# Multi-layered Security Technologies

for hyper-connected smart cities

## D4.3: M-Sec cloud and data level security - first version

December 2019

## Grant Agreement No. 814917

# Multi-layered Security technologies to ensure hyper-connected smart cities with Blockchain, BigData, Cloud and IoT

| | |
|---|---|
| **Project acronym** | M-Sec |
| **Deliverable** | D4.3 M-Sec cloud and data level security - first version |
| **Work Package** | WP4 |
| **Submission date** | December 2019 |
| **Deliverable lead** | CEA / YNU |
| **Authors** | CEA / YNU / KEIO |
| **Internal reviewer** | WLI / NII |
| **Dissemination Level** | Public |
| **Type of deliverable** | DEM |
| **Version history** | - v00, 08/11/2019, CEA, Full ToC, assignments and initial content<br>- v01, 26/11/2019, CEA, introduction and initial contribution<br>- v02, 10/12/2019, YNU, initial contribution<br>- v03, 17/12/2019, YNU, second contribution<br>- v04, 17/12/2019, CEA, second contribution<br>- v05, 17/12/2019, KEIO, initial contribution<br>- v06, 19/12/2019, CEA, final integration, ready for review<br>- v07, 30/12/2019, CEA, reviews integrated, ready for submission |

The M-Sec project is jointly funded by the European Union's Horizon 2020 research and innovation programme (contract No 814917) and by the Commissioned Research of National Institute of Information and Communications Technology (NICT), JAPAN (contract No. 19501).

# Table of Contents

# List of Tables



# List of Figures

# Glossary

| | |
|---|---|
| AES | Advanced Encryption Standard |
| CPU | Central Processing Unit |
| ECC | Elliptic Curve Cryptography |
| ECDSA | Elliptic Curve Digital Signature Algorithm |
| ECH | Elliptic Curve Diffie-Hellman |
| IDS | Intrusion Detection System |
| HMAC | Hash-based Message Authentication Code |
| HSM | Hardware Security Module |
| MCU | Micro Controller Unit |
| MPU | Micro Processor Unit |
| RSA | Encryption algorithm named after its inventors: Rivest, Shamir and Adleman |
| SPI | Serial Peripheral Interface |
| TCG | Trusted Computing Group |
| TPM | Trusted Platform Module |

# 1. Introduction

## 1.1 Scope of the document

In this document we present the demonstrators prepared in Task 4.2 of the M-Sec project related to cloud and data-level security. Three demonstrators are described:

- Hardware-based security, an asset which extends IoT device by bringing strong and complex encryption scheme as well as authentication capabilities to enforce the mutual verification of the connection between the device and the cloud
- Software-based "Visualization Tool for Security", an asset that utilizes the security data from IoT based Intrusion Detection System (IDS) for security monitoring purposes.
- GANonymizer which processes video streams in order to remove sensitive/private data such as people walking on streets and also cars.

## 1.2 Relationship to other work packages and tasks

The work done in task 4.2 is the result of many other tasks. In the first place, this task integrates the work of WP3, in particular those resulting from task 3.2 concerning architecture. Indeed, the work (described in Deliverable 3.3) specified a functional architecture and initiated an instantiation of this functional architecture with respect to the use cases defined in WP2 (indirect dependence). Another highly scalable task for data and cloud security is the risk analysis currently underway in Task 3.3. The first results of this task are used to identify and prioritize threats on the M-Sec reference architecture.

Finally, there is a close link with the other tasks of WP, in particular with the neighbouring tasks that are task 4.1 as well as task 4.3. Indeed, the objective of task 4.2 is to provide technologies to secure the exchange between data from IoT (4.1) to a remote registry (4.3). Concerning task 4.1, data security and a common problem. The encryption mechanisms depend directly on the nature of the IoT Device and we propose data security methods that rely on both, software and hardware integration developed in task 4.1. With respect to task 4.3, we are introducing through our developments means of strong authentication of data via encryption mechanisms. This strong authentication makes it possible to ensure to a distributed register the source of the data and thus to make a decision on the permanent inscription or not in the register.

## 1.3 Methodology followed

Regarding the overall topology as described in Figure 1, the work carried out in task 4.2 concerns the security from the devices to the cloud/application via the blockchain system. Given the use-cases, data flows commonly to a private environment (known devices, private / managed network, etc) to a public environment (cloud). The security technologies at the cloud and data level must take this change of perimeter into account and protect the private environment from attacks that may originate from the public network (attacks coming from the private network are treated in T4.1).

