Journal of Universal Computer Science, vol. 27, no. 7 (2021), 774-792
submitted: 13/5/2020, accepted: 26/5/2021, appeared: 28/7/2021 CC BY-ND 4.0

An Approach for Testing False Data Injection Attack on Data Dependent Industrial Devices

Mathieu Briland
(Flowbird Group & DISC/FEMTO-ST, University of Bourgogne Franche-Comté, France
https://orcid.org/0000-0003-1335-8450, mathieu.briland@{flowbird.group, univ-fcomte.fr})

Fabrice Bouquet
(DISC/FEMTO-ST, University of Bourgogne Franche-Comté, France
https://orcid.org/0000-0001-9181-1172, fabrice.bouquet@univ-fcomte.fr)

Abstract: False data injection is an attack in which an attacker injects fabricated data into a system with the objective to change the behaviour and the decision-making of the system. Many industrial data-based devices are vulnerable to such attacks, this work presents an approach for testing False Data Injection Attack. This approach uses a Domain-Specific Language to generate altered data with two objectives, to provide sophisticated attacks scenarios to increase the resilience of vulnerable systems against False Data Injection Attack and to train detection tools.

Keywords: False data injection, attacks, FDIA, internet of things, test, security
Categories: C.2.0, D.2.5, H.2.3
DOI: 10.3897/jucs.70326

1 Introduction

For any computer system, data security is essential, in terms of integrity and confidentiality. Data enables many different services to be offered to users, whether it is a simple consultation service, such as a website, or a vital service, such as life monitors in hospitals. The data used in these types of services are critical to their proper functioning. In the first case the risk may be, for example, an economic loss, in the second case a loss of human life. This criticality makes data interesting as an interface to execute cyber attacks. Many types of attacks specifically seek to target data to compromise systems. However, in this work we focus on a particular type of attack, called False Data Injection Attack (FDIA).

This work is carried out within the Flowbird company (http://www.flowbird.group). Flowbird is the world leader in on-street parking solutions. The parking meters manufactured by the company can be found in thousands of cities around the world and process thousands of data. These data are used to provide a variety of services such as parking solutions, notably offered through physical touch devices, or the monitoring of environmental conditions through the presence of numerous sensors in parking meters, such as noise or pollution sensors. An attack specifically targeting the data of the services provided by the company would then be critical, both in economic and reputational terms. Indeed, the services could then be unavailable or provide erroneous information.

The company's devices are connected to the internet via mobile networks and are linked to a data centre that sends and receives data from parking meters. The multiple services on the devices or in the data centre then use these data to provide the various services to the end users. The general architecture of the systems is shown in [Fig.1].

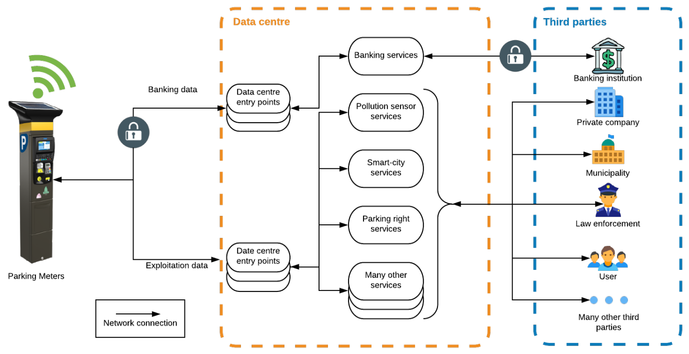

Figure 1: Simplified architecture of the parking management system

The system we study in this paper, the parking meters and their multiple services, share common characteristics with Internet of Things (IoT) devices, such as their architecture, physical vulnerability or services provided. Our systems are therefore susceptible to being attacked with the same attack as any IoT devices. In the following, we therefore consider the devices we are working on as part of the IoT, particularly in the state-of-the-art context. Nevertheless, the work is focused on the devices present within the company.

In this work, we focus on developing an approach and tools to protect our systems against FDIA. There are several levels to consider for the security aspect: data and their characteristics, data manipulation, and attack simulations to create credible scenarios. We identify the following research questions:

– **RQ1:** What are the data gathered by our systems?

  To answer this question, we are exploring our systems and looking for the types of sensors that exist, in order to deduce the nature of the data they measure and to characterise the information they carry. For example, formats, types or uses made of them.

– **RQ2:** How to effectively address the FDIA challenge on our systems?

  To answer this question, we present a typical architecture of our systems in order to understand their vulnerabilities to FDIA. We also present our approach through a multi-step workflow to address this challenge in our systems.

– **RQ3:** How to effectively describe FDIA scenarios with a Domain-Specific Language (DSL)?

  To answer this, we briefly present the DSL grammar used to model FDIA scenarios. Next, we explore the DSL through sample scenarios to demonstrate the modelling capabilities offered by the DSL, as well as its scalability.

This work is composed of 6 sections. In section 2, we present our context and the related works. In section 3, we explore the different data managed by our systems and characterise them. Section 4 introduces our approach and briefly outlines the DSL grammar used to model and produce FDIA. Section 5 explores the expressiveness of our approach through the presentation of multiple scenarios. Section 6 concludes the paper with a discussion and possible future work.

2 Context and Related Works

In this section, we present the devices we use and the link we establish with the IoT. This link allows us to explore existing work on data security issues and to justify why we are addressing the FDIA one. We then explore the FDIA in the literature and related works addressing them.

2.1 Security Aspects of Devices

Our Flowbird devices can be described as a five-layer architecture as IoT devices [Aazam et al., 2014]. Firstly, the perception layer collects data from the sensors, these data are then transmitted to the other layers via the network layer. It is connected to the internet and uses many different protocols. Then the middleware layer carries out the technical processing of the data. The application layer then uses this data to display it to the end user. Finally, the business layer is used to control the whole chain and uses the data to explore new business models.

The analogy between the general IoT architecture and our system architecture is shown in [Fig.2]. The description of the layers shows that data is at the centre of our (and IoT) architecture.

