UNIVERSITY OF TWENTE
PERVASIVE SYSTEMS GROUP

Towards Implicit Authentication in Smart Logistics
A Random Number Generator for Sensor PUFs in Resource Constrained IoT Devices

M.A.M. Hillerström

Supervisors:
prof. dr. ir. P.J.M. Havinga
Msc. I. Ullah
dr. ir. D.M. Ziener

June 26, 2021

Abstract

Over the years the freight transportation industry has become more complex, with increasing delivery demands and more parties involved. To increase the efficiency in the logistics sector it is necessary to manage goods during transportation, leading to a dynamic supply chain. This can be achieved by introducing the Internet of Things (IoT) into transportation making the logistics 'smart'. IoT objects can give real-time information on transit times, alternative routes and interact with operations by making decisions, which increase the flexibility of the routing system. One major challenge in smart logistics is securing the vast amounts of collected data stored on sensing devices or in a central database from unauthorized access. Since IoT objects consists of resource constrained sensing devices there is need for a secure and efficient authentication scheme. This work proposes to use a randomized version of the sensed data from the IoT objects itself for use in a Sensor PUF for authentication. Three Random Number Generators (RNGs) are designed and tested on a created dataset that is similar to transportation data. The tested randomization methods are based on bitwise XOR, multiplication with a natural constant or the Fast Fourier Transform (FFT). The RNGs are tested for their randomization abilities with the NIST statistical test suite, as well as their time complexity and time consumption. The results show that the best and most constant randomization results are achieved by first applying the FFT and subsequently a bitwise XOR operation on the sensed data. Further tests also indicate that this RNG works well on short term data gathering (one second), which is required for real-time applications.

1 INTRODUCTION

Over the years the freight transportation industry has undergone some significant changes. Transportation companies have more vehicles to manage, customers have higher delivery demands and the transportation network has become more complex [31]. The introduction of intermodal transportation has caused a reduction of costs and delivery times in logistics [20]. UNECE (United Nations Economic Commission for Europe) [51] has defined intermodal transportation as: "the movement of goods in one and the same loading unit or road vehicle, which uses successively two or more modes of transport (rail, sea, air, or inland waterway), without moving the goods themselves in changing modes". Intermodal transport causes more parties to be involved in one shipment, which influences the decision making time and process, leading to a suboptimal solution for the customer. To increase the efficiency in logistics further, it is necessary to be able to manage goods during transportation [20] (for example re-routing in case of a disruption) leading to a more dynamic supply chain.

A dynamic supply chain can be achieved by making logistics 'smart'. According to Verduijn and van de Loo [52] smart logistics contains three important features which lead to a more efficient and dynamic transportation model, i.e. connectivity, transparency and planning. Smart logistics can be achieved by introducing the Internet of Things (IoT) in the logistics sector. The Internet of Things (IoT) is a network of smart objects, which are equipped with sensors, processors and communication modules, to monitor their environment, communicate with other smart objects and their network and to reason to make their own decisions, to improve the operations and surroundings [15, 39, 42].

IoT applications can be found in all fields and industries, e.g. healthcare, transportation, agriculture, smart cities and smart homes. For example, in the agricultural industry IoT can be used to monitor the soil and crops and accordingly optimize water management. In logistics the smart objects can give real-time information on transit times, alternative routes and interact with the operations by making decisions, which increases the flexibility of the routing system. For a more comprehensive list of applications, see [43].

One of the challenges in IoT are the constrained resources of the smart objects, which means limitations must be taken into account during development. One of these challenges is limited power consumption, as the devices are often deployed in a remote area, running on batteries, which are not easy to renew. This means the device must use the limited energy source efficiently [15, 39].
Another challenge is the collection of vast amounts of sensed data, which may be stored on the device or transmitted to other devices or a central database. Many companies want to protect the data of the devices for unauthorized access with, among others, authentication measures.

Since the rising need for more secure communication in Wireless Sensor Networks (WSNs) and IoT, research focused more towards developing secure and efficient authentication schemes. Many authentication protocols use encryption keys to validate users [1, 3, 53]. In [18] encryption is based on attributes and key-exchange is based on landmarks, to achieve segmented access control. However, the authors have developed the scheme for mobile devices, which have more powerful resources than the constrained IoT devices. For smart logistics, the limited power, storage and computational resources of the IoT devices must be taken into account when developing an authentication scheme.

The authors of [41] and [47] have done this by reducing overhead with the use of Elliptic Curve (EC) cryptography. The benefit of EC cryptography is the smaller key size, compared to the more conventional RSA encryption keys, without compromising on security. The authors of [2] assure mutual authentication by exploiting hardware anomalies, in the form of Physical Unclonable Functions (PUFs). PUFs use manufacturing variations to generate a unique function output, similar to a fingerprint.
Another important requirement for authentication in smart logistics is authentication without human involve-

ment. This means that the IoT devices must be able to authenticate themselves and others in an implicit manner. This we refer to as implicit authentication, which can be achieved by using encryption keys. Random sequences play an important role in secure encryption keys. In this research we aim to use sensor data from an IoT sensor node to create a random sequence for cryptographic purposes.

The rest of the report is organized as follows. Section 2 explains terminology regarding Random Number Generators and cryptography. Section 3 explains Physical Unclonable Functions (PUFs) and discusses different PUF variants, including the Sensor PUF. Section 4 states the research problem this study addresses. Section 5 discusses the dataset creation, analyses the randomness of the created dataset and presents three Random Number Generators. Section 6 discusses the randomization performance of the presented randomization algorithms, as well as their complexity, time consumption and security against malicious attacks. Section 7 draws a conclusion.

2 Definitions

In this section terminology regarding random numbers and cryptographic mechanisms is introduced.

Random Number Generator (RNG)

A random sequence can be produced by a random number generator (RNG). This generated sequence cannot be determined, other than by random chance. There are two types of RNGs, a pseudo random number generator (PRNG) and a truly random number generator (TRNG).

Pseudo Random Number Generator (PRNG)

A PRNG is a deterministic function that produces a sequences of seemingly random numbers. The output sequence approximates randomness; the sequence appears random, but can be regenerated when the same input is given to the PRNG. This initial input is called the seed. A PRNG will produce the same output each time the exact same seed is applied.

True Random Number Generator (TRNG)

A TRNG uses physical phenomena which are assumed to be random as input source to create a sequence of random digits. Examples of such phenomena are atmospheric noise, cosmic background radiation and thermal noise in electrical components.

Entropy

Entropy is the amount of 'unexpectedness' in a sequence of digits. Entropy is expressed in bits, from zero to one bit. Zero bits entropy means there is no information in advance on what the outcome of the bit will be (0 or 1). One bit entropy means that it is known precisely what the outcome of the bit is [21].

3 Physical Unclonable Functions

As explained above, the introduction of smart objects in the transportation industry, so called Intelligent Products (IPs), induces the need for implicit authentication measures. PUFs (Physical Unclonable Functions)

are a very good candidate, as they are very secure by relying on uncontrollable manufacturing variations and are suitable for constrained devices. Therefore, this research will focus on the use of Random Number Generators for PUFs for implicit mutual authentication in smart logistics.

PUFs are based on the natural variabilities that emerge from the manufacturing process. These uncontrollable, device specific variations serve as a digital fingerprint to the device and can be used in device-identification & -authentication and in encryption key generation. The unclonability stems from the uncontrollable manufacturing variabilities, which make it impossible to create an identical device with the same circuit characteristics. When the manufacturing variations need to be explicitly added to the device, the PUF is called an 'explicit PUF'. Variations inherent to the device result in an 'implicit PUF'.

As mentioned, PUFs can serve different applications. During identification one wants to retrieve the identity (ID) of a known entity or give an ID to a new entity. With authentication the claimed device ID is verified. One way of authenticating a device is by using challenge-response-pairs (CRPs), a procedure based on the challenge-response authentication protocols (e.g. password authentication) often used in computer systems. PUFs can also be used for encryption-key generation. Their unique characteristics and unclonability makes it difficult for an adversary to get the secret key, which make PUFs very secure. The PUF response can be used as input to a key generation algorithm or directly as secret key. In the latter case, the response must be significantly large to be secure, possibly resulting in a large, expensive PUF.

3.1 PUF TYPES

Many different PUF types have been designed in the last decade. This section gives a non-inclusive overview of the different PUFs, a more complete overview can be found in [38].

The *optical PUF* consists of a transparent optical medium that is explicitly added to the device during manufacturing. This optical medium creates a unique speckle pattern when hit with a laser beam at a certain angle. The PUF challenges are formed by varying the laser beam angle and the response is the Gabor hash derived from the speckle pattern. The *Phosphor PUF* [38] uses a similar principle, in which the pattern created by phosphorescent particles is measured with UV light. The different challenges are formed by varying the measured section of the phosphorescent layer.

The *coating PUF* is an explicit PUF based on a coating layer added on the chip. This coating consists of dielectric particles that randomly vary in shape and size, making it unclonable [50]. The capacitance of the coating varies for different voltage levels. Thus the adjustable voltage is the PUF challenge and the measured capacitance the PUF response. The PUF is tamper proof, since any physical attacks on the coating alter the dielectric coating and consequently the PUF response [35].

The *magnetic PUF* [38] is used for the authentication of magnetic swipe cards. On the magnetic strip a ferromagnetic material is added, consisting of particles varying in size, shape and position. The formed particle distribution is unique for each magnetic swipe card, due to the manufacturing variations of the ferromagnetic particles.

Memory based PUFs are implicit PUFs based on the preferred stable state of memory cells. A memory cell transitions from an unstable state into a preferred stable state. This preference is determined by the physical mismatch between the transistors [8]. The state of the memory cell is a 1-bit PUF response. To achieve a secure PUF, many memory cells are needed to create a significantly large bit array as response. There are several kinds of memory PUFs, which differ slightly in their operation. The memory cells of the SRAM PUF [38] and the DFlipFlop PUF [35] settle to a stable state on device power-up. This means that in order to get a new PUF response during run-time, the device must first be powered off and on again. The Butterfly PUF [35] solves this problem by using latches which can be put in an unstable state, and

subsequently reach a stable state, while powered.

The *threshold voltage (V_t) PUF* [38] is based on the manufacturing variations of transistors. The V_t PUF measures the threshold voltage (V_t) of a transistor in the circuit. The challenge of the PUF is the selected transistor and the PUF response is the measured V_t. The output of this PUF only varies when a different challenge is given, therefore this PUF can only provide security for a CRP based application.

Another PUF exploiting the manufacturing variations of the transistor is the *carbon nanotube PUF* [38]. In this PUF carbon nanotube transistors are placed in two columns. One row is selected and with an arbiter circuit is determined which carbon nanotube transistor has more current flowing through. Correspondingly, the arbiter returns a '0' or a '1'. The PUF challenge is the selection of the carbon nanotube transistor and the response is the measured current. Since the transistors need to be placed in a specific configuration, the PUF is explicit.

In a *power distribution PUF* [38] the unique characteristics of power transfer lines in the power distribution grid in a circuit are used to identify a system. The measured resistance of each power transfer line is particular to that power line. To make sure only the resistance of the power transfer line is measured, the grid is modified to be able to bypass any existing components in the circuit. Therefore, this PUF requires additional manufacturing steps, resulting in an explicit PUF.

The *acoustical PUF* [38] uses acoustical delay lines of a circuit to characterize a system. Acoustical delay lines are used to delay a signal with the use of mechanical oscillation. Due to manufacturing variations each acoustical delay line generates a different constant frequency response, which serve as the PUF response. The PUF challenge is the acoustical delay line of which the response is measured.

