An Embedded AI-Based Smart Intrusion Detection System for Edge-to-Cloud Systems

Ritu-Ranjan Shrivastwa, Zakaria Bouakka, Thomas Perianin, Fabrice Dislaire, Tristan Gaudron, Youssef Souissi, Khaled Karray, Sylvain Guilley

An Embedded AI-based Smart Intrusion Detection System for Edge-to-Cloud Systems

Ritu-Ranjan Shrivastwa[1][0000−0002−8909−0406], Zakaria Bouakka[1], Thomas Perianin[1], Fabrice Dislaire[1], Tristan Gaudron[1], Youssef Souissi[1], Khaled Karray[1], and Sylvain Guilley[1,2][0000−0002−5044−3534]

[1] Secure-IC S.A.S., 15 Rue Claude Chappe Bât. B, 35 510 Cesson-Sévigné, France
{firstname.lastname}@secure-ic.com
https://www.secure-ic.com/
[2] Télécom-Paris, 19 place Marguerite Perey 91 120 Palaiseau, France
https://www.telecom-paris.fr/

Abstract. This article proposes a general purpose IoT framework usually applicable to all Edge-to-Cloud applications and provides an evaluation study on a use-case involving automotive V2X architecture, tested and verified on a toy smart-car in an emulated smart-car environment. The architecture in study is finely tuned to mimic actual scenarios and therefore the sensors available on the toy car encompasses almost all the sensors that assist a regular ADAS in smart cars of today. The cloud connectivity is maintained through the CoAP protocol which is a standard IoT connectivity protocol. Finally, the security solution proposed is that of a smart Intrusion Detection System (IDS) that is built using Machine Learning (ML) technique and is deployed on the edge. The edge IDS is capable of performing anomaly detection and reporting both detection results as well as sensor collected big data to the cloud. On the cloud side the server stores and maintains the collected data for further retraining of ML models for edge anomaly detection which is differentiated into two categories viz. sensor anomaly detection model and network anomaly detection model. To demonstrate Software update Over The Air (SW-OTA) the cloud in the evaluation setup implements a ML model upgrade capability from the cloud to the connected edge. This implementation and evaluation provides a Proof-of-Concept of the choice of ML as IDS candidate and the framework in general to be applicable to various other IoT scenarios such as Healthcare, Smart-home, Smart-city, Harbour and Industrial environments, and so on, and paves way for future optimization studies.

Keywords— Edge Computing, Artificial Intelligence, Cybersecurity services, Embedded security, Anomaly Detection, Intrusion Detection Systems, V2X, Internet of Things, Advanced Driver Assistance Systems) (ADAS).

1 Introduction

The connected device market is getting flooded as technology becomes more scalable and computing resources at the device level increased. Thus, the IoT is no longer a

fancy concept and has become the need to solve the real-time crisis as researchers observe possibilities to optimize livelihood in every sector of human civilization through a properly connected device infrastructure. This in turn is boosting the growth of standards and protocols to unify the development process which also, not so surprisingly, is opening new attack surfaces. Autonomous cars are running along-side manual cars and Artificial Intelligence is diagnosing medical symptoms in patients. The applications of sensing and actuation and end nodes of a IoT network are in fact pushing the Data Scientists to properly handle the generated Big Data streams and utilize it to improve the services for the end customers. Implementing sophisticated features implies that the security needs to be inherently robust to handle such complex system and therefore, prevent from compromising the whole system since, depending upon the use-case, there might be a safety risk involved. To prevent the exploitation of the smart solutions by adversaries on the edge side which is exposed to threat actors, it is important to have a smart solution that is able to track minute differences in operational environment and alert the mother system at the edge or cloud level.

The motivation behind this work is to propose a smart Intrusion Detection System (IDS) in a connected edge-to-cloud system that is capable of sensing from every sensory node available on board and aggregate the results of anomaly detection from each and report back to the cloud. This is achieved by mimicking the architecture of a smart car (V2X) ecosystem (where the car is connected to the cloud and its locomotion greatly depends on its on-board sensors) through a toy smart car and emulated environment with dedicated threats with a nearly full coverage on all sensing equipment on the toy smart car.

The idea is to emulate threats at both sensor and network level (all possible attacks to disrupt the functionality of the smart car) and develop a system capable enough to observe these differences in a real-time scenario. The experimental setup consists of a toy smart car with on-board sensor array connected to the internal IDS which is composed of two separate Machine Learning (ML) cores, one for detecting anomalies through the sensor data and one for detecting anomalies through the network data (packets). The toy smart car is connected to a cloud server through popular IoT connectivity protocol CoAP and transmits detection signal back to the cloud along with the collected data from the sensor and network. Through careful and lengthy evaluation of a multitude of scenarios and through a rigorous experimentation process, we find that the proposed IDS is suitable to a wide variety of input data and therefore can be universally applied to any IoT use-case.

The rest of the paper is organized as follows: the section 2 provides a generic background of topics linked with the proposed work by running through the problem statement and navigating through the solution while talking about various aspects of the system. It also provides some common standards that are (being) developed to streamline IoT product development. Next, the section 3 provides the details about the proposed method with details about the evaluation concept, experimental setup, and results. It also talks about the implications of the same in the real world. Finally, the section 4 presents other important topics that are extremely relevant, and acclaimed by technology providers, to the edge-to-cloud ecosystem. Eventually, Section 5 concludes the paper.

