NOKIA

Securing the Smart Cities Edge

Tapio Tallgren & Tina Tsou

April 5, 2019

Abstract, will be deleted!

Multi-Access Edge Computing (MEC) holds the promise for significant innovation and enriching our lives. But along with this innovation comes the threat of hackers- open hardware and software, third party software open vulnerabilities that can be exploited.

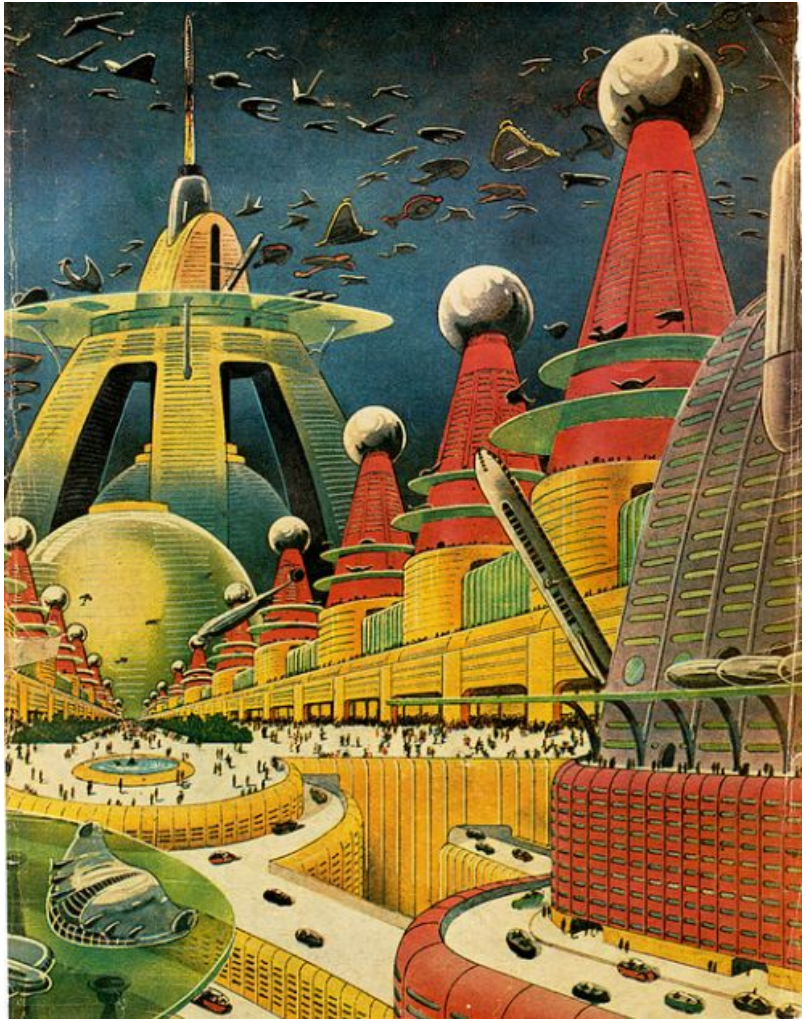
Within the Akraino Edge Stack Project, Nokia, Arm, & ecosystem partners have formed an edge blueprint for a Smart Cities platform called µMEC, targeted for a range of use cases. Devices based on the µMEC architecture can support sensors, connectivity protocols, etc. in different locations. As such, operating system level security is not enough. We need security for onboarding, securing data, and verifying the integrity of firmware, operating system and system software, as well as 3rd party applications.

In this session, we examine how technology such as Arm TrustZone and the OP-TEE software can be leveraged to achieve these goals.

Smart Cities

Smart cities? We all know what they look like.

Smart cities thrive on data...



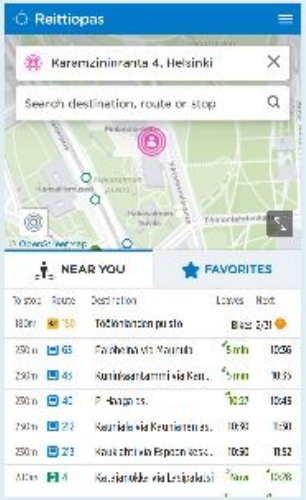
Open data

Open data is publicly available data that can be universally and readily accessed, used, and redistributed free of charge. It is structured for usability and computability. (Source: GovLab)