The *super high information content (SHIC) PUF* [38] uses nano-diodes in a matrix configuration, where each diode has a unique output. Because the nano-diodes are arranged in a matrix, a large number of CRPs are possible. The SHIC PUF has a minimal response time, which means that potential attackers have insufficient time to read out the CRPs. The PUF is explicit, since it requires the addition of a nano-diode matrix.

A *board PUF* [38] is an explicit PUF consisting of a layer of capacitors implemented on a printed circuit board (PCB), to authenticate a PCB or generate an encryption key. The capacitors vary based on manufacturing variations and thus the measured capacitance will vary per capacitor. The PUF challenge consists of a particular capacitor or group of capacitors and the PUF response is the measured capacitance.

A *delay based PUF* is an implicit PUF based on variations in delay of two identical paths in the chip circuit. In the *Arbiter PUF* [38] a comparator determines which path is the fastest and accordingly outputs a '0' or a '1' as PUF response. To get a multiple-bits response one can add the responses to sequential given challenges or combine multiple delay circuits in parallel. The *Clock PUF* [38] is very similar to the Arbiter PUF, as it determines the fastest path in the clock network of the circuit. The clock network distributes the clock signal throughout the circuit, to minimize the latency between clock signals in different parts of the circuit. Some latency always remains, which can be used to determine the clock difference between two parts of the circuit. The difference is unique per circuit, due to the manufacturing variations in the clock signal lines and can, therefore, serve as a PUF. A *Ring-Oscillator (RO) PUF* [38] measures the delay of two identical circuit paths in a different manner, as it is based on an oscillating frequency. From the oscillating frequency the delay is determined by edge-detectors and -counters, of which the value of the latter is the PUF response. The RO PUF is more reliable and easier to implement than the Arbiter PUF. The Arbiter PUF, however, is faster, smaller and consumes less power, making it more suitable for constrained devices [10].

A *Radio Frequency (RF) based PUF* uses the characteristics of a radio frequency wave to identify a system. The *RF-DNA PUF* [38] utilizes radio frequency scattering, similar to how an optical PUF utilizes light scattering. In the RF-DNA PUF the scattering of the radio frequency wave is caused by a randomly arranged copper wire grid. The scattering is influenced by the radio wave frequency and is measured by a RF scanner. Another RF based PUF is the *LC PUF* [38], where L is the symbol for inductance and C for

the capacitance. In the LC PUF a passive LC circuit is used to absorb power, by placing it onto an RF field. The amount of absorbed power is dependent on the amplitude and frequency, that in turn are determined by the values of the inductance and capacitance. Since the characteristics of the inductor and capacitor are unique, due to manufacturing variations, the CRPs are unique as well. Both RF based PUFs are explicit and only have a varying response, when the challenge is varied. Chatterjee et al. [12] did introduce an implicit RF-PUF, that can be used to identify and authenticate a radio transmitter. The PUF uses two artificial neural networks to learn both the data and signal characteristics of the transmitter nodes in the network. This requires more power and computational resources than a typical sensor node has, meaning that this PUF can only be implemented on the less-resource-constrained hubs. As such, the RF-PUF described in [12] is not suitable for mutual authentication in a remote wireless sensor network.

The *reconfigurable PUF* [35] is not a stand alone PUF, but an extension to an existing PUF. With this extension PUF the basic PUF can be randomly and irreversibly reconfigured, causing the CRPs to change. The alteration of CRPs can be used to prolong the PUF's life or to reset the PUF before using it in a different application. An example of such a PUF is a reconfigurable optical PUF [29]. By melting the transparent optical medium, the configuration of the medium changes, leading to changes in the speckle pattern. The reconfigured optical PUF will now give different responses to the unchanged challenges.

Another type of PUF is the *Sensor PUF*, where the manufacturing variations in sensors are used to create the PUF. There are several definitions and types of Sensor PUFs, which are discussed in the next section.

3.2 SENSOR PUF

The Sensor PUF is first mentioned in [44] where they use a twofold PUF, consisting of a sensing PUF and a conventional PUF. Combining two PUFs into one results in the combination of cryptography and sensing. The PUF challenge consists of both a binary challenge and a physical quantity (PQ), i.e the sensor measurement. The sensing part of the PUF is explicitly added to the PUF in the form of photo-diodes and a coating, making it an explicit PUF. A similar approach is used in [33] where a transducer is used to convert a PQ to a voltage level, which is then used together with a challenge as input to an RO-PUF. The output is the PQ measurement encrypted by the Ring-Oscillator (RO) PUF. The combination of PQ and RO-PUF is described as Sensor PUF.

In [13] the authors describe a single PUF which is used a both a sensor and a encryption device. They use DRAM memory cells to measure the temperature, as the DRAM cells will switch state due to current leakage. This current leakage is influenced by temperature and manufacturing variations of the DRAM cells. Because of the manufacturing variations the output can be used for authentication purposes as well. A similar concept is used in [48] where they use microelectromechanical (MEM) relays to measure pressure, as the voltage level of the relays depends on pressure and manufacturing variations. Both the PUFs from [13] and [48] are weak PUFs as they only have a 1 bit response per cell or relay, which means many sequential PUFs are needed to get a strong response. Furthermore, the resolution of the sensor aspect of the MEM relay PUF is too low for the PUF to function well as a sensor.

In [38] a PUF that uses existing sensors to create a Sensor PUF, for authentication or encryption purposes, is described. The PUF is based on MEMS sensing units, where an array of accelerometers and gyroscopes are used for the PUF. Due to the manufacturing variations each accelerometer or gyroscope behaves slightly different, even when they share the exact same movements. The MEMS PUF provides a challenge, an electrostatic pulse, to a subset of accelerometers/gyroscopes in an array of accelerometers/gyroscopes. The PUF response consists of the measured values. Considering a WSN sensing node typically contains just one sensor of a certain type and not an array of sensors, the MEMS PUF in this form is not suitable for a sensor node.

There are several papers in which only one accelerometer is used for authentication or key generation, by shaking the device. In [37] and [7] two handheld devices are authenticated by shaking them together, to create a common shared random secret, for example a cryptographic key. The same key is created on both devices only if the two devices are shaken together in one hand. Because each device creates its own symmetric key, there is no need to share any acceleration data or cryptographic keys for authentication. In [17] a similar Sensor PUF is used to authenticate the user to a wearable device. By shaking the wearable a 128 bit key is computed from the maximum and minimum accelerometer values, which is used for authentication.

The authors of [11] describe a Sensor PUF that besides authentication is used as a sensor. A CMOS image sensor is used to create an image and consequently use the feature vectors, timestamp and a PUF challenge to create a fingerprint on the image. The fingerprint can be used to determine from which sensor this image originated from.

3.3 PUF COMPARISON

In the previous sections an extensive overview of the different PUF types was provided and in this section we will compare these PUFs and discuss their limitations. In Table 3.1 an overview of the comparison is given. It shows that there are many possible parameters to use in a PUF. However, not all PUF implementations are feasible for an implicit mutual authentication PUF for smart logistics applications.

As can be seen from Table 3.1 most PUFs are explicit and have extrinsic evaluation. This is the case for the following PUFs: optical, phosphor, coating, carbon nanotube, power distribution, SHIC, board, RF-DNA, LC, MEMS and the Sensor PUF of [44]. This means that for all these PUFs additional manufacturing steps are needed, which costs valuable time and money. Extrinsic evaluation means the output is evaluated outside the PUF device, intrinsic evaluation happens on the PUF device. The magnetic-, arbiter-, clock-, RO- and RF-PUF are implicit, however, not intrinsically evaluative, which means there is still need for additional steps and there are additional costs.

The comparison Table 3.1 also shows that the sensor-related PUFs are not tamper proof. For PUFs used in logistics operations it is crucial to be tamper proof, since they will be unattended most of the time, leaving them vulnerable to physical attacks. Consequently, they must also be protected against modeling attacks. Instead of making the PUF tamper proof, one can make the sensor device tamper proof, with for example protective casing.

Several of the discussed PUFs are most suitable for CRP related applications, which requires the use of a database. These are the optical-, phosphor-, coating-, magnetic-, memory based-, threshold voltage-, carbon nanotube-, power distribution-, SHIC-, board-, delay based-, RF-DNA-, LC- and MEMS-PUF. In logistical WSNs a database is not always reachable, which means that the obligation of a database should be avoided. This can be done by using an unpredictable varying PUF output that does not need to be compared to an entity in a database or not need to be pre-generated and stored in a database. This output can be used to generated an encryption key for authentication, instead of using CRPs.

Most of the PUFs discussed are suitable for resource constrained devices. However, this is not the case for the RF-PUF, as it is designed to be implemented on the receiver-hub of the WSN. This also means that with the RF-PUF mutual authentication is not possible in an implicit way. Two of the memory based PUFs, the SRAM PUF and the DFlipFlop PUF, are not suitable for WSNs because of the need to power the device off and on for each authentication. This disrupts the usage of the sensor node in the network, making it less valuable to the WSN.

Now that we have defined the limitations of the current PUFs for implicit mutual authentication in smart logistics, we can define the focus of this research in the next section.

Table 3.1: PUF comparison. The FHD inter is the similarity between the output from two different PUFs to the same input. The FHD intra is the similarity between the output from one input given to the same PUF twice. In modeling attacks, the PUF can be cloned when input-output pairs are known.

PUF	Reference	Parameter	Implicity	Evaluation	FHD inter (%)	FHD intra (%)	Tamper evident	Modeling attack
Optical	[35, 40]	Light intensity	Explicit	Extrinsic	49.79	25.25	yes	Not possible
Phosphor	[14, 38]	UV light intensity	Explicit	Extrinsic	?	?	yes	?
Coating	[35, 49]	Capacitance	Explicit	Extrinsic	~50	<5	yes	Possible
Magnetic	[25, 38]	Magnetic field	Implicit	Extrinsic	?	?	yes	?
SRAM	[22, 35]	Transistor power-up state	Implicit	Intrinsic	49.97	3.57	?	Possible
D-FlipFlop	[34, 35]	FlipFlop power-up state	Implicit	Intrinsic	~50	<5	yes	Possible
Butterfly	[28, 35]	Logical stable states	Implicit	Intrinsic	~50	<5	?	Possible
Threshold Voltage	[32, 38]	Transistor voltage	Implicit	Intrinsic	50.00	1.30	?	?
Carbon Nanotube	[27, 38]	Transistor current	Explicit	Extrinsic	49.67	1.90	?	Not possible
Power Distribution	[24, 35]	Resistance	Explicit	Extrinsic	?	?	yes	?
Acoustical	[35, 54]	Frequency spectrum	Implicit	Extrinsic	?	?	yes	Possible
SHIC	[38, 46]	Voltage/current	Explicit	Extrinsic	?	?	?	Not possible
Board	[38, 55]	capacitance	Explicit	Extrinsic	47.21	3.63	yes	Not possible
Arbiter	[30, 35]	Signal delays	Implicit	Extrinsic	23.00	5.00	?	Possible
Clock	[38, 56]	Clock signal	Implicit	Extrinsic	50.30	5.07	yes	?
RO	[19, 35]	Frequency	Implicit	Extrinsic	46.00	0.48	?	Possible
RF-DNA	[16, 35]	Radio frequency scattering	Explicit	Extrinsic	?	?	yes	?
LC	[23, 35]	Power absorption	Explicit	Extrinsic	?	?	yes	?
MEMS	[4, 38]	Accelerometer values	Explicit	Extrinsic	42.64	92.17*	?	?
Sensor PUF	[44]	Characteristics photo diodes	Explicit	Extrinsic	?	?	no	?
Sensor Wearable PUF	[17]	Accelerometer values	Implicit	Intrinsic	?	?	no	?
RF-PUF	[12]	Tx characteristics	Implicit	Extrinsic	?	?	no	Possible

4 Research Proposal

To make a PUF more suitable for implicit authentication in smart logistics and given the study of the advantages and disadvantages of the different PUF types in the previous section, we propose a Sensor PUF that uses the physical properties of the sensors in the system for the creation of a secure encryption key. In literature there are several definitions of Sensor PUFs or PUF sensors, as can be seen from the paragraph on Sensor PUFs in Section 3.2. In this research we define the following terms:
A *PUF protected sensor* is a sensor that is protected against attacks with the use of a PUF. A *PUF based sensor* means a PUF that is used as a sensor [13]. A *Sensor PUF* is a PUF based on the intrinsic physical properties of a sensor. This remainder of this paper focuses on the latter, i.e. the Sensor PUF.
The Sensor PUF will use the varying sensor output in the generation of the encryption key. The use of cryptography for authentication eliminates the use of a database. Some sensors are more applicable to a Sensor PUF than others. This depends in part on their randomness, the update interval and number of measurement parameters, as will be discussed further in Section 4.2.

