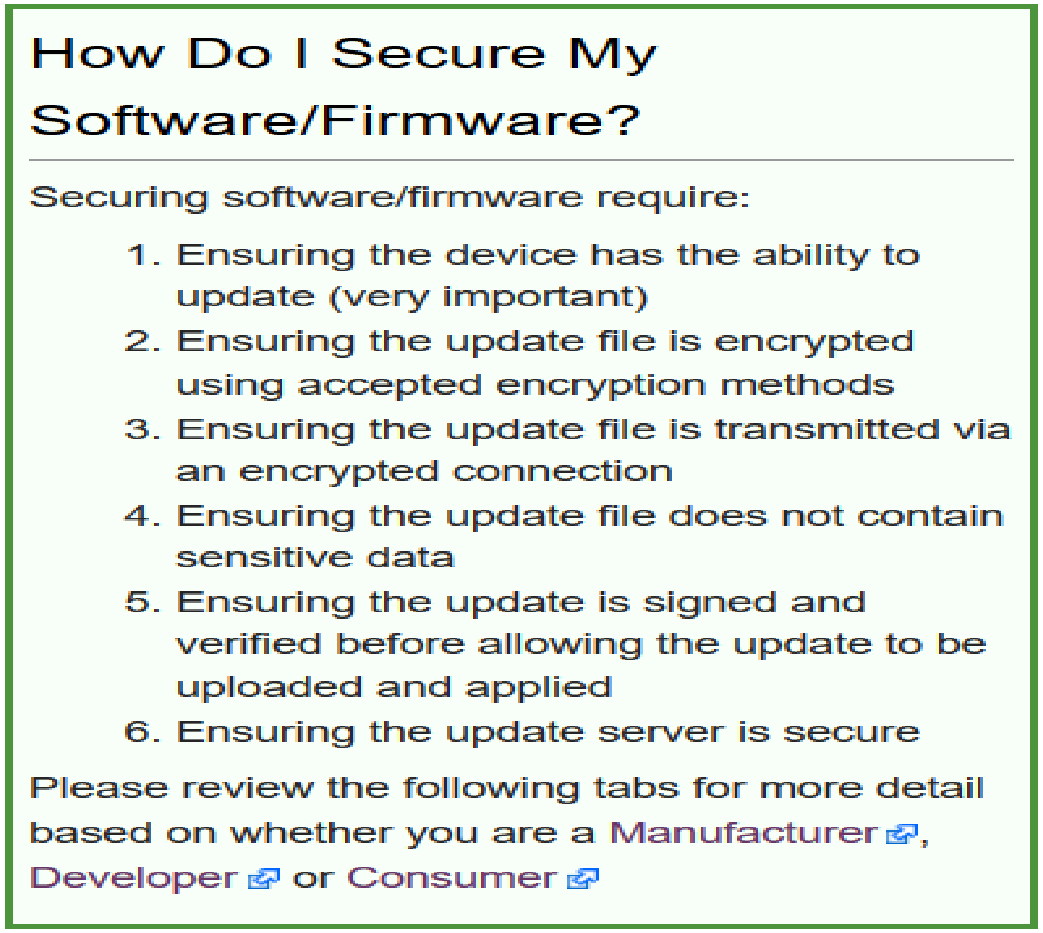
Scenario #2: Update file is unencrypted and human readable data can be viewed.

```
�v�ñ]��Ü��Qw�û]��ˇ3DP�Ö�∂]��ˇ3DPadmin.htmadvanced.htmalarms.htm
```

In the cases above, the attacker is able to either capture the update file or capture the file and view it's contents.

- Encryption Not Used to Fetch Updates
- Update File not Encrypted
- Update Not Verified before Upload
- Firmware Contains Sensitive Information
- No Obvious Update Functionality