2 General Background

2.1 Edge-to-Cloud

Today's devices are clearly becoming smarter by having more and more interactions with the outside world. Such interaction is offering much more capabilities and obviously opening new ways for new applications targeting most of technology ecosystems such as connected vehicles, Industry 4.0, smart cities, healthcare, smart agriculture, smart homes, etc. In the literature, those smart devices are often called edge devices if they have the capability to ensure back-and-forth connectivity with other devices or with a central system that we call Cloud server in the sequel. The edge device is basically composed of a processing unit that can be an MCU with low resources or an MPU with more power and computing resources. edge devices themselves can be used as bridges between a server and Internet of Things objects (IoT). Typically, we can define an edge-to-Cloud system as a technology composed of three main actors as follows:

- **Actor 1**: the edge device that comes with a connectivity module, alongside a host CPU for the software, and a hardware layer.
- **Actor 2**: the sever side is a machine with much more power and computing capabilities. It is a central element that talks with a fleet of edge devices. The server shall come with application services to manage and monitor the connected devices.
- **Actor 3**: the user interacting with the server to send requests to edge devices and monitor the fleet of edge devices. Users could have different privileges and roles with regards to the server.

An illustration of such system is depicted in figure 1.

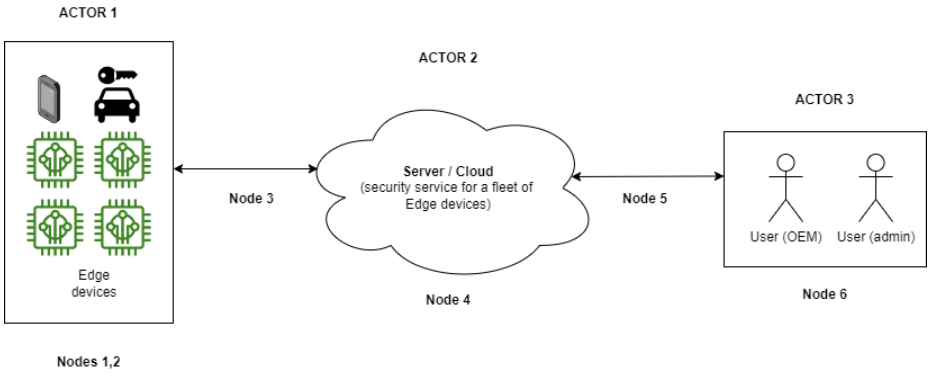

Fig. 1. edge-to-cloud main nodes and actors.

Very often, any similar technology faces two basic challenges, namely security and performance. Safety is a third challenge that shall be considered for related technologies as automotive and avionics. In the sequel we focus only on security. In fact, security is

the way how data is protected. The data is by default sensitive and can be everywhere, at rest or in-transit. Based on the figure 1, we can distinguish 6 nodes where end-to-end security shall be considered in edge-to-Cloud context as follows:

- **Node 1: at the hardware layer of the edge device.** The security here is generally managed by a technology dependent secure layer as a Trusted Execution Environment (TEE), Trusted Platform Module (TPM), or a dedicated Secure Element (SE), as described here [24]. Such layer ensures a strong security level as per data isolation, secure storage of secrets, etc. A TEE consists in separating the same Host processor into two spaces: normal space and secure space. All security operations shall run within the secure space by ensuring an isolation with the normal space. However, such security mechanism is less secure than a dedicated hardware as TPM or SE. In fact, within a TEE, the shared components as internal Host processor memories might leak sensitive data. The TPM is theoretically more secure than TEE as it comes with a separated hardware chip. However, the leakage might come from the link between the Host processor and the TPM. In fact, the data in transit might be probed and stolen if not encrypted. Finally, an integrated secure element shall be the most secure as it is embedded in the same SoC as Host processor. That said, physical attacks such as Side-channel attacks (SCA) [17] or Fault injection attacks (FIA) [5] are the first enemy against the hardware layer. Fortunately, countermeasures like data hiding exist.
- **Node 2: at the CPU Host layer of the edge device.** The security of data shall be considered by the Host processor that would implement a software bridge handling a secure channel with the server side. The Host processor shall be able to use cryptographic software engines if security hardware components are not available. For this purpose, the processor shall manage the secure communication with the server side by supporting software clients for security protocols as TLS and DTLS alongside crypto libraries as OpenSSL [29]. The processor might use cryptographic embedded hardware accelerators for performance purposes. Globally, the goal is to ensure that the edge device is identified, authenticated, and authorized relatively to the server side. For this purpose, the Host might hold and manipulate device IDs considered to be sensitive and that need to be protected. Moreover, the Host layer is in charge of all the software typically running at bare metal or OS level. That software might be obfuscated or signed and encrypted for more security. In fact, threats like malwares, binary reverse engineering are still redoubtable against Host layer.
- **Node 3: at the connectivity layer between the edge device and the server.** The connectivity layer is all about network stack ranging from the physical channel to applicative protocols. Edge devices are basically communicating over IP-based channels as Ethernet, WiFi, Cellular (4G, 5G, 6G, etc.), etc. Some RF protocols use an encapsulation technique to allow IP-based communication. The security shall consider all the layers of the network. The OSI model for instance suggests securing the lowest layers with MACSec (for data link) or IPSec (for transport). Then application protocols are proposed as TLS and DTLS. Higher applicative frameworks for connectivity like LwM2M, MQTTS come with a set of schemes to securely manage a device. As a matter of fact, LwM2M is based on CoAP and DTLS protocols to initiate a communication with an edge device.
- **Node 4: at the server core layer including its data storage components.** The server is the central element in the system. It manages the input and output data from edge devices. Security is a big matter and should at least be ensured

for the data at rest like edge devices' logs and users' credentials, often stored in databases; data in transit like direct requests from users to edge devices; or also the server components and interactions between those components themselves. In fact, the server is the most impacted node as it is exposed to internet. In other words, it is the target of a tremendous number of cyberattacks. As a matter of fact, a long list of cyberattacks is regularly updated by the OWASP web pentesting framework group. Hence, security shall be thoroughly checked from the server infrastructure level to applicative micro-services. The literature has recently proposed a new approach with several security requirements, called "zero-trust", that aims at maximizing the security at cloud server node.

