ICT Risk Assessment

Fabrizio Baiardi
f.baiardi@unipi.it

Syllabus

Definition of attack and countermeasures for the internet of things

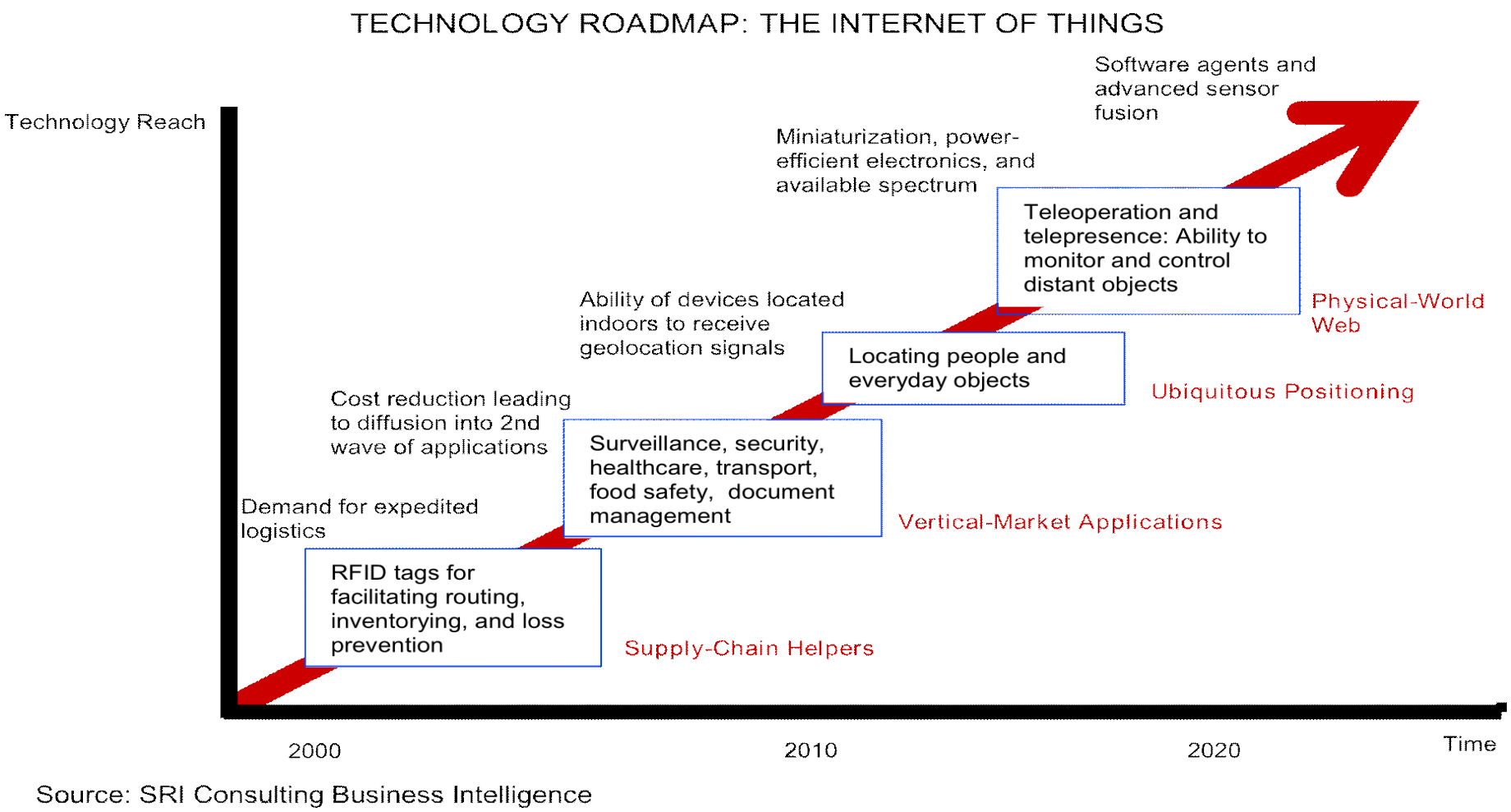
Internet of things

- Oxford dictionary 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* = 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)

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.
- Applications range from medical sensors, medication tracking, physical activity tracking, home heating, lighting and appliance intelligent 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



Source: SRI Consulting Business Intelligence

IOT Device Hardware

- IoT devices contain *sensors*, *actuators*, 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* Things through hardware in the device, 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. The data processing component of the IoT device coordinates data from multiple sensors or stores in flash memory (for whatever reason),

IOT Firmware - 1

The onboard software 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, i.e. the software that runs on the device. In many cases, the only cost-effective solution for manufacturers is to write embedded firmware to interact with the hardware.
- Embedded software engineers write 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

IOT Firmware - II

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

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) write device drivers,while application programmers (who do not need to understand the hardware) write the software that makes the device "smart".
- A popular OS choice for many manufacturers is Busybox, a stripped down version of Unix 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 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 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 Attack

- IOT Security is most often an afterthought because it is difficult to create a cheap reliable, resource-constrained device that can connect to a wireless network, use very little power. Security is the last thing to be considered
- Attack vectors
  - **Weak Password** To simplify the device setup and use, the manufacturer offers some login e.g. A single pair userid/password combination.
  - **Lack of encryption** Many IoT devices do not support encryption,
  - **Backdoor** Manufacturers put "hidden" access mechanisms to simplify the support . 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 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 scanning computer
- Never expose a device directly to the internet
  - When you consider 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 device classification - I

IoT Nodes may be classified by performance or functionally

Performance based classification:

1. Ultra-constrained node. RTOS or bare metal with 16K of RAM. Energy harvesting limits radio transmissions to conserve power.
2. Constrained node. RTOS with 32K-64K RAM. Most likely running on a battery and software optimised for battery life. Again minimizing radio transmission
3. Mainstream node. A feature rich RTOS with 128K RAM. More complex interaction with the context 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 + advanced software and runs from main power. Multiple radios to support the local network.

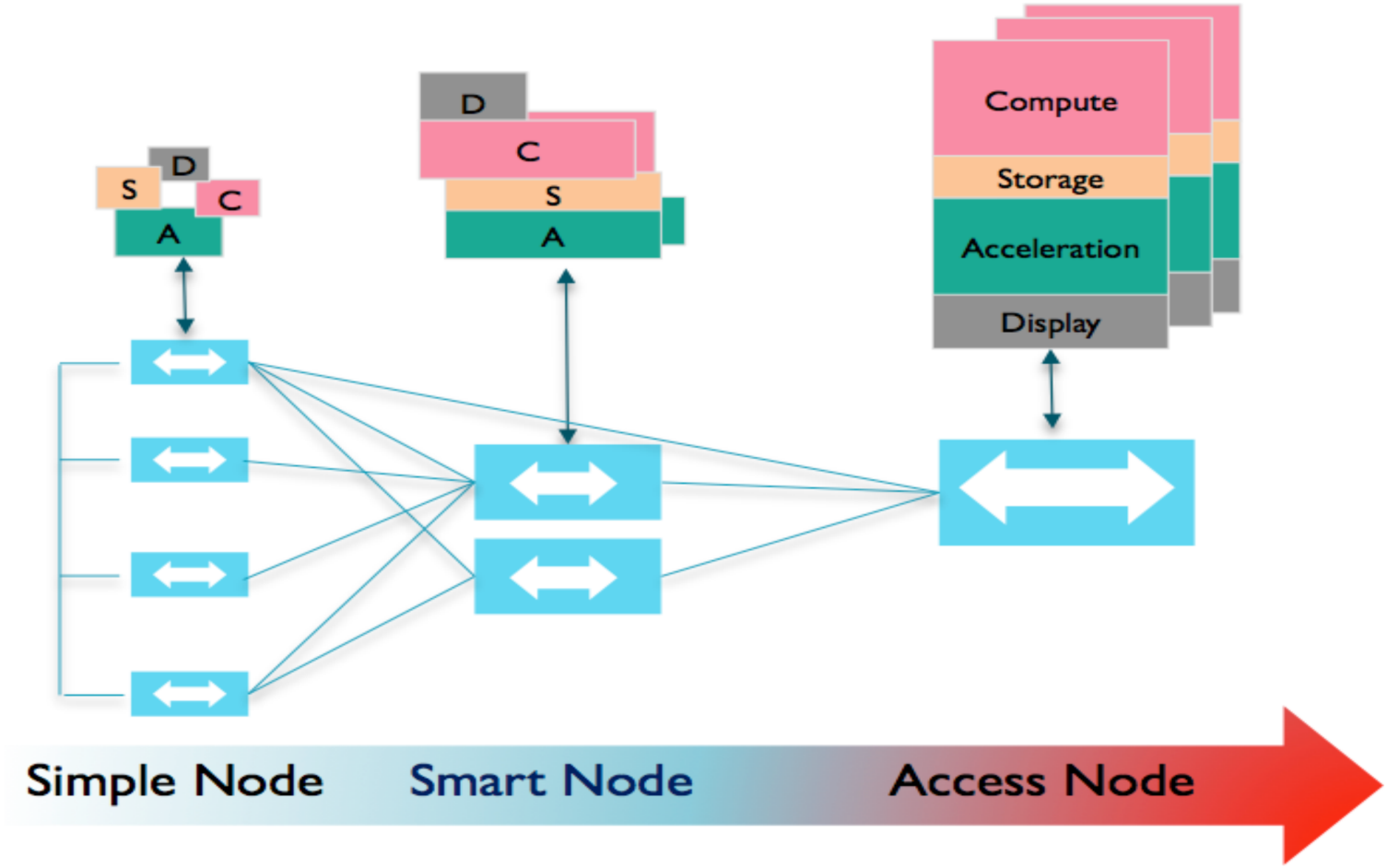
A device classification - II

Functional based classification:

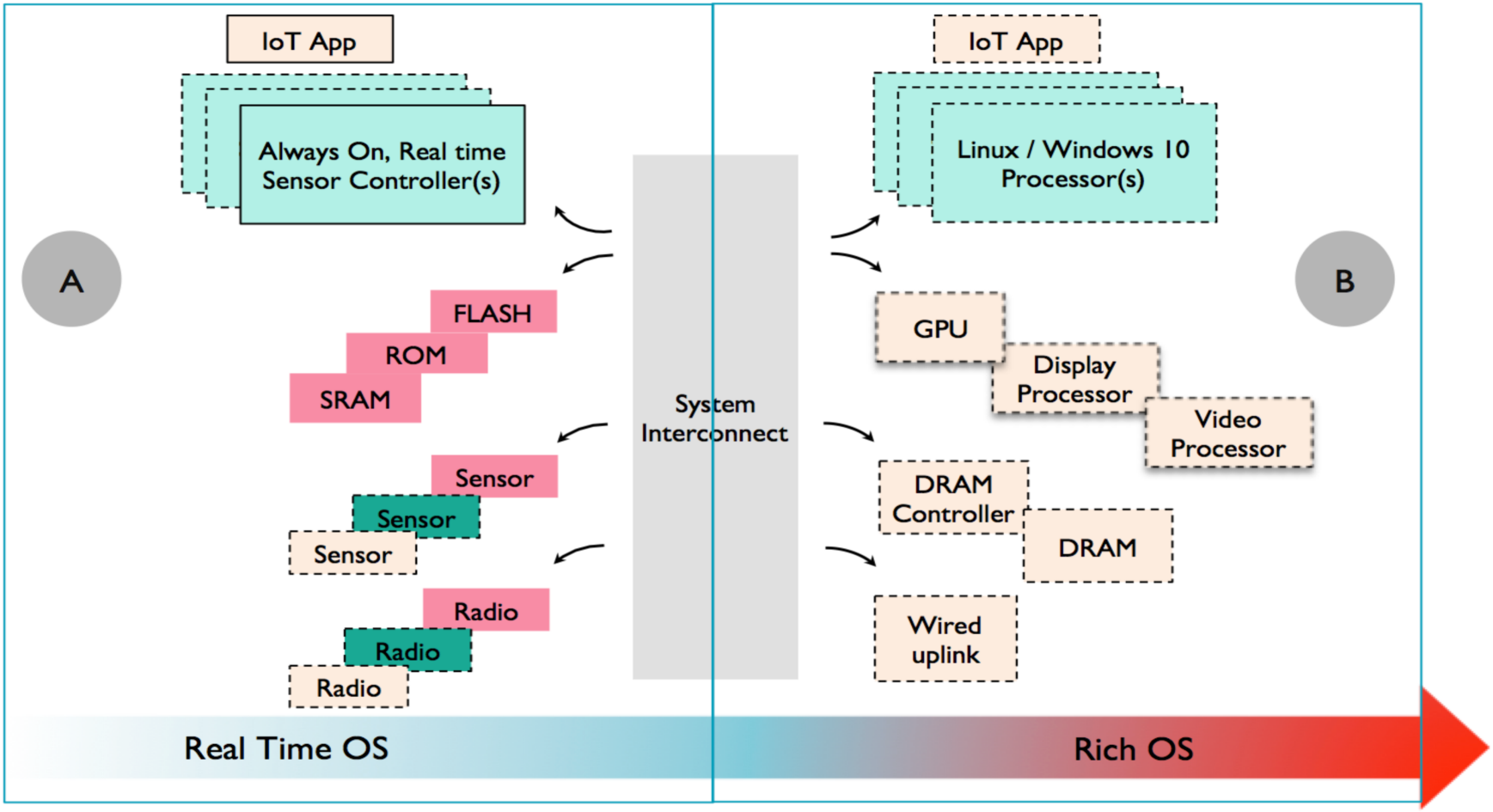
1. Simple node: not aware of the rest of the local network. It collects and reports information to the specified destination. Any of previous 1-3 may be a simple node.
2. Smart node: fully aware of all other network nodes mainly via software that understands mesh networks, local topologies and authorised interactions between nodes in the same network.
3. Access node: the edge box to connect the local network to the Internet via whatever broadband link is 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 device classification - III



Architecture of an IOT node



Node Architecture

- Software productivity requires 32 bit real time embedded processor to quickly develop and release software to handle connectivity, security and IoT applications while managing and controlling multiple sensors
- Software productivity is important since one node design may be used to release products for 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 system and network stack This reduces power consumption since the right amount of processing power is activated at any moment to handle the task of the moment. No need to wake up the main processor if the radio processor may check for incoming message
- It is very easy to use multiple processors of the same or different modelsince the same bus connects to the system fabric

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, the intended operation and software complexity
- Some designs exploit many segments since one memory size fits many applications and extra space simplifies future expansion.
- Only when an application requires a substantial memory increase, one creates distinct products to avoid burdening the smaller one with extra memory that definitely remains unused.