I9 | Insecure Software/Firmware | Make It Secure



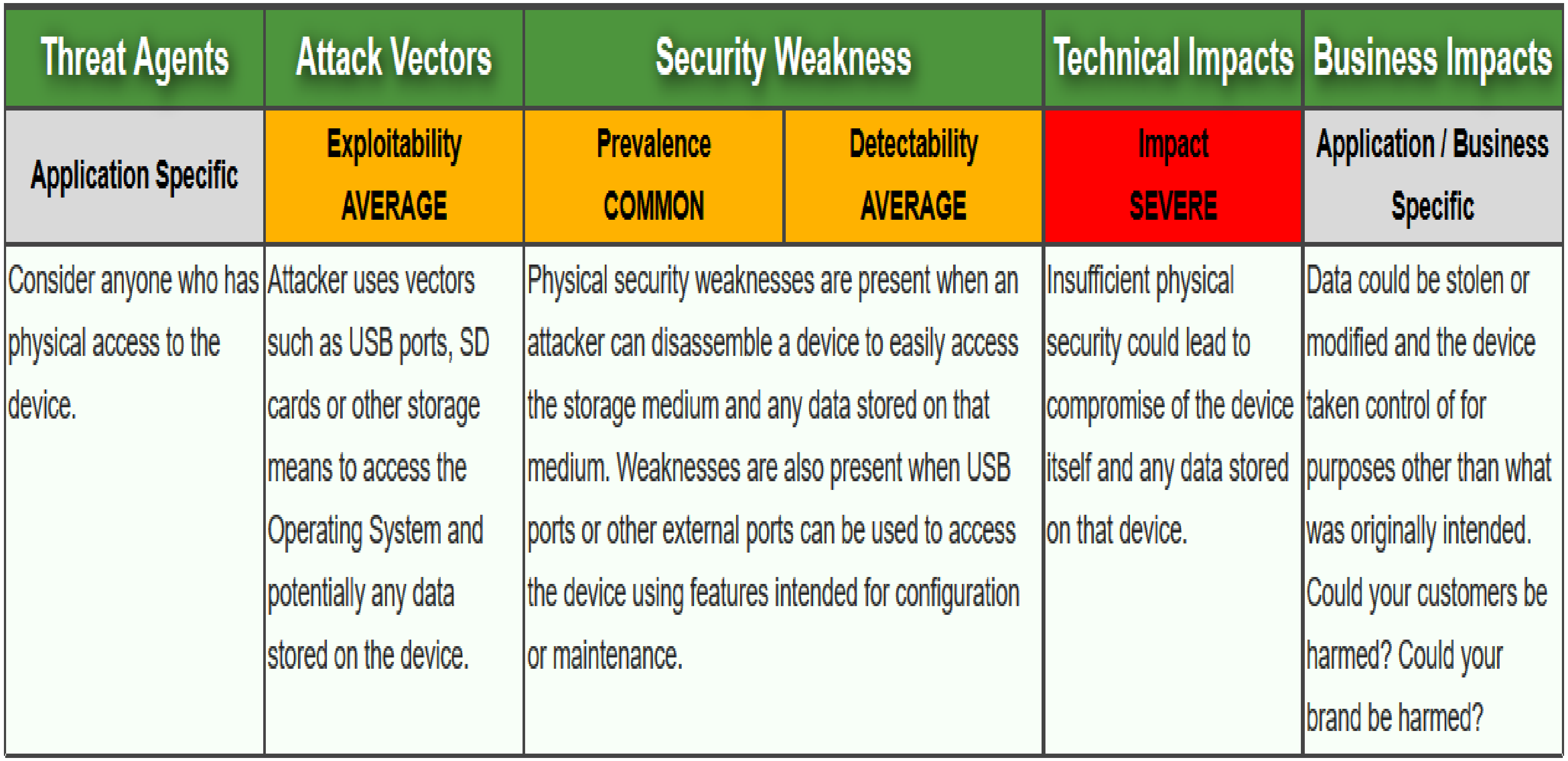
How Do I Secure My Software/Firmware?

Securing software/firmware require:

1. Ensuring the device has the ability to update (very important)
2. Ensuring the update file is encrypted using accepted encryption methods
3. Ensuring the update file is transmitted via an encrypted connection
4. Ensuring the update file does not contain sensitive data
5. Ensuring the update is signed and verified before allowing the update to be uploaded and applied
6. Ensuring the update server is secure