Explore open data resources

HSL Developer Community

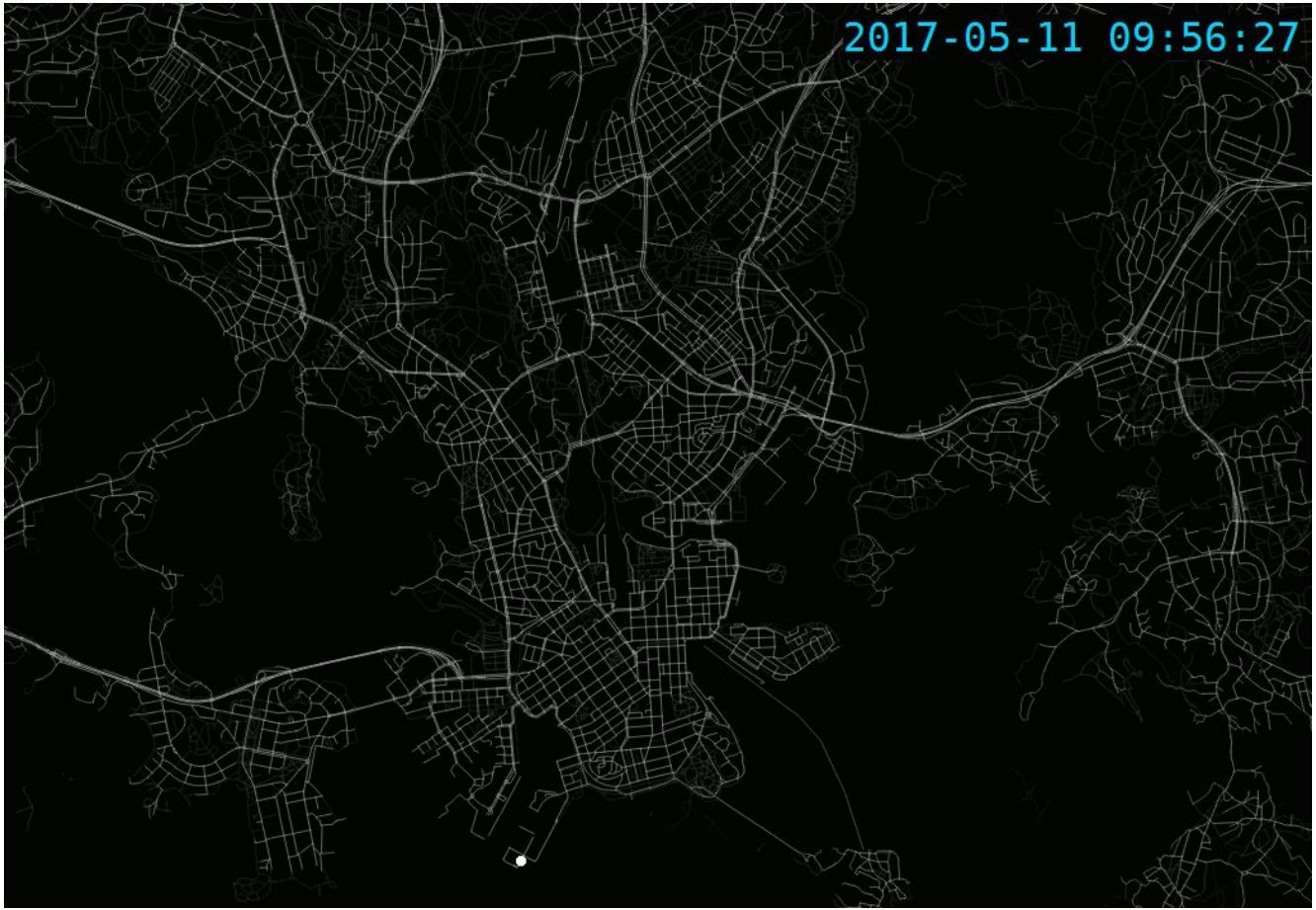
Contact Applications Journey planning Maps Ticket sales Travel card Other resources Related sites Archive



```
import paho.mqtt.client as mqtt
import json
import csv


# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    if rc==0:
        print("connected OK Returned code=",rc)
    else:
        print("Bad connection Returned code=",rc)

    # Subscribing in on_connect() means that if we lose the connection and
    # reconnect then subscriptions will be renewed.
    # This particular topic collects ALL traffic
    client.subscribe("/hfp/v1/journey/#")

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    j = msg.payload.decode('utf8').replace("'", '"')
    d = json.loads(j)
    v = d["VP"]

    # I don't want all of the data, so I define here the values I want
    values=v['long'],v['lat'],v['desi'],v['oper'],v['jrn'],v['line'],v['spd'], v['dl'], v['hdg'],v['drst'],v['veh'],v['tsi']

    # There are some nulls in there, so this is just a stupid way to exclude those
    if v['lat'] > 0:
        with open(r'hsl_mqtt.csv', 'a') as f:
            writer = csv.writer(f, quoting=csv.QUOTE_NONE, lineterminator='\n')
            writer.writerow(values)

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect("mqtt.hsl.fi", 1883, 60)

# Blocking call that processes network traffic, dispatches callbacks and
# handles reconnecting.
# Other loop*() functions are available that give a threaded interface and a
# manual interface.
client.loop_forever()
```