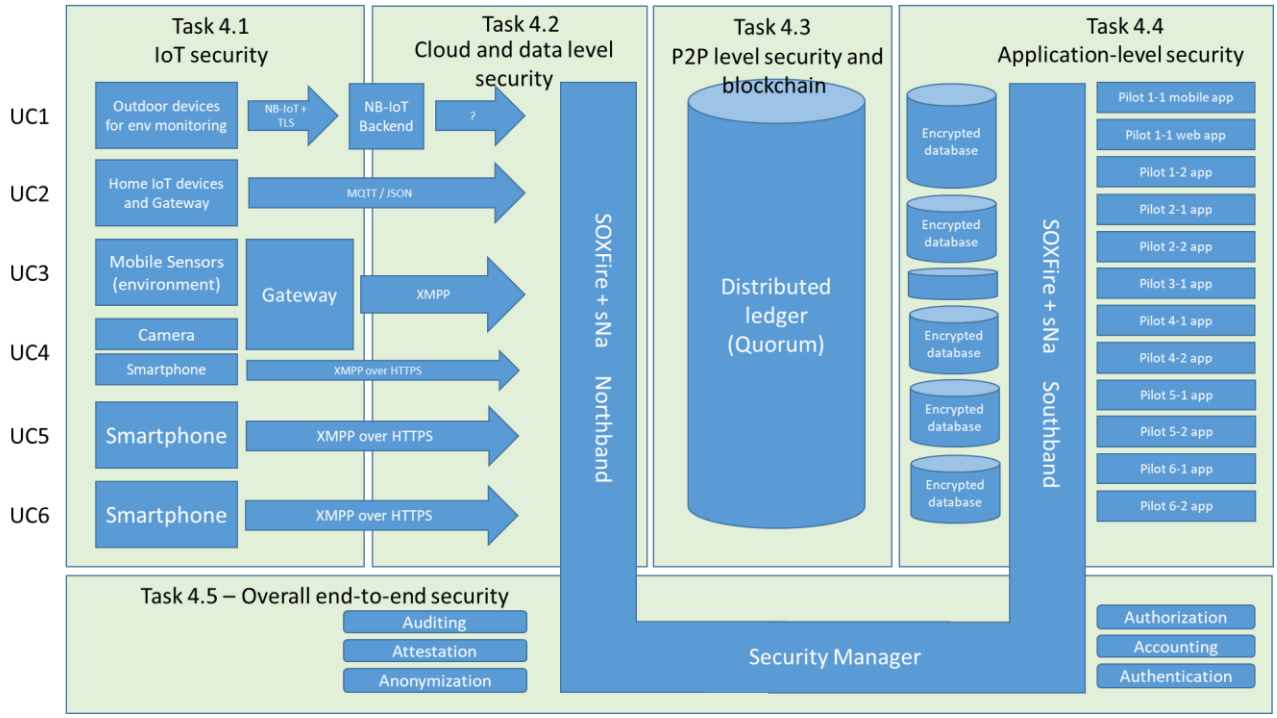


**Figure 1. Overall M-Sec topology**

The demonstrators presented in this deliverable aims to address risks, being assessed in Task 3.3, given the CIA (Confidentiality, Integrity and Availability) properties as follows:

- The hardware-based encryption provides a high level of encryption with good resilience towards cryptographic attacks. It mainly provides the **integrity** of the data and the encryption/decryption process.
- The threat monitoring increases the **availability** of the system by detecting attacks and enabling quicker responses.
- The GANonymiser provides **confidentiality** of the information by removing sensitive data in a stream. It is an essential component for the GDPR compliance as well as social acceptance.

# 2. Hardware based encryption

## 2.1 General Description of the Prototype

Data encryption is an operation that requires a lot of calculation steps. For an IoT product, generally constrained in computing resources and in energy, these operations are delicate because they can cause latency problems related to too-long computation time, or they can still use too much energy thus damaging a battery in a premature way. Similarly, encryption software implementations may be subject to vulnerabilities

that can be exploited. A typical example of such vulnerability is the “heartbleed” (CVE 2014-0160[1]) software vulnerability which was introduced in OpenSSL and has affected many devices such as smartphones, core internet routers and hard-drives firmware.