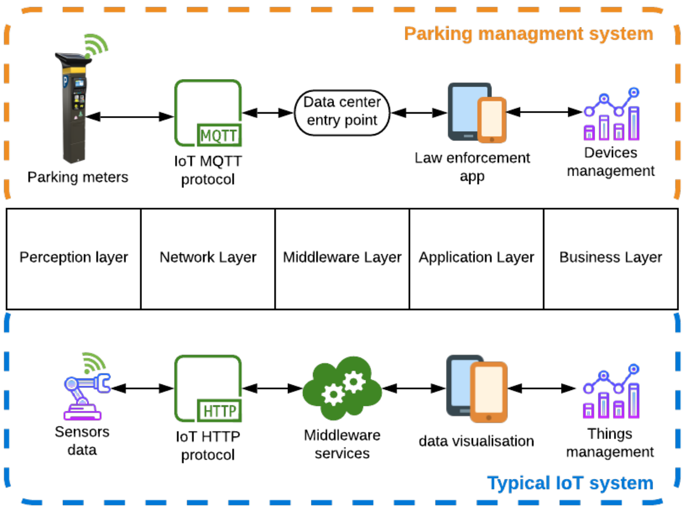

Figure 2: Analogy between IoT architecture with parking management system

Data integrity is a key element for the services provided by the company. In the field of IoT security, we seek to identify existing attacks in relation to the architectural layers.

Like the IoT, our devices are subject to many constraints (connectivity, autonomy, computing power, etc.). They also pose many security challenges, such as data privacy, data access permission, networks authentication, secure storage, secure data processing [Zhao and Ge, 2013]. Each layer of the architecture faces its own security concerns. [Tab. 1] summarises the attacks explored, classified in their respective layers. In the

Architecture layers	Security issues
Perception layer	Hijacked/fake node, False Data Injection Attack, Denial of Service (DoS) attack, Authentication attack, Cloning, Eavesdropping, Spoofing, Jamming, Inference attack, Physical attack
Network layer	State-of-the-art security attack. e.g. Sybil attack, Sinkhole attack, Sleep deprivation attack, DoS attack, False Data Injection Attack, Man in the middle attack, Traffic analysis, Routing attack, Selective forwarding
Middleware layer	Unauthorized access, DoS attack, Malicious insider, False Data Injection Attack
Application layer	False Data Injection Attack, DoS attack, Spear-Phishing attack, Sniffing Attack
Business layer	Same as Application layer

Table 1: Architecture layers and their security issues [Farooq et al., 2015, Zhao and Ge, 2013, Padmavathi and Shanmugapriya, 2009, Sikder et al., 2018]

table, the presence of FDIA in each layer shows its importance. Data integrity in our systems is a critical challenge, and FDIA is a major threat to this integrity. We have just seen the link between our systems and IoT, we can suppose that they share the same vulnerability to FDIA, which answers the first part of the question **RQ2**. In the next section, we look at FDIA in the literature.

2.2 False Data Injection Attack

First introduced by [Sencun Zhu et al., 2004] in the field of sensor networks and later in the field of Wireless Sensor Networks (WSN), especially smart grids [Liu et al., 2009], FDIA are attacks in which an attacker seeks to change a system's behaviour by modifying the data used for its services. For example, in the case of smart grids, the attacker seeks to inject errors into sensors state variables, which then leads systems to wrong power grid state estimation. In this particular area, this can lead to total power blackouts such as the one in Ukraine in 2015, where a FDIA was launched against the country's power grid. Three energy distribution companies were compromised and their services were disrupted, throwing part of Ukraine into a blackout [Liang et al., 2017].

More generally, FDIA is an intentional attack (malicious) or unintentional where a sensor network is compromised, usually by taking advantage of one or multiple sensors. The aim is to generate and send, data or events that don't represent the reality of the network environments. This type of attack can lead to service disruption, energy and network waste, money loss, and even physical destruction [Albright et al., 2010].

In the case of an intentional FDIA, the goal of the attacker is to disrupt the services provided by the application that gathers and processes the data sent by the sensors, such as the Ukraine attack.

In the case of unintentional FDIA, there is no attempt to harm a specific system. We're more likely talking about a False Data Injection (FDI). Usually FDI happens in the perception layer, it's an anomaly in the environment of the sensor or an error inside the sensor itself. However this type of FDI also needs some protection to prevent damage and disruption of the system. An example of unintentional FDI can be a worm on a humidity sensor that stops watering a crop, or a lorry parked next to a pollution sensor that triggers a specific event.

The vast majority of the research focusing FDIA has been conducted in the smart-grid and WSN field. In particular within power-grid state estimation, mainly for FDIA filtering and detection [Mo et al., 2010, Manandhar et al., 2014, Lee et al., 2010].

One of the difficulties of the FDIA challenge is to be able to develop, train and verify this FDIA filtering and detection techniques using real data from systems in production that have been attacked. Usually, the attacked data are either protected for confidential purposes or has simply not been detected, therefore not flagged as compromised.

To develop their attack mitigation systems and validate them through experimentation, various authors used several methods to generate data. [Yang et al., 2017] uses a pseudorandom generator to emulate the data collection and also a pseudorandom generator to emulate FDIA behaviour. [Yi Huang et al., 2011] uses for their system in normal state (no attacked) a Bayesian model of the random state variables with a Gaussian distribution and for the malicious data they changed the distribution. The data used by [Chaojun et al., 2015] are based on the data from the New York independent system operator (NYISO) from 2012, and generate the state data following a procedure. The attacked data are numerical and they apply a modification of 90%, 95%, 100%, 105%, and 110% of the original numerical value.

The main flaw in the use of these methods is usually the loss of correlation with reality. The use of non-real base data and arbitrarily designed attacks, result in the loss of both system-specific and attacker-specific behaviour.

The closest work but in a specific domain is made by [Cretin et al., 2018], who developed a DSL-based testing framework to perform FDIA on air traffic control systems. They used real data from air traffic control and perform FDIA on them by using a DSL adapted to the specificity of the aircraft domain, then, they reinject the altered data in air traffic control systems.

As far as we know, there is no related work to assess the resilience of industrial systems attacked by FDIA. This also applies to IoT systems [Bostami et al., 2019]. In the following section, we therefore explore various works that do not necessarily belong to the FDIA domain, which could bring some avenues of research to our own. In particular in terms of test methods and data generation.

2.3 Data Generation for FDIA

Once the attack model is made, it must be used to generate the data of the attack. In this section we will therefore review the methods used in the literature to generate data, in particular relevant test data.

We therefore wish to obtain synthetic but realistic data. There are several reasons for using synthetic data rather than data from real systems. The first reason is the amount of data. A system for multiple reasons may not produce enough data to use authentic data. For example, a system that is still in development, or a system that uses confidential data. The second reason is data labelling. It is difficult to categorise authentic data. Are the recovered data healthy or are they altered? This brings us to the third reason, it is difficult to have data that are definitely considered corrupted data. Especially because they have not been detected as such, and if they are, they are often kept confidential. And the fourth reason, the use of generated data allows the attack to target very specific behaviours.

It is obvious that there are shortcomings when using synthetic data. In particular, the main drawback is the realism of the generated data, and especially for correlated data or business logic rules.