24.882184,60.196806,58,18,296,75,8.31,0,108,0,258,1553257031
24.87259,60.19451,4,40,1250,32,0.0,180,113,1,425,1553257031
24.706416,60.225422,533,22,197,259,0.78,-60,192,0,1017,1553257031
24.662606,60.219641,565,20,82,817,0.09,-106,31,0,27,1553257031
24.918395,60.183417,14,12,80,41,0.0,-268,333,0,1020,1553257013
24.918395,60.183417,14,12,80,41,0.0,-268,333,0,1020,1553257008
24.918395,60.183417,14,12,80,41,0.0,-268,333,0,1020,1553257014
24.918395,60.183417,14,12,80,41,0.0,-268,333,0,1020,1553257009
24.75139,60.199804,549,22,437,875,9.47,-60,276,0,1028,1553257031
24.843166,60.258938,560,12,347,743,0.09,240,21,1,1520,1553257031
24.92998,60.169302,39,12,81,60,0.0,0,323,0,1304,1553257031
24.922533,60.184398,10,40,503,40,0.0,0,0,0,97,1553257031
24.920059,60.186691,4,40,1248,32,6.75,-64,144,0,433,1553257031
24.668287,60.201845,213,22,10,245,8.96,-7,223,0,922,1553257031
24.804902,60.270914,436K,22,108,812,2.79,-120,84,0,847,1553257031
25.070464,60.262974,75,12,57,102,5.59,-120,279,0,920,1553257031
24.99237,60.258984,79,22,511,110,11.5,-9,299,0,603,1553257031
24.998722,60.240917,71,55,324,94,0.39,-8,68,1,1219,1553257031
24.89843,60.203181,10,40,849,40,5.47,49,104,0,429,1553257031
24.801877,60.174496,548,22,340,873,0.27,120,80,0,1036,1553257031

NOKIA

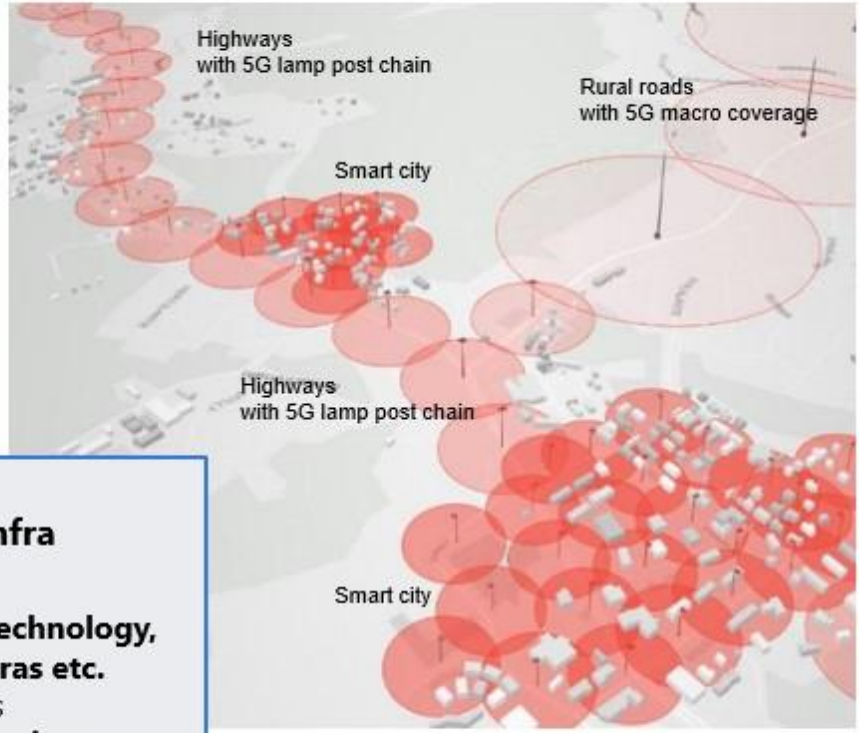


NOKIA

LuxTurrim5G

PROBLEM:
Capacity of mobile networks

- **Capacity of mobile networks is far too insufficient due to the increased number of users and new digital services built and planned**
- The problem can only be solved by using **small cell 5G** radio frequency technology **& higher frequencies**
- This requires **dense networks of antennas** setting new requirements for the network infrastructure



SOLUTION:
LuxTurrim5G

LuxTurrim5G project develops & pilots **smart light pole based 5G network infra** for smart cities

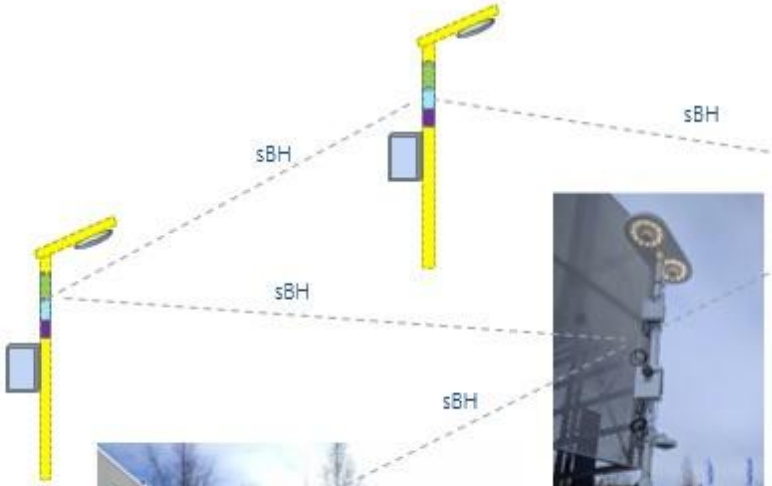
- **Smart light poles with integrated 5G technology, different sensors, active screens, cameras etc.**
- From proof of concepts towards business
- Ecosystem for piloting new **technical solutions, digital services & business models**