A countermeasure for such vulnerabilities is to use a hardware component, such as an HSM, a crypto-accelerator or a TPM. With such a device, it is expected to protect not only the private keys of the encryption, but also the process of encryption/decryption itself so it cannot leak data to unauthorized software. Our demonstration aims to show this enhancement on constrained devices, where the employment of secure elements is emerging. It relies on the developments carried out in T4.1 and presented in deliverable 4.1 where the same secure element is used to provide system integrity.

## Hardware target

This demonstration is made for typical IoT devices, powered by a micro-controller with energy limitation and using narrowband communication channel for direct connection to the cloud. Such a device is described in D4.1. It uses the STM32L4 microcontroller, this MCU plays the role of a connected object. For development purposes, we have used a NUCLEO-L476RG development boarding embedding such MCU. This device has a sensor attached to it, a potentiometer, simulating any analogue environmental sensor. It also has a STSAFE-TPM with an SPI connection to the MCU. These components are described in **Erreur ! Source du renvoi introuvable.**

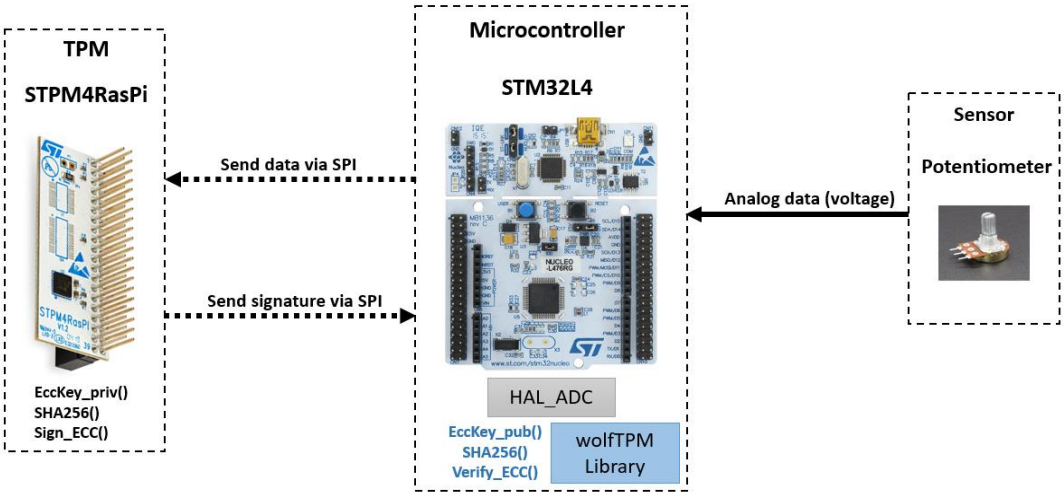


**Figure 2. Scheme for the hardware-encryption demonstrator**

## Software library

In order to integrate the TPM at the STM32L4 level some steps are necessary. First, we use the wolfTPM library from wolfSSL to integrate the TPM features. WolfTPM library is a compliant TPM 2.0 stack, designed for

[1] https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2014-0160 / https://en.wikipedia.org/wiki/Heartbleed

embedded use. It is highly portable, due to having been written in native C, having a single I/O callback for SPI hardware interface with no external dependencies, and its compacted code with low resource usage. This uses the TPM Interface Specification to communicate over SPI and Includes wrappers for Key Generation, RSA encrypt/decrypt, ECC sign/verify and ECDH and Symmetric AES encrypt/decrypt.

## 2.2 Package Information & Installation Instructions

### Required Tools and dependencies

We have used the "TrueSTUDIO for STM32[2]" tool in order to develop and build the demonstrator's firmware. Once the firmware is compiled from the source, it can be easily flashed to the device by simply copying the binary to the USB mass storage device which is mounted by plugging the device using a USB cable.

In order to run the demonstrator, the operator shall have the device, a USB cable with mini connectors and a computer with a serial terminal installed such as Putty under Windows or Minicom on Linux systems.