[Popić et al., 2019] reviewed data generator techniques and use cases. For the data modelling process, they present several methods such as [Hoag and Thompson, 2007] and [Rabl and Poess, 2011]. For their models, this type of data generator describes through an XML format the characteristics of the data. Then the data generation is made from the XML model. For example, [Anderson et al., 2014] present a framework for generating synthetic data for IoT. Their approach comes from the fact that it is difficult, in a big data context, to work on large amounts of data. Especially because IoT has completely heterogeneous data structures between the different IoT systems. Also, the sharing of such data brings data confidentiality issues in corporate contexts, and therefore cannot be released to the public. Their approach proposes to generate synthetic data from authentic data, keeping the structure and characteristics contained in the authentic data. To do so, they proceed in two main steps, data characterisation and data generation. On the data characterisation step, they extract two information from the data, the structure of the XML and the values present inside this structure. On the data generation step, they use the information harvested in the previous stage to reconstruct the data structure, and to populate it using random generator based on the distribution of the values.
The generation work here is very interesting. In particular the preservation of the data structure and the characterisation of the data values from real data to deduce statistics. Nevertheless, a temperature sensor next to a humidity sensor will have its data correlated with the previous ones but also with those of the humidity sensor. The approach proposed will probably deduce certain characteristics, such as an interval of temperature evolution, but the data will probably lose all correlation between them or the other sensor. This correlation between data is a critical aspect in our case. FDIA must be consistent with the data preceding or following it; otherwise it would be detected too easily.

We will also see later that our approach, although it may involve data generation, is largely based on data modification. In practice the generation of synthetic data from scratch is little used and the modification of existing data is much more used, because it allows us to keep the essence of the data and their connections.

The method we find the most interesting is the one of [Cretin et al., 2018], within the context of FDIA applied to the field of air traffic control. To assess the FDIA resilience of ADS-B, an air traffic control technology, they use a DSL to model FDIA scenarios. These scenarios are then applied through a framework to actual air traffic data to generate altered data. Although addressing the FDIA, their DSL is very oriented for the air traffic control domain. It includes language elements specific to the data present in aircraft recordings, such as altitude, ICAO aircraft model identification, or transponder code. The use of a DSL in this kind of case studies in highly specialised fields are interesting. This makes it possible for experts in the field to quickly get to grips with it by offering them great expressiveness. Moreover, to use a DSL it is not necessary to be familiar with general programming languages. In addition, the use of DSL is also found in synthetic data generation as in [Fremont et al., 2019]. The authors propose a probabilistic scenario description language for perception systems. They generate synthetic data from these scenarios describing car positioning scenes. This is in order to train and evaluate machine learning tools, especially on rare events or on their performance in particular conditions. DSL are very indicated to treat problems in very precise fields in an efficient way. They allow us to model and specify many aspects, including scenarios, so they are interesting for our case studies, where we want to model attack scenarios and then apply them to datasets.

3 Data Definition

Our Sensors type	Description	Sensors example	
		Our sensor	[White, 1987] Classes
Thermal sensor	Measure the thermal property, used for monitoring the sensors environment, or the sensor's device itself	Thermometer	Thermal (Temperature)
Proximity sensor	Measure the distance or detect the proximity of an object	Photo-sensor	Optical (Wave Amplitude)
Motion sensor	Measure the linear acceleration or rotation	Accelerometer	Mechanical (Acceleration)
Optical sensor	Measure optical property, lots of sensors operate in the infrared	QR code reader	Optical (Wave amplitude)
Chemical sensor	Measure the chemical environment of the sensors (gas, humidity, particle)	Particle sensor	Optical (Wave amplitude)
Acoustic sensor	Measure waves, as acoustic waves or ground waves	Microphone	Acoustic (Wave amplitude)
Position sensor	Measure a relative position	Compass	Magnetic field (Amplitude)

Table 2: Sensor classification in our system

To know what exactly we need to address for scenario generation and data falsification, we have to analyse the company ecosystem, mostly on the perception layer where the data are generated. The result presented in this section provides the answer to **RQ1**. We have worked on sensor classification, identification of data properties and data categorisation.

3.1 Sensor Classification

The identification of data property is based on the analysis of the entity that provides the data. In our devices the data are provided by the sensors. Sensors classification schemes have already been defined for electronic and engineering purposes [White, 1987, Hulanicki et al., 1991]. In the most comprehensive study they defined nine classes (acoustic, biological, chemical, electric, magnetic, mechanical, optical, radiation, thermal and others) that can encompass all sensors.

For this work, we have identified the sensors used in our systems and classified them under categories as shown in [Tab. 2]. The first column is our classification, the second a brief description, the third is the example of sensors found in our system, and the fourth the [White, 1987] classification of the sensor example. We have identified this classification because the behaviour of the data will change depending on the sensor type, and for data alteration we need to be as close to the real-life behaviour of the property as possible. Mainly to avoid being easily detected by possible intrusion detection tools.

In the following we present the seven classes, we have identified in our system.

Thermal sensors: Thermal sensors measure changes in thermal properties of a particular element, such as body temperature, ambient temperature or car engine temperature. In our use cases, this type of sensor is used to monitor the state of the environment around or inside parking devices.

Proximity sensors: Proximity sensors are used in a very wide variety of applications, the most common one is in smartphones to detect the presence of something near the screen (face, pocket) to avoid bad input. In our use case, this type of sensor is used to detect the presence of a person in front of parking devices.

Motion sensors: Motion sensors measure the position in space by means of linear acceleration and rotation of the monitored object. This is usually an accelerometer or gyro sensor. The most common use case is in the field of smartphones. In our case, this type of sensor is used to detect physical intrusion attempts on devices.

Optical sensors: Optical sensors deal with light by converting light properties in electrical signals. Many devices use this type of sensor for a wide range of applications. They can be found in a smartphone to adjust the screen luminosity, in a heartbeat sensor or in an infrared thermometer. In our use case, this type of sensor is used to read QRcodes.

Chemical sensors: Chemical sensors measure the chemical property, usually the concentration of a specific element. In our case, this type of sensor is used to measure the No2 concentration around devices.

Acoustic sensors: Acoustic sensors convert the wave amplitude of a sound. The most common example of an acoustic sensor is the microphone. In our use case, this type of sensor is used to measure the noise around devices.

Position sensors: The position sensors measure their position through space. The most common example of position sensors is the compass. In our use case, this type of sensor is used to detect the position of the devices.

We have also identified, by extending the search, two other categories that can accept sensors. But which are not present in our systems. Mechanical sensors, which measure a mechanical force (force, pressure, strain, torque), and level sensors, which measure a liquid or solid level (water, grain, powder).