NOKIA

Digital Smart City: LuxTurrim5G

LuxTurrim 5G

5G mmW access points in every light pole

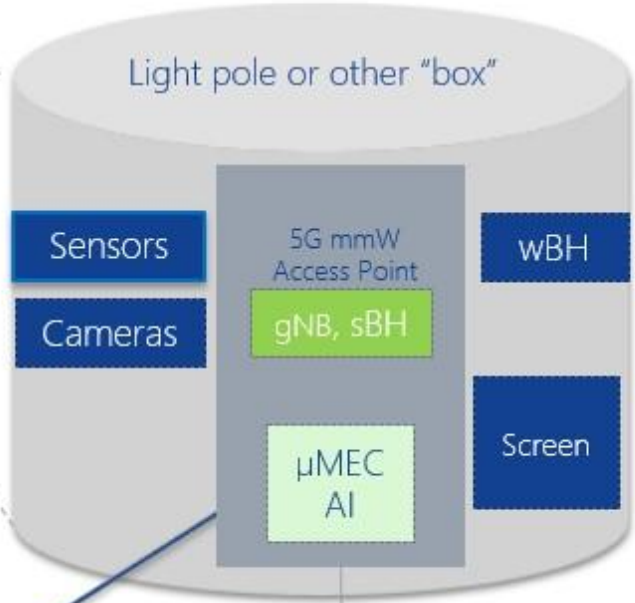


Nokia HQ Espoo campus

Fiber

On 73 GHz frequency, 15 Gbps per sector, four sectors

5G Base station with sensors, cameras, sBH,...



Local AI services:
-sensor fusion
-video analytics
-autonomous traffic
-RN optimization
-...

Fiber(/wireless)

5G Core

NOKIA

Akraino project

Akraino Edge Stack is an open source software stack that improves the state of edge cloud infrastructure for carrier, provider, and IoT networks.

Akraino Edge Stack offers new levels of flexibility to scale edge cloud services quickly, to maximize the applications or subscribers supported on each server, and to help ensure the reliability of systems that must be up at all times.

Akraino Edge Stack also provides processing power closer to endpoint customer devices to meet application latency requirements of less than ~20 milliseconds.

This open source software stack intend to provide critical infrastructure to:

- Enable line speed processing
- Enable high throughput
- Reduce latency
- Improve availability
- Lower operational overhead
- Provide scalability
- Address security needs
- Improve fault management

Akraino Edge stack community is focused on Edge APIs, Middleware, Software Development Kits (SDKs) and will allow for cross-platform interoperability with 3rd party clouds. The Edge stack will also enable the development of Edge applications and create an application w/ Virtual Network Function (VNF) ecosystem.

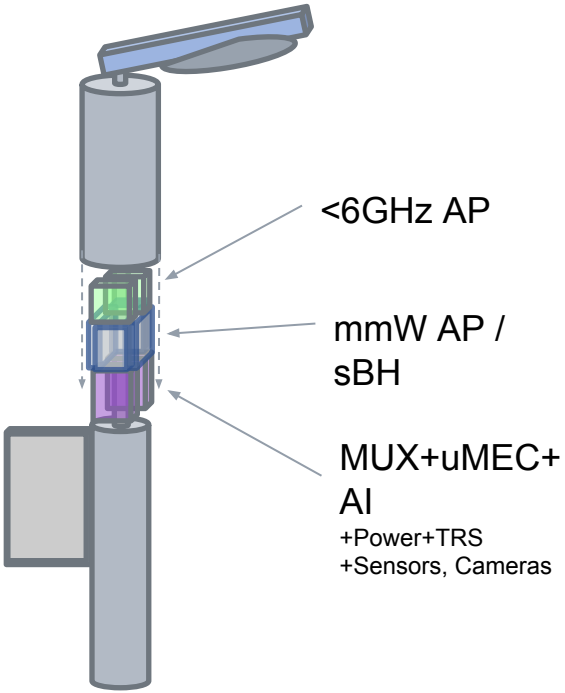
The **Akraino Wiki** is a collaboration tool for the Akraino community to work together and publish documents.

Presentation in ONS NA!