### User Manual

The TPM is connected and powered with the STM32L4 via SPI and on the other part the potentiometer is plugged into via ADC pins. In particular, the TPM shall be wired as described in Table 1

**Table 1. Connection layout between the MCU board and the TPM daughter-board**

| Function / signal | Pin on board | Pin on daughter board |
|---|---|---|
| **SPI1_SCK** | PA5 | TPM_pin_23 |
| **SPI_MISO** | PA6 | TPM_pin_21 |
| **SPI1_MOSI** | PA7 | TPM_pin_19 |
| **SPI1_CS** | PB6 | TPM_pin_24 |
| **rst_tpm** | PA9 | TPM_pin_18 |
| | VCC | VCC |
| | GND | GND |

To start the demonstration, we need to plug the board to a computer having a terminal using a USB-mini cable. The device will be powered-up directly from the USB port and will execute its firmware. Then, we can connect to its console using a terminal with the following parameters:

- Speed baud = 115200
- Data bits = 8
- Stop bits = 1
- Parity = None
- Flow control = None.

[2] https://atollic.com/truestudio/



In order to have the initial output of the firmware we need to reset the device using the black push-button located on the board as illustrated in Figure 3.

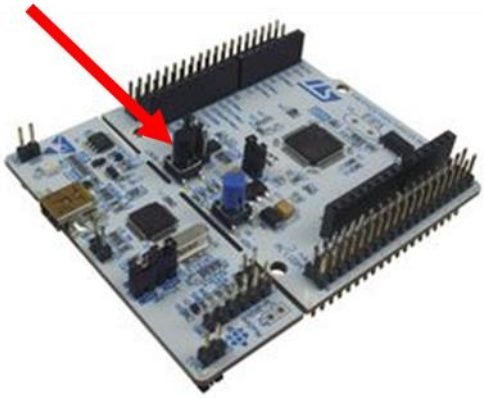

**Figure 3. Location of the reset button on the Nucleo-L476RG board**

Pressing reset of the STM32L4 gives this on the HyperTerminal as shown below:

```
//////////////////////////////////////M-SEC Demo using wolfTPM
API's//////////////////////////////////////

Enter G to start
```

At this stage, the device is unprovisioned and the first step is to provision the TPM with keys, keys for the session management and keys for the application.

First, we generate keys in order to preserve the privacy of the exchange between the TPM and the MCU. Indeed, as this exchange of information occurs on an SPI bus, an attacker can eavesdrops this communication. We prepare an authenticated session for the further commands from the CPU to the TPM which will encrypt most of the parameters in the next steps. To initiate authentication with the TPM, a session can be opened. When a session is started, the TPM processes the command and generates a session handle, computes a TPM nonce and calculates a session key. This key is used to generate HMACs, encrypt command parameters and decrypt response parameters. After the session is created, the session key remains the same for the lifetime of the session. The session handle and the TPM nonce are returned by the command.

The next step is to lunch the creation of the primary key derived from the static seed Endorsement Key (EK). This new primary derivation key is transferred and stored in an NV (Non-volatile) ram in the TPM. Subsequently, all the keys that will be used for the security primitives will be derived from the primary key.

- Then we generate a private key. To make a signature of the data, we use ECC (Elliptic-Curve Cryptography) asymmetric algorithm as a cryptographic scheme. This scheme begins to be widely used at the IoT level thanks to its energy performance. It has good efficiency in terms of implementation. Data coming from the sensor will be encrypted using this private key which will be kept into the memory of the component. This key is stored at the handle 0x80000004.

```
//////////////////////////////////////TPM provisioning//////////////////////
/////////////////
```

```
RESET---------------------------------> TPM
TPM2: Caps 0x30000495, Did 0x0000, Vid 0x104a, Rid 0x4e
TPM2_Startup pass
TPM2_SelfTest pass
TPM2_StartAuthSession: sessionHandle 0x3000000
TPM2_PolicyGetDigest: size 32
TPM2_CreatePrimary: Endorsement 0x80000000 (314 bytes)
TPM2_CreatePrimary: Storage 0x80000002 (282 bytes)
TPM2_LoadExternal: 0x80000004
TPM2_MakeCredential: credentialBlob 68, secret 256
TPM2_ReadPublic Handle 0x80000004: pub 314, name 34, qualifiedName 34
TPM2_Create: New ECDH Key: pub 88, priv 126
TPM2_Load ECDH Key Handle 0x80000004
TPM2_ECDH_KeyGen: zPt 68, pubPt 68
TPM2_ECDH_ZGen: zPt 68
TPM2 ECC Shared Secret Pass
TPM2_ECC_Parameters: CurveID 3, sz 256, p 32, a 32, b 32, gX 32, gY 32, n 32, h 1
TPM2_Create: New ECDSA Key: pub 88, priv 126
---------------ECC public key---------------
c8c619e4f69a3f5188c6568
fd735185a384fe8fae3f6b61
31baccd6fbc5773da57ff5c
7cc9da04e83e727d17e6dd
f81ecbb85da220dd8dd82cb5
1ebcf6518b037c73ac97429
705228338af7307748c5f178
b6c48469
---------------ECC private key---------------
02044da6792e442925da3
73de74e4661cc38335d58e7
ada0347c76dccb1f5735010
fe9ff41dd36020d79fb7530
244ac75c31875e1db8bfa0d0
fcd9f4c6e4187e58a2fdcd28
d285ec8a82acc633f8b4cc4
55b75ab92b4844ce15da7a2d
1a89f2114eb7263e8f741838
2bac83325be56c8c137c2272
1550d014deff
TPM2_Load ECDSA Key Handle 0x80000004
//////////////////////////////////////RUN TIME APPLI//////////////////////////////////////
```