An important opinion (Bruce Schneier)

Security in a world where everything is connected

(no connected car but a computer with four wheels, no connected refrigerator but a computer that keep things cold)

1. Most software is poorly written
2. Internet was never been designed to be secure
3. Extensibility is a problem = A computer is an universal Turing machine
4. Complexity helps the attacker
5. New vulnerabilities to interconnection
6. Attacks always get faster, smarter and cheaper

Changes in the scenario

- Automation, autonomy, physical agents
- Patching becomes much more complex or impossible
- Integrity and availability become more important than confidentiality (leaking health records vs hacking a body device)
- The same vulnerability and the same attack work both in ICT and internet of things

and Schneier final opinion is “nothing new”

Nothing new?

While it is true that old vulnerabilities strike even in IOT the physical nature of IoT, the number of devices, the existence of preloaded code may result in something new

Two prospectives on IOT Security

1. Any smart device increases your system attack surface and its connection can result in new attacks (attacks to the device, device as an intermediate step)
2. Any smart device can store some malware to attack your system (attacks from the device)

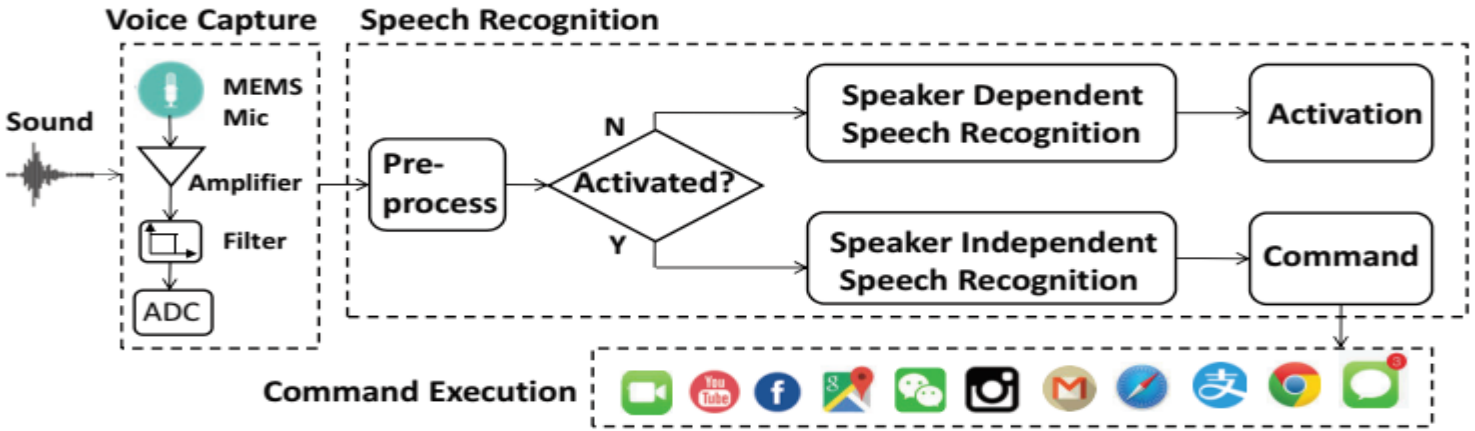
- While most of security research has focused on the first issue, the second one is becoming more and more critical
- Internet 4.0 includes a huge number of devices with code that cannot be easily accessed and tested
- Fuzzing and reverse engineering will become more and more important to preserve confidentiality and integrity

IoT : A new perspective on attacks

- The interaction of an IoT device with the physical world enables to exploit physical properties to attack an ICT system in new 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, Cortana and Alexa. Proof-of-concept attacks are activating Siri to initiate a FaceTime call or Google Now to switch the phone to the airplane mode, and manipulating the navigation system in an Audi car

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 target characteristics are public
- **No Owner Interaction.** 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. The attacker cannot ask owners to perform any operation, ie pressing a button or unlocking the screen.
- **Inaudible. T**o inject voice commands without being detected, the attacker will use sounds inaudible to human, i.e., ultrasounds ($f > 20$ kHz). No frequencies from 18 kHz to 20 kHz, still audible to kids
- **Attacking Equipment.** 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, a remote controllable speaker is around the victim's desk or home or someone carrying a portable speaker while walking by the victim

IoT : A new perspective – Threat Device

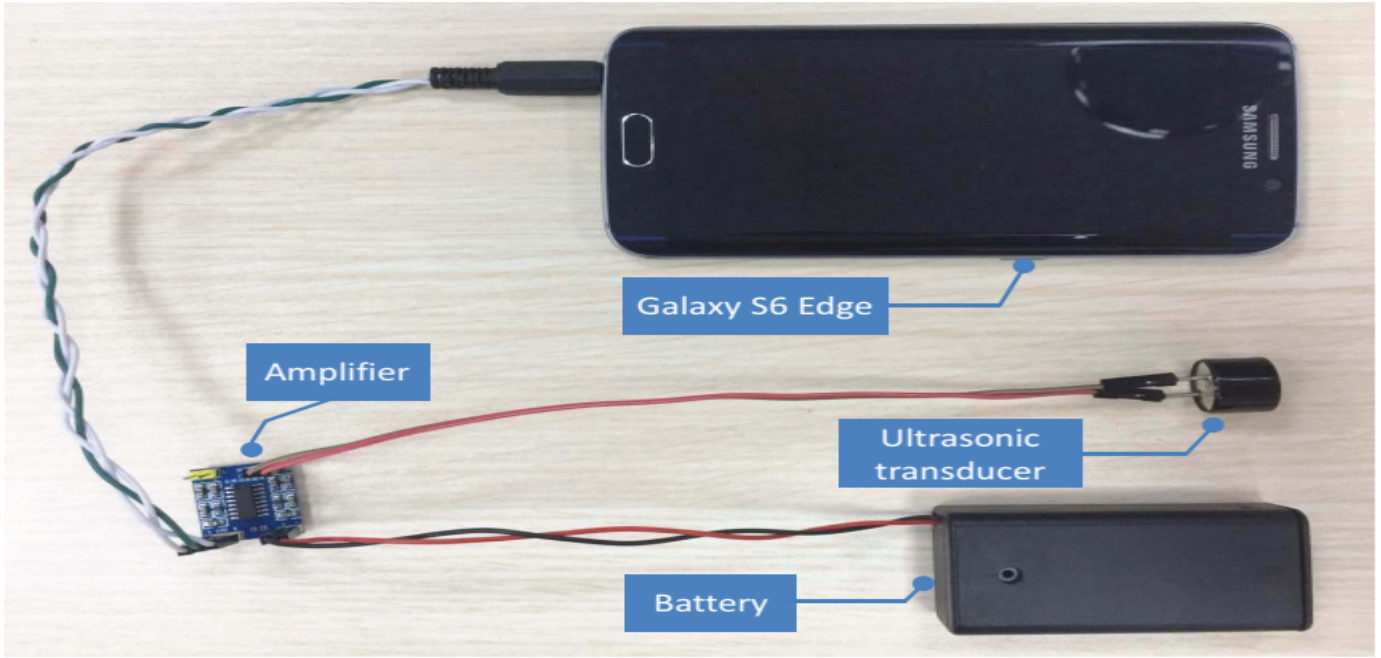
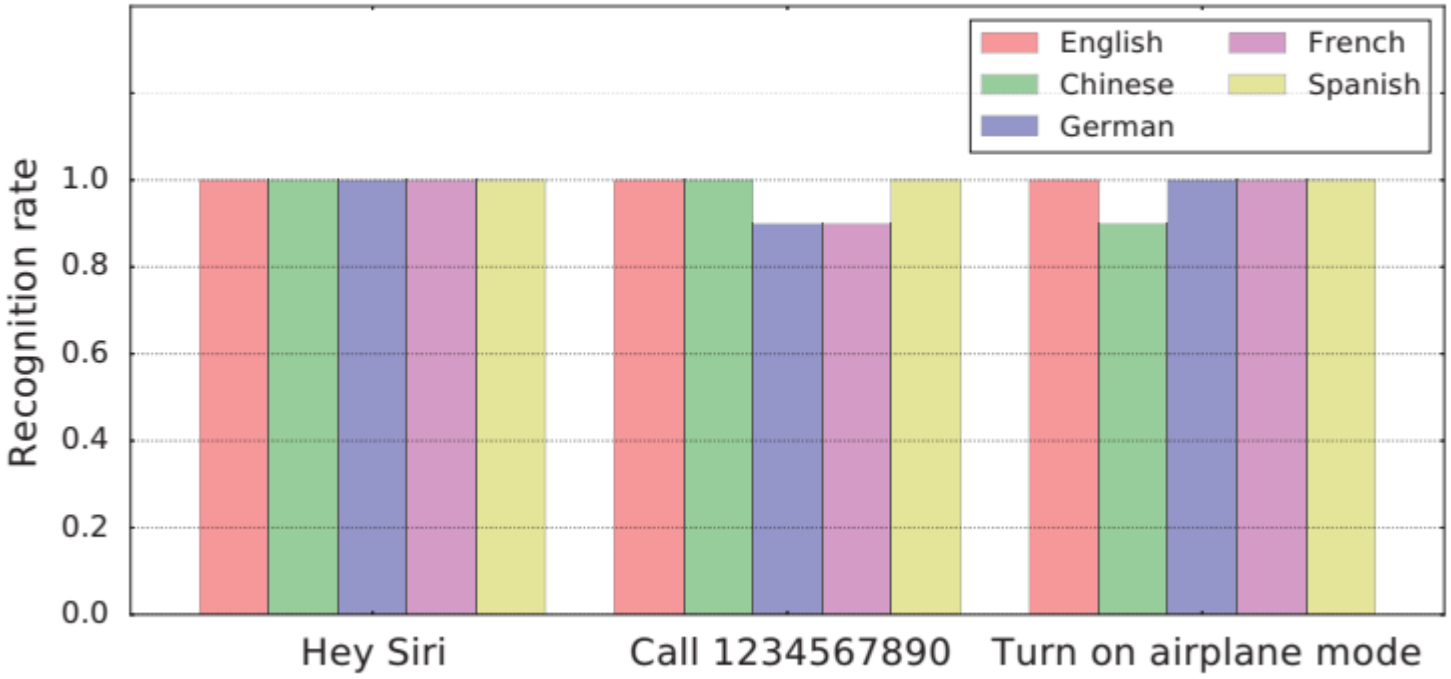


Figure 11: Portable attack implementation with a Samsung Galaxy S6 Edge smartphone, an ultrasonic transducer and a low-cost amplifier. The total price for the amplifier, the ultrasonic transducer plus the battery is less than $3.

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 taxonomy

- A proposed taxonomy on attacks 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 any physical functionality of the IoT device, and considers it only as a standard computing device connected to the LAN and/or to the internet.
- For example, the attacker combine compromised IoT devices into a botnet to send spam or 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 will be many cheap devices from a variety of small companies with minimal security protections, and that cannot be upgrade or patched
- These attacks are a serious threat but the least interesting ones because they are applicable to any networked device and are not unique to IoT devices. (nothing new)

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.
- Some consequences of lost functionality are more serious, e.g. 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 (integrity more important)

Misusing the functionality

- These attacks use rather than destroys a functionality of the physical device, but 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 turns on the lights and open all the faucets as soon as the user leaves for a long vacation.
- However, most of these attacks are likely to be irritating pranks rather than serious problems (integrity is more important).

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.
- Some unexpected applications of connected LEDs, that have been used to transmit data by changing the light intesity (something new)

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 may happen …

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 th 'Smart' features, like the learning thermostat or the smoke detector that emails you when the battery needs changing.

Each piece 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.