We discuss how a sensor node's sensor data can be used in a Random Number Generator for use in a Sensor PUF. The Random Number Generator will use a sensor or a combination of multiple sensors to

produce the random sequence. It is assumed that the sensing devices are tamper proof and consequently that the final Sensor PUF is secure to physical attacks. To conduct this research, several research questions are formulated in the next section.

4.1 RESEARCH QUESTIONS

In particular, this thesis will examine the following main research question:
How can a sensor node in a smart logistics wireless sensor network be used to create a Random Number Generator for use in a Sensor PUF?

To answer this question, a set of three sub-research questions has been formulated:

1. *Which sensors or combinations of sensors of an Intelligent Product (i.e. sensor node) are applicable to create random data for a Sensor PUF?*
2. *How can we obtain useful data from these (combinations of) sensors for encryption purposes?*
3. *Is the extraction method suitable for constrained devices, as found in a smart logistics wireless sensor network?*

4.2 HYPOTHESIS

The suitability of sensors for a RNG in a Sensor PUF depends on the randomness of the output data, the sensing interval and the number of measurement parameters. The randomness indicates how uniformly distributed the output in the sensor is, which is important for secure key generation. A higher randomness makes it more difficult to guess an encryption key, which makes the key more secure. The update interval indicates how quickly the physical quantity changes over time. For example, the temperature changes less over time than accelerometer values. The number of measurement parameters is based on the different output parameters of one sensor. For example, a temperature sensor has one measurement parameter, the temperature, whereas an accelerometer has three measurement parameters, the acceleration on the x-, y- and z-axis.

Table 4.1 gives an estimation of which sensors can be used for the ideal Sensor PUF RNG. The estimation contains the randomness, update interval and measurement parameters. Together these three aspects are used to form the final score. As the table shows, it is expected that certain sensors are more applicable to a Sensor PUF than others. Sensors with a higher update interval are likely to produce more varying output over time than sensors with a low update interval. The randomness in the measurement values is also important, since with a repetitive output, the encryption key is easier to predict. The table shows that we expect the accelerometer, gyroscope and gas sensor to be valuable candidates for a PUF sensor. Of these three, we expect the accelerometer and gyroscope most likely to be present in a smart logistics application. It is improbable that the gyroscope alone will be sufficient for a functioning Sensor PUF, based on that a gyroscope does not generate uniform distributed data and the results of [17]. We do expect that combining the gyroscope with other sensors will result in a more ideal Sensor PUF, opposed to when only one sensor is used.

Table 4.1: The ideal Sensor PUF

Sensor	Randomness	Update interval	Measurement parameters	Final Score
Temperature	--	--	--	--
Accelerometer	++	++	++	++
Gyroscope	--	++	++	+
Gas	+	-	+	+
GPS	+	-	-	-
Light	+	-	-	-
Rain	+	--	-	-

Additionally, in terms of efficiency we expect the computation time and energy consumption of the extraction method to be proportional to the size of the data to be randomized. A larger packet size results in a larger computation time and more energy consumption. This also follows from the correlation between the computation time and the energy consumption.

The next step is to create a dataset of sensor data, distributed uniformly, to form the input of the Sensor PUF. This dataset should contain single sensors and combinations of sensors. Since the Sensor PUF will run on a constrained device (a sensor node), the Random Number Generator must be designed for limited energy consumption, memory usage and processing time. It is important that the extracted values are uniformly distributed, therefore this property should be tested with standard statistical tests. When the distribution has been proven to be uniform, the Sensor PUF output is suitable for usage in authentication systems. In the next section the process to create the sensor-data dataset is described in more detail, as well as the different extraction methods used.

5 Data Randomization

As described previously, the mechanism to obtain random sensor data for encryption purposes must be designed for constrained devices. This means that for data extraction and processing the limitations for computational power and memory usage must be taken into account. This section aims to construct and implement a mechanism with which a dataset of random sensor data can be obtained. First, data will be gathered with a sensor device, after which its randomness is evaluated with statistical tests. When this unaltered sensor data proves to be non-random, an algorithm will be used to generate random data from the sensor dataset.

5.1 Creation of dataset

In the previous section it was concluded that the dataset should contain data from single sensors as well as combinations of sensors. Table 4.1 shows us that the accelerometer and gyroscope are good candidates for a Sensor PUF in smart logistics. These two claims combined result in the choice for an Inertial Measurement Unit (IMU) to gather data, see Figure 5.1. An IMU uses an accelerometer, gyroscope and magnetometer to determine, with an on-board processor, the linear and angular motion of the object it is attached to.

The IMU used to create the dataset for this research is the Sparkfun 9DoF Razor IMU M0[1], which

[1] `https://learn.sparkfun.com/tutorials/9dof-razor-imu-m0-hookup-guide/all`, accessed 12-12-2020

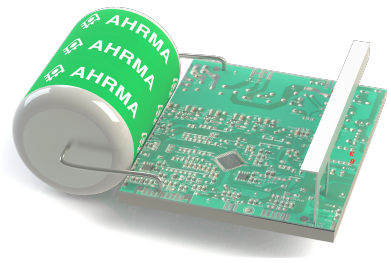


Figure 5.1: Inertial Measurement Unit (IMU) is used to gather the movement data.

contains, besides the three sensors, a LiPo battery connector and a SD card slot. This makes the IMU portable, which is needed to be able to generate similar data as an Intelligent Product (e.g. smart pallet) in the logistics sector would. The Razor IMU M0 uses the on-board processor to calculate the quaternions, yaw, pitch, roll and heading of the device. This gives a set of both single sensor data and data of combinations of sensors, namely: accelerometer x-, y- and z-axis, gyroscope x-, y- and z-axis, magnetometer x-, y- and z-axis, quaternions w-, x-, y- and z-axis, yaw, pitch, roll and heading. See Appendix E for the full description of the dataset.

Before collecting the data, various pre-tests were done to determine the optimal sensor configuration on the IMU for the generation of the dataset. The test results led to a minimal required accelerometer range of $\pm$2g and a gyroscope range of $\pm$500 dps (degrees per second). The default setting of the non-configurable magnetometer is $\pm 4800 \mu$T (Tesla). The data was sampled at 100Hz. The data was obtained by driving in a car (to mimic road transportation), in trips ranging from 50 km to 150 km. The dataset contains subsets from different car trips, each with the same IMU configurations and positioning of the IMU in the car console.

Now that a dataset with sensor data is created, the next step is to see if this sensor data is random. The performed randomness test is described in the next section.

5.2 Data randomness test

The proposed technique is to use the Sensor PUF as an input for the encryption/authentication. Therefore, the sensor data in the created dataset should be random. This means the data should have a uniform distribution of 0's and 1's, should be non-repetitive and should be unpredictable. This is tested with the NIST test suite [6]. The NIST test suite contains multiple statistical tests, designed for cryptographic purposes, that analyses a binary sequence for its randomness. The test suite performs the followin 15 tests; Frequency, Block Frequency, Runs, Longest run of ones, Rank, Fast Fourier Transform, Non Overlapping Template, Overlapping Template, Universal, Linear complexity, Serial, Approximate Entropy, Cumulative Sums, Random Excursions and Random Excursions Variant. Some tests perform multiple subtests. For example the Cumulative Sums test consists of two subtests, performing the test in both forward and backwards direction. The Random Excursions and Random Excursions Varient tests are only exectued under certain circumstances. For these tests to be performed, the Frequency tests must have been passed and the length of the input bits must be greater than 100000 [36].

Before the data could be tested with the NIST test suite, post-processing of the data was needed. The

first and last 30 seconds of data of each trip were removed, because during this time the car is assumed to be stationary, making it very unlikely random data would be created by the sensors. At a sampling rate of 100Hz, 30 seconds of data amounts to 3000 samples. After removing these samples, the data on the different axes of sensors with multiple axes was combined into one data stream. For example, the accelerometer generates data on three axes; x-axis, y-axis and z-axis. This was combined into one accelerometer data stream as follows: Acc_{x_1}, Acc_{y_1}, Acc_{z_1}, Acc_{x_2}, Acc_{y_2}, Acc_{z_2}, Acc_{x_3}, etc. The same was done for the gyroscope-, magnetometer-data and the quaternions. The latter has data on four axes, which was combined in the same manner. The yaw, pitch and roll (YPR) data were combined in a similar manner too. Before combining the YPR data into one stream, the decimal values were extracted, to be used for the tests. The integer values were discarded, since visual inspection showed that these were not random values. Lastly, the same was done for the heading data. In the remainder of this thesis only the decimal values of the YPR and heading data are considered.
Besides combining the data of one sensor, a data combination of multiple sensors was tested too, to study the effect of combined sensor streams on randomness. These combinations were accelerometer + gyroscope + magnetometer, gyroscope + magnetometer and magnetometer + quaternions. The data was combined in a similar manner; Acc_{x_1}, Gyro_{x_1}, Mag_{x_1}, Acc_{y_1}, Gyro_{y_1}, Mag_{y_1}, Acc_{z_1}, Gyro_{z_1}, Mag_{z_1}, Acc_{x_2}, Gyro_{x_2}, Mag_{x_2}, etc. The last step was to convert the data to binary values to be suitable for the NIST test suite. A list of all tested combinations can be found in Table 5.1.

For all NIST tests performed in this study the standard configuration was used, which means that all available statistical tests were performed. The data was formatted as ASCII's 0's and 1's and was tested in one sequence. The results of the tests on the sensor data are shown in Appendix A.1. The data of two driving trips were tested.