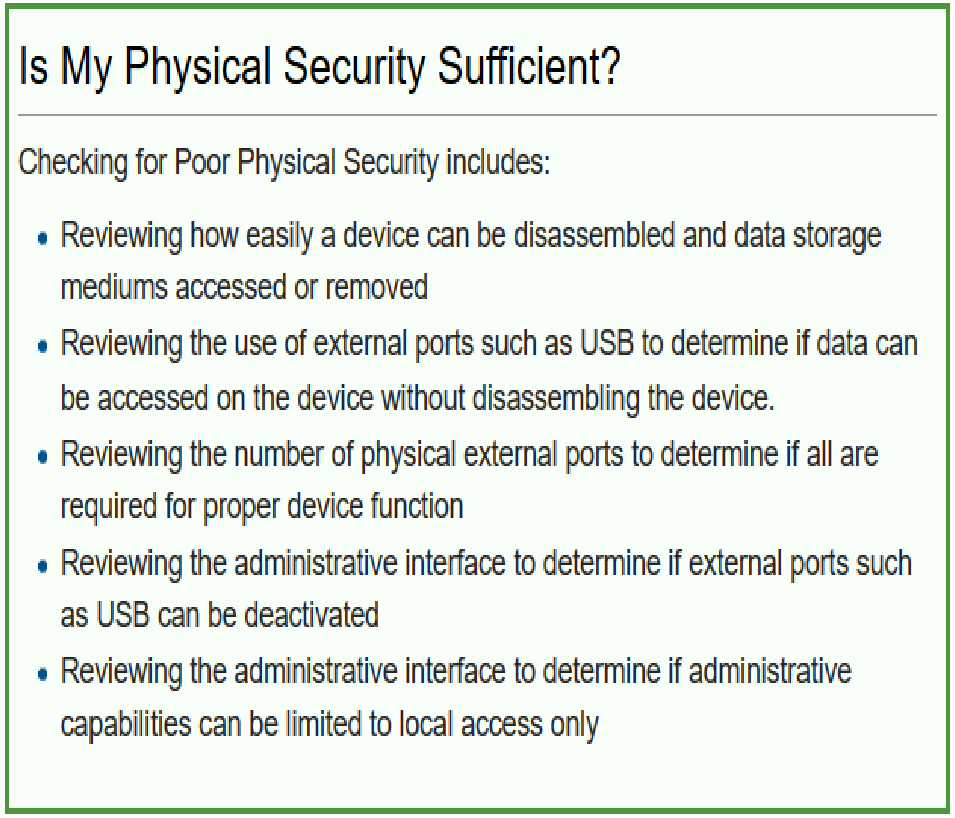
Please review the following tabs for more detail based on whether you are a Manufacturer, Developer or Consumer

I10 | Poor Physical Security



Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	Application / Business Specific
Consider anyone who has physical access to the device.	Attacker uses vectors such as USB ports, SD cards or other storage means to access the Operating System and potentially any data stored on the device.	Physical security weaknesses are present when an attacker can disassemble a device to easily access the storage medium and any data stored on that medium. Weaknesses are also present when USB ports or other external ports can be used to access the device using features intended for configuration or maintenance.		Insufficient physical security could lead to compromise of the device itself and any data stored on that device.	Data could be stolen or modified and the device taken control of for purposes other than what was originally intended. Could your customers be harmed? Could your brand be harmed?

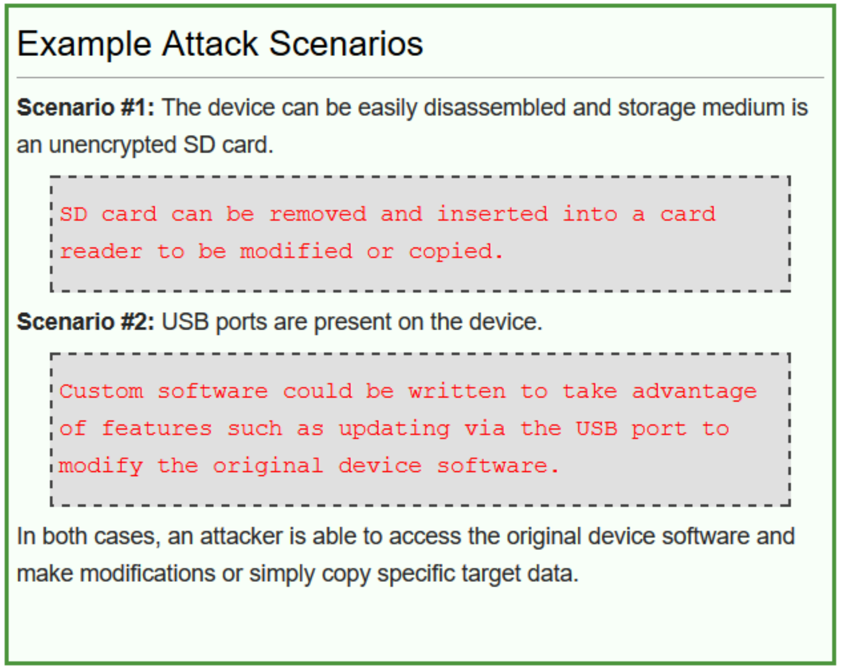
I10 | Poor Physical Security | Testing



Is My Physical Security Sufficient?