Once the provisionning phase done, the application will loop over the following sequence

1) Read the analog value from the potentiometer
2) Send the analog value to the TPM for encryption
3) Retreive the encrypted value from the TPM

It corredponds to the following output

```
HAL_ADC_Start ---------------------------------------> AnalogValue : 807
Message to be signed ECC : F8 20 69 80 47 FE E7 BD E8 F0 81 64 20 E0 60 E0 68 41 1E 0 28 E1 60 FA
D1 70 47 68 68 60 61 20
TPM2_Sign: ECC S 32, R 32
TPM2_VerifySignature: Tag 32802
```

The value retrieved via the ADC is sent via SPI in the TPM, and then a SHA256 value is calculated in the TPM. Once the hash fingerprint is generated, it is encrypted with the private key. In order to verify the validity of

the previously signed data in the TPM, the public key is sent to the MCU, the signature is returned from the TPM to the STM32L4 and then decrypted using the public key generated in the TPM. A hash is calculated in the STM32L4 and compared with the one generated from the signature. If the two hashes are identical, the signature is validated. As a result, the integrity and authenticity of the data has been proven through the TPM.

### Licensing

WolfSSL, the crypto library with TPM support used in this demonstration is available under two licensing models, either open-source or commercial. More information is available on the editor website: https://www.wolfssl.com/license/

### Summary

This demonstration illustrates the use of a hardware component to ensure encryption of devices, in a context where it can also provide other security functions such as device integrity presented in deliverable 4.1. Having hardware-based encryption offers protection against software attacks targeting the encryption process itself or the storage in which the private keys are stored.

### Next Steps

During the second year, this work will be continued toward a remote attestation scheme, involving three roles such as prover, identifier and verifier. We think that this scheme is suited to the M-Sec architecture as it can provide an anonymization capability with Zero-Proof Knowledge implementations such as ECDAA (Elliptic Curve based Direct Anonymous Attestation). Initial tests have been carried out and need to be confirmed in order to overcome the pairing step specific to elliptic Curves cryptography.

# 3. Software-based Threat Monitoring

## 3.1 General Description of the Prototype

In order to stay vigilant and monitor threats to the IoT devices layer from anywhere in the cloud, an analysis tool is required that can translate the data into easy-to-understand graphical information. This visualization tool is a software-based solution that collects and examines activity from IoT layer or agents embedded in IoT gateway devices. This tool will not only help with the security health checks by providing insight into how the security is being maintained at IoT gateways, but also helps in further analysis of devices under attack. Thereby, providing 24/7 security threat monitoring and alerts.

## Visualization Tool

The software solution consists of four modules:

1) **Data Collection Agents**
   A software solution embedded in the IoT Gateways is used to collect logs generated by the IDS at the IoT security layer and forward the security data to the aggregator module.
2) **Aggregator Module**
   Aggregator agent in the cloud organizes the data in real-time for further analysis.
3) **Data Analysis Module**
   Data analysis engine examines all the data by matching with well-known attack patterns and forwards suspicious activity to the visual graphics module.
4) **Visual Graphics Module**
   Visual graphics module converts the data to easy-to-understand information for further investigation at the security monitoring station.