- **Node 5: at the connectivity layer between the server and the user machine.** Same as for node 3, the connectivity here is more about the relationship between the user and the server. The security of this node is crucial as it deals with user credentials and devices registration initial inputs alongside secret data as keys and certificates. Thankfully, a known approach called IAM that stands for Identity Access Management, comes with a set of tools, protocols, and frameworks to securely authenticate, and authorize users to access the server. We cite 0Auth2 [14] for instance. In addition to that, security could be reinforced by a double authentication technique as it is proposed by FIDO2. Moreover, the security could be maximal by combining such software-based solutions with hardware tokens. The attack surface is about letting data to be transferred as plaintext without any mutual authentication or privilege mechanism. The attacks are numerous like data sniffing, probing, fuzzing, man-in-the-middle, etc.
- **Node 6: at the user machine level.** This node regards the user machine that interacts with the server side. Most commonly known attacks are performed on software web browsers and interfaces. Technically, this represents the front-side of the server solution that can be a web interface, a web application, a command line interface, an exposed API, etc. The security scope is about all the known attacks as SQL injections against databases, cross-site scripting attacks (XSS), traversal directory attacks, etc.

2.2 AI and Machine Learning for Anomaly and Intrusion Detection

Artificial Intelligence (AI) and particularly the Machine Learning (ML) subfield provide powerful prediction algorithms that constitute state-of-the-art techniques in several research areas: image processing, natural language processing, medical diagnosis, etc. Naturally, those methods are also drawing increasing interest in the cybersecurity landscape. Indeed, the advanced modelling capabilities of ML algorithms allow to leverage on large quantities of available data and knowledge to improve security systems in various fields of application. ML approaches are perfectly suited for attack or failure detection applications as they allow creating a model of the normal predictable behavior of a system [25]. After this profiling phase, it becomes possible to detect significant deviations from the model. A typical example of application are intrusion detection systems which analyse a network traffic to detect, block and report malicious packets. A "traditional" IDS uses a database of known malicious signatures that compares with the incoming packets to detect attacks. This approach presents a significant drawback: it detects attacks based on known threats and is unable to handle new attacks. On the other hand, it is possible to use ML algorithms to create a model of the normal behaviour of a network and to detect abnormal activities based on the observed devia-

tions from the base profile. This approach has the advantage of detecting unknown or zero-day attacks.

The same idea can be applied to sensor data analysis. Standard deployment of fleet of sensors requires calibration, and threshold-based analysis is necessary to process sensor values, which often leads to false positives. AI-based sensor aggregation and analysis enable detection of fault injection attacks, anomalies and failures, and advanced diagnosis [11], while reducing the number of false alerts. To build such a system, a test chip is characterized in controlled environment, in order to generate sample data and train a detection model to be deployed on the final chip. Then, in operation, the model classifies new data, provide useful information (attack? anomaly? failure? type of attack?) and report to upper layers. Based on desired security policy or user feedback, the detection sensibility can also be adapted after deployment.

It is worth mentioning that AI techniques can also be used as tools to conduct attacks or assist security evaluations. For example, in the field of side-channel analysis, ML-based methods are employed to process measurements obtained during the execution of a security target to extract secret values such as private keys [21]. However, Machine Learning systems can also be the target of attacks, Typically, fault injections attacks can be used to fault the computation of a neural network and to bypass a security verification. More recently, a whole new class of attacks, called adversarial attacks [13], have been designed to fool ML algorithms by crafting malicious inputs which can go through neural network-based detection.

2.3 Connectivity

Over the past years, the rise of IoT market highlighted the need for efficient communication layers with small footprint, allowing low end devices like sensors or cameras to ensure low power consumption, low network bandwidth and a long living time if powered by a battery.

In the edge-to-cloud context, some protocols emerged and were standardized by the industry for machine to machine (M2M) communication. We can mention the most commonly used ones:

- Message Queue Telemetry Transport (MQTT) [28], based on the publish-subscribe methodology allowing one-to-many communication through brokers.
- Constrained Application Protocol (CoAP) [27], a client-server protocol offering a RESTful interface allowing a client node to directly command another node.

Both protocols are IP-based and come with activable security features: MQTTS using TLS for MQTT, and DTLS or IPSec for CoAP, ensuring transport encryption and mutual authentication.

More recently, higher level protocols including an application layer have been presented as standards. OMA Lightweight M2M (LwM2M) [4] is a protocol for device management and service enablement, defining the application layer communication protocol between a server and a client, which is the IoT device. In the same way, the Matter protocol (previously known as Project CHIP) [2] for home automation connectivity is being standardized. It promises interoperability among smart home devices from different vendors and IoT platforms.

2.4 Security Standards and Frameworks

The aforementioned edge-to-cloud type IoT systems involve massive data transfer through the network that includes sensitive information, unauthorized access to which might jeopardize the whole system leading to security and even safety related hazards. Therefore, such systems are equipped with state-of-the-art cyber-security mechanisms to deal with security threats and intrusions. To ensure a secure development and assure the consumer of the degree of competence and robustness of the systems in terms of security features, it is essential to go through the certification process of such products. Any edge-to-cloud system might fall under a typical IoT framework and therefore it becomes quintessential to secure the system based on standard practices. The widely acclaimed certification schemes that would help reach such standards of security are listed below:

1. **Common Criteria (CC)** - The [7] (ISO/IEC 15408) provides seven assurance levels based on which any general purpose product can be certified. The framework is based on the IT product security but the presence of protection profiles makes it more suited for different market verticals. Certification with CC is gaining momentum for IoT products as more and more companies enter to compete with this product line, since IoT has already reached to our homes.
2. **SESIP** [12] is a platform-level certification scheme for IoTs, promoted by the Global Platform association. It is meant to be similar in the rigor to the CC albeit with a more simple applicability.
3. **FIPS 140-3** - The FIPS 140-3 [19] (ISO/IEC 19790) from the NIST (USA) is an upgrade over the extremely popular FIPS 140-2 standard. It provides a technical baseline for security products and can be easily framed around IoT ecosystem. With four distinct levels of security, the Cryptographic Module Verification Program (CMVP) ensures that the security of the product is optimized as per the security scope.
4. **ENISA Cloud certification scheme** ENISA [9] is a popular and respected European agency for cybersecurity that provides useful standards from time to time. In 2020, it provided a draft certification scheme for cloud based applications targeting the IoT edge-to-cloud products.
5. **Eurosmart IoT certification scheme** This standard [10] based on the European Cybersecurity Act, provides a targeted framework for IoT products with three different levels of security such as basic, substantial, and high.
6. **PSA Certified** Similar to the CC, the PSA Certified [22] provides third-party lab evaluation for security assurance for IoT product vendors and manufacturers with three distinct assurance levels.