If your smart lights are off, your thermostat has turned down your central heating and all your doors and windows are locked its pretty obvious you aren't at home and the history of your thermostat setting shows when you are likely to return.

There is a big neon sign to advertise when you are out.

IOT = a fridge that sends spam

In 2014 a fridge has been sending out spam after a web attack compromised smart gadgets. More than 100,000 devices 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 campaign and were routed through the compromised gadgets.

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

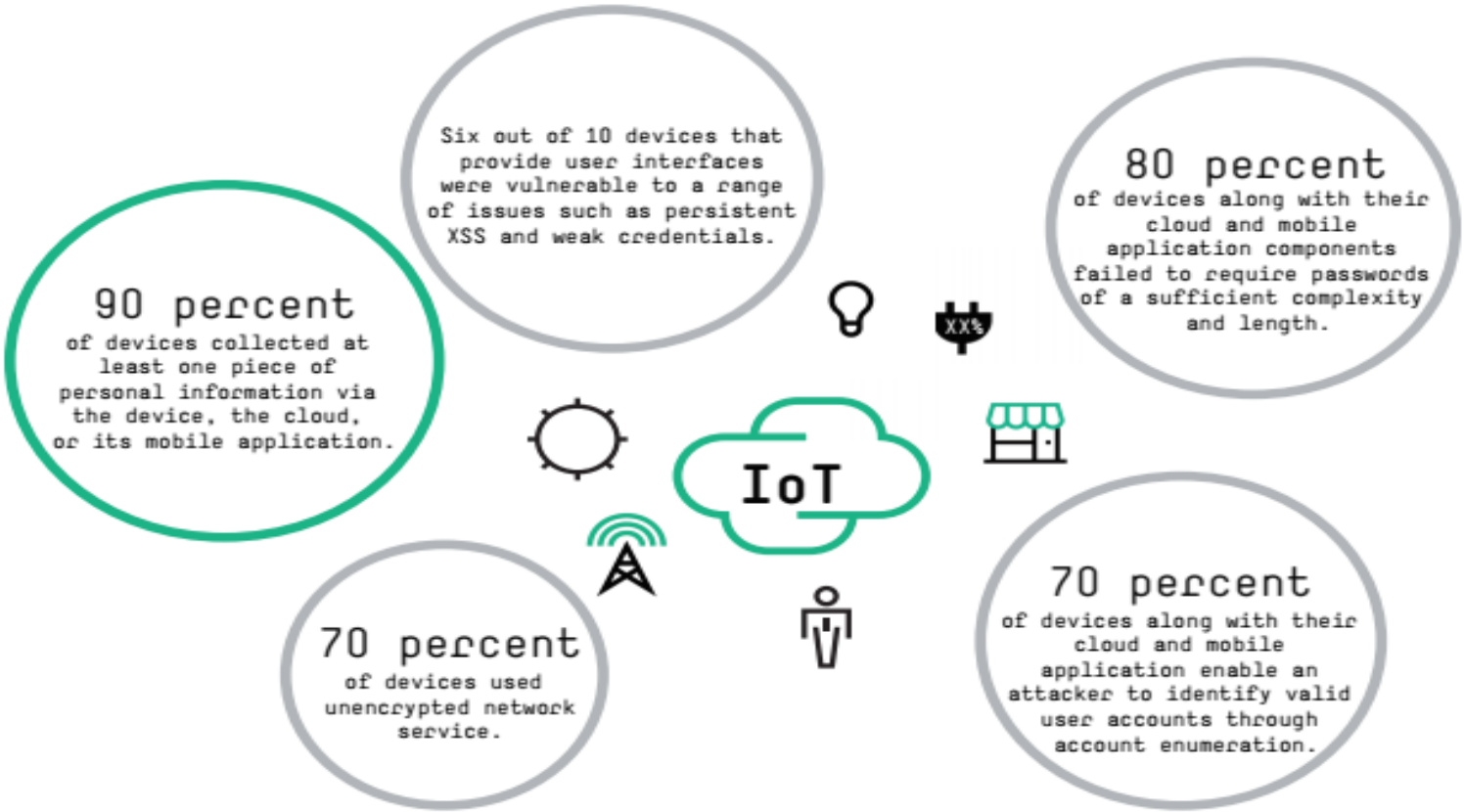
Many of these gadgets have computer processors onboard and act as a self-contained web server to handle communication and other sophisticated functions.

What has happened: the Mirai botnet

Composed mainly of embedded and IoT devices, In late 2016 it overwhelmed high-profile targets with massive DDoS attacks

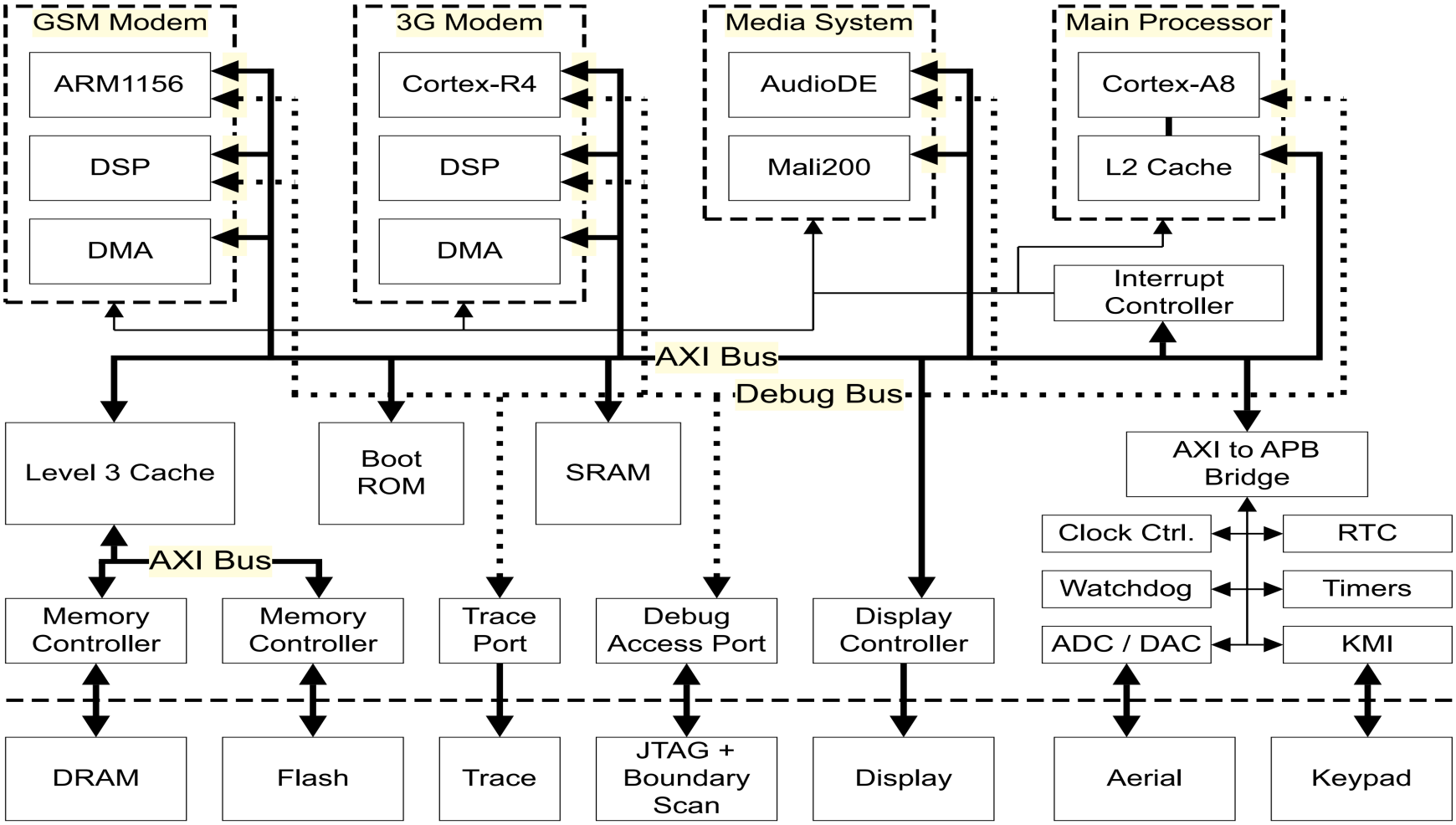
- Node scans IPv4 address space for devices that run telnet or SSH, and attempt to log in using a hardcoded dictionary of 46 unique passwords, some traceable to a vendor. It primarily targeted IP cameras, DVRs, and consumer routers
- The bot sends the victim IP address and credentials to a server, which 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
- Mirai launched 15,194 attacks between September 27, 2016–February 28, 2017. These include Application-layer and Volumetric attacks, and TCP State exhaustion, all of which are equally prevalent

What has happened …



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

IOT - 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 multiple threats
- TrustZone technology does not provide a fixed one-size-fits-all security solution, but an infrastructure foundations so that a SoC designer can choose from a range of components that can fulfil specific functions within the security environment.
- The main security architecture goal is simple;

  enable the construction of a programmable environment to protect from attacks to confidentiality and integrity of almost any asset
- A platform with these characteristics can be used to build a wide set of security solutions which are not cost-effective with traditional methods

TrustZone: The three features

1. A partition of all hardware and software resources in one of two worlds
   - Secure world for the security subsystem,
   - Normal world for everything else.

   Hardware logic ensures a strong security perimeter between the two ensures that Normal world components cannot access Secure world ones. By placing sensitive resources in the Secure world, and by robust software on secure cores, we protect almost any asset against possible attacks (as in Intel Enclaves)
2. Extensions in the processor cores 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
3. A security-aware debug infrastructure which can enable control over access to Secure world debug, without impairing debug visibility

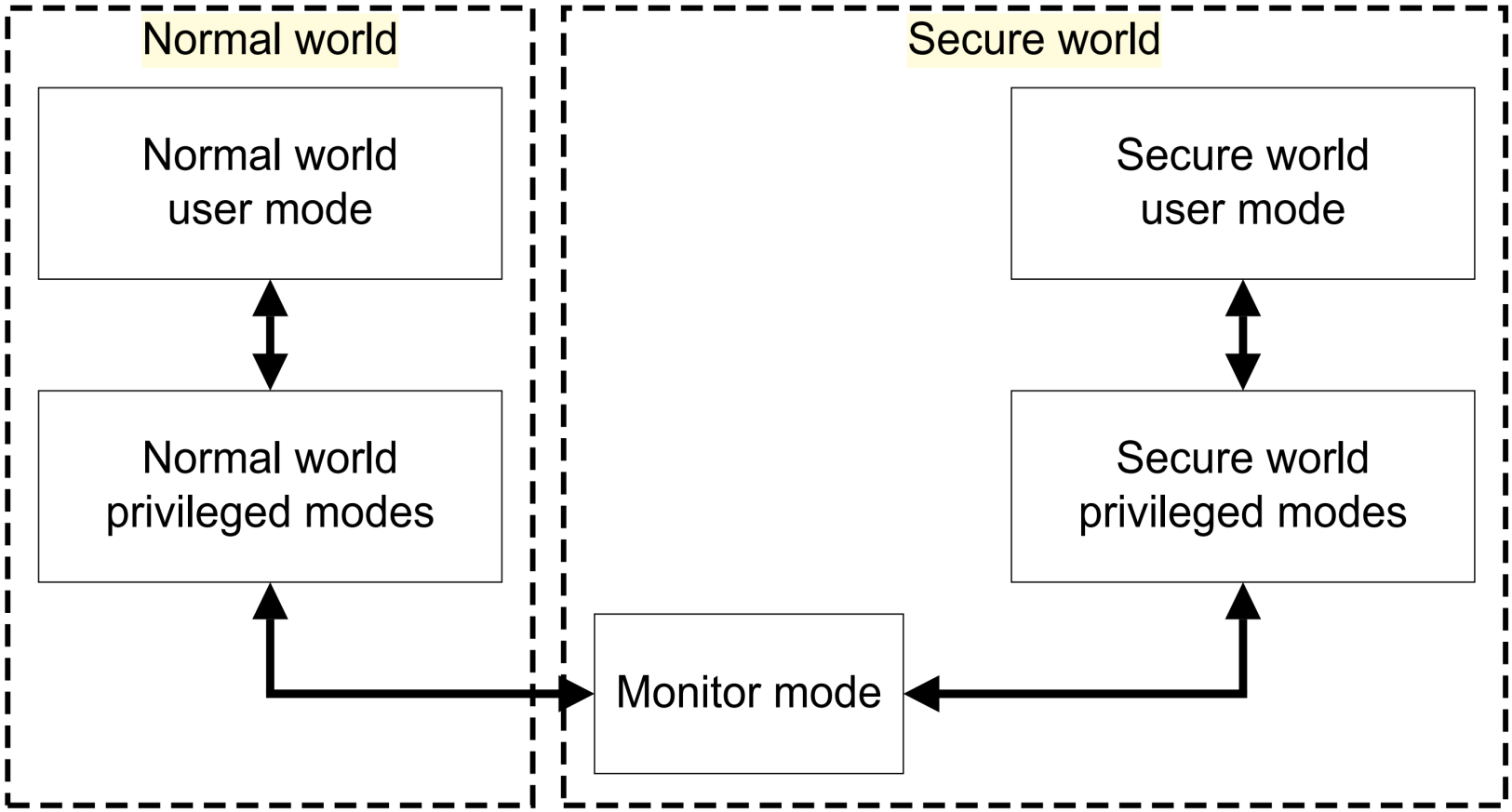
System architecture – system bus

- 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 on a new transaction, and the bus or slave decode logic interpret them to ensure 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 NS bit value 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.