The results of the tests are consistent for all tested combinations. In Figure 5.2 the percentage of statistical tests passed for each tested combination is shown. For a combination to be called random it should pass at least seven NIST tests [36], which yields a 50% pass rate in Figure 5.2. We can conclude that none of the tested data from sensors and combinations of sensors are random. All tested configurations failed the following tests: Frequency, Block Frequency, Cumulative Sums, Runs, Longest Run, FFT (Fast Fourier Transform), Non Overlapping Template, Overlapping Template, Universal, Approximate Entropy and Serial. Since none of the combinations passed the Frequency tests, the Random Excursions- and Random Excursions Variant-test are not applicable either, therefore there are no results of these tests. As can be seen in Appendix A.1 only 13 out of 16 combinations passed two tests, the Rank and Linear Complexity tests, and thus none of the tested sequences pass the required seven or more NIST test. The Rank test checks the linear dependency of substrings to the input sequence [6]. The Linear complexity test determines if the "sequence is complex enough to be considered random" according to the length of a linear feedback shift register [6]. As none of the tested combinations passed more than 20% of the tests, we conclude that the gathered dataset is not random and thus as such not suitable for cryptographic usage.

Despite this, sensor data can still be valuable for authentication or cryptography purposes in smart logistics. With a randomization algorithm the sensor data can be randomized enough to pass the required seven or more NIST tests. To randomize the extracted sensor data we have come up with three algorithms. These algorithms are described in the next section. The results on the randomness introduced into the data by these algorithms is described in Section 6.

Table 5.1: The combinations of sensor data from the dataset used in the NIST test to determine the randomness of the dataset. Acc = accelerometer, Gyro = gyroscope, Mag = magnetometer, Q = quaternions. Gyro+Mag and Mag+Quaternions were only tested for the 07082020 subset.

Tested combinations dataset
Acc_x, Acc_y, Acc_z
$Gyro_x$, $Gyro_y$, $Gyro_z$
Mag_x, Mag_y, Mag_z
Q_w, Q_x, Q_y, Q_z
Yaw, Pitch, Roll
Heading
Acc, Gyro, Mag
Gyro, Mag
Mag, Quaternions

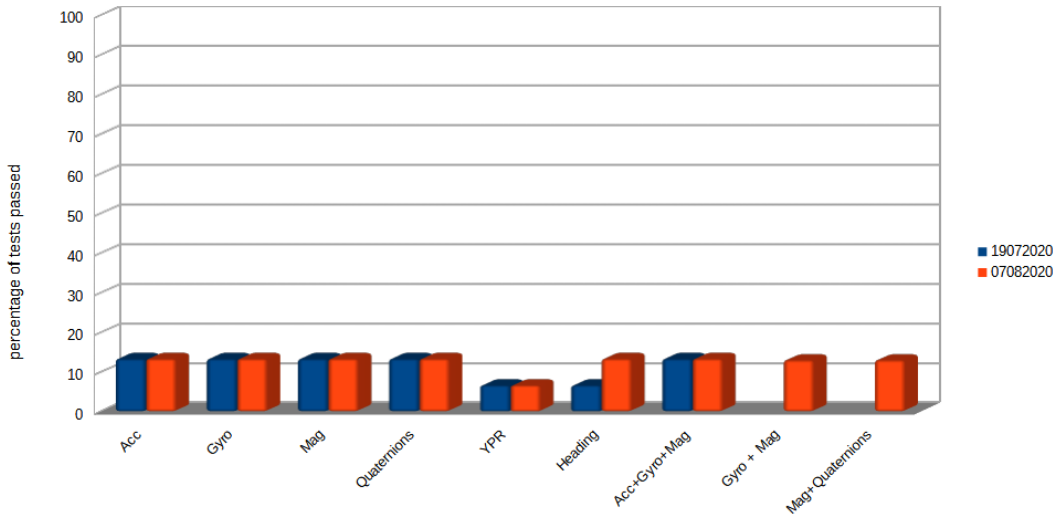


Figure 5.2: The results of the NIST randomness tests for the gathered dataset. On the y-axis the percentage of tests passed is shown for each tested sequence, for both the 19072020 part and 07082020 part of the dataset.

5.3 Data Randomization Algorithms

In this section the three randomization algorithms designed to randomize the data in the sensor dataset are discussed. The goal of the algorithms is to achieve a uniform distribution of 0's and 1's in the dataset, such that the tested sequences passes seven or more tests from the NIST test suite. Furthermore, the algorithm must be minimized in complexity, which is expressed as the number of computations, and in memory usage, while obtaining maximum randomness. All three algorithms were implemented in Matlab.
The unprocessed sensor data of Section 5.2 were generated in one sequence, which took ≈ 40 minutes. This means that the randomness results are based upon long term data gathering. In practice the Intelligent Product might not have the time to generate data for more than a few seconds before outputting a random sequence to be used in cryptography. Moreover, a constrained device is unlikely to have the required amount of memory available to store this data. Therefore, the algorithms must also provide a random sequence with a limited amount of sensor data. Consequently, Algorithm 3 will also be tested with 100 and 500 data samples (one and five seconds of data gathering, respectively), as the outcome of this algorithm depends on the number of datapoints inputted. Next, each of the three algorithms is described in detail.

5.3.1 Algorithm 1: bitwise XOR

The first algorithm performs a bitwise XOR operation on various combinations of data samples. In order to do so, the text files containing the data of one driving trip are loaded into Matlab and the first and last 30 seconds of data is removed. As described above, this is done to exclude the data from when the car is stationary. Next, the absolute value of all data is taken and in some occasions the decimal parts of the Yaw, Pitch and Roll and Heading data is extracted, to be used in the algorithm. Each datapoint is converted to a binary value, to be able to perform the bitwise XOR operation. Up until this point no data streams of (different) sensors have been combined. Next the XOR operation takes place, in various combinations. In Table 5.2 the different XOR combinations are given. The bitwise XOR has been performed in the order stated in this table. For example; Acc_x, Gyro_y, Mag_z means that first the Acc_x data was XORed with the Gyro_y data and the result was XORed with Mag_z. After the XOR operation, the result is saved to a text file, to be later used to test for its randomness with the NIST test suite. The pseudo-code in shown in Algorithm 1.

As discussed in the beginning of Section 5.3, real-time data randomization is a requirement for Intelligent Products, as long term data gathering is not feasible. Therefore, the algorithms should not only be tested on large datasets, but especially on smaller sets as well, to mimic real time processing. However, since the XOR is a bitwise operation and thus data is processed per bit, the output of Algorithm 1 will not change when processing the data in different sized chunks.

5.3.2 Algorithm 2: constant multiplication

The second algorithm aims to randomize the sensor data by multiplying it with a set of three decimal digits of the constants e or π. For both constants up to a trillion digits are known[2], therefore this algorithm does not have to reuse the digits for a considerable time. Even if the algorithm would randomize a data stream of one million datapoints, the constants e and π would last at least, without reusing digits, 10 million and 16 million times, respectively.
The basis for the second algorithm is the same as for the first algorithm. First, the text files containing the data of one driving trip are loaded into Matlab and the first and last 30 seconds of data are removed. Next the

[2] `http://www.numberworld.org/digits/E/`, `http://www.numberworld.org/digits/Pi/`, last accessed 10/12/2020.

Algorithm 1: Random Sequence generation with XORing of sensor data

```
Input: SensorData
Output: RandomSequence
for i > 3000 to i < length(SensorData)-3000) do
    DataToProcess ← SensorData
for i < length(DataToProcess) do
    AbsDataToProcess ← abs(DataToProcess)
for i < length(DataToProcess) do
    DataToProcess.Yaw ← extract_decimals(DataToProcess.Yaw)
    DataToProcess.Pitch ← extract_decimals(DataToProcess.Pitch)
    DataToProcess.Roll ← extract_decimals(DataToProcess.Roll)
    DataToProcess.Heading ← extract_decimals(DataToProcess.Heading)
for i < length(AbsDataToProcess) do
    BinaryData ← convert_to_binary(AbsDataToProcess)
for i < length(BinaryData) do
    XORResult_1 ← BinaryData_1 ⊕ BinaryData_2
    XORResult_2 ← XORResult_1 ⊕ BinaryData_3
RandomSequence ← XORResult_2
```

decimal parts of the Yaw, Pitch, Roll and Heading data are extracted, to be used in the algorithm and the first five million digits of both constants e and π are loaded into Matlab. For both constants their decimals are grouped per three decimals. This yields that each datapoint will be multiplied with a value ranging from 0 to 999. Next, the data is multiplied with the constants decimal values, from either e or π. In one data stream, each datapoint is multiplied with one group of three digits. For example, multiplication of combination Acc_x, Acc_y, Acc_z with constant e is as follows: $Acc_{x1} * e_{1-3}$, $Acc_{y1} * e_{4-6}$, $Acc_{z1} * e_{7-9}$, $Acc_{x2} * e_{10-12}$, $Acc_{y2} * e_{13-15}$, etc. The data stream combinations applied in Algorithm 2 are shown in Table 5.2. All of these combinations in the table have been multiplied with both e and π. The pseudo-code is shown in Algorithm 2. Similar to Algorithm 1, the outcome of Algorithm 2 is not dependent on the size of data blocks that are multiplied. Multiplication happens per datapoint, therefore, multiplying 100, 500 or all samples with a constant is all equivalent.

After the multiplication the absolute value of the result is taken and the result is converted to binary. The binary stream is saved to a text file, to be tested later on with the NIST test suite.

5.3.3 Algorithm 3: Fourier Transform

The third algorithm is an extension of the first algorithm. This algorithm combines frequency analysis with a bitwise logical operation. First, the Fast Fourier Transform (FFT) is applied to the data, after which the XOR operation from algorithm one is performed. The basis of this algorithm is, again, equal to the other two algorithms. First, the text files containing the data of one driving trip are loaded into Matlab. Subsequently, the first and last 30 seconds of data is removed and the decimal parts of the Yaw, Pitch, Roll and Heading data are extracted. Next the FFT of each data stream is taken separately. This means that no sensors are combined and that the multiple data streams from one sensor are kept separated as well. The result of a FFT consists of a real and an imaginary part, both decimal numbers. The algorithm is only based on the real number, so the imaginary number is discarded. Besides this, the integer part is discarded as well. Even though the real numbers can be negative, there is no need to take the absolute value if we only keep the decimal part of the

Table 5.2: The sensor combinations applied in Algorithm 1, Algorithm 2 and Algorithm 3. Acc = accelerometer, Gyro = gyroscope, Mag = magnetometer, Q = quaternions.

Algorithm 1	**Algorithm 2**	**Algorithm 3**
Acc_x, Acc_y, Acc_z	Acc_x, Acc_y, Acc_z	Acc_x, Acc_y, Acc_z
$Gyro_x$, $Gyro_y$, $Gyro_z$	Acc_x	$Gyro_x$, $Gyro_y$, $Gyro_z$
Mag_x, Mag_y, Mag_z	$Gyro_x$, $Gyro_y$, $Gyro_z$	Mag_x, Mag_y, Mag_z
Acc_x, $Gyro_y$, Mag_z	$Gyro_x$	Acc_x, $Gyro_y$, Mag_z
Q_w, Q_x, Q_y, Q_z	Q_w, Q_x, Q_y, Q_z	Q_w, Q_x, Q_y, Q_z
Yaw, Pitch, Roll	Q_w	Yaw, Pitch, Roll, Heading
Yaw, Pitch, Roll, Heading	Yaw, Pitch, Roll	Q_w, Yaw, Heading,
Q_w, Yaw, Heading,	Yaw, Pitch, Roll, Heading	Q_w, Yaw, Heading, Acc_x, $Gyro_y$, Mag_z
Q_w, Yaw, Heading, Acc_x, $Gyro_y$, Mag_z	Heading	

Algorithm 2: Random Sequence generation by multiplication with π

```
Input: SensorData
Output: RandomSequence
for i > 3000 to i < length(SensorData)-3000) do
    DataToProcess ← SensorData
PiDecimals ← load_file(DecimalsPi)
for i < (length(PiDecimals) - 3) do
    GroupedPiDecimals ← ((PiDecimals(i) x 100) + (PiDecimals(i + 1) x 10) + PiDecimals(i + 2))
    i = i + 3
for i < length(DataToProcess) do
    DataToProcess.YPR ← extract_decimals(DataToProcess.YPR)
    DataToProcess.Heading ← extract_decimals(DataToProcess.Heading)
for i < length(DataToProcess) do
    MultipliedData ← DataToProcess(i) x GroupedPiDecimals(i)
for i < length(MultipliedData) do
    AbsMultipliedData ← abs(MultipliedData)
RandomSequence ← AbsMultipliedData
```

number.