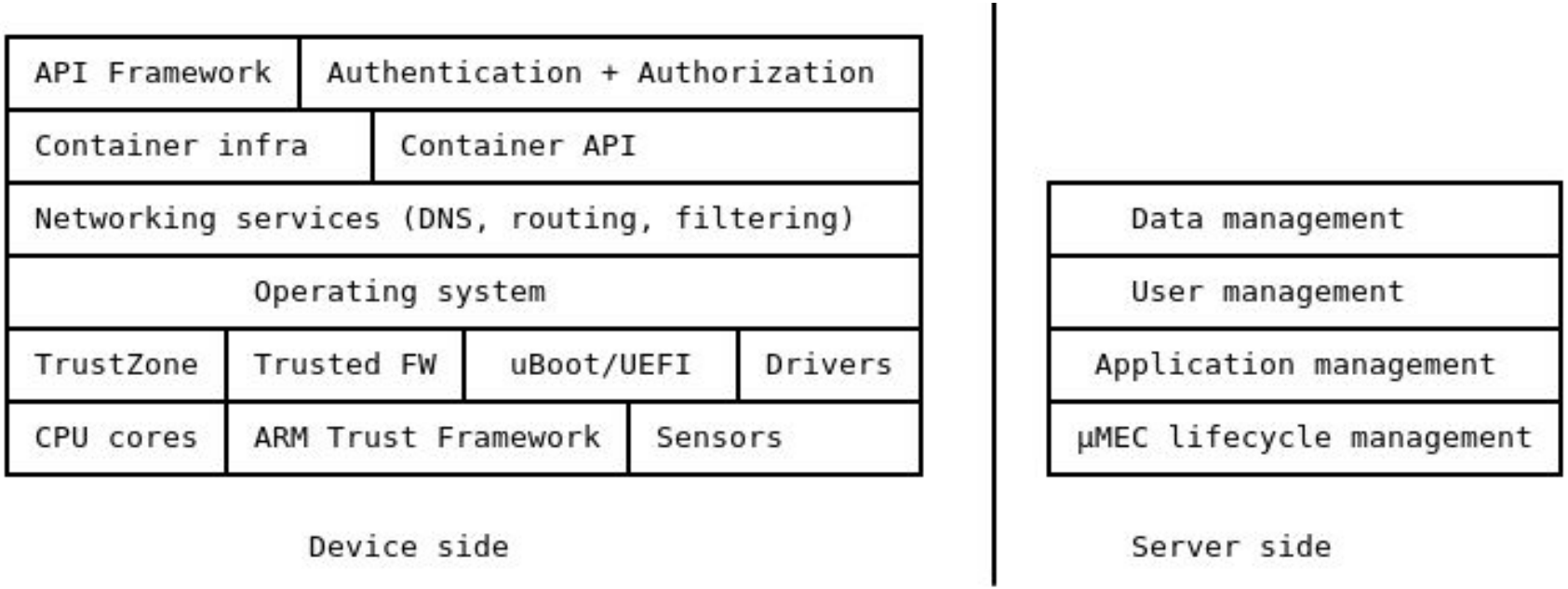
NOKIA

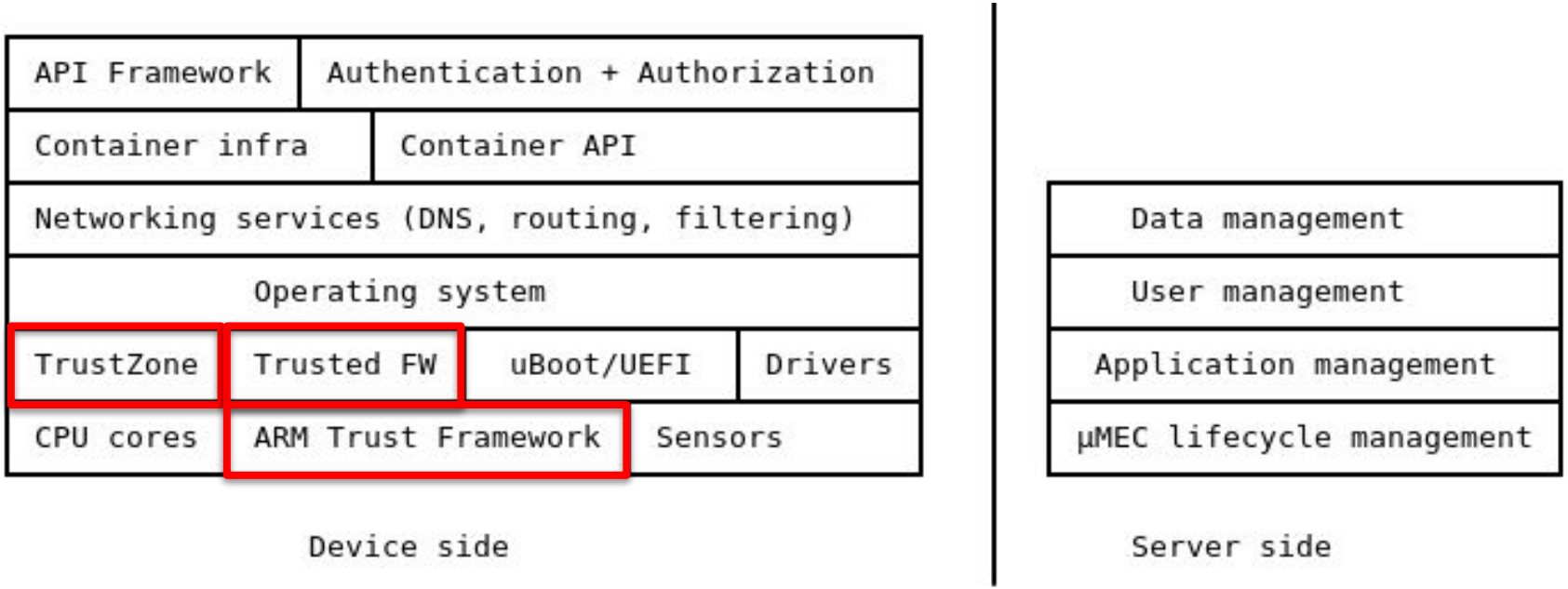
µMEC concept

- µMEC complements the emerging 5G radio networks by enabling new applications
- µMEC is a small form factor HW+SW platform for especially the Smart City services on Ultra Far Edge
- It can use 5G, WLAN or fiber connection
- It can be installed on light poles, vehicles, etc
- The µMEC proof-of-concept is based on LuxTurrim5G and open source components



µMEC deployment example: LuxTurrim5G


API Framework
Authentication + Authorization
Container infra
Container API
Networking services (DNS, routing, filtering)
Operating system
TrustZone
Trusted FW
uBoot/UEFI
Drivers
CPU cores
ARM Trust Framework
Sensors
Device side
Data management
User management
Application management
µMEC lifecycle management
Server side


API Framework
Authentication + Authorization
Container infra
Container API