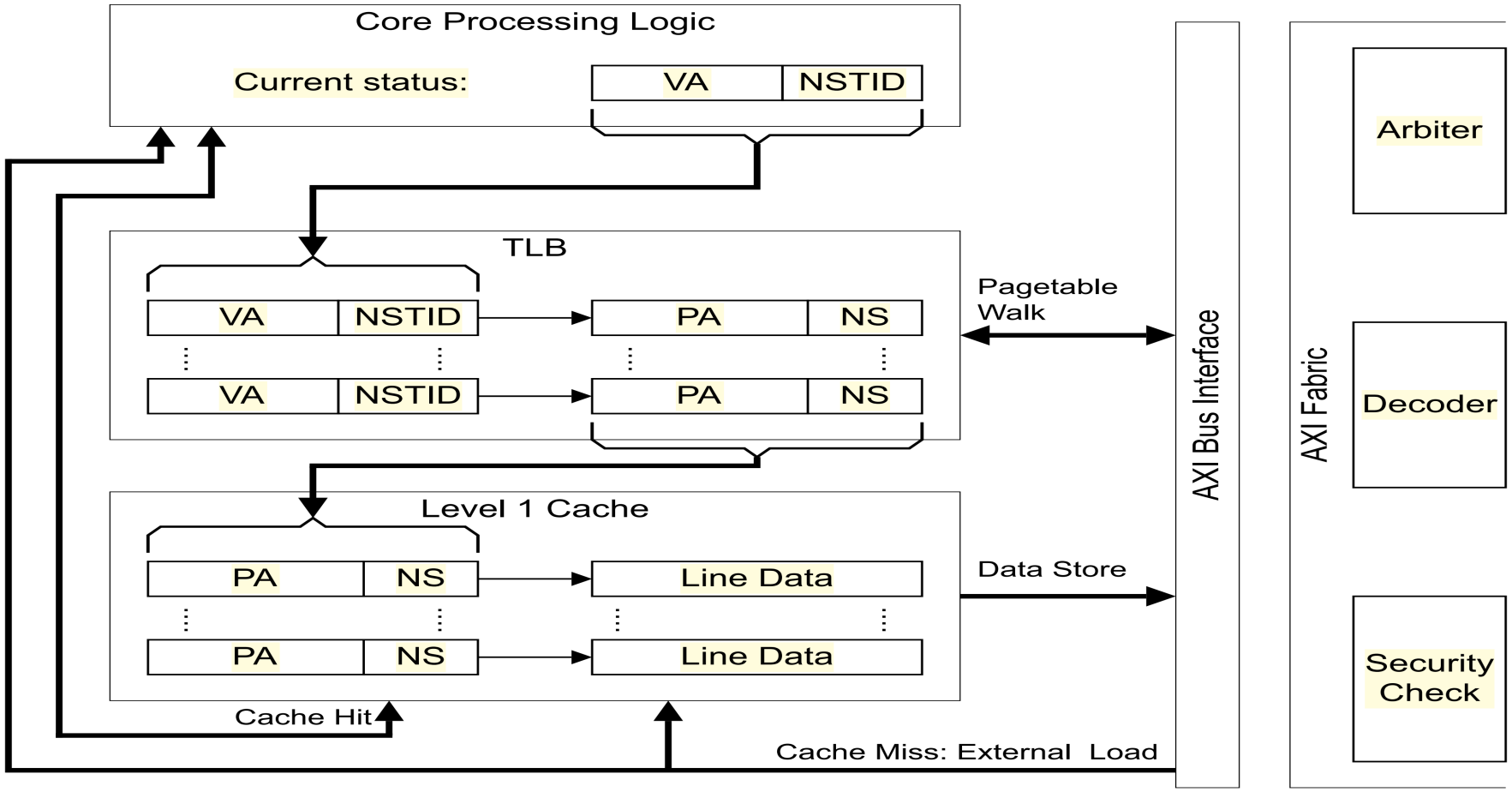
System architecture – memory subsytem

- Two virtual MMUs exist, one for each virtual processor. Each world has local set of translation tables, giving independent control over 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. Non-secure and Secure entries co-exist in the TLBs, enabling faster switching as there is no flush of 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 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 cache content with regard to the security state, is dynamic. Any non-locked down cache line can be evicted 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.

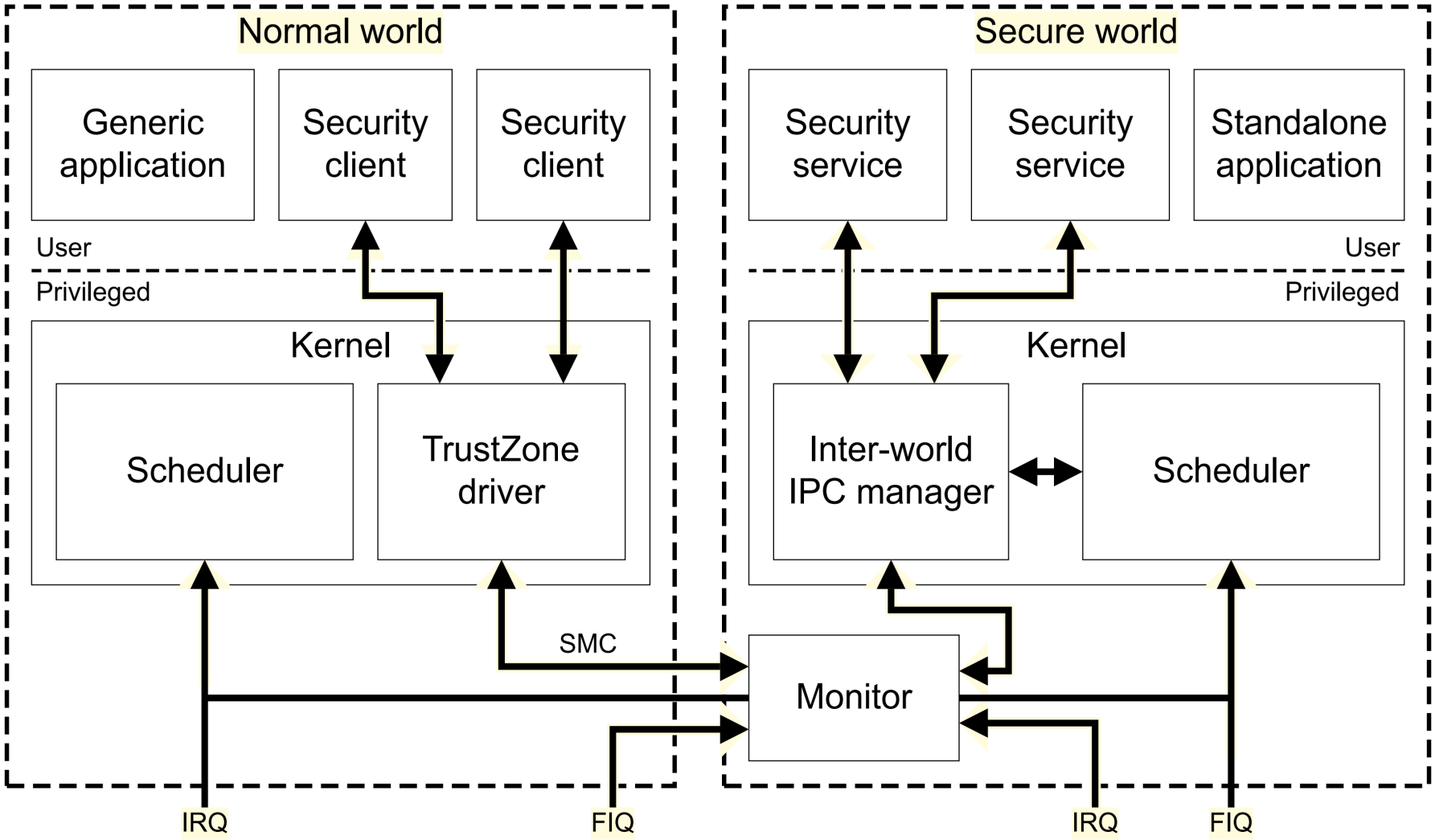
System architecture – debug

- 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 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

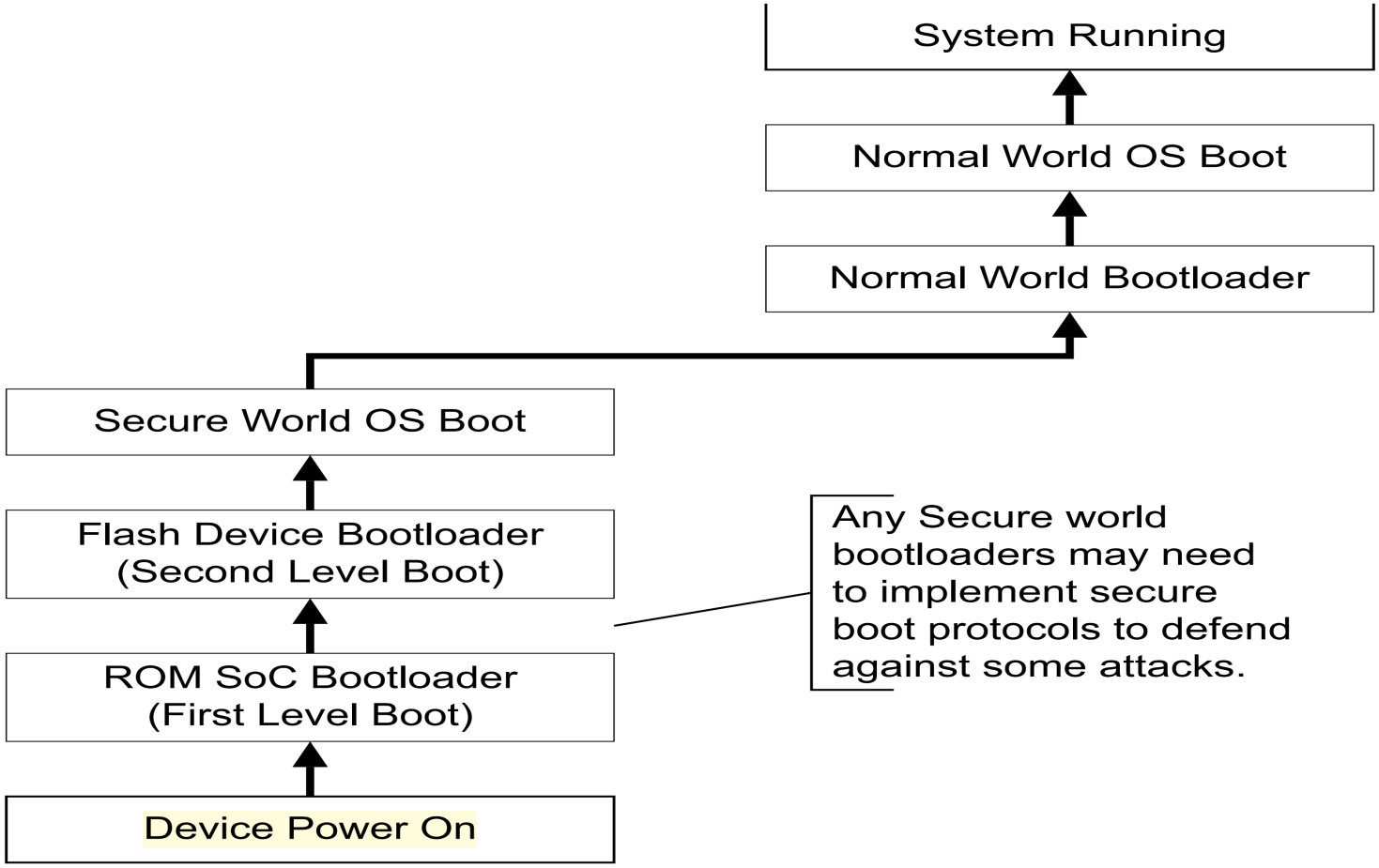
Software Architecture – A secure OS

- 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.
- Each virtual processor runs 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