The next step is to convert every data stream to binary values, to be able to perform the bitwise logical XOR operation. Similar to Algorithm 1, the XOR operation is applied to various data combinations. These combinations are shown in Table 5.2. The final step is to save the data to a text file, applicable for the NIST test suite. The pseudo-code in shown in Algorithm 3.

Algorithm 3: Random Sequence generation with XORing of FFT processed sensor data

```
Input: SensorData
Output: RandomSequence
for i > 3000 to i < length(SensorData)-3000) do
    DataToProcess ← SensorData
for i < length(DataToProcess) do
    AbsDataToProcess ← abs(DataToProcess)
for i < length(DataToProcess) do
    DataToProcess.YPR ← extract_decimals(DataToProcess.YPR)
    DataToProcess.Heading ← extract_decimals(DataToProcess.Heading)
for i < length(DataToProcess) do
    FFTData ← take_fft(DataToProcess)
    RealPartFFTData ← FFTData.real
    AbsRealFFTData ← abs(RealPartFFTData)
for i < length(AbsRealFFTData) do
    DecimalsAbsRealFFTData ← extract_decimals(AbsRealFFTData)
for i < length(AbsDataToProcess) do
    BinaryFFTData ← convert_to_binary(DecimalsAbsRealFFTData)
for i < length(BinaryData) do
    XORResult_1 ← BinaryFFTData_1 ⊕ BinaryFFTData_2
    XORResult_2 ← XORResult_1 ⊕ BinaryFFTData_3
RandomSequence ← XORResult_2
```

6 Results & Discussion

The previous section discusses the design of three algorithms to randomize sensor data. For each of these three algorithms tests were executed in which the data from the dataset, as described in Section 5.1, was fed into the algorithm. The results were tested for their randomness with the NIST test suite and the aim was to have randomized the data. As a comparison for how more random or less random the data has become, we use the randomness tests on the dataset from Section 5.2. This chapter discusses the results of the randomness tests performed with the NIST test suite.

6.1 Randomness results

For Algorithm 1, based on XORing data-points, most individual tests are passed and consequently most tested sequences (combinations) are considered random. This is visualized in Figure 6.1. For both the 19072020 and 070820202 data subsets the results are similar. The results per test for Algorithm 1 are shown in Appendix B.1. Compared to the raw data, more sequences, five out of nine for both subset, were tested with the Random

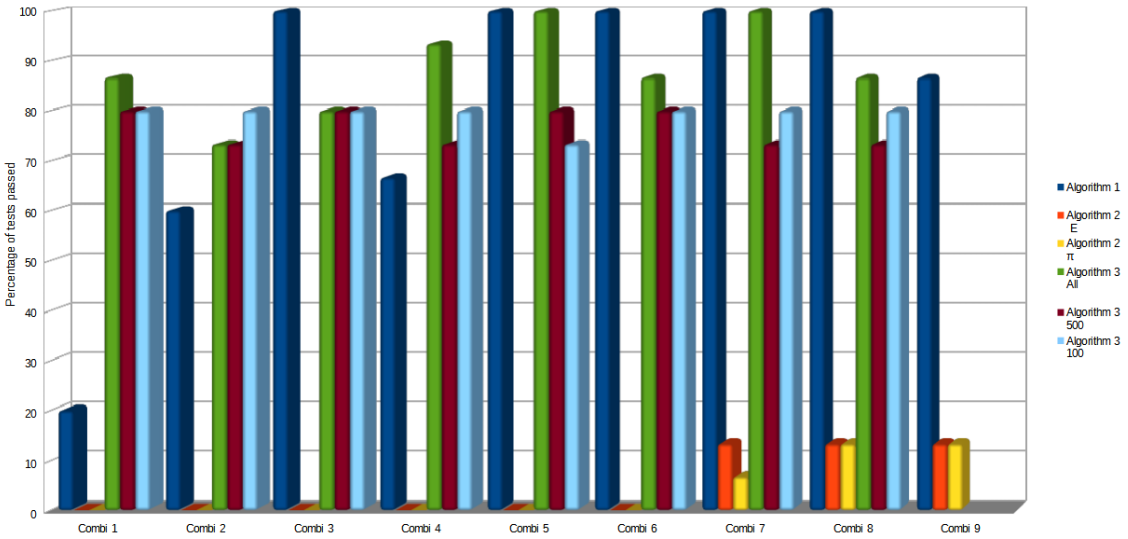
Excursions and Random Excursions Variants tests. This is related to the fact that more combinations for Algorithm 1 passed the Frequency test. Each time these two tests were performed, the tests passed. For both subsets the sequence Acc_x, Acc_y, Acc_z (for Algorithm 1; Combi 1 in Figure 6.1) is not randomized by Algorithm 1. A plausible explanation for this could be that the accelerometer values on the z-axis do not vary significantly. Most absolute values are between 15000 and 17999. This means that for all z-axis values the first digit is a 1 and the second digit is likely a 5, 6 or 7. This leads to repetition in the sequence, which instantly means the sequence is not random. All other tested combinations passed more than 50% of the tests, some with little margin, and thus are random.

For Algorithm 2 the tests were only performed on subset 07082020, for both multiplication by constants e and π. Figure 6.1a shows that the majority (six out of nine) of the tested combinations for both constants passed none of the statistical tests. The remaining three combinations passed < 20% of the tests. Subsequently, none of the sequences generated by Algorithm 2 are random and therefore we conclude that Algorithm 2 is a poor randomization algorithm. In Appendix C.1 the results per test for each combination are shown. For three out of nine tests the Rank and Linear Complexity tests passed, for both e or π.
When comparing Figure 5.2 and Figure 6.1a we can see that even less tests were passed after manipulation by Algorithm 2, compared to the raw data as tested in Section 5.2. From this we can conclude that multiplying with the digits of the constant e and π adds predictability and pattern to the data. This can be explained by the fact that natural constants are not truly random numbers [5].

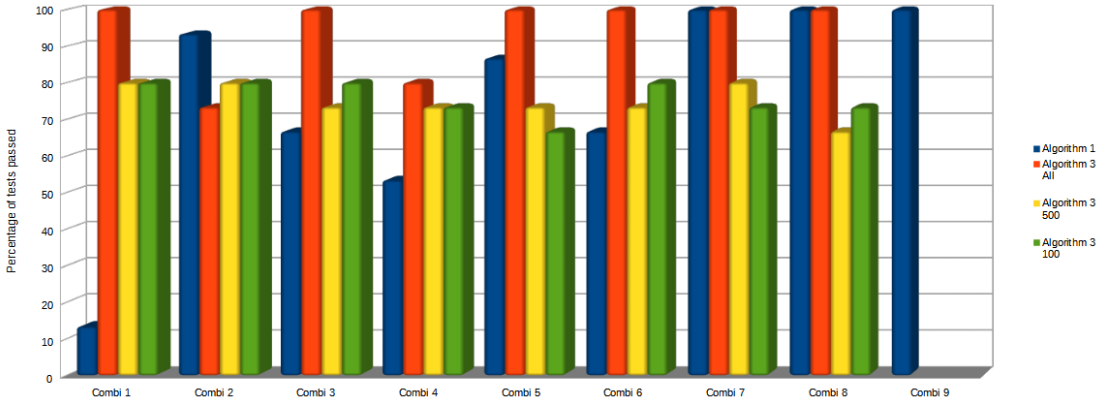
For the tests of Algorithm 3 the same sequences were used as for Algorithm 1, except for Algorithm 3 the Yaw⊕Pitch⊕Roll was not calculated. Therefore, in Figure 6.1 Combi 9 does not contain Algorithm 3 results. Algorithm 3 has been applied on the full dataset and a subset of 5 seconds of data (500 data-points) and 1 second of data (100 data-points). The exact test results are shown in Appendix D.1, D.2 and D.3, respectively. It can be seen from the graphs in Figure 6.1 that all tested sequences are random by a large margin. Compared to Algorithm 1, Algorithm 3 does randomize sequence Acc_x, Acc_y, Acc_z. For this particular sequence we can conclude that the Fourier Transform adds additional randomness to the sequence with respect to the XOR operation.

Compared to Algorithm 1 and Algorithm 2, the Random Excursions and Random Excursions Variants tests are executed often for Algorithm 3. Each time these tests were executed, the tests passed. From the graphs in Figure 6.1 it can be concluded that Algorithm 2 is the worst performing algorithm regarding randomization and should not be used for authentication purposes. The randomization abilities of Algorithm 1 and Algorithm 3 are fairly close to each other. The graphs show that several combinations Algorithm 1 have a higher percentage of passed tests than 'Algorithm 3 - 100' and 'Algorithm 3 - 500'. This is mostly because the Random Excursions and Random Excursions Variant tests are not performed for the smaller datasets of Algorithm 3, likely due to the smaller sequence length, whereas these tests are often performed for Algorithm 1. Overall, all three runs of Algorithm 3 give more consistant results than Algorithm 1, which is important for authentication purposes. When the algorithm sometimes does not succeed at randomization, there is a chance that the sequence used for authentication is not random. This makes the system vulnerable to attacks, as further discussed in Section 6.4.

As discussed before, IoT devices operate in real-time and therefore, the randomization algorithm must be able to generate a random sequence within a few seconds. To analyse This effect of short term data gathering on the randomization, the most succesfull algorithm, Algorithm 3, is also tested with 100 and 500 datapoints. In the next section we will discuss these results.



(a) Randomness results for dataset 07082020.



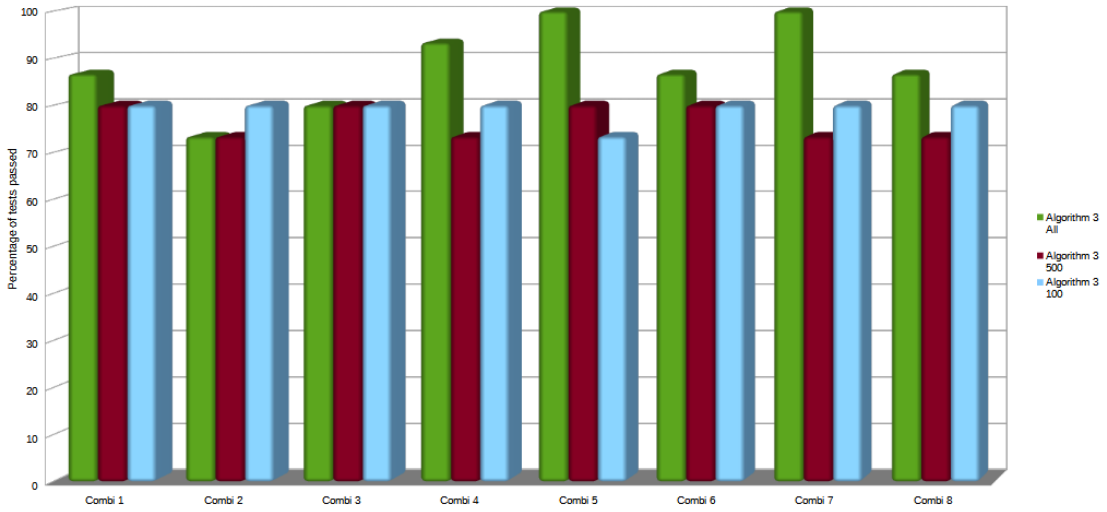
(b) Randomness results for dataset 19072020. Algorithm 2 was not tested for this dataset.