Networking services (DNS, routing, filtering)
Operating system
TrustZone
Trusted FW
uBoot/UEFI
Drivers
CPU cores
ARM Trust Framework
Sensors
Device side
Data management
User management
Application management
µMEC lifecycle management
Server side

arm

Platform Security Architecture

PSA

What is the Platform Security Architecture (PSA)

- Platform Security Architecture is a combination of hardware and software delivering key platform security functions within a common security model across the various ARM architecture profiles
  - Supports A, R and M profiles

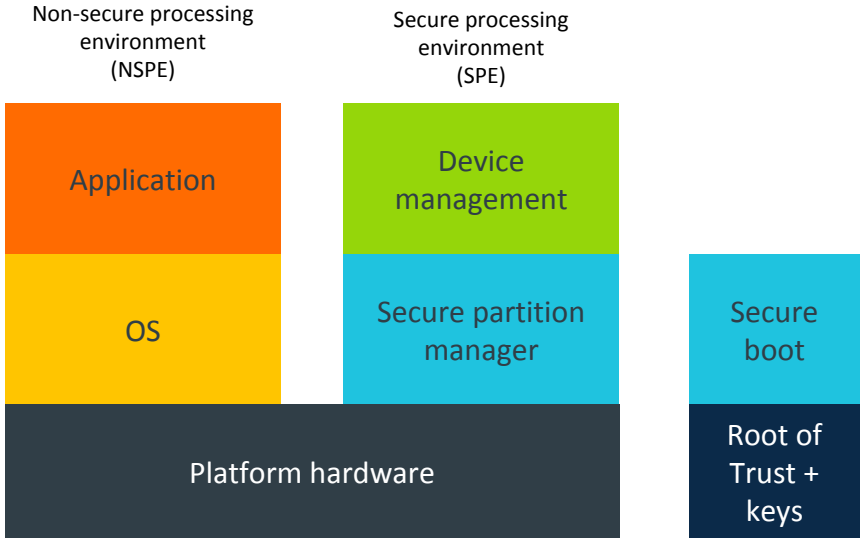
- Delivers a set of guaranteed security related services to any operating system
  - Flexibility of hardware system design, including CPU Core
  - Flexibility of choice of operating system

- Standard ‘trusted application model’
  - Avoid the problems experienced in the A-profile world