Rogue Firmware Attack – I

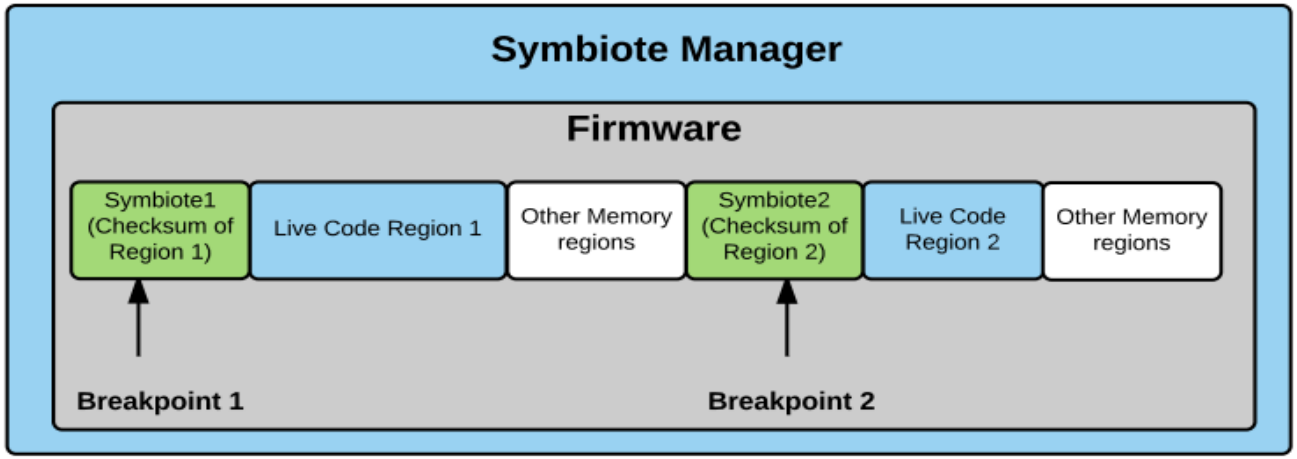
- After a lab attack, an erroneous update or because of a malicious software developer a device can store some malicious software that attacks the system or offers an hidden backdoor
- Some run time attacks can be discovered by memory introspection or attestation, other ones can be discovered by an intrusion detection system

Doppelganger

- 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

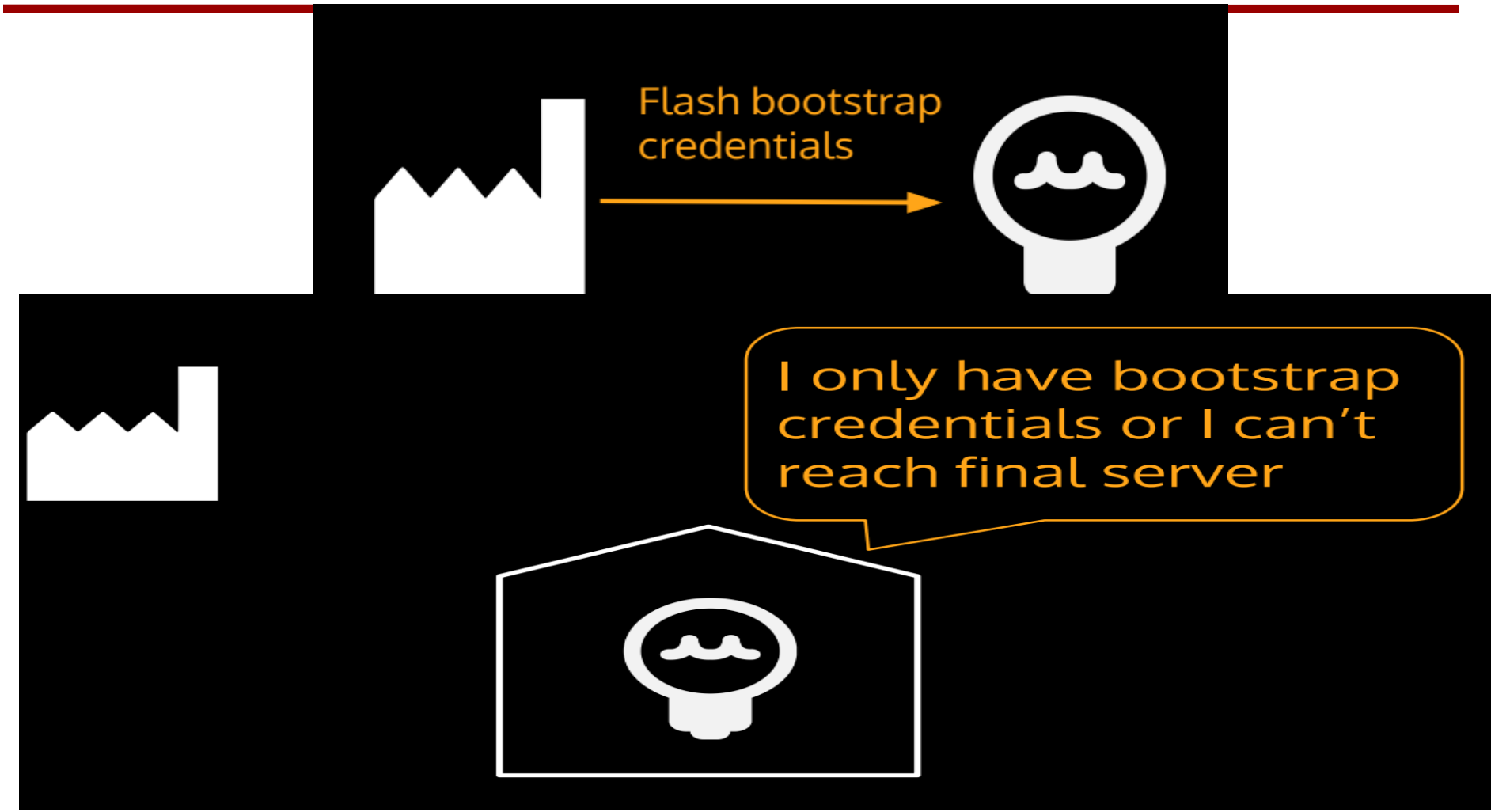
- Doppelganger adds its symbiote manager to the beginning of the firmware.
- The symbiote manager may be seen as a debugger that runs
  - the firmware of the embedded system
  - Doppelganger in a different context of the OS to make it resistant to attacks against its runtime.
- During the firmware execution, every time the symbiote manager detects a symbiote in memory,
  - it stops the execution process (symbiote=breakpoint),
  - compares the current memory area checksum of the symbiote one,
  - Dopplganger considers a mismatch an evidence of a modification attack and it prevents the processor from running the code.
- Doppelganger does not defend against attacks that load code in the dynamic memory

Managing an IOT device

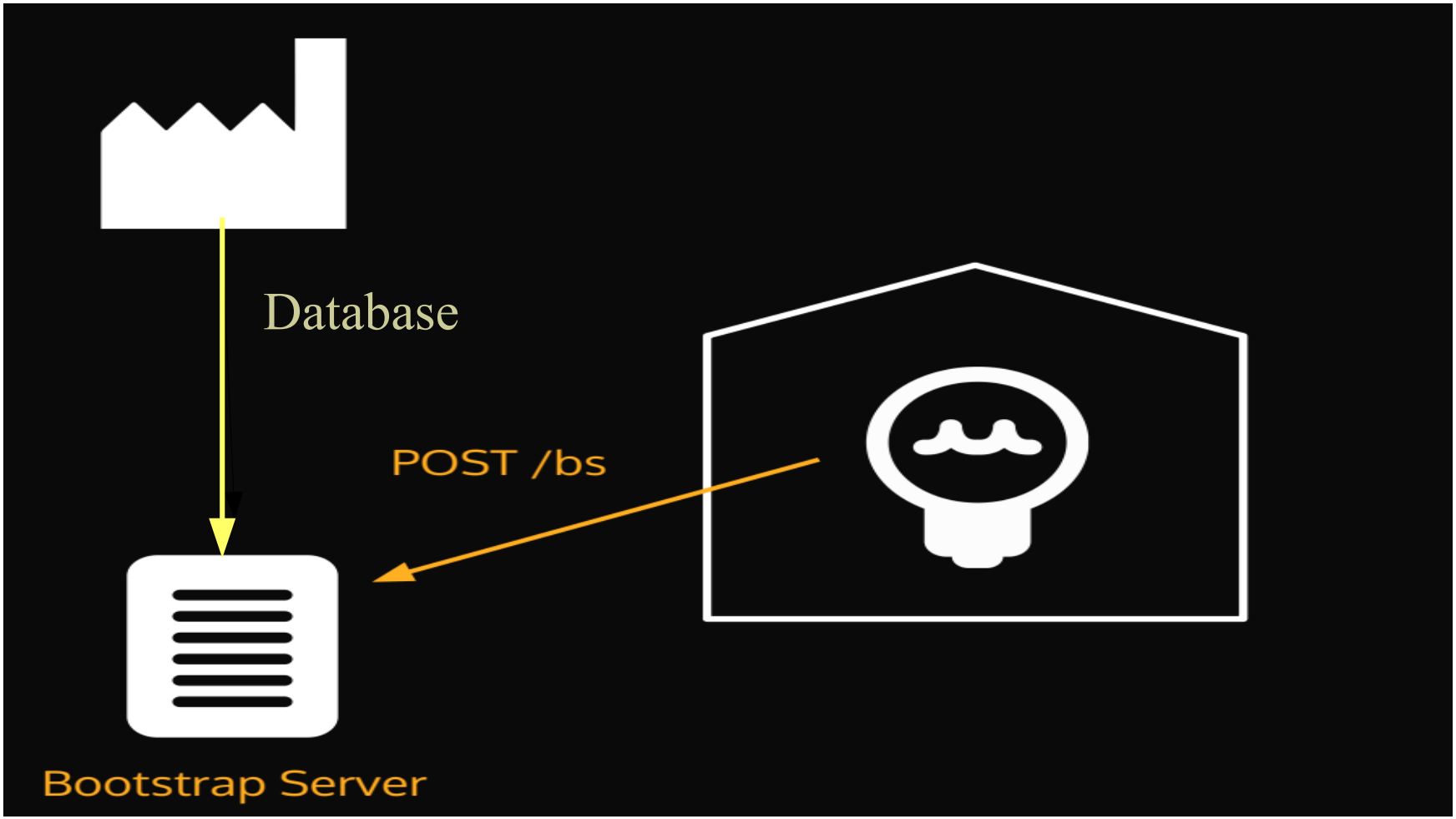
- Several IOT devices are managed by a remote server that can
  - Update
  - Patch

  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 arises the problem of the trust on the manufacturer of the device