Usually, the sensors are used for two main intentions: 1- to gather information to perform control and operate actuators, 2- to monitor elements, either for ensuring the correct functioning or to take a decision based on the monitoring. For FDIA, a malicious adversary carries out attacks through these two main objectives. For example, in the case of a thermal sensors an attacker can disturb the function of an asset by modifying the temperature returned by the sensor, or by generating an anomaly to trigger the maintenance of the part attacked.

A big challenge of FDIA is that sensors are generally not alone in their environment. It depends on the type of the systems. A system for monitoring the street environment (temperature, hydrometry, particles, pollution) of a city will need several sensors to get sufficient and reliable coverage (one device per street, about a hundred metres between the sensors). A smart car will need several sensors but close to each other to analyse the road (multiple sensors per side of the car, a few centimetres between the sensors). Due to

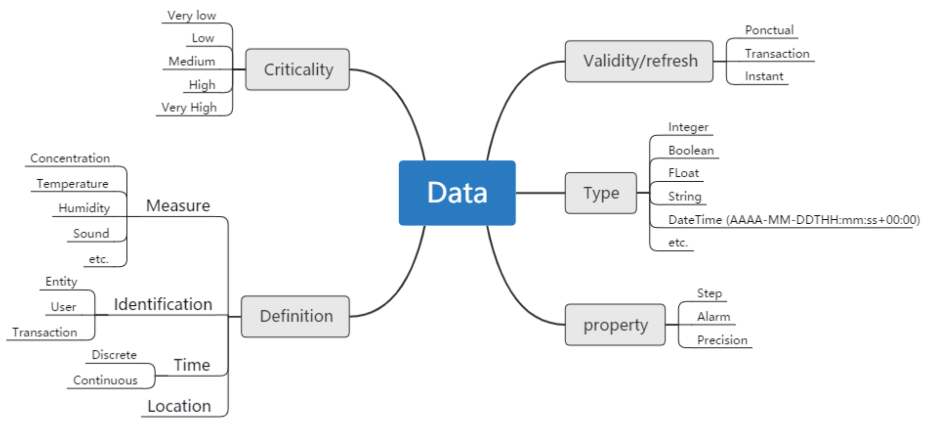

Figure 3: Categorisation of a data

this usual configuration of multiple devices, the spatial propagation of the measurements must be addressed. For example, in thermal sensors the temperature propagation must be considered for the attack to remain coherent. In some cases, if two sensors are close to each other, the data from the two sensors must be altered according to their distance and the power of the alteration made. It also depends on whether the attacker tries to attack a local property or a more global property.

3.2 Data Classification

After identifying the sensors' categorisation, we need to analyse what categorises the data returned by the sensors. We have analysed multiple data sets from our systems to find out which property defines them.

Through the research we define that a data is characterised through five properties: the definition, the criticality, the validity, the property and the type. This classification is summarised in [Fig. 3].

Definition: is the representation of the data, it can be a data of *identification*, *time*, *location* or a *measure*.

The data of *identification* is always present in our datasets, this data serve to identify, either the device itself, the transaction or the user of the device. For FDIA, identification data are important, their main purpose is to select the target that the attacker wants to attack.

The *time* data are also always present in our dataset. They are usually in a timestamp format. These data are used to situate other correlated data over time. For FDIA they are useful for creating scenarios within a specific time frame. Moreover time data needs a high degree of consistency to remain undetected.

Location data represents the location across space of the device. The location is not present in all dataset. However, for effective tampering and attack propagation through the system, it is important to have this data available, it can be added by the attacker. This data should be considered differently than location data that does not represent the position of the object concerned. Such data would be found in the next category.

The *measure* category contains all the data returned by the sensors described in the previous section. Generally, the decisions taken by the application layer are based on those data, therefore they are usually the main target of attackers.

Criticality: is a scale of the severity of the impact that could occur when specific data are attacked and altered by FDIA. This data property is useful to have a good categorisation of the probability of an attack on the data. Moreover this property helps the scenario maker to build the attack. This scale is defined through five levels, very low, low, medium, high and very high. This scale is typically used in the standard ISO/IEC 27005 for information security risk management.

Type: is the data type. Through our analysis we found that all data types that can be found in languages are used in our data (Integer, Float, String, Char, Bool, etc.). However the vast majority of the data rely on a JSON Data format for their data transaction. Therefore, most types used are String, Number and Boolean.

Validity: is the data validity through the different transactions. Usually the data is valid only during its proper transaction, but occasionally some data can be sent for information or configuration purpose and can be one-time data.

Property: represents a particular property of the data, we have identified specific properties such as the step, the precision or if the data is an alarm.

All these properties can evolve. They are based on the data we have analysed, keeping in mind to be the most exhaustive.

In this section we have answered the **RQ1**. We have identified the sensors used in our systems and defined their produced data with their properties. We have also identified the different characteristics of the data. This allows us to better understand the data and the attacks that may be perpetrated against it. This is used to define our approach as well as the DSL.

4 Approach and DSL

To address the FDIA challenge and to complete the answer to **RQ2** started in section 2.1, we have defined a complete approach and a DSL to model and execute the attacks. The workflow of our approach is shown in [Fig. 4].

Data acquisition is the first step of this workflow, it consists of two processes: either importing a dataset under a file format such as a CSV or JSON, or intercepting the devices data flow by performing a man in the middle or eavesdropping attack to perform a live attack.

Data configuration is made by the expert of the System Under Test (SUT). It consists of the property definition of the data present in the dataset or in the flow defined in the data acquisition step.

The conversion is the step where the input data are transformed to an internal format for a better processing of the data as well as to be able to process a wide heterogeneity of data and systems. This converted data are stocked in a database.

Designing scenario is made by the expert of the system under test. We provide a textual DSL for designing a scenario of FDIA. The expert uses this DSL, and selects the specific dataset he wants to alter.

Tampering generator is the phase where we read the scenario describing FDIA, and transform it into a memory representation.