PSA defines a common security model

Based on security by separation

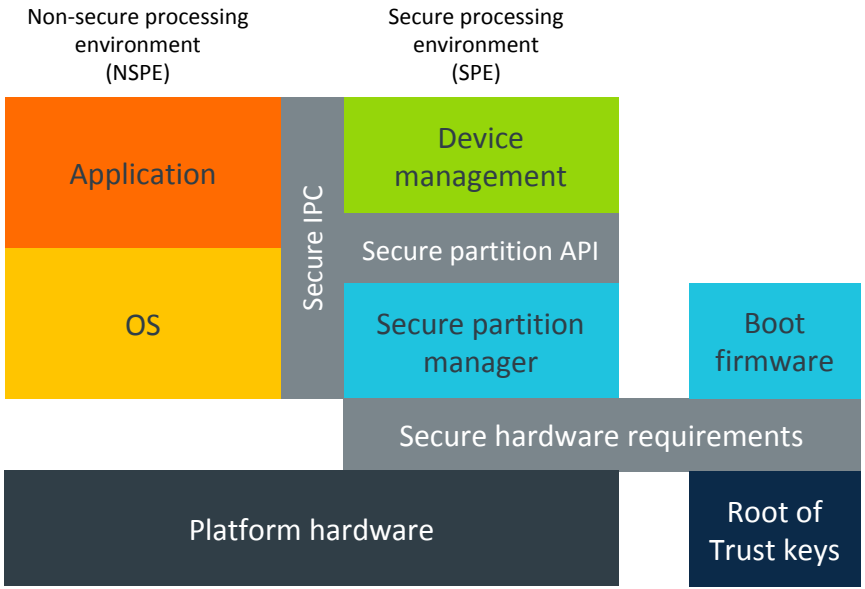
- PSA protects sensitive assets (keys, credentials and firmware) by separating these from the application software and hardware
- PSA defines a Secure Processing Environment (SPE) for this data, the code that manages it and its trusted hardware resources
- The application software runs in the Non-secure Processing Environment (NSPE)
- PSA requires a secure boot process so only authentic, trusted firmware runs in the SPE
- PSA depends on secure installation of the initial keys and firmware during manufacture



arm

Enabling reusable and shareable implementations

- **PSA specifies interfaces to decouple components**
  - Enables reuse of components in other device platforms
  - Reduces integration effort
- **Partners can provide alternative implementations**
  - Necessary to address different cost, footprint, regulatory or security needs
- **PSA provides an architectural specification**
  - Hardware, firmware and process requirements and interfaces



arm

arm

Trusted Boot

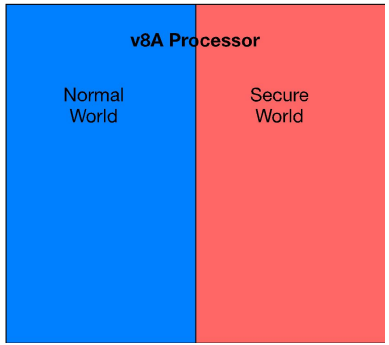
h/w Root-of-Trust
(hRoT)

- Initial Root-of-Trust (iRoT) / hardware Root-of-Trust (hRoT)
  - Immutable 1st load of boot in ROM gets control in EL3 and establishes the iRoT
  - Chain-of-Trust established from 1st load of boot through loader/OS boot
    - Other Roots of Trust: e.g., RoT of measurement, RoT of reporting, RoT of authentication, RoT of confidentiality, RoT of authorization, …
- Verified Boot
  - Cryptographically authenticate each signed load of boot including boot loader
    - boot loader can load and verify OS/hypervisor
- Trusted Boot
  - Immutable 1st load of boot in ROM loads and verifies the next load of boot
    - 1st load trusted because it's immutable – provisioned in ROM at manufacture
  - Each load continues to initialize the TrustZone environment
    - Including SEL2 and the SEL1/SEL0 Secure Partitions
  - Final load of firmware boot loads & instantiates UEFI secure boot/uboot/grub/etc.
    - Instantiates and transfers control to the loader then hypervisor (or bare metal OS) in the Normal World
- Secure Firmware Update
  - UEFI/EDK2 Secure Boot (+ ACPI) MM mode, Capsule Update

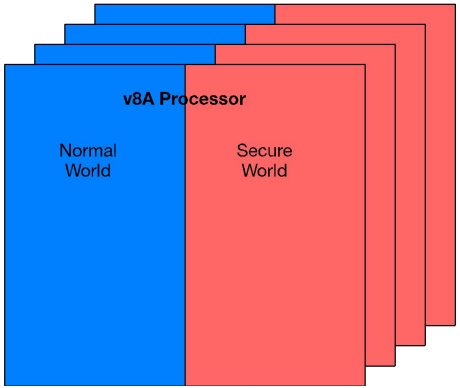
arm

TrustZone

- Hardware isolated Secure (trusted) and Normal (non-trusted) worlds
  - Secure hardware resources are only accessible to software running in the Secure World (TEE)
    - Encompasses memory, software, bus transactions, interrupts, and peripherals
    - Provides access to secure services running in TZ
    - Provides confidentiality and integrity for secure assets
    - Statically allocated resources at boot time
- Runs on each Application Processor (AP)
  - Transitions to Secure World via an architected instruction: SMC
    - Think of it like an SVC or HVC
  - NOT like intel SMM