Checking for Poor Physical Security includes:

- Reviewing how easily a device can be disassembled and data storage mediums accessed or removed
- Reviewing the use of external ports such as USB to determine if data can be accessed on the device without disassembling the device.
- Reviewing the number of physical external ports to determine if all are required for proper device function
- Reviewing the administrative interface to determine if external ports such as USB can be deactivated
- Reviewing the administrative interface to determine if administrative capabilities can be limited to local access only



Example Attack Scenarios

Scenario #1: The device can be easily disassembled and storage medium is an unencrypted SD card.

```
SD card can be removed and inserted into a card
reader to be modified or copied.
```

Scenario #2: USB ports are present on the device.

```
Custom software could be written to take advantage
of features such as updating via the USB port to
modify the original device software.
```

In both cases, an attacker is able to access the original device software and make modifications or simply copy specific target data.

- Access to Software via USB Ports
- Removal of Storage Media

OWASP IoT framework assessment

- A vendor agnostic set of evaluation criteria for developers and architects to measure relative security strengths of IoT development frameworks.
- A useful benchmark for vendors to produce more robust IoT development frameworks to address the security issues of IoT.
- Evaluation criteria are broken down into four sections that are representative of typical IoT system archetypes. Each section has specific security related concerns that are outlined in the framework evaluation criteria for that section. These sections are:
  - Edge
  - Gateway
  - Cloud Platform
  - Mobile

OWASP IoT FA: Edge

- The physical device that makes up the IoT ecosystem.
- In many deployments it is heterogeneous, meaning it is made up of any number of types of devices with different hardware, operating systems, networking or communications capability and resources.
- An ideal framework will provide cross platform components so that edge code can be deployed anywhere from bare metal, to an embedded operating system, to a mobile OS, to a full blown desktop computer, and so on.

OWASP IoT FA: Edge – I

- Communications encryption
  - Encrypted communications should occur end-to-end wherever possible. Encryption allows endpoints to validate identity to ensure that communications cannot be intercepted or redirected. Some communications may pass through a barrier, a gateway or load balancer, which may impact end-to-end encryption.
- Storage encryption
  - Sensitive data on the edge is liable to theft or exposure unless it is stored with proper security considerations. Frameworks should offer secured local storage to protect data from local malicious applications, compromised operating systems, or malicious owner/operator. Sensitive data can include sensor reading, configuration settings, authentication credentials, or cryptographic keys.
- Strong logging
  - The framework should offer robust logging, including security event logging. The log events should be customizable and should report sensitive events in a usable format for end users, managers, and operators.

OWASP IoT FA: Edge – II

- Automatic updates and/or version reporting
  - The framework should clearly identify the running version and allow for software patches and updates. An automatic updating process frees users from having to manually update systems, raising the likelihood that systems will be kept up to date.
- Update verification
  - Updates should be delivered over a secured channel and verified after download to ensure they are legitimate. Binary signing (and checking) and hashes delivered over a verified, encrypted, channel ensure that malicious updates aren't installed on a device.
  - A physical access may allow an attacker to place a binary directly on a device so updates should be verified prior to installation rather than simply checking a download.
- Cryptographic identification capabilities
  - Frameworks should support cryptographic capabilities to verify trusted components and include lifecycle management. This supports the issuing, and re-issuing, of cryptographic material, expiration of certificates, a revocation and revocation checking mechanism, and a system from signing key material. This enables strong cryptographic authentication, which is particularly important with machine to machine (M2M) authentication and communications encryption.