Figure 6.1: The results of the NIST randomness tests. On the y-axis the percentage of tests passed is shown for each tested sequence per algorithm. Algorithm 3 was only tested on 8 combinations.

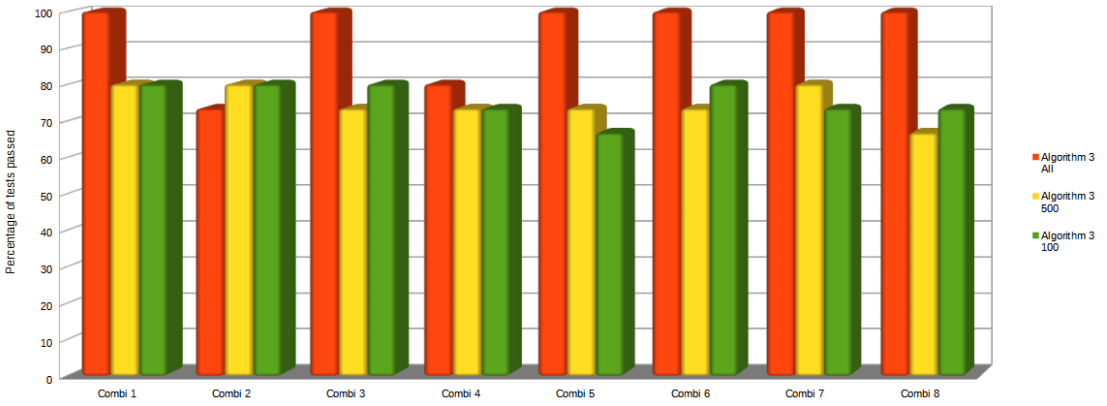
6.1.1 Results short term data gathering

Algorithm 3 was not only applied to the full dataset, but also to a subset of 100 datapoints and 500 datapoints. This is equivalent to 1 second and 5 seconds of data gathering, respectively, in the case of our dataset. The comparison of randomness between the three tested sets of Algorithm 3 is shown in the graphs in Figure 6.2. The graphs clearly show the smaller datasets pass less statistical tests of the NIST test suite. However, as can be seen from the detailed test results in Appendix D, the tests Random Excursions and Random Excursions Variant are not performed for the 100 and 500 datapoints, which explains the smaller percentage of tests passed in Figure 6.2. The results of all three tested variants of Algorithm 3 are consistant and sufficiently random. This means that Algorithm 3 can also randomize short data streams that are gathered in one second.

Therefore, Algorithm 3 is expected to work in real-time Intelligent Products in Smart Logistics.



(a) Randomness results for dataset 07082020.



(b) Randomness results for dataset 19072020.

Figure 6.2: The results of the NIST randomness tests for Algoritm 3 for all datapoints and a subset of 100 and 500 datapoints. On the y-axis the percentage of tests passed is shown for each tested sequence

6.2 Complexity

Besides the ability of the algorithms to randomize data, their complexity is also of importance when evaluating their usefulness in smart logistics. Intelligent products used in smart logistics are constrained devices, regarding their computational and memory resources. This means the algorithm used to randomize the sensor data cannot be too powerful and cannot use unconstrained amounts of memory. In this section we will evaluate and compare the complexity of the designed algorithms. In this evaluation the amount of function calls is taken into account, as well as the time complexity of the functions and the memory-storage and -usage

required.

Algorithm 1 and Algorithm 3 are related to each other, as Algorithm 3 in an extension of Algorithm 1. Algorithm 1 only uses a XOR operation on each data-point, after conversion to binary, Algorithm 3, on the other hand, takes the fourier transform of each sensor stream prior to the XOR task. The Fourier transform used is the build-in Matlab function, FFT, defined as follows: $Y(k) = \sum_{j=1}^{n} X(j)W_n(j-k)(k-1)$, where $W_n = e^{(-2\pi\imath)/n}$. The time complexity of the FFT function is $O(NlogN)$, where N is the number of data-points. The time complexity of a bitwise XOR is $O(N)$, where N is the length of the binary sequence. The graph in Figure 6.3 shows that the computational steps for implementations of $O(NlogN)$ complexity (FFT) increase more with increasing input size N, compared to implementations with $O(N)$ complexity. This means that for Algorithm 3 the number of computations increases more for increasing input size than for Algorithm 1. Moreover, after the FFT function is applied in Algorithm 3, the real and imaginary part of the results must be separated and subsequently the mantissa and integer part. Therefore, Algorithm 3 does not only use more complex functions than Algorithm 1, but also more function calls in general. This makes Algorithm 3 relatively heavy computational wise, compared to Algorithm 1.

The amount of preprocessing and post processing done in Algorithm 2 is similar to Algorithm 1. However, Algorithm 2 uses a multiplication operation, which, depending on the implemenation, can have a time complexity as bad as $O(N^2)$, where N is the number of digits of the numbers to be multiplied. For example, when multiplying 100 times 200, N is 3. This means with n datapoints for input in Algorithm 2, the time complexity is $O(n \cdot N^2)$. The comparison of the complexities $O(N^2)$, $O(NlogN)$ and $O(N)$ is shown in Figure 6.3.

Because Algorithm 3 uses the Fourier transform, where the data is used in a summation, more memory usase is required compared to Algorithm 1, which it is an extension of. Algorithm 1 does not need noteworthy memory usage. Algorithm 2 does not need a lot of memory usage during computation, however, it does require storage of the digits of a natural constant (e or π in our case). For the tests combinations of Table 5.2 this varies from 0.8 MB to 3.0 MB.

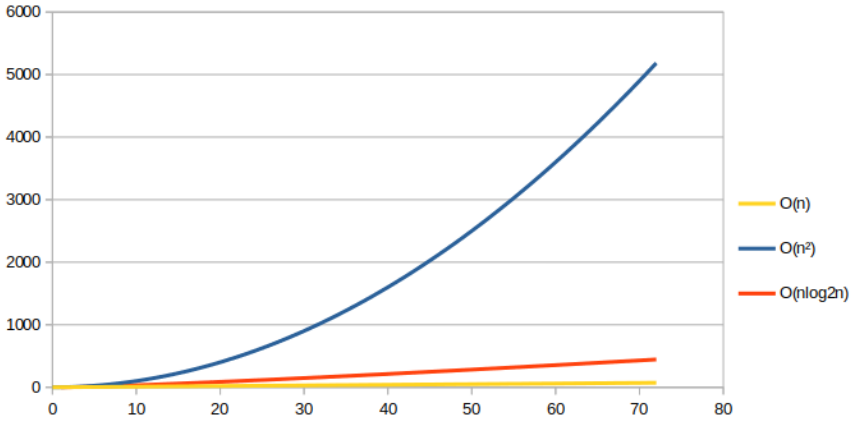


Figure 6.3: The complexity of the algorithms, for increasing n (x-axis).

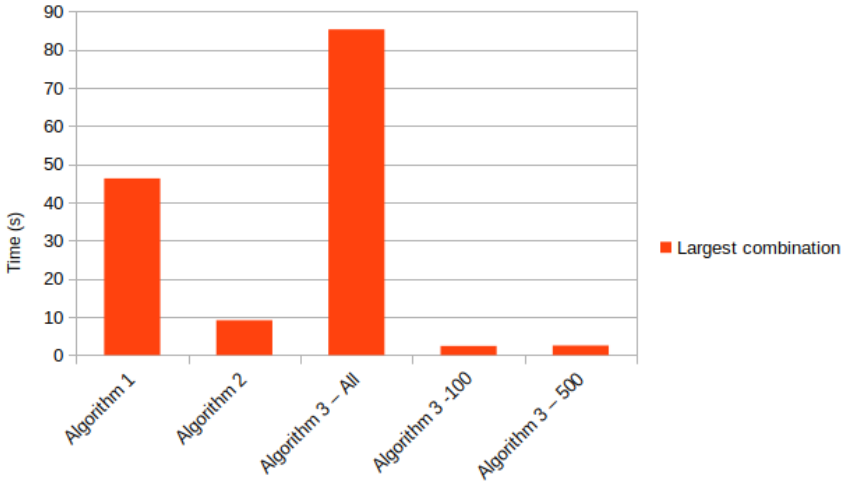
6.3 TIME CONSUMPTION

Another important aspect of the algorithms is timing, i.e. how long the algorithm takes to generate a random sequence. This is important as the user does not want to wait too long before, for example, an encrypted data transmission can be established. Ideally we would want this to happen in real-time, so it becomes possible to do real-time data gathering and monitoring with the Intelligent Products. The time consumption of the three algorithms was determined based on calculating random sequences for all tested combinations (see Table 5.2) and for the largest combination (most sensor streams combined). The timing results are shown in Figure 6.4, where each time result in seconds is the average of ten measurements. The results are grouped; Figure 6.4b shows the time consumption for calculating random sequences for all tested combinations and Figure 6.4a shows the time consumption for only generating the largest sequence. For the timing of Algorithm 1, Algorithm 2 and 'Algorithm 3 - All', 260k datapoints are used. For 'Algorithm 3 - 100' and 'Algorithm 3 - 500', respectively, 100 and 500 datapoints are used. The measurements were done on a laptop with an AMD Ryzen 7 4700U processor running Windows 10, with a clockspeed of 2.0 GHz, 8 cores and 15.4 GB RAM.

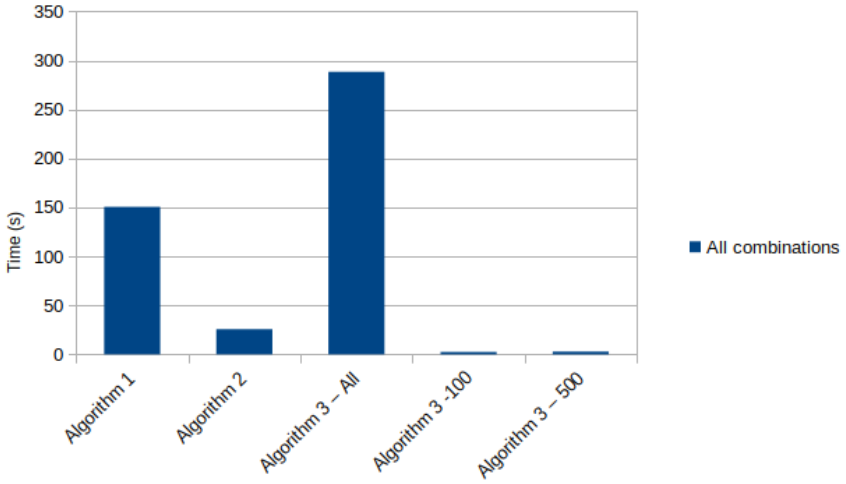
Both subfigures in Figure 6.4 show that Algorithm 2 is significantly faster than Algorithm 1 and Algorithm 3, when using the same amount of datapoints as input. The difference in time consumption for using π or e for Algorithm 2 is negligible, therefore these results are combined in one bar in the graph.
It is unexpected that Algorithm 2 is significantly faster, as the time complexity analysis of the previous section shows that Algorithm 2 is significantly more complex than the other algorithms. A reason for this could be the way in which Matlab implements multiplication and the functions *bitxor()* and *fft()*. The implementation of *bitxor()* and *fft()* could be suboptimal, causing Algorithm 1 and Algorithm 3 to take significatly longer than Algorithm 2.