Scenario execution is the process of applying the memory representation of the scenario written by the expert to the dataset. It results in a new dataset with tampered

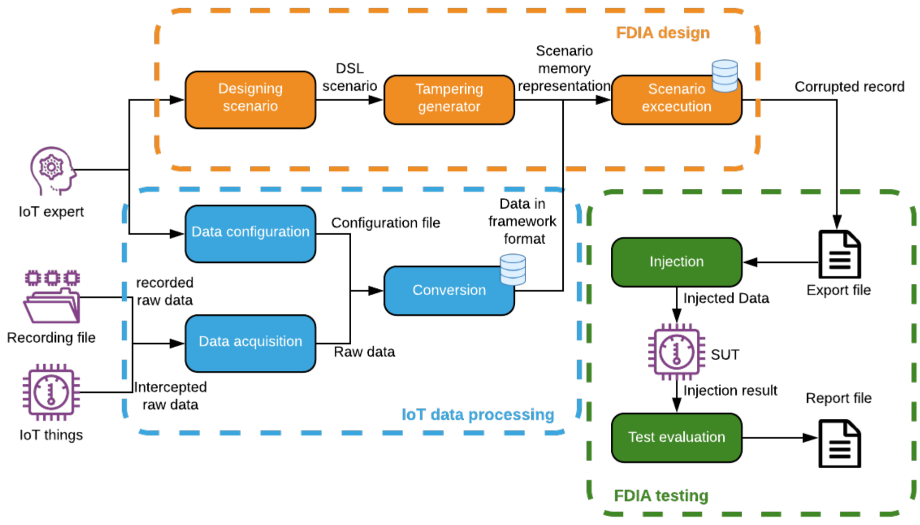

Figure 4: Workflow for FDIA testing - Data acquisition, designing and testing

data stored in a database. After this step the expert can choose to export this dataset, in order to teach the AI learning machine to detect false data injected in the dataset.

Injection is the process of converting the data back to their original form, and inject them in the SUT. It can be either an injection of all the data tampered, or an injection respecting the timing of the different message in the dataset.

The SUT represent the system where data altered by FDIA are injected.

Test evaluation aims to verify the impact of the injection in the SUT with a test oracle. This test evaluation should be exported to a report, for the purpose of either correcting the SUT or improving the FDIA written by the test expert.

Domain-Specific Languages (DSL) are languages tailored to a particular domain and application [Mernik et al., 2005]. They have more expressiveness than the General Programming Languages (GPL) such as C or Java. The main advantage of using a DSL is the gain in productivity when used by people experts in the domain covered by the DSL. It allows users not fluent in general computer languages to learn easily and perform complex action on a specific domain.

To address the challenge posed by FDIA, we have defined a DSL that aims to describe the full scenario of the attack. A FDIA scenario is a complete, step-by-step description of the actions taken by the attacker to carry out an attack. To define this DSL we reviewed literature best practices [Fowler, 2010, Czech et al., 2018] and used the methodology defined by [Mernik et al., 2005]. This 4-step methodology answers the questions, ”When and how to define a DSL”. For the purpose of brevity, we will not present the methodology in this work. Nevertheless, we will briefly present the DSL obtained at the end of the methodology in the following.

A scenario is made up of two parts, the scenario property and the scenario actions as shown in [Fig. 5] which describes the DSL grammar. The scenario property part is

```
⟨scenario⟩ ::= ⟨scenarioDeclaration⟩ ⟨executionList⟩

⟨scenarioDeclaration⟩ ::= Scenario ⟨string_literal⟩ ticker ⟨decimal_literal⟩ geolocation ‘(’
      ⟨real_literal⟩ ‘,’ ⟨real_literal⟩ ‘)’

⟨executionList⟩ ::= ⟨execution⟩ ‘;’ (⟨execution⟩ ‘;’)*

⟨execution⟩ ::= ((create things
      ⟨alterationCriteria⟩ ⟨timeframe⟩)
   | (alter things ⟨selectionCriteria⟩ ⟨alterationCriteria⟩ ⟨timeframe⟩)
   | (delete things ⟨selectionCriteria⟩ ⟨timeframe⟩)
   | (copy things ⟨selectionCriteria⟩ ⟨alterationCriteria⟩ ⟨timeframe⟩))

⟨selectionCriteria⟩ ::= where ⟨selectionCriterion⟩ (and ⟨selectionCriterion⟩)*

⟨selectionCriterion⟩ ::= ⟨id⟩ ‘=’ ⟨type⟩
   | ⟨id⟩ ‘>’ ⟨type⟩
   | ⟨id⟩ ‘<’ ⟨type⟩
   | ⟨id⟩ ‘!=’ ⟨type⟩
   | ⟨id⟩ isInsideCircle ‘(’ ⟨real_literal⟩ ‘,’ ⟨real_literal⟩ ‘,’ ⟨decimal_literal⟩ ‘)’
   | ⟨user_function⟩

⟨timeframe⟩ ::= from ⟨decimal_literal⟩ to ⟨decimal_literal⟩

⟨attenuationCriteria⟩ ::= with attenuation of ⟨real_literal⟩

⟨alterationCriteria⟩ ::= set ⟨alterationCriterion⟩ (and ⟨alterationCriterion⟩)*

⟨alterationCriterion⟩ ::= ⟨id⟩ ‘=’ ⟨type⟩
   | ⟨id⟩ ‘=’ ⟨evol⟩
   | ⟨id⟩ ‘+=’ ⟨real_literal⟩
   | ⟨id⟩ ‘*=’ ⟨real_literal⟩
   | ⟨id⟩ ‘+=’ ⟨evol⟩ ⟨attenuationCriteria⟩?
   | ⟨user_function⟩

⟨evol⟩ ::= ‘(’ ⟨decimal_literal⟩ ‘->’ ⟨decimal_literal⟩ ‘,’ ⟨decimal_literal⟩ ‘)’

⟨type⟩ ::= ⟨id⟩
   | ⟨string_literal⟩
   | ⟨decimal_literal⟩
   | ⟨real_literal⟩

⟨decimal_literal⟩ ::= ( ’0’-’9’ )+

⟨id⟩ ::= ( ’a’-’z’ ’A’-’Z’ ) ( ’_’ ’a’-’z’ ’A’-’Z’ ’0’-’9’ )*

⟨string_literal⟩ ::= ‘"’ (’_’ ’a’-’z’ ’A’-’Z’ ’0’-’9’)* ‘"’

⟨real_literal⟩ ::= ( ’0’-’9’ )+ ’.’ ( ’0’-’9’ )+
```

Figure 5: Grammar of the DSL

used to describe properties applied to the whole scenario. The scenario actions part is a description of the actions carried out by the scenario.

The scenario’s actions are composed of multiple part:

Primitive: They determine the action performed, and the form of the following elements of the rule. They are: create, alter, copy and delete.

Selection criteria: It is the selection of the specific records that the attack designer wants to alter. Usually it is an identifier of a device, but it can also be a specific condition such as a temperature value above a threshold. So it can also be seen as a trigger. Specific selection depending on the sensors targeted can occur such as the selection inside a circle defined by geographical coordinates and a radius.