OWASP IoT FA: Edge – III

- No default passwords
    - The framework should support custom credentials that can be created, set, and reset by the operator. It should eschew default or shared credentials across the ecosystem.
- Strong local authentication
    - The framework should provide strong authentication of operators to the edge. Where possible this should include complex passwords and multi-factor authentication. The authentication mechanism should report or log failed authentication attempts and provide a exponential delay or lock out mechanism to prevent brute force attacks.
- Offline security features
    - The framework should assume that the edge component may lose connectivity and fall back to local security features in the absence of network resources. Offline security features should be just as robust as online features to prevent attackers from disrupting communications so as to degrade security countermeasures.
- Configurable root trust store
    - Cryptographic roots of trust are critical for using certificates for identity validation. These stores should be configurable in order to add new certificates and expire or remove revoked certificates to maintain forward compatible security. The framework should enforce checks on the ability to manipulate the trust root.

OWASP IoT FA: Edge – IV

- Device and owner authentication
    - The framework should recognize that the device may need to authenticate as itself or broker identity of an owner or operator. The framewor identity model should recognize the unique access and authentication needs for both the component and the user(s).
- Transitive ownership considerations
    - IoT devices are often sold or ownership is transfered. The framework should allow the device to be wiped, reset, or otherwise have data compartmentalized or destroyed to protect owner information. Whether the device is a set piece in a physical location whose owner might change, or physically transferable to a hostile or competitive owner, the framework should consider the transitive nature of the device and allow for information protection accordingly.
- Defensive capabilities
    - The framework should provide mechanisms to detect malicious and anomalous activity or integrate easily into device side malware protection or anomaly detection products.

OWASP IoT FA: Edge – V

- Secure M2M capabilities
  - The framework should support machine to machine trust, authorization, verification, and authentication. This support should extend to offline capabilities to avoid a single point of failure in a platform or gateway.
  - The framework might support transitive trust, An owner might certify a number of devices which could then authenticate and trust based on the owner, independent of the device or platform.
- Secure web interface
  - The interface for edge components should addresses the OWASP Top 10 at a minimum. To the extent possible, the web interfaces development frameworks should ensure countermeasures against common vulnerabilities such as authentication bypass, XSS and CSRF.
  - Web interfaces should use TLS (HTTPS) and not use self signed or invalid certificates.

OWASP IoT FA: Edge – VI

- Utilize established, tested networking stacks and protocols
    - This avoid scommon security vulnerabilities in newer, untested, or exotic stacks and protocols. Frameworks should limit the number of protocols to the minimum possible and provide protocols or stacks in a disabled-by-default state to limit attack surface.
- Use latest, up to date third party components
    - Frameworks should use up to date 3rd party components as well as the capability to report on versions and update these components as they age or security updates become available.
- Capability to utilize hardware devices
    - The framework should support the use of any hardware security features such as Hardware Security Modules (HSM's), TPM's, and cryptographic coprocessors. The framework may not require these components, but should utilize them if available.

OWASP IoT FA: Edge – VII

- Support multi-factor authentication
  - The framework should support multi factor authentication
- Support temporal and spacial authentication and functionality
  - IoT devices might be moved and the framework should fine tune permissions based on space and time. It should support location aware permissions utilizing any sensors on an edge device and should also support a permissions model that can change based on rules of time.
- Tracks and contains data from potentially tainted (insecure) sources
  - The framework should allow for some data tagging or sanitization to track and contain untrusted data from channels that cannot be secured..
- Features (interfaces) are disabled by default
  - The framework should strive to disable as many services and features as possible by default, allowing developers and deployment configuration to enable features in order to minimize attack surface.

OWASP IoT FA: Edge – VIII

- Written in a type safe programming language or subject to scrutiny
    - Framework components for edge devices should be written in programming languages that posses security countermeasures and demonstrate a history of strong security. Other components should be scrutinized to remove code vulnerabilities
- Does not employ secrets in code
    - The framework edge components should employ defensive countermeasures to protect any secrets from reverse engineering and physical compromise
- Device monitoring and management capabilities
    - The framework should enable device platform monitoring, and possibly management, capabilities to dececct weaknesses or vulnerabilities in other components on the edge.