Since Algorithm 1 and Algorithm 3 are related to each other, comparing their timings is more interesting. For calculating all tested sequences as well as only the largest sequence 'Algorithm 3 - All' takes about twice as long as Algorithm 1. This is in line with the complexity analysis of Section 6.2, where it was shown that the FFT implementation does not only have a larger time complexity than bitwise XOR but Algorithm 3 also takes more computational steps altogether.

With an average of 84 seconds for the longest combination and 256 seconds for all combinations, the time consumption of 'Algorithm 3 - All' is too long for real time applications. Therefore, it was also tested to generate a random sequence with Algorithm 3 with less datapoints, namely 100 and 500 datapoints. This is respectively 1 second and 5 seconds of data gathering in the used dataset. The computation times of 'Algorithm 3 - 100' and 'Algorithm 3 - 500' are much more promising for use in real time Intelligent Products in the logistics sector. For both 500 and 100 datapoints and both calculating for all tested sequences or only the largest tested sequence, the time consumption was only 3 seconds. This is acceptable for real time applications.



(a) Time consumption of the algorithms computing the largest tested sequence.



(b) Time consumption of the algorithms computing all tested sequence at once.

Figure 6.4: Time consumption of the algorithms for all tested sequences and only the largest tested sequence. The computations of Algorithm 1, 2 and '3 - All' were done on 260k datapoints. Time consumption of 'Algorithm 3 - 100' was calculated with 100 datapoints and that of 'Algorithm 3 - 500' was calculated with 500 datapoints.

6.4 Security analysis

The Intelligent Products in smart logistics are constantly in transportation and thus accessible to an adversary who aims to tamper with the device and possibly access the data. Therefore, it is important that the device is

protected from physical attacks [26, 45]. Access to the data or randomization algorithms is a security concern, as one could recreate the random sequence with this data. One way to read out any data is by a physical attack called probing. In a probing attack the adversary aims to directly read out the data signal by probing onto the data lines in the device [26]. As probing requires expensive, specialized tools and a laboratory setting, it is unlikely that such an attack will take place.

Another possibility to access the sensor data, is by trying to recreate the same data by using the exact same sensors and placing them in the same location. However, due to the manufacturing variations in the sensors, the measured values will not be exactly the same, which will influence the outcome of the randomization algorithms and thus the generated key. As encryption keys need to be a perfect match, recreating the sensor data to obtain an encryption key is not a security concern.

If an attacker would be able to get access to the sensor data, without having knowledge of the randomization algorithm, it may still be possible to successfully predict the random output sequence. The likelihood for this to happen depends on the Hamming Distance, which is the number of bits that differ in two sequences of equal length [9]. This means that with a sufficiently large Hamming Distance, the sequences are uniformly distributed, meaning they have an equal probability of occurring. As the Hamming Distance has not been calculated in this research, it is not possible to state how likely it is to be able to predict the output sequence when obtaining the sensor data.

7 Conclusion

As securing data becomes more important with the rise of smart logistics, the need for generating random sequences on IoT devices for authentication and cryptographic purposes increases. The limitations of these constrained devices make it more challenging to develop a mechanism to generate random sequences. We have shown that a Sensor PUF can generate random sequences by using combinations of sensors already present on the Intelligent Product. As the raw sensor data itself is not random, an algorithm is needed to randomize the data. Multiplication of unprocessed sensor data by a constant does not result in random data. Applying an XOR operation on combinations of sensor streams does randomize the data. The most random sequences, however, are created by applying the FFT before the XOR operation, as is done with Algorithm 3. We have shown that this algorithm works well for short term data gathering on a non resource constrained device, as it can randomize one second of sensor data, sampled at 100Hz. The computational costs of Algorithm 3 are limited, making this algorithm suitable for use on constrained IoT devices in smart logistics.

This research did not investigate the actual time consumption of the algorithms on a resource constrained device. As the computational power is likely to be less than the processor used in this study for the timing analysis, it is necessary to investigate further what the time consumption on a resource constrained device will be. In case the randomization takes longer on a constrained device, it is worth examining if even less datapoints can be used to generate a random sequence.

References

[1] Rifaqat Ali, Arup Kumar Pal, Saru Kumari, Marimuthu Karuppiah, and Mauro Conti. A secure user authentication and key-agreement scheme using wireless sensor networks for agriculture monitoring. *Future Generation Computer Systems*, 84:200–215, 2018.

[2] Muhammad Naveed Aman, Kee Chaing Chua, and Biplab Sikdar. Mutual authentication in iot systems using physical unclonable functions. *IEEE Internet of Things Journal*, 4(5):1327–1340, 2017.

[3] S Anandhi, R Anitha, and Venkatasamy Sureshkumar. Iot enabled rfid authentication and secure object tracking system for smart logistics. *Wireless Personal Communications*, 104(2):543–560, 2019.

[4] Aydin Aysu, Nahid Farhady Ghalaty, Zane Franklin, Moein Pahlavan Yali, and Patrick Schaumont. Digital fingerprints for low-cost platforms using mems sensors. In *Proceedings of the Workshop on Embedded Systems Security*, page 2. ACM, 2013.

[5] David H Bailey and Richard E Crandall. On the random character of fundamental constant expansions. *Experimental Mathematics*, 10(2):175–190, 2001.

[6] Lawrence E Bassham III, Andrew L Rukhin, Juan Soto, James R Nechvatal, Miles E Smid, Elaine B Barker, Stefan D Leigh, Mark Levenson, Mark Vangel, David L Banks, et al. *Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications*. National Institute of Standards & Technology, 2010.

[7] Daniel Bichler, Guido Stromberg, Mario Huemer, and Manuel Löw. Key generation based on acceleration data of shaking processes. In *International Conference on Ubiquitous Computing*, pages 304–317. Springer, 2007.

[8] Christoph Böhm and Maximilian Hofer. *Physical unclonable functions in theory and practice*. Springer Science & Business Media, 2012.

[9] Christoph Böhm and Maximilian Hofer. Testing and specification of pufs. In *Physical Unclonable Functions in Theory and Practice*, pages 69–86. Springer, 2013.

[10] Heike Busch, Miroslava Sotáková, Stefan Katzenbeisser, and Radu Sion. The puf promise. In *International Conference on Trust and Trustworthy Computing*, pages 290–297. Springer, 2010.

[11] Yuan Cao, Le Zhang, and Chip-Hong Chang. Using image sensor puf as root of trust for birthmarking of perceptual image hash. In *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*, pages 1–6. IEEE, 2016.

[12] Baibhab Chatterjee, Debayan Das, Shovan Maity, and Shreyas Sen. Rf-puf: Enhancing iot security through authentication of wireless nodes using in-situ machine learning. *IEEE Internet of Things Journal*, 6(1):388–398, 2018.

[13] Shuai Chen, Bing Li, and Yuan Cao. Intrinsic physical unclonable function (puf) sensors in commodity devices. *Sensors*, 19(11):2428, 2019.

[14] Cheun Ngen Chong, Dan Jiang, Jiagang Zhang, and Long Guo. Anti-counterfeiting with a random pattern. In *2008 Second International Conference on Emerging Security Information, Systems and Technologies*, pages 146–153. IEEE, 2008.

[15] Mehiar Dabbagh and Ammar Rayes. Internet of things security and privacy. In *Internet of Things from hype to reality*, pages 211–238. Springer, 2019.

[16] Gerald DeJean and Darko Kirovski. Rf-dna: Radio-frequency certificates of authenticity. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 346–363. Springer, 2007.

[17] Kazuhide Fukushima, Seira Hidano, and Shinsaku Kiyomoto. Sensor-based wearable puf. In *Secrypt*, pages 207–214, 2016.

[18] Qi Gao, Junwei Zhang, Jianfeng Ma, Chao Yang, Jingjing Guo, and Yinbin Miao. Lip-pa: A logistics information privacy protection scheme with position and attribute-based access control on mobile devices. *Wireless Communications and Mobile Computing*, 2018, 2018.

[19] Blaise Gassend, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas. Silicon physical random functions. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 148–160. ACM, 2002.

[20] Vaggelis Giannikas and Duncan McFarlane. Product intelligence in intermodal transportation: The dynamic routing problem. In *Dynamics in Logistics*, pages 59–69. Springer, 2013.

[21] Robert M Gray. *Entropy and information theory*. Springer Science & Business Media, 2011.

[22] Jorge Guajardo, Sandeep S Kumar, Geert-Jan Schrijen, and Pim Tuyls. Fpga intrinsic pufs and their use for ip protection. In *International workshop on cryptographic hardware and embedded systems*, pages 63–80. Springer, 2007.

[23] Jorge Guajardo, Boris Škorić, Pim Tuyls, Sandeep S Kumar, Thijs Bel, Antoon HM Blom, and Geert-Jan Schrijen. Anti-counterfeiting, key distribution, and key storage in an ambient world via physical unclonable functions. *Information Systems Frontiers*, 11(1):19–41, 2009.

[24] Ryan Helinski, Dhruva Acharyya, and Jim Plusquellic. A physical unclonable function defined using power distribution system equivalent resistance variations. In *2009 46th ACM/IEEE Design Automation Conference*, pages 676–681. IEEE, 2009.

[25] Ronald S Indeck and Marcel W Muller. Method and apparatus for fingerprinting magnetic media, November 15 1994. US Patent 5,365,586.

[26] Oliver Kömmerling and Markus G Kuhn. Design principles for tamper-resistant smartcard processors. *Smartcard*, 99:9–20, 1999.

[27] ST Choden Konigsmark, Leslie K Hwang, Deming Chen, and Martin DF Wong. Cnpuf: A carbon nanotube-based physically unclonable function for secure low-energy hardware design. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 73–78. IEEE, 2014.

[28] Sandeep S Kumar, Jorge Guajardo, Roel Maes, Geert-Jan Schrijen, and Pim Tuyls. The butterfly puf protecting ip on every fpga. In *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 67–70. IEEE, 2008.

[29] Klaus Kursawe, Ahmad-Reza Sadeghi, Dries Schellekens, Boris Skoric, and Pim Tuyls. Reconfigurable physical unclonable functions-enabling technology for tamper-resistant storage. In *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 22–29. IEEE, 2009.

[30] Jae W Lee, Daihyun Lim, Blaise Gassend, G Edward Suh, Marten Van Dijk, and Srinivas Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No. 04CH37525)*, pages 176–179. IEEE, 2004.

[31] Seokgi Lee, Yuncheol Kang, and Vittaldas V Prabhu. Smart logistics: distributed control of green crowdsourced parcel services. *International Journal of Production Research*, 54(23):6956–6968, 2016.

[32] Keith Lofstrom, W Robert Daasch, and Donald Taylor. Ic identification circuit using device mismatch. In *2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No. 00CH37056)*, pages 372–373. IEEE, 2000.