**Figure 4. Visualization Process**

As shown above in Figure 4 the process is simple. IDS in IoT Gateways monitors the public interfaces by matching traffic with known signatures. These signatures are extracted from various testbeds, including research and analysis involving the honeypot (IoTPOT). Embedded agent (filebeat) in IoT gateway uploads the logs to the aggregator module in the cloud for further analysis. The aggregator module collects all the data from various IoT gateways and passes it to the data analysis engine, which examines all the data with the help of the threat detection module. The results are flagged as alarms in the visual graphics module for further investigation and easy-to-understand security threat monitoring.

# 3.2 Package Information & Installation Instructions

## Required Tools and dependencies

Following are required for this demonstrator:

- Laptop/Desktop with CPU (i3 2.4GHz or above), Memory (16GB or above), Hard-disk (512GB or above), and an internet connection.

## User Manual

An installer has been compiled to download and install all the required open source software files along with customized settings for use case 3 and 4 pilots on the IoT gateway device. The software runs in the background and information is displayed on the monitoring screen.

The installation file is named as “installer.sh” that can be downloaded securely over any internet connection with the following command:

```
$ scp -P 64295 -i <key> rainforest@xxx.xxx.xxx.xxx:~/iot-k/installer.sh .
```

After downloading, check and confirm the integrity of the downloaded file using MD5 hash, as follows:

```
$ md5sum installer.sh
MD5: 5cb38fb1754267c7d699565f00e3262c
```

*Installation Guide*

1) The “installer.sh” file should be downloaded into the “/root” directory and the directory should look something like this:

```
root@iot:~# ls -l
total 40712
-rw-r--r-- 1      root root          0 Sep 27  2017 1
-rwxr-xr-x 1      root root          41678664 Nov 26 16:59 installer.sh
-rw------- 1      root root          1675 Nov 26 16:32 key.pem
root@iot:~#
```

2) Launch installer shell as follows:

```
root@iot:~# chmod +x installer.sh
root@iot:~# ./installer.sh
installer/
installer/start_ips.sh
installer/suricata-4.1.5.tgz
installer/root.tgz
:        :        :        :
:        :        :        :
```

3) Wait for it to complete updates, download, and install the program. It will finish installation and return the prompt as follows:

```
:        :        :        :
Index setup finished.
Loading dashboards
Loaded dashboards
Loaded machine learning job configurations
Loaded Ingest pipelines
root@iot:~#
```

4) Confirm that filebeat program is running as follows:

```
root@iot:~# /etc/init.d/filebeat status
```


If not running, then start it with the following command:

```
$ /etc/init.d/filebeat start
```

### *Running Demonstrator*

PC client can connect with the filebeat server from anywhere using secure pre-set credentials and shall look something like as shown in Figure 5 and 6. This is a visual tool for security-related events or alarms. User needs to login with the pre-set credentials and watch for security events/alarms and examine security reports. If further investigation is needed, then a security expert should examine the logs/alerts and take appropriate actions, ranging from further data analysis to updating threat signature patterns on the IDS.