OWASP IoT FA: Gateway

- The gateway will often support weak edge devices, or allow edge devices a bridge networks to cloud components.
- Gateways
  - can serve as a communications aggregation and control bottleneck and offer an easy interface between an insecure, but trusted, local network, and a secure connection to the untrusted public internet.
  - will support range limited or proprietary protocols from edge devices
  - have a security critical role because of greater resource availability than edge devices and run full operating system stacks.
  - in many ecosystems the gateway and the edge might be synonymous, with sensors communicating to the edge which brokers those communications into the IoT ecosystem.
- A gateway may, or may not, have any sort of user interface, which can present benefits and limitations to the device.

OWASP IoT FA: Gateway - I

- Multi-directional encrypted communications
  - The gateway should enforce secure communications so as to not degrade the security of messages in any direction wherever possible.
  - A gateway may bridge secured and unsecured communications channels and should consider attacsk on insecure endpoints. It should provide capabilities to segment and isolate communications.
- Strong authentication of components (edge, platform, user)
  - Where possible the gateway should authenticate multidirectionally to the edge and to the cloud. Cryptographic capabilities in gateway authentication should be a strong component of the solution.
- Storage
  - The gateway may be a single point of failure and should store only the minimum amount of information, in an encrypted format if possible

OWASP IoT FA: Gateway - II

- Denial of service and replay attack mitigation
    - The gateway should be able to detect and resist attacks from the edge including spoofing, replay, and excessive communications.
- Logging and alerting
    - The gateway should be able to log and alert based on event logging. It might include integration with standard logging services or intrusion detection systems. The framework may support alternative methods for alerting in the gateway (such as SMS).
- Anomaly detection and reporting capabilities
    - The framework should support the gateway as it is uniquely suited to monitor traffic to and from the cloud and should support anomaly detection or integrate easily with anomaly and intrusion detection systems. A strong gateway might even support intrusion prevention capabilities to exclude suspicious actors from the ecosystem.

OWASP IoT FA: Gateway - III

- Use latest, up to date third party components
    - Frameworks should use up to date 3rd party components as well as the capability to report on versions and update these components as they age or security updates become available.
- Automatic updates and/or version reporting
    - The framework should clearly identify the running version and allow for software patches and updates. An automatic updating process frees users from having to manually update systems, raising the likelihood that systems will be kept up to date.

OWASP IoT FA: Cloud

- The cloud component of an IoT ecosystem
  - refers to the central data aggregation and management portion of the ecosystem
  - will typically consist of a data storage layer (such as a database), analytics and reporting, ecosystem management, a web interface, and other components such as e-mail, backups, etc.
  - may or may not be hosted on public cloud infrastructure.
  - includes a command and control (C2) component to delivery and distribute updates and extensions.
- Access to the cloud component is typically restricted, especially to the supporting infrastructure.
- The cloud component carries significant risk because it is the central point of aggregation for most data and it should contains extensive and effective security controls since it is the keystone of most IoT ecosystems.

OWASP IoT FA: Cloud - I

- Encrypted communications
    - It should support encrypted communications including security certificates to identify itself to other components in the ecosystem and to identify other components as well
- Secure web interface
    - It should be build using technology to avoid common web application vulnerabilities in the code and mitigate the OWASP Top 10 at a minimum.
- Authentication
    - It should support complex authentication including multi factor authentication. The interface should
        - include brute force and anti-account enumeration features
        - not ship with default credentials
        - allow users to easily set, and safely reset, account information.

OWASP IoT FA: Cloud -II

- Secure Authentication Credentials
    - Authentication credentials, in any form should be appropriately salted and hashed, on encrypted, prior to storage
    - Storage mechanisms should be uniformly strong and extend beyond passwords to address machine authentication credentials in any form.
- Encrypted storage
    - The cloud component is often the system of record and aggregation for the entire deployment. Wherever possible the framework should support data encryption at rest as well as in any export or backup mechanism.
- Capability to utilize encrypted communications to storage layer
    - Communications between the cloud interface and data aggregation layer and the data persistence layer should be encrypted. The framework should encrypt communications by default

OWASP IoT FA: Cloud -III

- Data classification capabilities and segregation
  - This component will collect a variety of data from other components. The framework should support data classification and protect data dependent on classification. Interface controls should limit access and exposure.
- Security event reporting and alerting
  - Due to the greatest visibility into ecosystem function, security controls are critical at this layer. For this reason, it should
    - have robust security event monitoring, reporting, and alerting capabilities.
    - detect and react to malicious activity
    - segregate bad actors, limit access to malicious parties, and integrate easily with third party logging and intrusion detection and prevention systems.