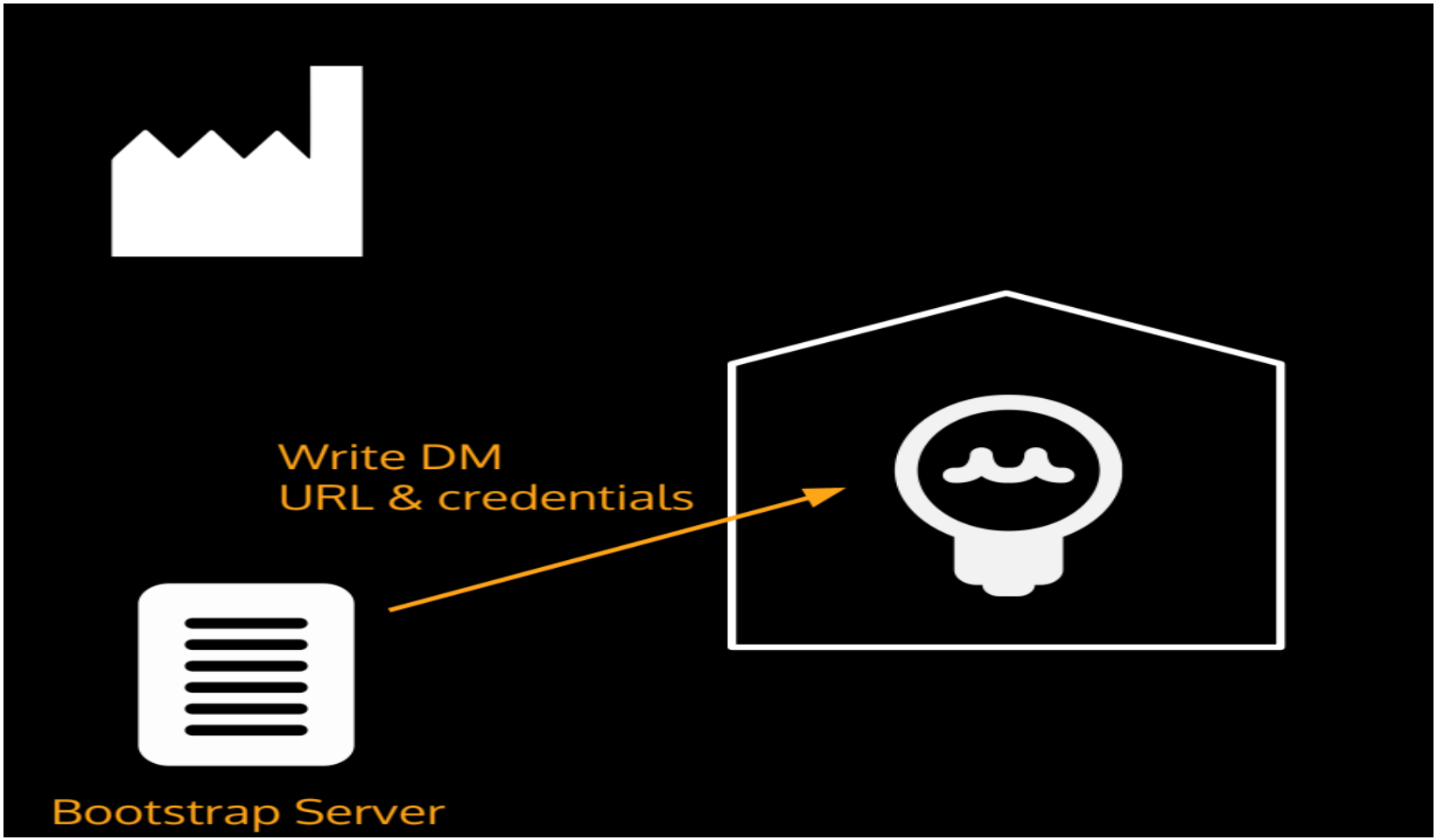
Secure Management of a Device - I



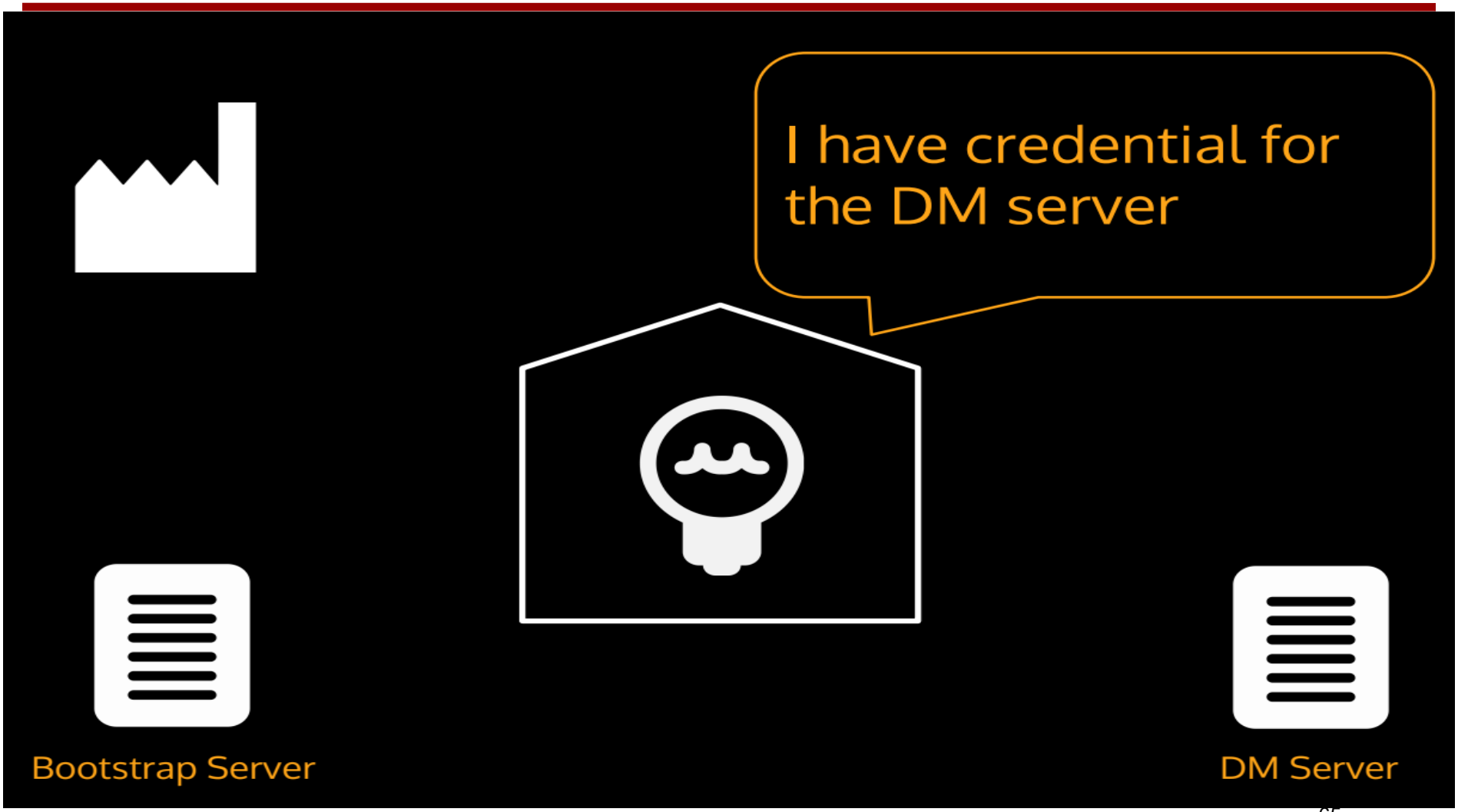
Secure Management of a Device - II



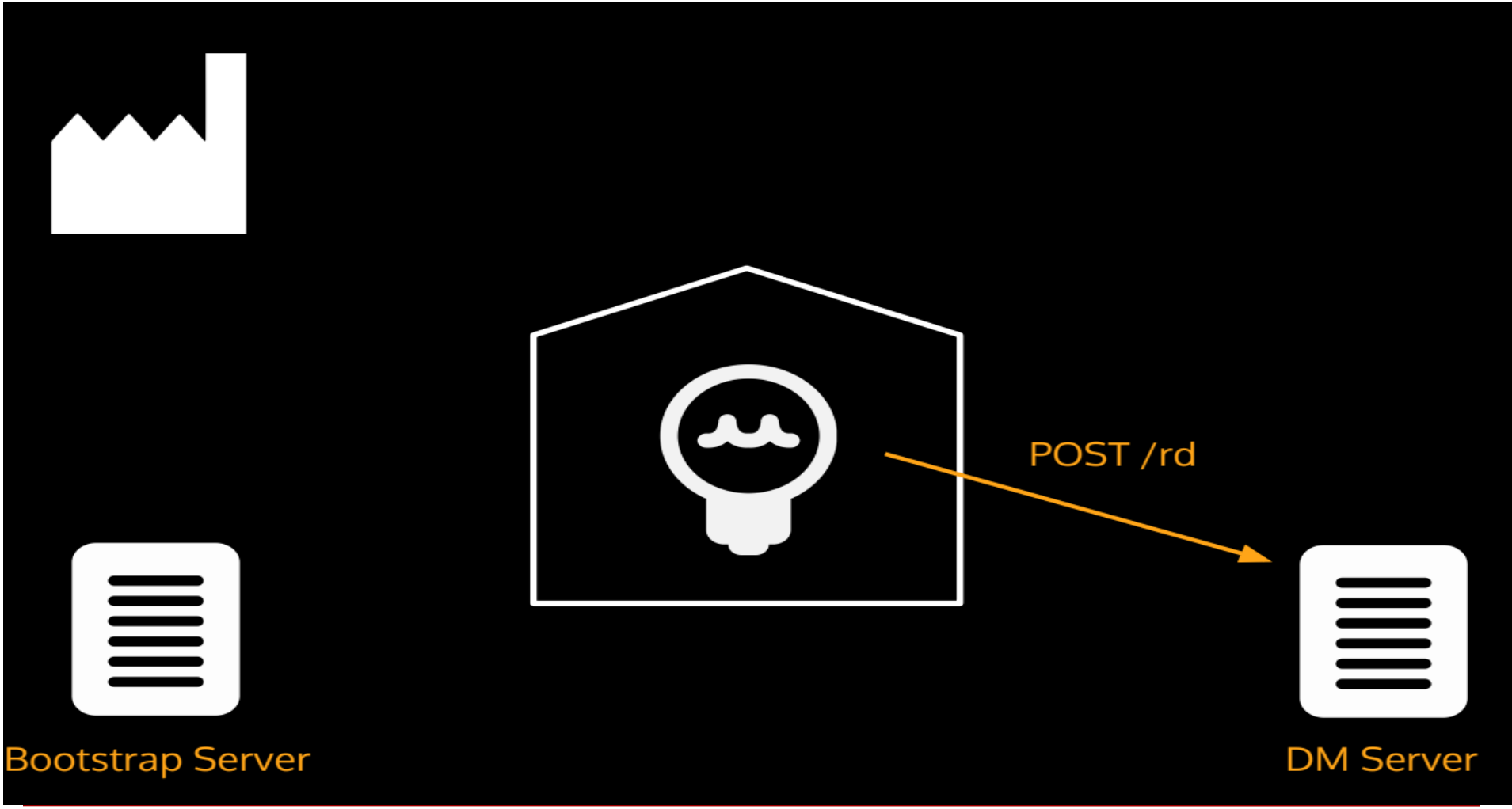
Secure Management of a Device - III



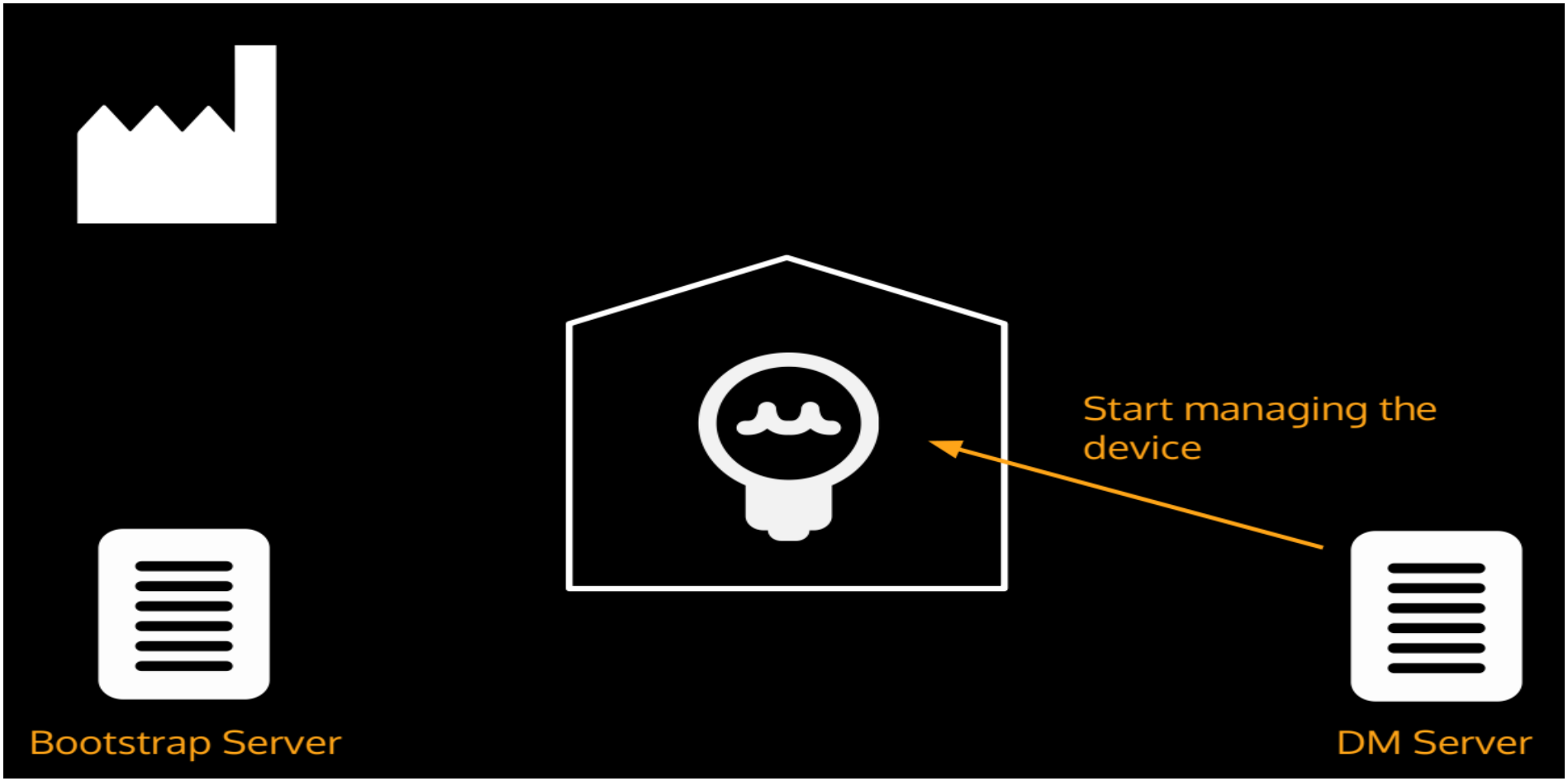
Secure Management of a Device - IV



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 and they fill the need for secure, flexible, and analyzable lightweight block ciphers.
- Most block ciphers were designed to perform well on a single platform and not to offer 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 platform, and is amenable to analysis using existing techniques.
- Both perform well across the 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, microcontrollers, 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 a minimal throughput requirement is 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.
- Sizes of 64 and 128 bits are prevalent in 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