3 Monitoring Cyber Use-Case

This section describes our proposed embedded IDS architecture and implementation, designed to be deployed on fleets of devices in the context of IoT monitoring.

3.1 Typical Architecture

The proposed IDS is comprised of multiple anomaly detection cores, each one being in charge of processing a different type of inputs. The IDS can be deployed with a

variable number of detection cores, depending on the target device. The architecture described in this paper contains two cores: for WLAN connectivity and for sensors. Additional cores can be plugged, for CAN intrusion detection for instance. Each core has access to a local storage where a detection model is stored. The global architecture is summarized in Figure 2 for an automotive use-case which we chose for the evaluation in our work.

A COAP server is used for bi-directional communication with the cloud server side: on one hand, for sending notifications to the cloud and on the other hand, for configuring the edge IDS from the cloud server. The cloud is equipped with tools for monitoring real-time data from the connected device and displays anomaly notification.

The edge, along with the telemetry of core anomaly detectors, also uploads the aggregated sensor data. This data is collected in the cloud in a desired format which is later used for offline-training to further tune the Machine Learning core. Finally, the cloud has the capability, through the CoAP channel, to push a new ML model with newly trained/tuned parameters based on more collected data on the edge, and update the Intrusion Detection System. This is a typical Software update Over The Air (SW-OTA).

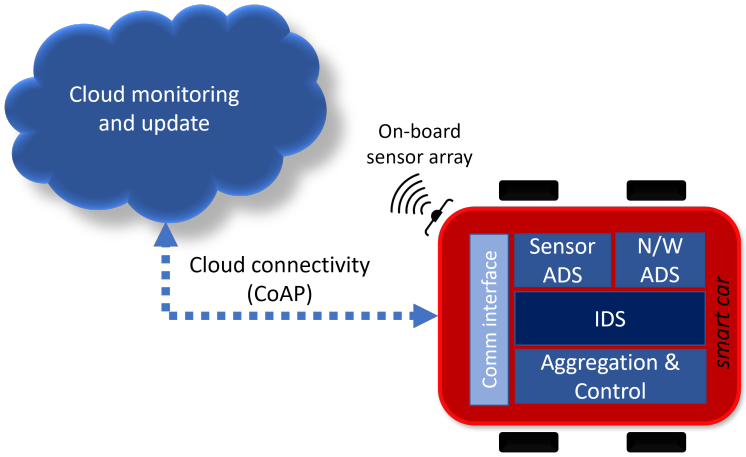

Fig. 2. Typical edge-to-cloud architecture.

3.2 Security Threats and Attack Surface

The attack surface of an edge device varies from case to case, depending on its connectivity features (WiFi, Bluetooth, etc), its hardware and software architectures (OS, bare metal, microcontroller or pure hardware) but in most cases, we can identify the following threats:

– Fault Injection Attacks (FIA). This class of attack consists in actively stressing a system in order to compromise its security. In short, when perturbing a security

system, an attacker can induce faults during a computation or generate bit-flips in memory cells. Those effects can then be exploited for sensitive variable recovery, for example with differential fault analysis (DFA) [8] or to skip specific instructions in order to bypass a security mechanism. There are several physical channels that can be used to generate the perturbation: power glitching, clock glitching, by temperature, electromagnetic injection, laser injection, etc, as well as software or hybrid methods [23].
Fault tolerant systems can be designed at the cost of performance: those systems use redundancy as countermeasure for fault injection attacks, in various ways: statically (e.g. second order statements, step counters), or dynamically (e.g. checksums, control-flow graph redundancy). Active defense against fault injection consists of analysing sensors values to detect attacks at runtime: this approach, enhanced with machine learning, is one of the focus of the IDS presented in this paper.

– Connectivity related cyberattacks: these attacks target communication interfaces of the devices. Multiple attacks can be realized, with various objectives and few examples are detailed here.
Denial of Service (DoS) attacks aim at flooding a service with traffic in order to prevent the device to operate correctly, for example by occupying all the available bandwidth, consuming all the device resources, making the system crash or preventing legitimate traffic to reach its destination.
Address Resolution Protocol (ARP) spoofing is a different type of attack where the attacker aims at impersonating a valid host device, causing the target device to send any traffic meant for the true host to the attacker instead. The attacker can then listen to the packets, discard them, or falsify them before sending them to the true host, achieving a Man-in-the-Middle (MiTM) position. ARP is a protocol used in Ethernet and WiFi to resolve a MAC address given an IP address. Devices can broadcast ARP requests to a network when they need the MAC address associated to a certain IP (in this case, the IP of the host). Anyone connected to the network can reply with an ARP response. Since ARP does not support any authentication mechanism, an attacker can send fake ARP responses containing its own MAC address, causing the target to send packets to the attacker instead of the host. However, during such an attack, the attacker generates unusual activity on the network which can be detected by an intrusion detection system.
Lastly, **port scanning** consists in scanning each port in a network in order to discover which ports are open and whether they give access to vulnerable applications. While actually not an attack, this malicious behaviour can be detected by an intrusion detection system.
The IDS framework presented in this paper is designed to handle connectivity related cyberattacks with a focus on the TCP/IP network interface.