Alteration criteria: These are the alterations that are applied to the messages in the record selected by the selection criteria. As for the selection, some specific alterations need to be defined depending on the sensors. Attenuation can be added to an alteration to simulate the geographical distance between the different selected records.

Time frame: It represents the time windows affected by the linked scenario action. Its representation is in absolute time in relation to the recording file.

In this section we finish to answer the **RQ2**. We have developed a comprehensive multi-step approach to meet the FDIA challenge. We have also defined a Domain-Specific Languages that aims to define complex scenarios of FDIA based on the characteristics of our systems and their data. Source files and a prototype of our approach can be found at https://gitlab.com/mbriland/fdia-simulation.

5 Experimentation on the DSL Expressiveness

In this section, we want to demonstrate the interest of the approach in its expressiveness, and therefore in its ability to describe specific FDIA scenarios. The section contributes to answering the **RQ3**. To this end, we will give examples of real attack scenarios and we will represent them in our dedicated language.

5.1 Primitives and Data Usages

The first cover of expressiveness is the primitive one. We have defined four types of primitives to define the elementary operations that a FDIA could perform on real data. Creating, modifying, deleting and copying.

Our first scenario is therefore the data creation. This type of attack makes sense when attacking alarm values, as it requires little or no correlation with other data. It is within this primitive that the use of synthetically generated data is interesting. Mainly for complex attacks. For example, a scenario for a low battery alarm on our system:

```
scenario "alarm triggering"
create things where meter_code="521" set LowBatteryAlarm=true
     from 0 to 9999;
```

Second primitive is the data alteration. This attack primitive is in practice the most widely used, as it is the one that allows the greatest expressiveness on the data. It allows modifying the greatest range of system behaviour by modifying the data values. For example, a scenario to simulate the malfunctioning of a sensor on our system:

```
scenario "failsensor"
alter things where meter_code="521" set temperatureTC=200
    from 0 to 9999;
```

Third primitive is the data deletion. This attack makes sense when the attacker tries to make information disappear. This can be interesting to cancel an intrusion alarm. For example, a scenario to deactivate an open door alarm on our system:

```
scenario "alarmCanceling"
delete things where DoorOpenAlarm=true from 0 to 9999;
```

The fourth primitive is a copy of data. This attack is interesting when the attacker need to reproduce the real behaviour of other devices or other specific data. For example, some kind of spoofing scenario, where data is retrieved from another device to create a new one with the same data:

```
scenario "clone"
copy things where meter_code=500 set meter_code=499 from 0 to
     9999;
```

So these are the four basic types of alterations that can be made with this DSL. In terms of data types coverage by DSL, we can cover all the types of data present in our datasets. An example scenario for involving different data types:

```
scenario "dataCoverage"
alter things where meter_code=500 and area="park03" set
    temperature*=0.03 and isAlive=true from 0 to 9999;
```

5.2 Complex Usages

The previous examples focused mainly on a single device. The strength of FDIA lies in the subtle alteration of several entities to achieve the desired behaviour. In our DSL we can effectively act on several entities. For example, the following scenario selects several entities that lie within a circle through their geographical location:

```
scenario "MultiDevices selection"
alter things where location isInsideCircle
    (47.213865,5.968195,500) set temperature*=0.03 from 0 to
    9999;
```

With all these elements we can build more complex attacks, associating them in scenarios. The scenario in the following describes an attack using multiple primitive to perform a complex attack. Line 2 sets the property that allows you to locate the application point of the scenario. Line 3 describes an alter primitive scenario action, it selects the devices present in a circle defined by a radius in metres and their GPS location. Then, it applies to this selection an increase in the quantity of particles measured over time, attenuated with respect to the distance between the devices and the point of application of the scenario. Line 4 describes a create primitive scenario action, it creates a message containing a pollution alarm when the particle measure in over a threshold. Line 5 describes a delete primitive scenario action, it deletes all messages with a particle number greater than 34000.

```
1 scenario "ComplexAttack"
2 geolocation (47.213865,5.968195)
3 alter things where location isInsideCircle
      (47.213865,5.968195,500) set particles
      +=(0.0->99999.0,10.0) with attenuation of 10.0 from 0 to
      9999;
4 create things where particles>4000 set particleAlarm=true
      from 0 to 9999;
5 delete things where particles>34000;
```

This attack makes it possible to simulate a complex failure attack. Firstly, the number of particles measured will gradually increase (respecting a propagation pattern between devices thanks to attenuation). This can lead to reactions from the system, for example, display within a city of dangerous air quality. Secondly, an alarm is emitted when the value is over 4000. Thirdly, if the particulate data exceeds a certain threshold then messages sent by the devices are suppressed, indicating a failure or malfunction this usually generates alerts to the system operator, probably caused by the particular increase in particle measurement.