OWASP IoT FA: Cloud -IV

- Automatic updates and update verification
  - The framework should support easy (automatic)updates and update verification of the cloud component. The framework should have an easy interface for reporting versions and any available updates. Automated alerting of updates out of band (SMS or e-mail) is desirable for non-automatic updates
- Use latest, up to date third party components
  - Frameworks should use up to date 3rd party components as well as the capability to report on versions and update these components as they age or security updates become available. Any updates should be distributed over a secured channel and verified before installation.

OWASP IoT FA: Cloud -V

- Plugin or extension verification, reporting and updating
  - The component will often have enhancements and customization options in the form of extensions and plug-ins that should be monitored and updated in a modular. The framework should ship with a minimal set of features enabled by default to limit attack surface. An easy administrative interface for extensions and plug-ins should be available.
- Interface segregation and isolation based on utility
  - The component will often communicate with various other components. The one to communicate with an embedded device will necessarily differ from a web interface. Communications channels should be protected segregated and protected to enforce least privilege to limit access based on role and use.

OWASP IoT FA: Cloud -VI

- Application level firewall and defensive capabilities
    - The component should block some actors, throttle malicious activity and respond to threats. It should also perform mass credential resets, deprecations, and other disaster and breach response actions
- Ensure ecosystem segregation in the case of multi-tenant solutions
    - The framework should provide appropriate segregation and data protection for distinct customer, e.g. dedicated data storage layers per customer, or tagging

OWASP IoT FA: Cloud -VII

- Stack security considerations (no web UI to execute arbitrary code)
  - The framework should support full stack security countermeasure in the cloud component. This includes integrations on all layers of the cloud component, and with cloud provider security countermeasures. The component should include secure configuration management and automatic updates.
- Audit capability
  - Frameworks should provide mechanisms to ensure delivery of targeted messages to specific edge components. This will allow confidence in audit and to support delivery guarantees of security sensitive instructions or data. This audit should be bidirectional

OWASP IoT FA: Mobile

- Mobile interfaces in IoT deployments vary in capabilities and integration. While some applications merely provide limited data reporting from specific edge devices, others allow for the manipulation of edge components, and still others provide a full view analytics and cloud management capabilities.
- Particular care and attention should be paid to mobile components in IoT ecosystems since they may be deployed beyond the boundaries of device management, can grant privileged access to alter, adulterate or expose sensitive information, may have the capability to actuate edge devices, and can easily fall into malicious hands.
- Mobile components may carry many of the same risks as cloud components but are often given less security consideration and are exempt from the robust physical and access security controls that can be placed on cloud components.

OWASP IoT FA: Mobile - I

- Authentication requirements equal or greater to other components
  - The framework should ensure that mobile authentication mechanisms don't degrade auth requirements.
- Local storage security considerations
  - The threat of theft or loss also means that local storage could fall into malicious hands. The framework should strictly limit the amount of data stored on the device and the data should be encrypted where possible.

OWASP IoT FA: Mobile - II

- Capabilities to disable or revoke mobile components in the case of theft or loss
  - The framework should support an easy deprovisioning of mobile components quickly and easily to support response to mobile device theft or loss.
- Strong audit trail of mobile interactions
  - Because a mobile device might fall into malicious hands a security audit trail of its application interactions should be preserveed . The framework should support robust logging to track interactions from mobile components to support forensics if mobile devices were used maliciously after the fact.

OWASP IoT FA: Mobile - III

- Mobile should perform cryptographic verification and validation of other components
  - Where possible a mobile should support cryptographic verification and validation of the other components during interactions. Proper certificate checking and authentication should always take place.
- Encrypted communications channels
  - Mobile application should operate under the assumption of a hostile observer who will attempt to inspect, interdict, interrupt, replay and manipulate traffic.

OWASP IoT FA: Mobile - IV

- Multi-factor authentication
  - To perform multiple factors of authentication, the framework should adpot sensors and biometrics for extended security checking on the mobile platform.
- Utilize mobile component to enhance authentication and alerting for other components
  - Where possible the mobile component should integrate into authentication and alerting for events at other components. Edge, gateway, or cloud components might alert to the mobile framework, or the mobile framework might allow for multi factor authentication or enhance authentication to other components.