– Side Channel Attacks (SCA). Side Channel Attacks are a type of passive attacks where the attacker "listens" to a system during a sensitive computation, through a physical channel like the power consumption or EM emanations, in order to discover the sensitive information being processed. This often implies using physical equipment to generate measurements and employing statistical tools to process those traces and extract the secret values. Some side-channel attacks can be conducted entirely by software, with no physical access to the device [1].
Side-channel attacks, because they passive, are not in the scope of the detection framework presented in this paper. However, runtime detection of cache-based side channel attacks is an area of research as the malicious processes performing those attacks have observable side effects on hardware performance counters, which can

be characterized as abnormal behavior. In future works, it could be considered to extend the scope of the proposed IDS to specific types side channel attacks.
– Application level attacks. On rich edge devices containing an operating system and applications, attacks can target directly vulnerabilities present in software applications, such as stack buffer overflows. This part of the attack surface is not covered by this paper. In many cases, those attacks can be prevented by source code analysis or dynamic testing.

3.3 On-Board Intrusion Detection

The main challenge on the edge is to aggregate all the sensor information from various channels and detect abnormalities or falsified perturbation and detect a difference (glitch) using ML, pertaining to the whole system. In this work we try to classify a normal scenario (un-perturbed case) and an anomalous scenario while the smart car is in motion. In order to ensure good training, the data is significant. Therefore, separate sessions were run to collect different types of data. As mentioned earlier, two anomaly detectors are embedded within the edge viz. sensor anomaly detector and network anomaly detector. The sensors data recorded is from a variety of sensors including:

1. **External**: Ultrasonic ranger, Camera, LDR sensor, IR sensors (for lines-tracing in the test area), Gyroscope, Accelerometer, Magnetometer, Barometer, Temperature and Humidity Sensor;
2. **Internal**: CPU temperature, clock, voltage, memory split, throttle status.

Apart from the sensors, the Network packet data is also monitored for anomalous activity. In order to collect precise data without any phase difference, an additional on-board system called the data aggregation unit is installed to collect inputs from all sensors and stack them together balancing the phase. Once the data is collected, the training for the sensor and network anomaly detectors is carried out. The training process is completed as shown in the figure 3.

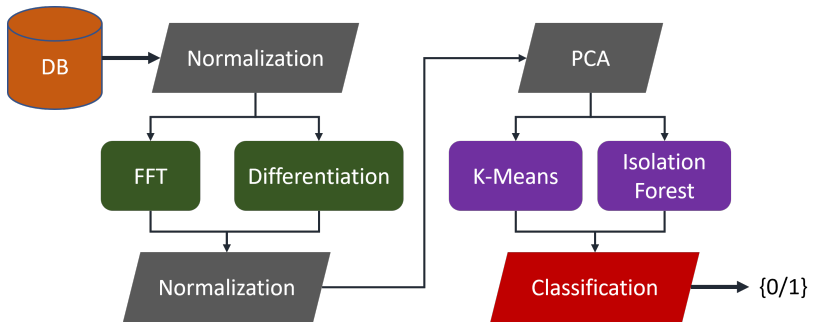

Fig. 3. Anomaly detection ML core training and model generation flow diagram.

ATTACK IMPLEMENTED	Non-contact to ground (accidental lift-off)	Collision	Humidity	Ultrasonic sensors spoofing	TCP DoS	Nmap	ARP poisoning
IDS DETECTION	Very good	Very good	Very good	Not at all	Very good	Very good	Poor
DETAILED RESULT (PRECISION, RECALL)	Ø	Ø	Ø	Ø	(71%, 83%)	(52%, 92%)	Ø
SENSORS INVOLVED (A PRIORI)	pressure sensor, thermometer, humidity sensor, photoresistors, accelerometer, gyroscope, magnetometer, ultrasonic sensors	pressure, temperature, humidity, photoresistors, accelerometers, gyroscopes, magnetometers, ultrasonic sensors	humidity sensor	ultrasonic sensors	network	network	network

Fig. 4. Results table.

Results The results obtained from testing the robot smart car with different attacks involving both sensors as well as network, are highlighted below in the figure 4.

The on-board IDS could easily detect both the sensor anomalous activity as well as the network anomalous packet activity. Thus, securing the Proof-of-Concept (PoC) of the designated motivation of having a generic framework of Intrusion Detection System for IoT systems (in our case a **V2X** scenario) by running smart filtration of sensor and network activity (surfaces prone to threats from different adversarial actors) including both human-error (accidents, improper operation, etc.) as well as induced faults, to detect and classify them against normal operation. Additionally, it connects to the cloud for sending both real-time (with only latency due to connectivity) anomaly detection status as well as collected sensor data which is used to re-train existing anomaly detection ML models or new models on the cloud and push new ML core models to the edge. The capabilities of the cloud services available for our use-case are shown in figure 5.

```
* The user's guide:

* First, enter the IP address of the server, e.g: 192.168.43.245

* Second, choose an operation. There are:

        - 1: To post the model parameters for sensors AD
        - 2: To post the model parameters for network AD
        - 3: To activate or disactivate the model
        - 4: To get the output of the ML model on the edge device
        - 5: To get the recorded sensors data
        - 6: To get the recorded network data
================================================================

- Enter the IP address of the server:
```

Fig. 5. Cloud remote monitoring and management console.

3.4 Device Fleet Monitoring with AI

The architecture of the embedded IDS presented above is designed for a deployment on single devices, meaning that the IDS is deployed on an embedded system and analyses the data of that system only. In the context of IoT, the challenge is to monitor and to ensure security to large fleets of heterogeneous IoT devices, equipped with different sensors or different interfaces. In this case, it is of course possible to deploy the IDS on compatible devices, each one endowed with an IDS running at the edge and reporting to the central cloud server. This is necessary to provide real-time detection capabilities, but the constrained resources on the edge limit the usage of powerful AI-based detection model. However, on the server side, alerts and data from the whole fleet can be aggregated and high computation resources are available, enabling a lot of new analysis possibilities: from intrusion, anomaly or failure detection, to smart visualization and business intelligence.