V8A AP Normal and Secure World, single processor



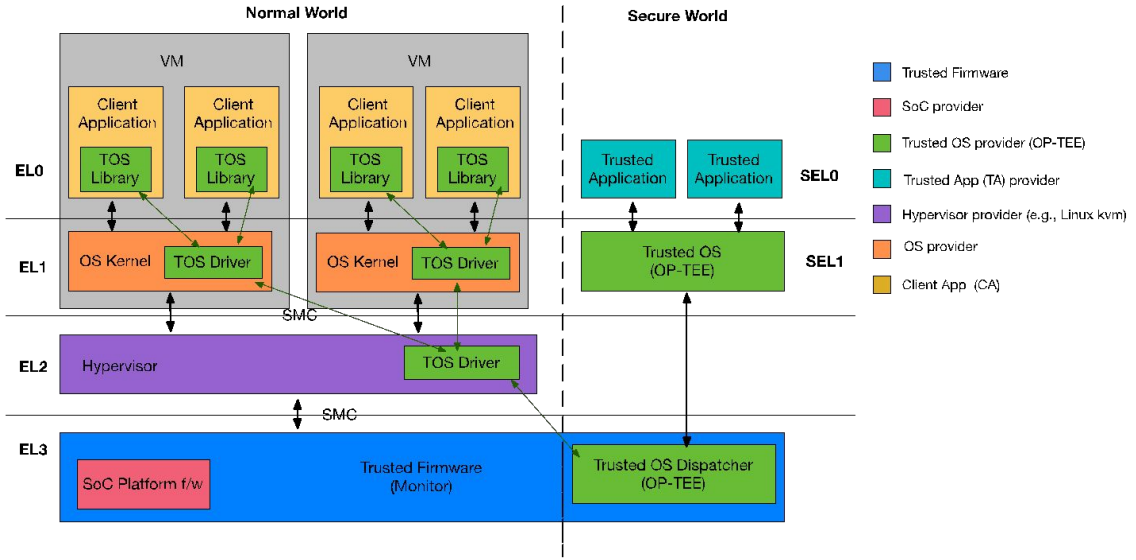
Multiple V8A AP Normal and Secure Worlds, one set per processor

Normal World

- Exception Levels
  - EL0: user mode – applications
  - EL1: kernel mode – OS kernel
  - EL2: hypervisor mode – hypervisor
  - EL3: Secure Monitor mode – manages secure transition from/to Secure World
    - Power State Coordination Interface (PSCI) platform code by SiP

Secure World

- Exception Levels
  - SEL0: secure user mode – trusted apps
  - SEL1: secure kernel mode – trusted OS



V8A AP Normal and Secure World – with OP-TEE as Secure OS

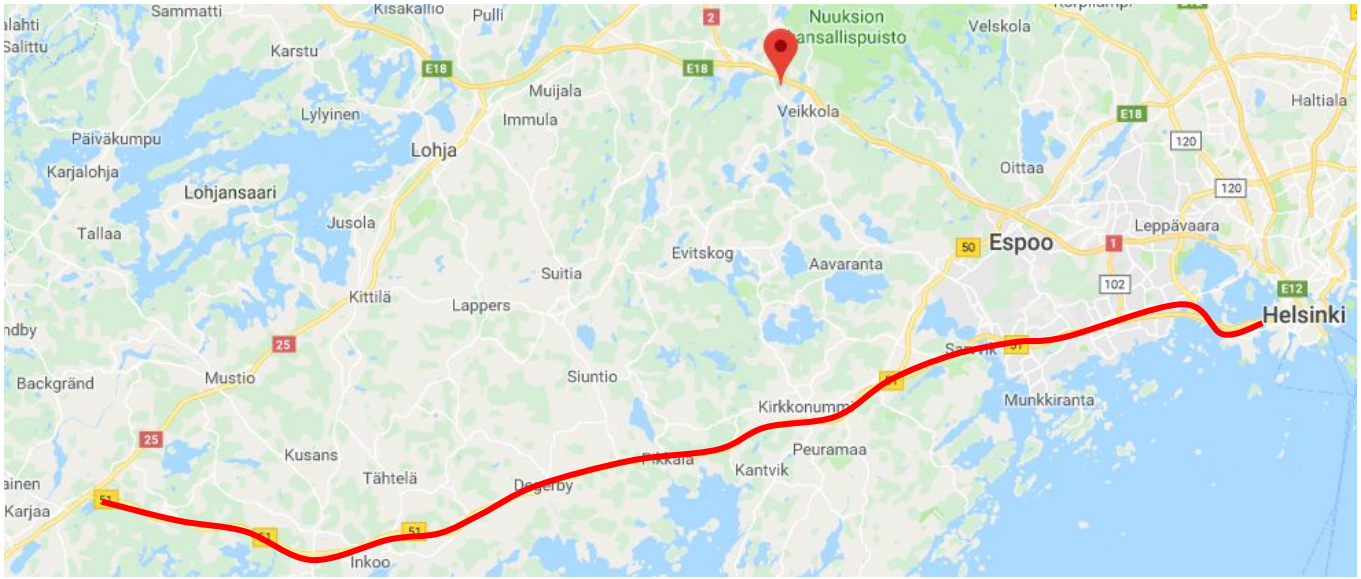
NOKIA

Demo!

Demo!

Finland's 5G plans: 'We want to build world's smartest road'

LED-light poles, base stations, and antennas will create a 5G network with uses ranging from self-driving cars to helping avoid moose collisions.



Demo!