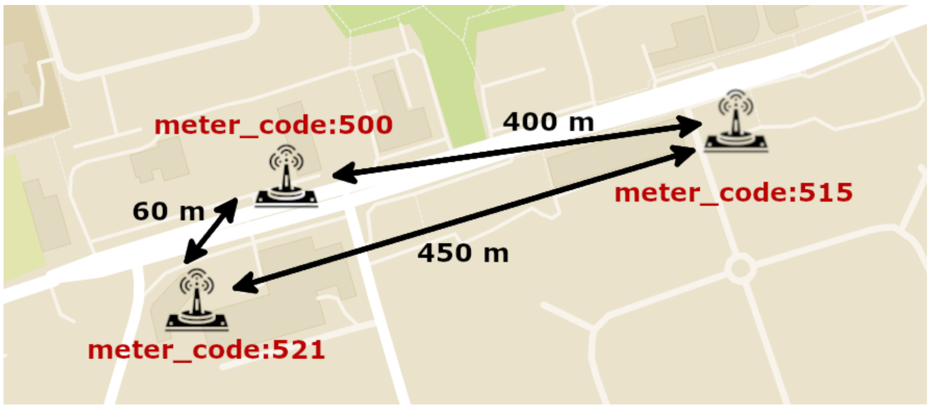

Figure 6: Sensors' geographical locations

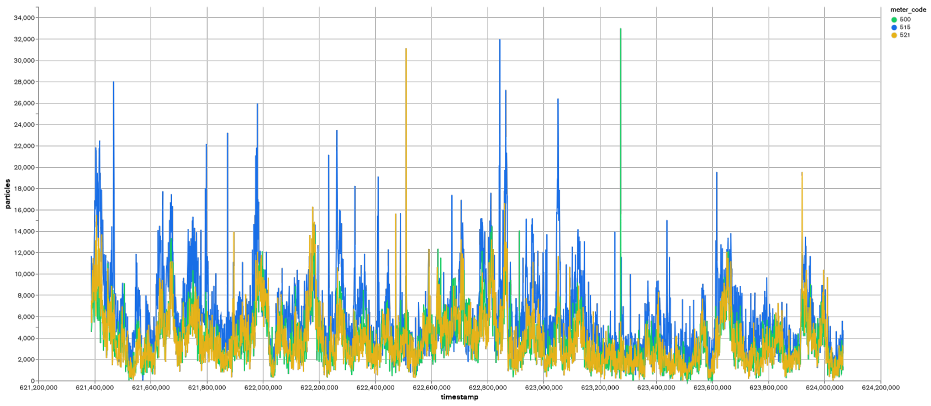

Figure 7: Particle data before running the scenario

[Fig. 7] and [Fig. 8] shows the execution of the ”complexAttack” scenario on one month of particle data collected by the three devices presented in the map in [Fig. 6]. The triggering of the alarm is not represented in the figure. The [Fig. 7] represents the original data and the [Fig. 8] shows the result of the FDIA attack scenario. We see the effect of gradually increasing the number of particles over time and clipping of all particles above 34000.

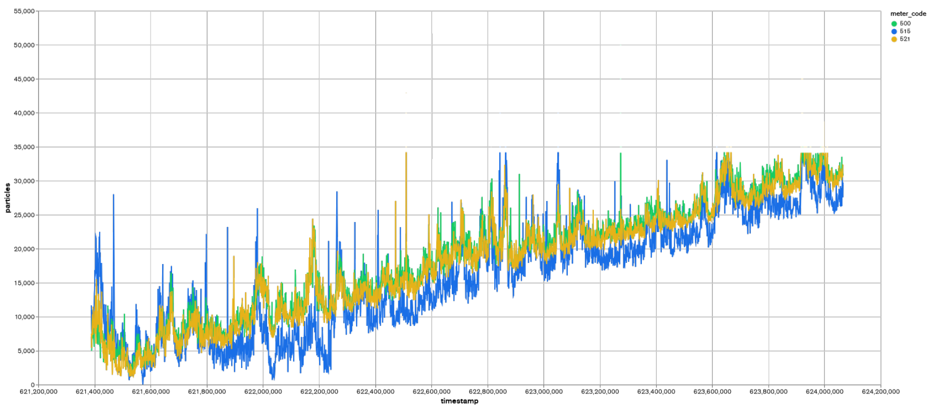

Figure 8: Particle data after running the scenario

5.3 Other Domains Usages

We can therefore perform many different attack scenarios, using the basic functionality of the language. However, this type of attack is strongly linked to the context in which it is carried out and may require specific primitives or selection/alteration functions. A new system may therefore require the evolution of the DSL in order to perform interesting FDIA. The DSL is designed to support this scalability. In the research of [Cretin et al., 2018], the authors identified six specific FDIA scenarios for their air traffic control domain. If we wanted to address these scenarios some would be achievable natively through the DSL, such as the dispartition of aircraft using the delete primitive which would allow messages to be deleted from an aircraft under certain conditions.

However, other more complex scenarios would require the evolution of the DSL, such as the ghost aircraft flooding scenario where several simulated aircraft are created in order to saturate the airspace around selected aircraft. With our actual DSL this attack with the creation of three aircraft would look like this:

```
scenario "saturate"
copy things where ICAO=39AC47 set ICAO=25AC48 and altitude
    +=1000 and longitude+=0.03 from 0 to 9999;
copy things where ICAO=39AC47 set ICAO=26AC49 and altitude
    +=-500 and latitude+=0.015 from 0 to 9999;
copy things where ICAO=39AC47 set ICAO=27AC50 and altitude
    +=1600 and longitude+=0.03 and latitude+=0.026 from 0 to
    9999;
```

With three aircraft the objective is still attainable, but if the saturate attack needs much more aircraft, there is a need to define a new primitive specific for this attack scenario. This scenario also needs specific parameters, selection and alteration function for a correct realisation. This could take this form:

```
scenario "saturate"
saturate things where isInsideSphere
    (47.213865,5.968195,1000,500) set ICAO=random(icaoDB)
    with FloodAmount=50 from 0 to 9999;
```

The action is defined by the primitive named ”saturate”, followed by a new geographical selection (inside a sphere). It may be interesting to develop other types of geographical selections for this specific domain. Then we define an alteration criterion with a notion of randomness for the identification of the simulated aircraft. Followed by a parameter used for the action, which is the number of aircraft to simulate for each selected aircraft. As the scenario is already feasible with the native DSL, the new implementation can be easily added.

This shows us that our DSL allows covering basic and general alterations that could occur in our attacked systems. It also allows us to extend it with more complex and domain-specific alterations by defining new alteration primitives and new alteration functions targeted to that domain. This gives us good confidence in the expressiveness of the DSL and answer the **RQ3**.

6 Conclusion and Discussion

In this work, we addressed the FDIA challenge that could take place on our systems. To do this, we propose an approach centred on the use of a DSL to model the attacks. We then run the scenarios described using the DSL on real data, which allows us to keep the essence of the data. This altered data can then be used, for example, to test our systems by injecting them, or to train machine learning detection systems.

The DSL allows simple selections like selection by identifier, but also more complex selections such as geographic selections. The DSL also allows simple and more complex alterations such as distance-based alteration attenuation. It allows a high expressivity to address FDIA affecting multiple data or systems. In particular, it has an upgrade capability to handle new systems and domains vulnerable to FDIA.

As mentioned in the introduction, our systems have much in common with those in the IoT field. We therefore think it would be interesting to explore how our DSL could help the IoT field to be more resilient to FDIA, which is also very vulnerable to such attacks.

There are multiple objectives to improve this prototype. The first objective is the overall amelioration of the DSL through a better expressiveness with more selection and alteration function, this may involve the use of multiple domain systems to reveal the most interesting attacks in multiple specific areas. One objective is also the implementation of a link between the prototype and the SUT to perform tests directly from the tool. This assertion system could take shape within the DSL to specify and validate that the attack scenario triggers or not certain properties within the SUT. In the longer term, the goal would be to improve the attacks to make them more effective based on the data attacked, this would allow a better test coverage, and help the expert to select the best scenarios. To make them more effective, the use of artificial intelligence through machine learning methods would be interesting. Moreover, this development could be coordinated with the development of a tool to detect FDIA. For this purpose, the use of Generative Adversarial Networks (GANs) would be very indicated, which would improve FDIA generation and detection in both directions.