Cloud IDS. Since the edge computation resources and storage are limited, it is not always possible to use cost intensive ML models at edge level. For instance, large deep neural networks sometimes require several gigabytes of memory, or rely on GPU to run in reasonable computation time. On the edge, lightweight models are sometimes preferred. However, it is possible to deploy powerful and expensive models on the server side, with the goal of verifying and confirming alerts generated on the edge side. In other words, when an alarm is generated on the edge, it is notified on the server along with some metadata and on the server side, it is analyzed a second time by the "twin" detection model. With this architecture, one can provide real-time, lightweight detection systems at the edge while fully exploiting the capabilities of AI based detection on the server side, while minimizing the amount of transmitted data.

Fleet anomaly detection. At server level, it also becomes possible to aggregate the alerts and other information from multiple devices to improve the security analysis. When analysing the relations of similar devices between each other, if we assume a nominal operation for the fleet of devices, we can detect devices behaviors diverging from the general tendency [15]. So doing so, we can build an anomaly detection system at fleet level. Depending on the monitored data used as input of this system, it is possible to vary the scope of detected anomalies, from attacks, missuses, environment induced variations, or failures.

Visualization and business intelligence. After aggregation and detection, the next use-case of fleet monitoring is smart visualization. Smart visualization should allow the user to view the fleet in a condensed way providing insight regarding what is happening and what is going to happen, in order to take action if necessary. When dealing with IoT devices, the data is heterogeneous and difficult to interpret: the edge side provides notifications from the IDS and from other components, logs and and in some cases sensor values. So there is a need to aggregate this information in a form meaningful for an end user. For this purpose, AI-based dimensionality reduction have shown to be efficient [6], with algorithms like Principal Component Analysis, LLE, t-SNE and its variants, etc. Those methods reduce data in high dimension into 2-D or 3-D dimensional maps, while preserving similarities and dissimilarities of the original inputs. Dimensionality reduction can directly bring out devices with different distributions than others, giving hints at possible failures, or providing valuable insight regarding the devices behaviour or their life-cycle (a device may for example degrade

at a different pace depending on its geographical location). For the same purpose, clustering methods can be used.

3.5 Case-Studies: Automotive and Healthcare

In this sub-section we provide two similar state-of-the-art research studies that suggest similar solutions. However, with our advanced edge capabilities to monitor multiple different anomalies through sensor aggregation as well as monitoring the network for threats, we provide additional value to the existing IoT solutions or propositions.

1. **Healthcare:** The authors in [20] propose a cloud-based intelligent healthcare monitoring system that focuses on providing a smart solution to locate human organs to aid in the transplant surgical processes in the Hospitals. It is a classic approach of using IoT infrastructure in delivering life saving solutions in healthcare.
2. **Automotive:** Similarly to the approach in healthcare, as proposed in this work, the automotive industry greatly gains from the edge-to-cloud approach in the V2X infrastructure. In the paper [16], the authors, from Denso Corporation and Nanzan University in Japan, share their insights and experiences about the growing influence of cloud-based solutions in the Automotive sector. They claim that the Automotive software is evolving to become Automotive Cloud Service System (ACSS) and will continue to do so in the coming days.

4 Discussion: Other Security Services for edge-to-Cloud

Edge devices have a life cycle like human beings. The life of an edge device starts at design, then semiconductor and Original Equipment Manufacturer (OEM) level, to pass through manufacture that will oversee feeding the device with its identity, software program, applications, and services. When the edge device is ready, it is shipped by the system maker to the market by distribution supply chains. Hence, the edge device will be able to start its mission in the field. From security viewpoint, several security services shall be considered in edge-to-cloud context. We mainly mention assets provisioning, secure firmware update over the air (SFUOTA) and device identity. Those can be seen as micro-services fully managed by a remote side that is the server. An illustration can be depicted in figure 6.

4.1 Assets Provisioning

An asset is any sensitive data that is used to derive or manipulate secrets. Those assets could be:

- Chip/device private asymmetric key along with signed certificate for the public key.
- Chip/device symmetric master key.
- Device unique identifier.
- Chip maker/OEM public key certificate.

We can distinguish two provisioning phases in the life cycle of an edge device: before shipping (i.e. at manufacture stage) and after shipping (i.e. in the field stage). Before shipping, the provisioning is generally made locally in a safe zone at the manufacture

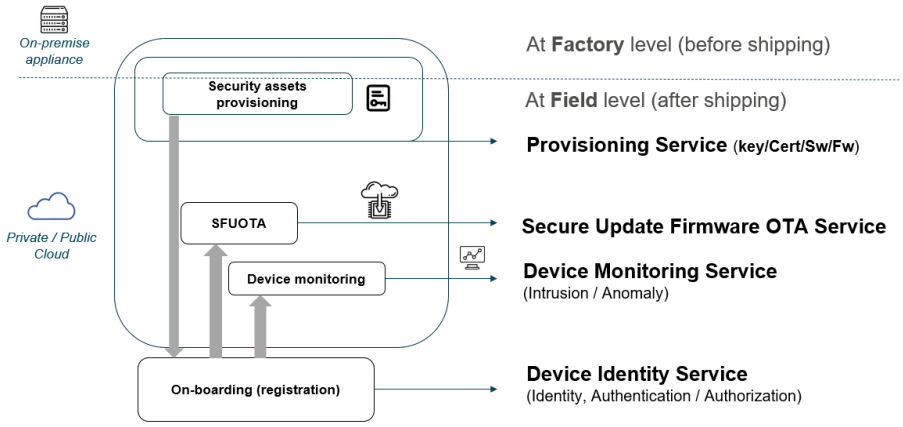

Fig. 6. Edge-to-cloud security services.