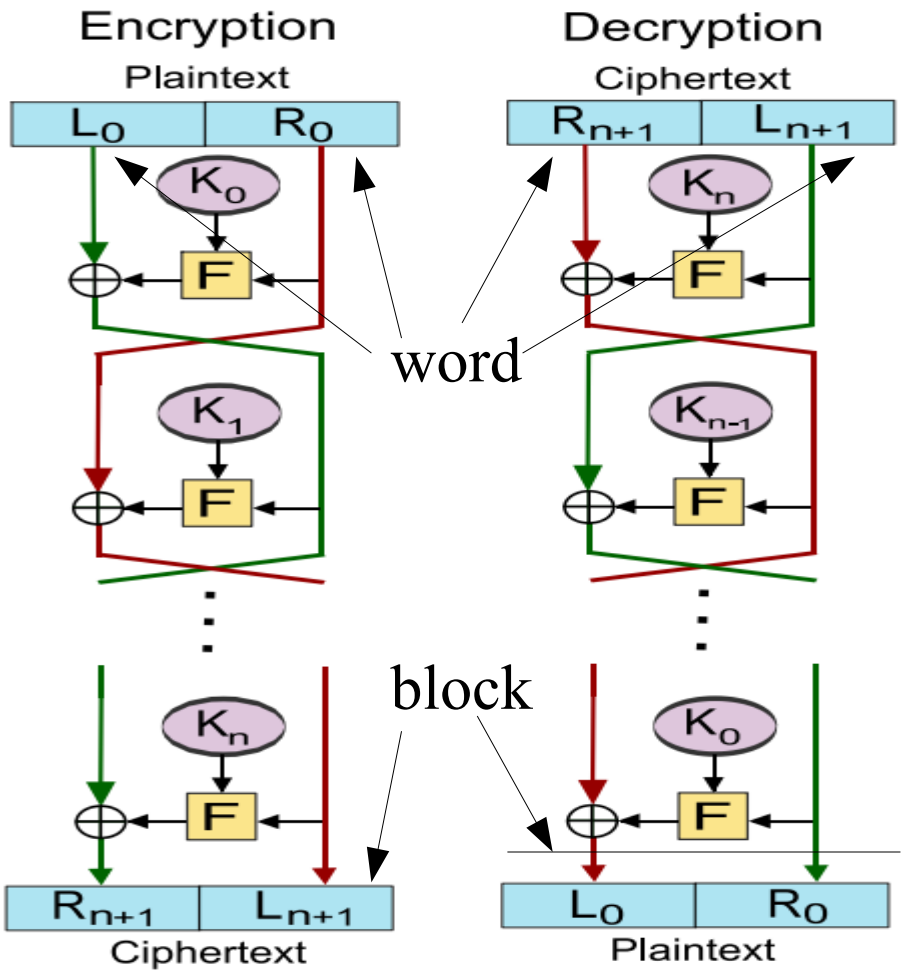
block size	key sizes
32	64
48	72, 96
64	96, 128
96	96, 144
128	128, 192, 256

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

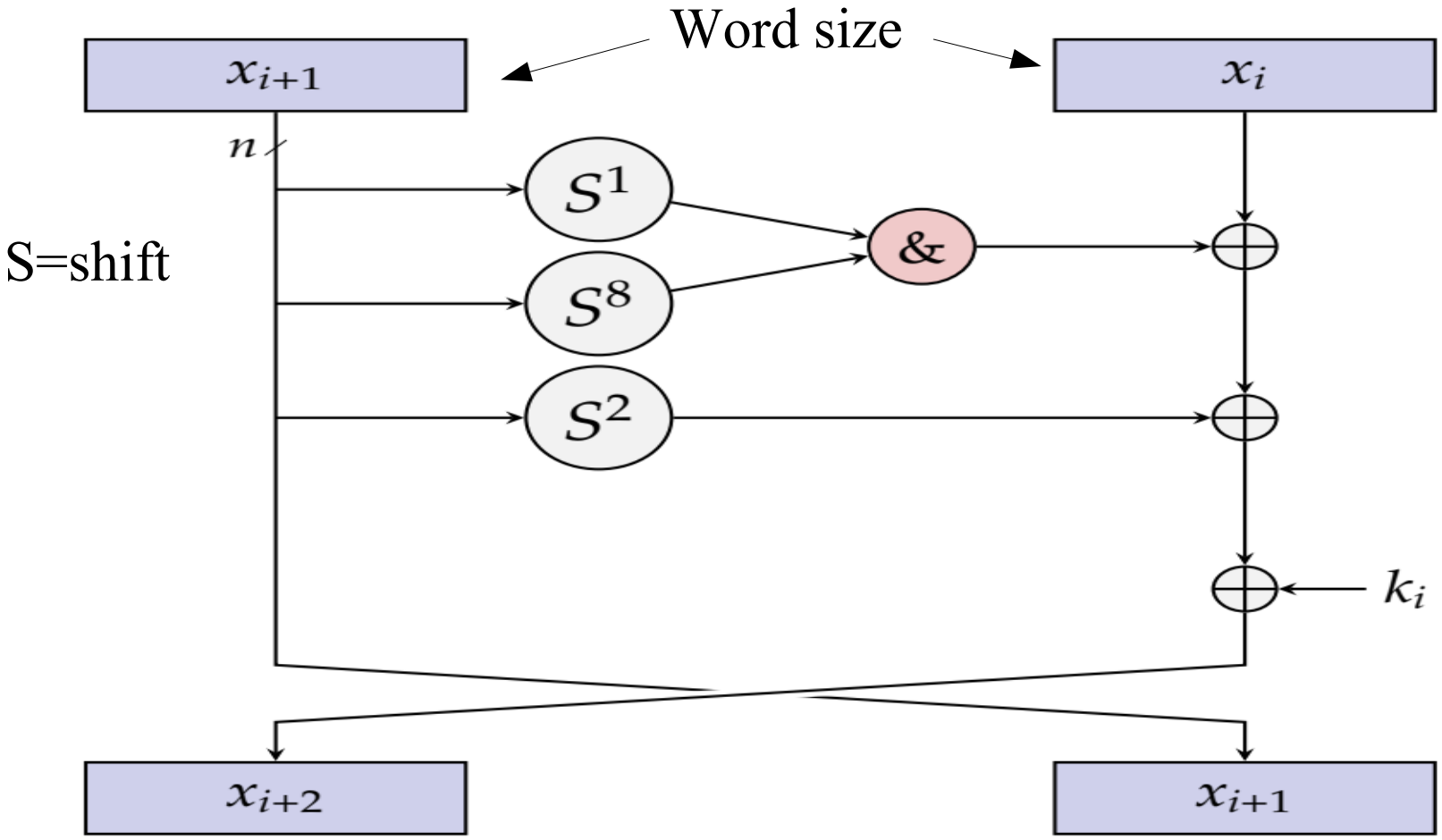
This implies that n $\in$[16-64] because the block size $\in$[32-128]

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 $u = u_0 u_1 u_2 \ldots = 11111010001001010110000111001 10\ldots,$

z_1 $v = v_0 v_1 v_2 \ldots = 10001110111110010011000010110 10\ldots,$

$z_2 = (z_2)_0 (z_2)_1 (z_2)_2 \ldots = 10101111011100000011010010011 00$

$01010000100011111100101101100 11\ldots,$

$z_3 = (z_3)_0 (z_3)_1 (z_3)_2 \ldots = 11011011101011000110010111100 00$

$00100100010100111001101000011 11\ldots,$

$z_4 = (z_4)_0 (z_4)_1 (z_4)_2 \ldots = 11010001111001101011011000100 00$

$00101110000110010100100111011 11\ldots,$

SIMON Algorithm – The Key Sched - 1

z_0 $u = u_0 u_1 u_2 \ldots = 11111010001001010110000111001 10\ldots,$

z_1 $v = v_0 v_1 v_2 \ldots = 10001110111110010011000010110 10\ldots,$

$z_2 = (z_2)_0 (z_2)_1 (z_2)_2 \ldots = 10101111011100000011010010011 00$

$01010000100011111100101101100 11\ldots,$

$z_3 = (z_3)_0 (z_3)_1 (z_3)_2 \ldots = 11011011101011000110010111100 00$

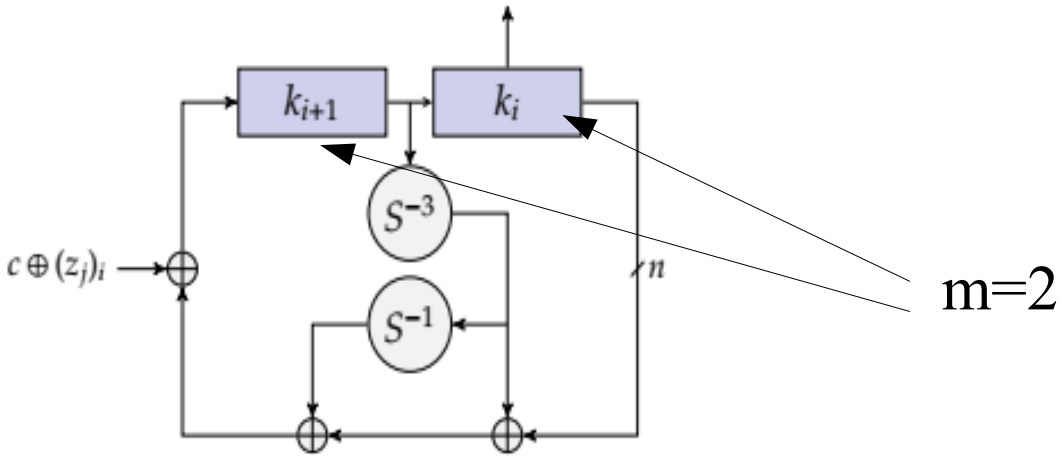
$00100100010100111001101000011 11\ldots,$

$z_4 = (z_4)_0 (z_4)_1 (z_4)_2 \ldots = 11010001111001101011011000100 00$

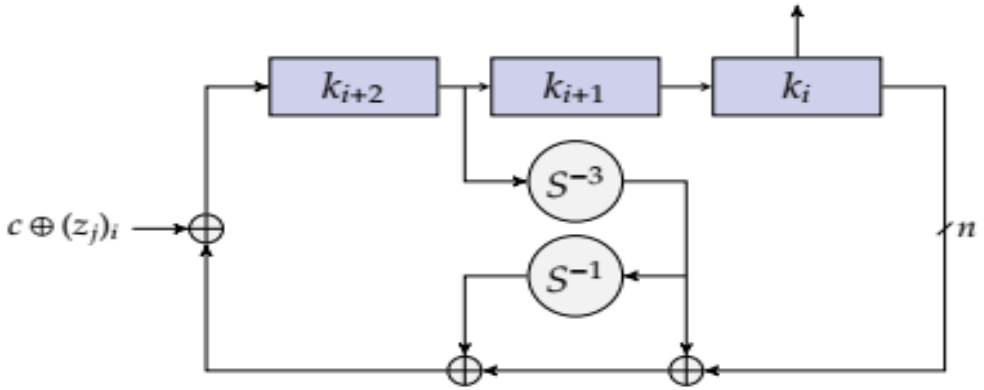
$00101110000110010100100111011 11\ldots,$

SIMON Algorithm – The Key Sched - 2

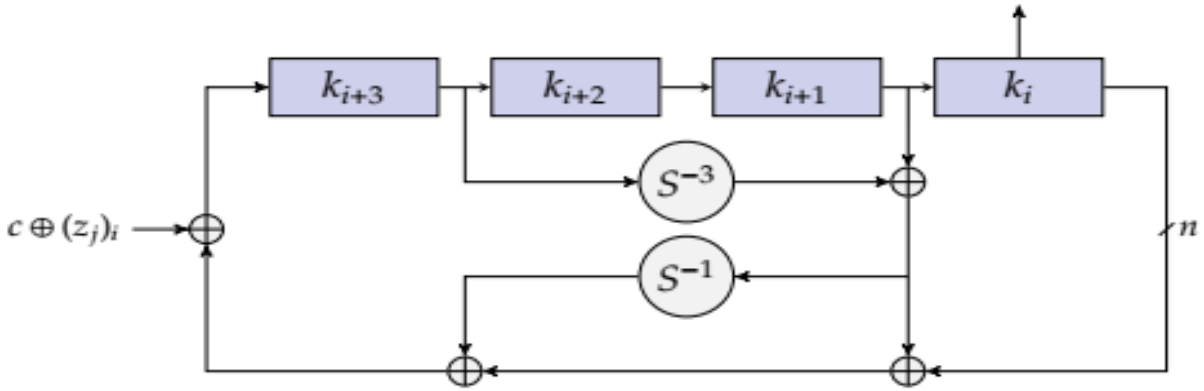
$$k_{i+m} = \begin{cases} c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+1}, & \text{if } m = 2, \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+2}, & \text{if } m = 3, \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})(S^{-3}k_{i+3} \oplus k_{i+1}), & \text{if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



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:

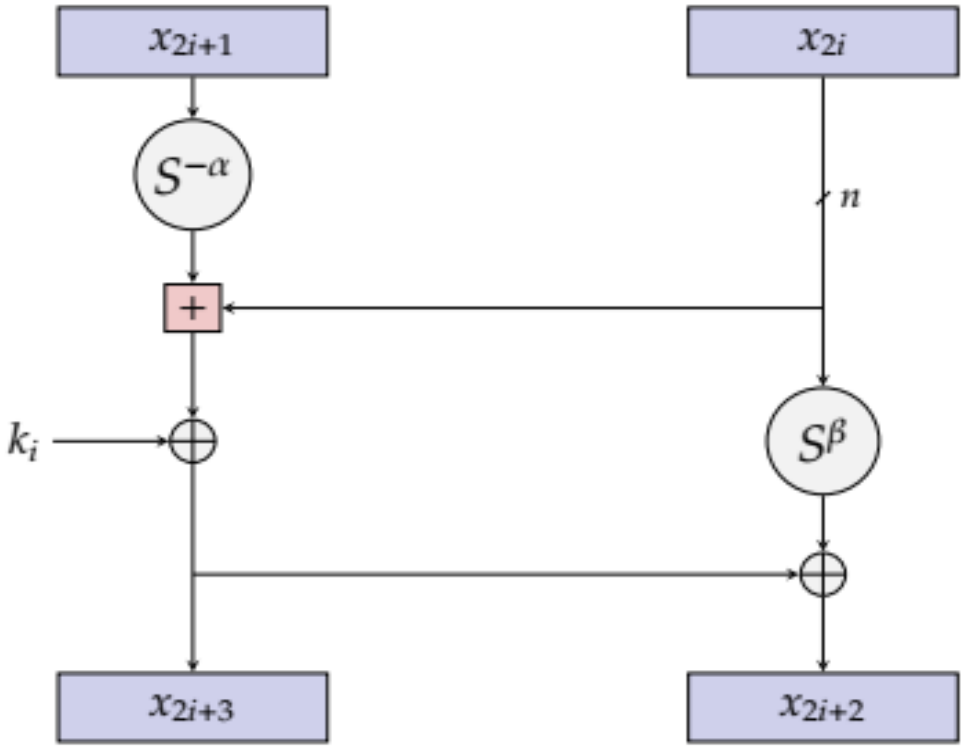
- bitwise XOR, $\oplus$,
- addition modulo 2^n, $+$, and
- left and right circular shifts, S^j and S^{-j}, respectively, by j bits.