Acknowledgements

Thanks to Luc Porchon, Abbas Ahmad and Jérémy Bourdé for their feedback. Work supported by the App2-ISITE BFC and FEDER SARCoS project, ANR Gelead project and EIPHI Graduate School (contract ANR-17-EURE-0002)

References

[Aazam et al., 2014] Aazam, M., Khan, I., Alsaffar, A. A., and Huh, E.-N. (2014). Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved. In *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST)*, pages 414–419, Islamabad, Pakistan. IEEE.

[Albright et al., 2010] Albright, D., Brannan, P., and Walrond, C. (2010). Did Stuxnet Take Out 1,000 Centrifuges at the Natanz Enrichment Plant? Technical report, Institute for Science and International Security.

[Anderson et al., 2014] Anderson, J. W., Kennedy, K. E., Ngo, L. B., Luckow, A., and Apon, A. W. (2014). Synthetic data generation for the internet of things. In *2014 IEEE International Conference on Big Data (Big Data)*, pages 171–176. IEEE.

[Bostami et al., 2019] Bostami, B., Ahmed, M., and Choudhury, S. (2019). False Data Injection Attacks in Internet of Things. In Al-Turjman, F., editor, *Performability in Internet of Things*, pages 47–58. Springer International Publishing, Cham.

[Chaojun et al., 2015] Chaojun, G., Jirutitijaroen, P., and Motani, M. (2015). Detecting false data injection attacks in AC state estimation. *IEEE Transactions on Smart Grid*, 6(5):2476–2483.

[Cretin et al., 2018] Cretin, A., Legeard, B., Peureux, F., and Vernotte, A. (2018). Increasing the Resilience of ATC systems against False Data Injection Attacks using DSL-based Testing. In *8th International Conference on Research in Air Transportation (ICRAT 2018)*, pages 1–4.

[Czech et al., 2018] Czech, G., Moser, M., and Pichler, J. (2018). Best practices for domain-specific modeling. a systematic mapping study. In *44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 137–145.

[Farooq et al., 2015] Farooq, M. U., Waseem, M., Khairi, A., and Mazhar, S. (2015). A Critical Analysis on the Security Concerns of Internet of Things (IoT). *International Journal of Computer Applications*.

[Fowler, 2010] Fowler, M. (2010). *Domain Specific Languages*. Addison-Wesley Professional, 1st edition.

[Fremont et al., 2019] Fremont, D. J., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A. L., and Seshia, S. A. (2019). Scenic: A language for scenario specification and scene generation. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, page 63–78.

[Hoag and Thompson, 2007] Hoag, J. E. and Thompson, C. W. (2007). A parallel general-purpose synthetic data generator. *SIGMOD Rec.*, 36(1):19–24.

[Hulanicki et al., 1991] Hulanicki, A., Glab, S., and Ingman, F. (1991). Chemical sensors: definitions and classification. *Pure and Applied Chemistry*, 63(9):1247 – 1250.

[Lee et al., 2010] Lee, H. Y., Cho, T. H., and Kim, H.-J. (2010). Fuzzy-Based Detection of Injected False Data in Wireless Sensor Networks. In Bandyopadhyay, S. K., Adi, W., Kim, T.-h., and Xiao, Y., editors, *Information Security and Assurance*, volume 76, pages 128–137. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Liang et al., 2017] Liang, G., Weller, S. R., Zhao, J., Luo, F., and Dong, Z. Y. (2017). The 2015 Ukraine Blackout: Implications for False Data Injection Attacks. *IEEE Transactions on Power Systems*, 32(4):3317–3318.

[Liu et al., 2009] Liu, Y., Ning, P., and Reiter, M. K. (2009). False Data Injection Attacks against State Estimation in Electric Power Grids. *Proceedings of the 16th ACM Conference on Computer and Communications Security*, page 16.

[Manandhar et al., 2014] Manandhar, K., Cao, X., Hu, F., and Liu, Y. (2014). Detection of Faults and Attacks Including False Data Injection Attack in Smart Grid Using Kalman Filter. *IEEE Transactions on Control of Network Systems*, 1(4):370–379.

[Mernik et al., 2005] Mernik, M., Heering, J., and Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM Computing Surveys (CSUR)*, 37(4):316–344.

[Mo et al., 2010] Mo, Y., Garone, E., Casavola, A., and Sinopoli, B. (2010). False data injection attacks against state estimation in wireless sensor networks. In *49th IEEE Conference on Decision and Control (CDC)*, pages 5967–5972. IEEE.

[Padmavathi and Shanmugapriya, 2009] Padmavathi, D. G. and Shanmugapriya, M. D. (2009). A Survey of Attacks, Security Mechanisms and Challenges in Wireless Sensor Networks. *International Journal of Computer Science and Information Security, IJCSIS, Vol. 4, No. 1 & 2, August 2009, USA*, 4(1):10.

[Popić et al., 2019] Popić, S., Pavković, B., Velikić, I., and Teslić, N. (2019). Data generators: a short survey of techniques and use cases with focus on testing. In *IEEE 9th International Conference on Consumer Electronics*, pages 189–194.

[Rabl and Poess, 2011] Rabl, T. and Poess, M. (2011). Parallel data generation for performance analysis of large, complex rdbms. In *Proceedings of the Fourth International Workshop on Testing Database Systems*, DBTest '11, New York, NY, USA. Association for Computing Machinery.

[Sencun Zhu et al., 2004] Sencun Zhu, Setia, S., Jajodia, S., and Peng Ning (2004). An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 259–271, Berkeley, CA, USA. IEEE.

[Sikder et al., 2018] Sikder, A. K., Petracca, G., Aksu, H., Jaeger, T., and Uluagac, A. S. (2018). A Survey on Sensor-based Threats to Internet-of-Things (IoT) Devices and Applications. *CoRR*, abs/1802.02041.

[White, 1987] White, R. (1987). A Sensor Classification Scheme. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 34(2):124–126.

[Yang et al., 2017] Yang, L., Ding, C., Wu, M., and Wang, K. (2017). Robust detection of false data injection attacks for data aggregation in an Internet of Things-based environmental surveillance. *Computer Networks*, 129:410–428.

[Yi Huang et al., 2011] Yi Huang, Li, H., Campbell, K. A., and Zhu Han (2011). Defending false data injection attack on smart grid network using adaptive CUSUM test. In *45th Conference on Information Sciences and Systems*, pages 1–6. IEEE.

[Zhao and Ge, 2013] Zhao, K. and Ge, L. (2013). A Survey on the Internet of Things Security. In *2013 Ninth International Conference on Computational Intelligence and Security*, pages 663–667, Emeishan 614201, China. IEEE.