and not remotely. This is to reduce the risk of tampering with the initial assets. Ideally, an HSM, Hardware Security Module local server, is needed to ensure a maximal security by considering a certification authority (CA) that will be the unique guardian of signing certificate, as requested from OEMs generally, and storing secret keys. Then those assets are injected to the devices through a physical access such as JTAG, on a specific machine called the tester. One other approach to be used for sake of convenience, is to map the HSM server with a remote central server to allow remote provisioning to several manufacturer distant sites. The server itself may hold an HSM server as CA. Such HSM would serve the same purposes as the local HSM. In practice, such scheme could be applied when a central site as head quarter oversees provisioning several manufacturers that he may own or not. After shipping, the edge device may need to update or revoke its assets. In that case, in-field provisioning is obviously needed. Here, the remote server shall be able to provision the edge device provided that the latter is reachable from the network. Technically, a secure channel is initiated and established by the user after properly verifying the mutual identities of both parties: the server and the edge device. Therefore, the server could generate new assets and send them to the edge device. Now, the edge device will make the Host processor handle the received assets to safely store them and send them to a secure hardware layer as TPM or SE as described previously. Besides, the remote server shall be able to revoke those assets and suspend the device's activity.

4.2 Secure Firmware Update

The edge device runs a software which is a piece of code that needs to be provisioned before shipping and then updated, similarly to the key assets. More precisely, the software, called firmware, comes packed as a binary image stored in some persistent memory like the Flash. When the device starts, it boots on an embedded code from the static memory (ROM) that allows loading the firmware. The firmware itself is composed of the system code with initial applications and services needed by the kernel image or the full operation system to work properly. Before shipping, the OEM basically needs

to provision the device with the firmware. That could be performed either locally by interacting directly and simultaneously with many devices at factory/OEM level; or remotely based on a management server. Such server is necessary when the edge device is already shipped and in the wild. Such mechanism would make the life of chip makers and OEMs much easier. As a matter of fact, for automotive, the conventional situation today is to drop your car off at the repair shop to get it updated with new software version. Updating the firmware remotely is a sensitive task. For this purpose, several standardization bodies are about to push upward a unified solution. We mention for instance the IETF SUIT framework [18].

4.3 Device Identity

Before the heavy presence of connected objects in our landscape, security consisted in protecting users and their access to shared resources by ensuring their identities based on security standards. Nowadays, it has become similarly important to ensure the protection of data generated by these connected objects.

Authentication The principle of authenticating objects is the same as authenticating users, but the technologies to be implemented are quite different. Obviously, both cases share the same objectives. Authentication is the process of verifying the identity of a user or an object by comparing the credentials provided as an input with those stored in a database. These data are called authentication factors, which form the basis of authentication protocols and methods available in the cybersecurity world. We can categorize these said methods into four types:

- Hardware: includes any physical device that stores or generates a secret key on a real time basis.
- **Memorial**: such as passwords.
- **Corporeal**: which uses the human characteristics that only the rightful user has, the example of facial recognition and biometric authentication. As far as an IoT is concerned, the equivalent of a biometric is a "PUF" (Physically Unclonable Function) [26], whose security is standardized as per ISO/IEC 20897.
- **Reactional**: includes everything that is unique, that only the user can produce, like a signature or a gesture.

Based on these methods we can distinguish between three types of authentications:

- **Mono-Factor Authentication**: which consists of using a single (factor/method) to validate both the user and the object identity.
- **Multi-Factor Authentication**: which consists of using more than one method to validate both the user and the object identity.
- **Single-Sign On (SSO) authentication**: it is a mechanism that allows the user and the object to access one or more resources at the same time without having to go through the authentication service each time.

Authorization Authorization is the mechanism of determining whether the authenticated user can gain access to resources or perform specific actions. This function is called RBAC (Role Based Access Control), which is a security concept for managing access rights in a computerized system. Access rights are not managed by an administrator but delegated to an IAM (Identity Access Management) solution also called

User-IAM that is based on authentication and authorization. The aims of an User-IAM can be applied by analogy to the Device-IAM. The difference lies in the tools and the implementation of the concept. As for IAM-Device, unique identifiers such as IMEI serial numbers, UUIDs, MAC addresses, etc., allow to establish a first level of Mono-Factor Authentication of devices. Some devices reinforce their security by storing these identifiers in secure elements in the hardware (such as in Hardware Security Modules or HSMs, Trusted Execution Environments or TEEs, Secure Elements, etc.). However, this mono-factor authentication aspect can be improved. Like User-IAM, this mechanism can be strengthened by adding a second layer of authentication through PUF (Physically Unclonable Function). PUF is a technology that extracts a unique identifier from the intrinsic properties of each device. The output of a PUF is unique to a device and reproducible, and therefore constitutes an identifier.

Practical Use-Case In general practice, a server-side microservice implements a standard connectivity protocol, such as Lightweight M2M (LwM2M) or FIDO device on-boarding [3]. It allows to connect the device at its first connection to authenticate and authorize it to access the server resources. More precisely, the process is realized in two phases:

- On-boarding: First, a primary server-side microservice will ask the Device to communicate its identifier. This identifier will be verified by another microservice. In a second step, the primary microservice will request the PUF image which will be compared to its exact value in a secure database in the cloud.
- Authorization: Once, and only if, the authentication is verified, the primary microservice will assign a token to the device to allow it to access the server's resources.

5 Conclusion

We propose a novel approach for anomaly and intrusion detection, based on Artificial Intelligence in the context of edge-to-cloud security monitoring. This framework is motivated by the need to provide security services, like monitoring, device identity management or secure firmware update, to fleets of IoT devices in various applications fields: automotive, healthcare, smart-homes, etc. In those ecosystems, millions of edge devices need to embed real-time security systems to prevent attacks and intrusions, all reporting to a central server. We propose to enhance such systems with machine learning-based anomaly detection methods in order to improve the detection scope and capabilities and to make an overall better usage of the complex and heterogeneous data processed at the edge. We introduce a customizable edge IDS, monitoring network interfaces and sensors to detect multiple types of threats including but not limited to fault injections, and network cyberattacks like DoS or ARP spoofing attacks. The scope of our framework extends to advanced analytics: artificial intelligence can be used at its best on the cloud server side for fleet monitoring by aggregating and correlating data from millions of devices to detect anomalies, failures, to provide smart visualizations and eventually gain valuable insight for business intelligence.