For $k \in GF(2)^n$, the key-dependent SPECK2n round function is the map $R_k : GF(2)^n \times GF(2)^n \to GF(2)^n \times GF(2)^n$ 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



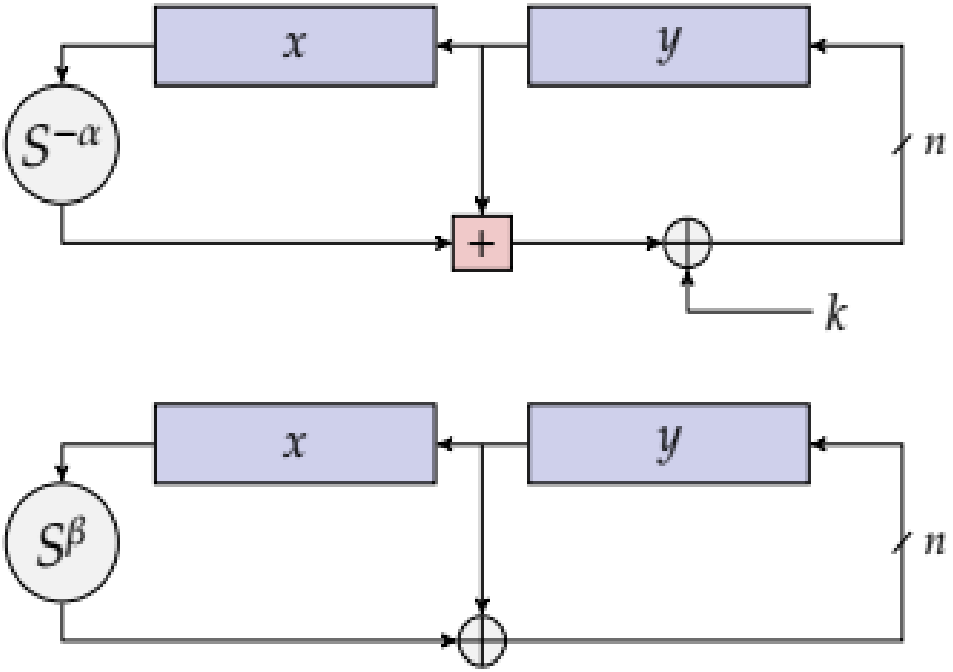
SPECK Family – Decryption - Parameters

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

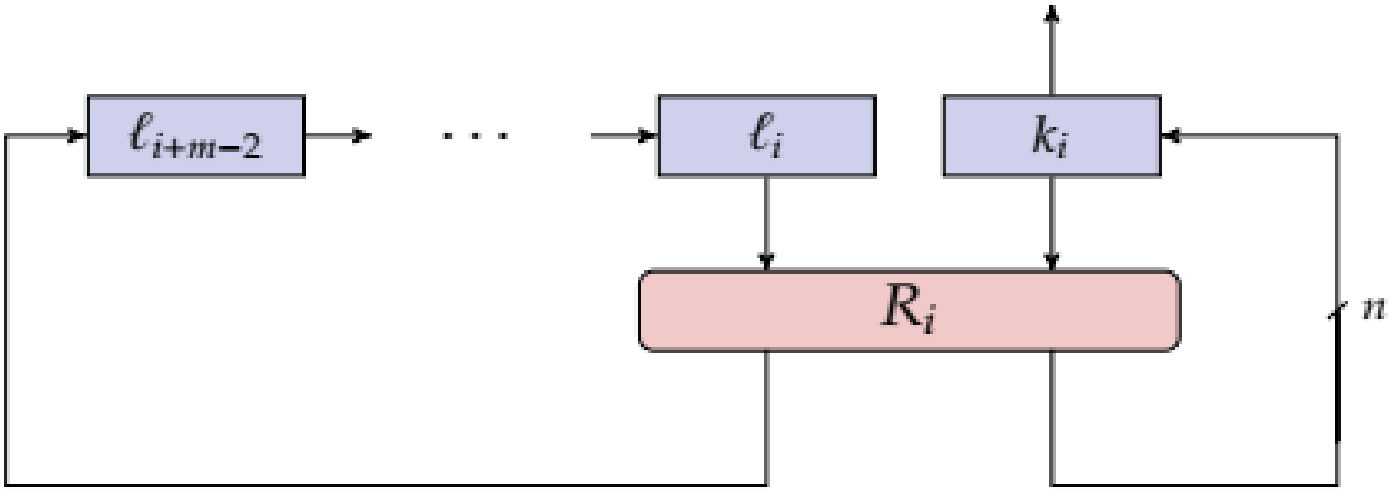
SPECK Round – Composition of two Feistel



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 $\ell_i, k_0 \in \mathrm{GF}(2)^n$, for a value of m in $\{2, 3, 4\}$. Sequences k_i and ℓ_i are defined by

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



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

  (α,β) = (7,2)  if n = 16
          (8,3)  otherwise

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

```
---------------------------- key expansion ------------
for i = 0..T-2
    ℓ[i+m-1] ← (k[i] + S^-α ℓ[i]) ⊕ i
    k[i+1]   ← S^β k[i] ⊕ ℓ[i+m-1]
end for
---------------------------- encryption --------------
for i = 0..T-1
    x ← (S^-α x + y) ⊕ k[i]
    y ← S^β y ⊕ 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)
SIMON32/64	523	5.6
	535	11.1
	566	22.2
	627	44.4
	722	88.9
SIMON48/72	631	5.1
	639	10.3
	648	15.4
	662	20.5
	683	30.8
	714	41.0
	765	61.5
	918	123.1

algorithm	area (GE)	throughput (kbps)
SPECK32/64	580	4.2
	642	8.3
	708	16.7
	822	33.3
	850	123.1
SPECK48/72	693	4.3
	752	8.5
	777	12.8
	821	17.0
	848	25.5
	963	34.0
	1040	51.1
	1152	192.0

Space/throughput trade off

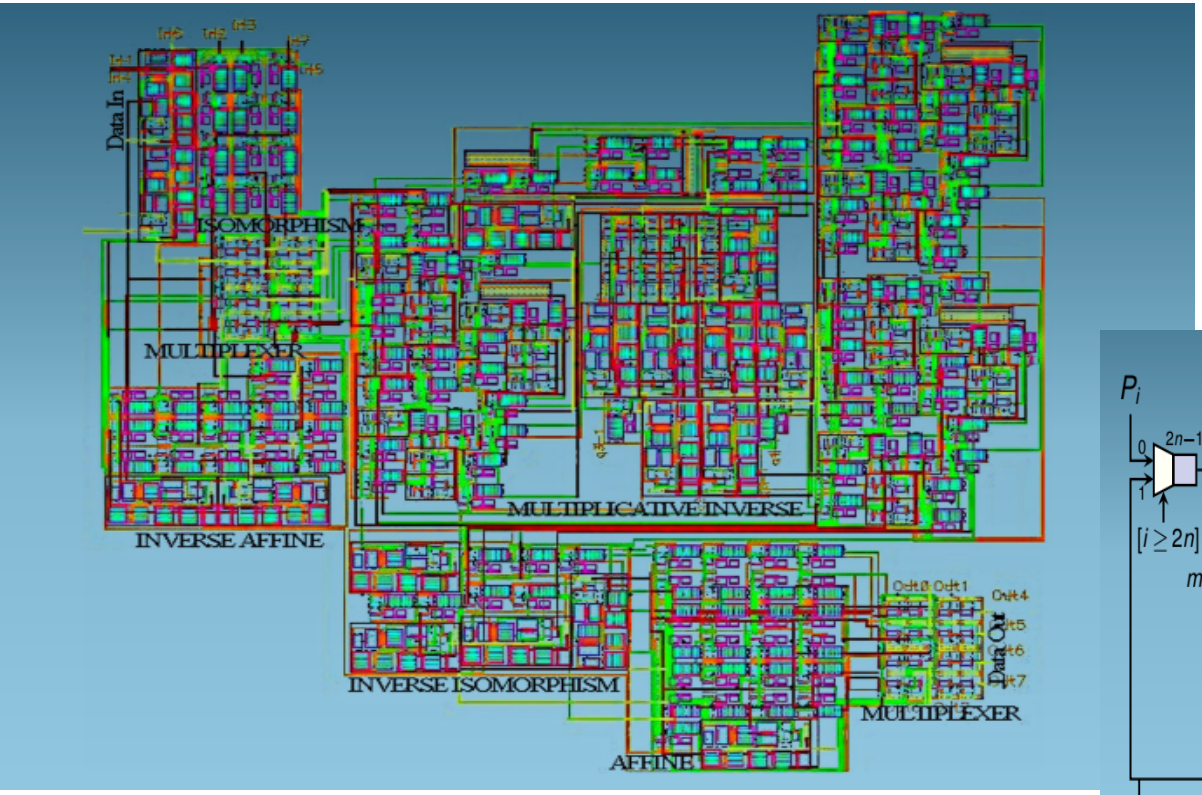
SIMON/SPECK Software (8 bits devices)

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

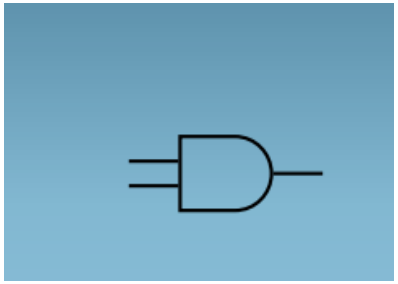
size	name	flash (bytes)	SRAM (bytes)	enc. cost (cycles/byte)
32/64	SIMON	130	0	205
	SPECK	92	0	140
48/72	SIMON	196	0	220
	SPECK	130	0	130
48/96	SIMON	196	0	220
	SPECK	134	0	136
64/96	SIMON	274	0	239
	SPECK	182	0	144
64/128	SIMON	282	0	250
	SPECK	186	0	150
96/96	SIMON	454	0	284
	SPECK	276	0	148
96/144	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

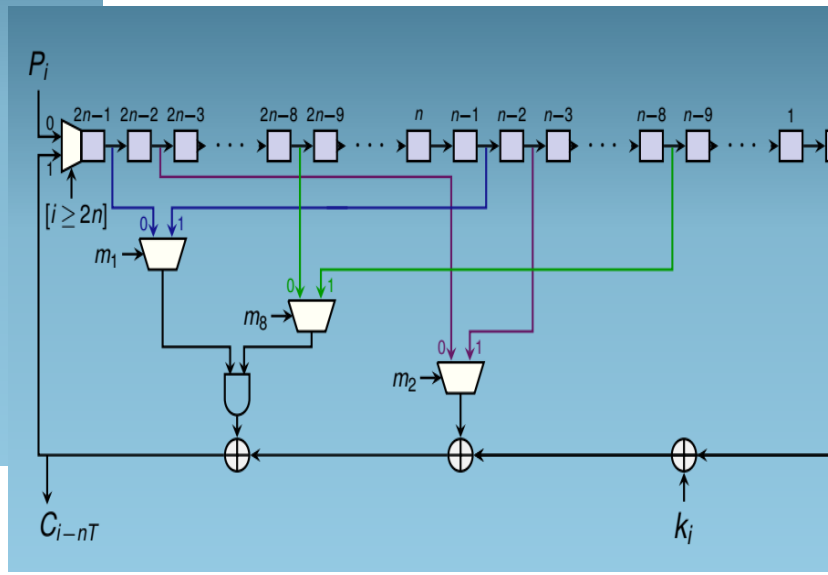
SIMON/SPECK Comparison

AES



Simon





SIMON/SPECK Comparison

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."