**Figure 5. Client Dashboard – Alerts Overview**

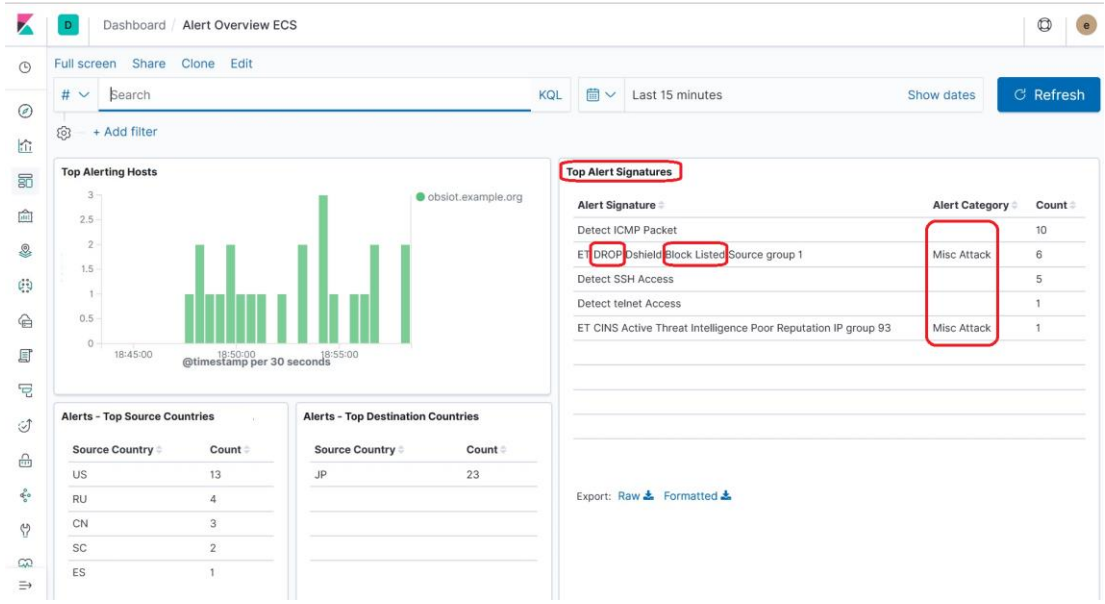


**Figure 6. Client Dashboard – Events Overview**

### Licensing

- This tool is based on Filebeat and Kibana open source code, customized according to M-Sec needs and requirements.

### Reference use case

Use-case 3 and 4, both benefits from this security threat monitoring tool for the secured mobile sensing platform.

### Summary

This is a security monitoring tool that provides visual graphics for easy-to-understand information related to ongoing security activities at the IoT layer. This tool sits in the cloud and collects logs for threat monitoring. The collected data not only provides a security health check of IoT gateways, but can also be used for further threat detection.

### Next Steps

Deploy and test it in the field as part of preparations for Pilot 3.1 and 4.1.

# 4. Privacy Management Tool

## 4.1 General Description of the Prototype

Use cases for handling video data in various situations in smart city solutions are increasing. For example, surveillance video for security, video monitoring of transport infrastructure and social infrastructure, people flow analysis, such as disaster prevention measures, etc. But in many cases, such video data includes privacy information that can identify an individual. It is, therefore, necessary to remove such personal information that can become a violation of General Data Protection Regulation (GDPR) and Japanese regulations. Hence, an application called "GANonymizer" is being developed as a solution that automatically removes objects related to personal information using Deep Learning.

### Video Image Privacy - GANonymizer

GANonymizer is an imaging processing tool to remove (make transparent) the pedestrians and cars recorded in the driving record videos. Its objective is to avoid the privacy leakage when distributing and utilizing the videos. The GANonymizer is developed using deep learning-based object detection techniques and is currently designed to run securely on the KEIO's secured mobile sensing platform for use case 3 & 4, for addressing the privacy issue.

**Figure 7: GANonymizer Test Results**

The results of applying GANonymizer are shown in Figure 7.

- Upper row images are original video images
- Lower row images are after removing privacy objects, like cars and human objects.

GANonymizer consists of two parts of neural networks as shown in Figure 8.

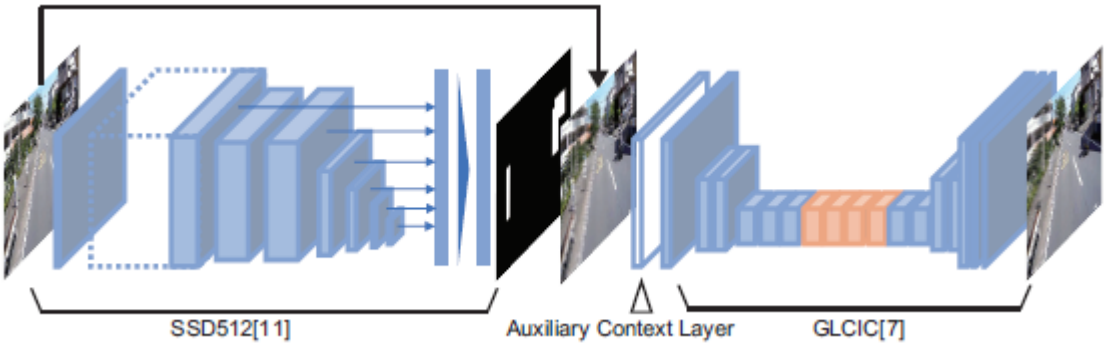


**Figure 8: GANonymizer Architecture**

In order to detect the target objects from the input image, which might violate the privacy, we adopt the deep neural networks: Single Shot Multibox Detector (SSD). And in order to generate a more natural image, we adopt Globally and Locally Consistent Image Completion (GLCIC) which is one of the most successful models in image completion.