References

1. Acıiçmez, O., Schindler, W., Koç, Ç.K.: Cache based remote timing attack on the aes. In: Cryptographers' track at the RSA conference. pp. 271–286. Springer (2007)

2. Alliance, C.S.: Matter home automation connectivity standard. https://csa-iot.org/all-solutions/matter/ (2022)
3. Alliance, F.: Fido device onboard: A specification for automated, secure iot provisioning technology. April-2021 Available online https://fidoalliance.org/intro-to-fido-device-onboard (2021)
4. Alliance, O.M.: Lightweight machine to machine technical specification. Aug-2020 Available online https://www.openmobilealliance.org/release/LightweightM2M/ Website https://omaspecworks.org/what-is-oma-specworks/iot/lightweight-m2m-lwm2m/ (2020)
5. Barenghi, A., Breveglieri, L., Koren, I., Naccache, D.: Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. Proc. IEEE **100**(11), 3056–3076 (2012), http://dblp.uni-trier.de/db/journals/pieee/pieee100.html#BarenghiBKN12
6. Bergmeir, P.: Enhanced machine learning and data mining methods for Analysing large hybrid electric vehicle fleets based on load spectrum data. Springer (2018)
7. Criteria, C.: Common Criteria. https://www.commoncriteriaportal.org/index.cfm, https://www.commoncriteriaportal.org/index.cfm
8. Dusart, P., Letourneux, G., Vivolo, O.: Differential fault analysis on aes. In: International Conference on Applied Cryptography and Network Security. pp. 293–306. Springer (2003)
9. ENISA: EUCS – Cloud Services Scheme. https://www.enisa.europa.eu/publications/eucs-cloud-service-scheme/
10. Eurosmart: Eurosmart IoT Certification Scheme. https://www.eurosmart.com/eurosmart-iot-certification-scheme/
11. Facon, A., Guilley, S., Ngo, X., Nguyen, R., Perianin, T., Shrivastwa, R., Secure-IC, S., Rennes, F.: High Precision EMFI Detector using Machine Learning and Sensor Fusion
12. Global Platform: Security Evaluation Standard for IoT Platforms (SESIP). Version 1.0. Document Reference: GP_FST_070 (March 2020)
13. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
14. Hardt, D.: The OAuth 2.0 Authorization Framework. RFC 6749 (Oct 2012). https://doi.org/10.17487/RFC6749, https://www.rfc-editor.org/info/rfc6749
15. Hendrickx, K., Meert, W., Mollet, Y., Gyselinck, J., Cornelis, B., Gryllias, K., Davis, J.: A general anomaly detection framework for fleet-based condition monitoring of machines. Mechanical Systems and Signal Processing **139**, 106585 (2020)
16. Iwai, A., Aoyama, M.: Automotive cloud service systems based on service-oriented architecture and its evaluation. In: 2011 IEEE 4th International Conference on Cloud Computing. pp. 638–645. IEEE (2011)
17. Krasovsky, A., Maro, E.: Actual and historical state of side channel attacks theory. pp. 1–7 (09 2019). https://doi.org/10.1145/3357613.3357627
18. Moran, B., Tschofenig, H., Brown, D., Meriac, M.: A Firmware Update Architecture for Internet of Things. Internet-Draft draft-ietf-suit-architecture-12, Internet Engineering Task Force, https://datatracker.ietf.org/doc/draft-ietf-suit-architecture/12/, work in Progress
19. NIST-USA: Federal information processing standard 140-3. https://csrc.nist.gov/publications/detail/fips/140/3/final
20. Parane, K.A., Patil, N.C., Poojara, S.R., Kamble, T.S.: Cloud based intelligent healthcare monitoring system. In: 2014 international conference on issues and challenges in intelligent computing techniques (ICICT). pp. 697–701. IEEE (2014)

21. Perianin, T., Carré, S., Dyseryn, V., Facon, A., Guilley, S.: End-to-end automated cache-timing attack driven by machine learning. Journal of Cryptographic Engineering **11**(2), 135–146 (2021)
22. PSA: PSA Certified. https://www.psacertified.org/getting-certified/
23. Qiu, P., Wang, D., Lyu, Y., Qu, G.: Voltjockey: Breaking sgx by software-controlled voltage-induced hardware faults. In: 2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST). pp. 1–6 (2019). https://doi.org/10.1109/AsianHOST47458.2019.9006701
24. Sabt, M., Achemlal, M., Bouabdallah, A.: Trusted execution environment: What it is, and what it is not. In: 2015 IEEE Trustcom/BigDataSE/ISPA. vol. 1, pp. 57–64 (2015). https://doi.org/10.1109/Trustcom.2015.357
25. Shahid, M.R., Blanc, G., Zhang, Z., Debar, H.: Anomalous communications detection in iot networks using sparse autoencoders. In: 2019 IEEE 18th International Symposium on Network Computing and Applications (NCA). pp. 1–5. IEEE (2019)
26. Shamsoshoara, A., Korenda, A., Afghah, F., Zeadally, S.: A survey on physical unclonable function (puf)-based security solutions for internet of things. December-2020 Available online https://www.sciencedirect.com/science/article/pii/S1389128620312275 (2020)
27. Shelby, Z., Hartke, K., Bormann, C.: The constrained application protocol (coap)(rfc 7252). Jun-2014 Available online. http://www.rfc-editor.org/info/rfc7252 (2014)
28. Standard, O.: Mqtt version 5.0. Jun-2019 Available online. https://mqtt.org/mqtt-specification/ (2019)
29. The OpenSSL Project: OpenSSL: The open source toolkit for SSL/TLS (April 2003), www.openssl.org