## Single Shot multibox Detector (SSD)

SSD is one of the popular models that can detect the object with high accuracy. Especially, we select SSD512 which is the variant SSD model and performs higher than any others. Since the target objects are general and those are contained in PascalVOC dataset, we use the model weights which are trained by PascalVOC.

## Globally and Locally Consistent Image Completion (GLCIC)

After the target objects are detected, GANonymizer replaces the area where the target objects exist as if there are no objects. There are a lot of completion methods using computer vision technology. We adopt the in-

painting methods which adopt the deep neural networks to succeed in generating the images more realistic and natural.

GLCIC is based on Generative Adversarial Networks (GAN) and consists of three networks: the completion network, a local discriminator network, and a global discriminator network. Since GLCIC requires an image and corresponding binary mask for its input, GANonymizer creates the mask based on the bounding boxes which are the outputs from SSD512. Then GLCIC reconstructs the mask part of the input image based on the whole image and is trained by the procedure of GAN. The local discriminator assesses the quality of the mask part of the input image which is completed by the completion network. Simultaneously, the global discriminator assesses the quality of the entire image which is completed by the completion network. The training is terminated when the discriminator networks cannot distinguish between the original input image and the image which is reconstructed by the completion network, that is when the completion network becomes able to reconstruct the mask part of the input image realistically and naturally.

In terms of object removal, it is significant to naturally reconstruct masks based on the various background of images, Hence, for our GLCIC, we apply the model trained with the places dataset, which contains the pictures of the various place, so that it can reconstruct the mask more naturally.

## 4.2 Package Information & Installation Instructions

Currently, the prototype has been made only for testing purposes. There is no installation package of GANonymizer yet, but we are planning some installation package in the future.

### Required Tools and dependencies

The development environment for the current prototype of GANonymizer is as follows:

- CPU: Intel(R) Core(TM) i7-6950X @ 3.00GHz, GPU: NVIDIA GeForce GTX 1080 – 4 cores, RAM: 64GB, and OS: Ubuntu

### User Manual

This will be included within the installation package, in the next phase.

### Licensing

Currently, the GANonymizer includes some proprietary code, but open source is planned in the next phase.

### Reference use case

The reference use case of GANonymizer is Use Case 3, but we also can apply GANonymizer in other UCs which use video including privacy image.

### Summary

There are many situations that use video data that also includes privacy data, in general. So, the GANonymizer solution has been developed to address such privacy concerns. By removing identifiable information at the IP camera video source before transmitting the data reduces the risks associated with privacy leakage. Therefore, it could be applied for not only current use-cases in the M-Sec project, but also in other smart city pilots in the future.

### Next Steps

Test it in the use-case pilot 3.1 in 2020. Further improve its processing speed.

# 5. Conclusion

The three demonstrations presented in this deliverables enhance the security of data between the devices and their respective backends in complementary ways. At first we show how encryption on IoT devices can be made in a safer manner in a domain in which the integration of complex algorithms is quite often neglected due to a lack of resources. Then, we have presented a tool that can monitor deployed devices and enables detection/responses facing attacks. Finally, we present a privacy-enhancing technology that removes any individual or cars on video streams that can be captured in the public space.

At this stage, the security technologies demonstrated in this deliverable are planned to be integrated with the use cases as described in Table 2

**Table 2: Demonstrators and its correlation with Use Case Pilots**

| Demonstrator | Type | Use Case Pilots | Purpose |
|---|---|---|---|
| **Hardware-based encryption** | Hardware-based solution | Use Case 1<br>Pilot 1.1<br>Pilot 1.2 | Provide embedded security layer to IoT devices |
| **Software-based Threat Monitoring** | Software-based solution | Use Cases 3 & 4<br>Pilot-3.1<br>Pilot-4.1 | Secure IoT mobile sensing platform by monitoring and preventing cyberattacks |
| **Image Processing Tool for Video Privacy (GANonymizer)** | Software-based solution | Use Case 3<br>Pilot-3.1 | Enable privacy on the video feeds from IP cameras |

We expect these demonstrators to be enhanced in the next period for the final demonstration with more integration within the other tasks, especially regarding the T4.1 and T4.3 which are neighbours in the architecture. Finally, we will improve the management of security by integrating the functions provided by the T4.5.