[33] Hua Ma, Yansong Gao, Omid Kavehei, and Damith C Ranasinghe. A puf sensor: Securing physical measurements. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 648–653. IEEE, 2017.

[34] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. Intrinsic pufs from flip-flops on reconfigurable devices. In *3rd Benelux workshop on information and system security (WISSec 2008)*, volume 17, page 2008, 2008.

[35] Roel Maes and Ingrid Verbauwhede. *Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions*, pages 3–37. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[36] Kinga Marton and Alin Suciu. On the interpretation of results from the nist statistical test suite. *Science and Technology*, 18(1):18–32, 2015.

[37] Rene Mayrhofer and Hans Gellersen. Shake well before use: Authentication based on accelerometer data. In *International Conference on Pervasive Computing*, pages 144–161. Springer, 2007.

[38] Thomas McGrath, Ibrahim E Bagci, Zhiming M Wang, Utz Roedig, and Robert J Young. A puf taxonomy. *Applied Physics Reviews*, 6(1):011303, 2019.

[39] Subhas Chandra Mukhopadhyay and Nagender K Suryadevara. Internet of things: Challenges and opportunities. In *Internet of Things*, pages 1–17. Springer, 2014.

[40] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical one-way functions. *Science*, 297(5589):2026–2030, 2002.

[41] Pawani Porambage, Corinna Schmitt, Pardeep Kumar, Andrei Gurtov, and Mika Ylianttila. Two-phase authentication protocol for wireless sensor networks in distributed iot applications. In *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2728–2733. IEEE, 2014.

[42] Ammar Rayes and Samer Salam. Internet of things from hype to reality. *Springer*, 2017.

[43] Ammar Rayes and Samer Salam. Iot vertical markets and connected ecosystems. In *Internet of Things From Hype to Reality*, pages 239–268. Springer, 2019.

[44] Kurt Rosenfeld, Efstratios Gavas, and Ramesh Karri. Sensor physical unclonable functions. In *2010 IEEE international symposium on hardware-oriented security and trust (HOST)*, pages 112–117. IEEE, 2010.

[45] Debapriya Basu Roy and Debdeep Mukhopadhyay. *Security of Crypto IP Core: Issues and Countermeasures*, pages 67–114. Springer International Publishing, Cham, 2017.

[46] Ulrich Rührmair, Christian Jaeger, Christian Hilgers, Michael Algasinger, György Csaba, and Martin Stutzmann. Security applications of diodes with unique current-voltage characteristics. In *International Conference on Financial Cryptography and Data Security*, pages 328–335. Springer, 2010.

[47] Jian Shen, Shaohua Chang, Jun Shen, Qi Liu, and Xingming Sun. A lightweight multi-layer authentication protocol for wireless body area networks. *Future generation computer systems*, 78:956–963, 2018.

[48] Jack Tang, Ramesh Karri, and Jeyavijayan Rajendran. Securing pressure measurements using sensorpufs. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1330–1333. IEEE, 2016.

[49] Pim Tuyls, Geert-Jan Schrijen, Boris Škorić, Jan Van Geloven, Nynke Verhaegh, and Rob Wolters. Read-proof hardware from protective coatings. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 369–383. Springer, 2006.

[50] Pim Tuyls and Boris Škorić. *Secret Key Generation from Classical Physics: Physical Uncloneable Functions*, pages 421–447. Springer Netherlands, Dordrecht, 2006.

[51] UNECE. Terminology on combined transport. *New York and Geneva: United Nations Economic Commission for Europe*, 2001.

[52] Thierry Verduijn and Babiche van de Loo. Intelligent logistics concepts: Improving your supply chain with collaboration and ict. In *Intelligent Logistics Concepts*, chapter 1, pages 1–7. Eburon Publishers, Delft, 2003.

[53] Pandi Vijayakumar, Victor Chang, L Jegatha Deborah, Balamurugan Balusamy, and PG Shynu. Computationally efficient privacy preserving anonymous mutual and batch authentication schemes for vehicular ad hoc networks. *Future generation computer systems*, 78:943–955, 2018.

[54] Serge Vrijaldenhoven et al. Acoustical physical uncloneable functions. *Philips internal publication PR-TN-2004-300300*, 2005.

[55] Lingxiao Wei, Chaosheng Song, Yannan Liu, Jie Zhang, Feng Yuan, and Qiang Xu. Boardpuf: Physical unclonable functions for printed circuit board authentication. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 152–158. IEEE Press, 2015.

[56] Yida Yao, MyungBo Kim, Jianmin Li, Igor L Markov, and Farinaz Koushanfar. Clockpuf: Physical unclonable functions based on clock networks. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 422–427. EDA Consortium, 2013.

A NIST RANDOMNESS RESULTS: RAW DATA

Table A.1: Results of randomness for raw data. The percentages depict percentage of sub tests passed. ✓ represents the NIST test is passed, while ✗ represents the NIST test is failed. A – means the NIST test suite gives no result. On average, the number of bits input to NIST test suite are 800000. The combinations Gyro+Mag and Mag+Quaternions are only tested for the 07082020 dataset.

Dataset	Data stream	Frequency	Block Frequency	Cumulative Sums	Runs	Longest Run	Rank	FFT	Non Overlapping	Overlapping	Universal	Approx. Entropy	Random Excursions	Ramd. Excu. Variant	Serial	Linear Complexity	Random
19072020	Acc	✗	✗	✗	✗	✗	✓	✗	3%	✗	✗	✗	–	–	✗	✓	✗
	Gyro	✗	✗	✗	✗	✗	✓	✗	5%	✗	✗	✗	–	–	✗	✓	✗
	Mag	✗	✗	✗	✗	✗	✓	✗	0%	✗	✗	✗	–	–	✗	✓	✗
	Quart	✗	✗	✗	✗	✗	✓	✗	11%	✗	✗	✗	–	–	50%	✓	✗
	YPR	✗	✗	✗	✗	✗	✗	✗	0%	✗	✗	✗	–	–	✗	✓	✗
	Heading	✗	✗	✗	✗	✗	✗	✗	0%	✗	✗	✗	–	–	✗	✓	✗
	Acc+Gyro+Mag	✗	✗	✗	✗	✗	✓	✗	0%	✗	✗	✗	–	–	✗	✓	✗
07082020	Acc	✗	✗	✗	✗	✗	✓	✗	4%	✗	✗	✗	–	–	✗	✓	✗
	Gyro	✗	✗	✗	✗	✗	✓	✗	5%	✗	✗	✗	–	–	✗	✓	✗
	Mag	✗	✗	✗	✗	✗	✓	✗	0%	✗	✗	✗	–	–	✗	✓	✗
	Quart	✗	✗	✗	✗	✗	✓	✗	6%	✗	✗	✗	–	–	✗	✓	✗
	YPR	✗	✗	✗	✗	✗	✗	✗	0%	✗	✗	✗	–	–	✗	✓	✗
	Heading	✗	✗	✗	✗	✗	✓	✗	7%	✗	✗	✗	–	–	✗	✓	✗
	Acc+Gyro+Mag	✗	✗	✗	✗	✗	✓	✗	0%	✗	✗	✗	–	–	✗	✓	✗
	Gyro+Mag	✗	✗	✗	✗	✗	✓	✗	0%	✗	✗	✗	–	–	✗	✓	✗
	Mag+Quaternions	✗	✗	✗	✗	✗	✓	✗	2%	✗	✗	✗	–	–	✗	✓	✗

B NIST RANDOMNESS RESULTS: ALGORITHM 1

Table B.1: Results of randomness for Algorithm 1. The percentages depict percentage of sub tests passed. ✓ represents the NIST test is passed, while ✗ represents the NIST test is failed. A – means the NIST test suite gives no result. On average, the number of bits input to NIST test suite are 2500000.

Dataset	Data stream	Frequency	Block Frequency	Cumulative Sums	Runs	Longest Run	Rank	FFT	Non Overlapping	Overlapping	Universal	Approx. Entropy	Random Excursions	Ramd. Excu. Variant	Serial	Linear Complexity	Random
19072020	$Acc_x \oplus Gyro_y \oplus Mag_z$	✓	✓	✓	✗	✓	✓	✓	56%	✗	✓	✗	–	–	50%	✓	✓
	$Q_w \oplus Q_x \oplus Q_y \oplus Q_z$	✓	✓	✓	✓	✓	✓	✓	99%	✓	✓	✓	–	–	✓	✓	✓
	Yaw⊕Pitch⊕Roll	✓	✓	✓	✗	✗	✓	✓	64%	✓	✓	✗	100%	100%	✗	✓	✓
	Yaw⊕Pitch⊕Roll⊕Heading	✓	✓	✓	✓	✓	✓	✓	100%	✓	✓	✓	100%	100%	✓	✓	✓
	$Q_w \oplus$ Yaw⊕Heading	✓	✓	✓	✓	✓	✓	✓	97%	✓	✓	✓	–	–	✓	✓	✓
	$Q_w \oplus$ Yaw⊕Heading⊕$Acc_x \oplus Gyro_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	99%	✓	✓	✓	88%	100%	✓	✓	✓
	$Acc_x \oplus Acc_y \oplus Acc_z$	✗	✗	✗	✗	✗	✓	✗	25%	✗	✗	✗	–	–	✗	✓	✗
	$Gyro_x \oplus Gyro_y \oplus Gyro_z$	✓	✓	✓	✓	✓	✓	✓	100%	✓	✓	✓	100%	100%	✓	✓	✓
	$Mag_x \oplus Mag_y \oplus Mag_z$	✓	✓	✓	✗	✓	✓	✓	51%	✓	✗	✗	100%	100%	50%	✓	✓
07082020	$Acc_x \oplus Gyro_y \oplus Mag_z$	✗	✓	✗	✗	✓	✓	✓	98%	✓	✓	✓	–	–	✓	✓	✓
	$Q_w \oplus Q_x \oplus Q_y \oplus Q_z$	✓	✓	✓	✓	✓	✓	✓	100%	✓	✓	✓	100%	100%	✓	✓	✓
	Yaw⊕Pitch⊕Roll	✓	✓	✓	✓	✓	✓	✓	99%	✓	✓	✓	100%	100%	✓	✓	✓
	Yaw⊕Pitch⊕Roll⊕Heading	✓	✓	✓	✓	✓	✓	✓	100%	✓	✓	✓	100%	100%	✓	✓	✓
	$Q_w \oplus$ Yaw⊕Heading	✓	✓	✓	✓	✓	✓	✓	99%	✓	✓	✓	100%	94%	✓	✓	✓
	$Q_w \oplus$ Yaw⊕Heading⊕$Acc_x \oplus Gyro_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	99%	✓	✓	✓	–	–	✓	✓	✓
	$Acc_x \oplus Acc_y \oplus Acc_z$	✗	✗	✗	✗	✓	✓	✗	31%	✗	✗	✗	–	–	✗	✓	✗
	$Gyro_x \oplus Gyro_y \oplus Gyro_z$	✗	✓	✗	✗	✓	✓	✓	97%	✗	✓	✓	–	–	✓	✓	✓
	$Mag_x \oplus Mag_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	99%	✓	✓	✓	100%	100%	✓	✓	✓

C NIST RANDOMNESS RESULTS: ALGORITHM 2

Table C.1: Results of randomness for Algorithm 2. The percentages depict percentage of sub tests passed. ✓ represents the NIST test is passed, while ✗ represents the NIST test is failed. A – means the NIST test suite gives no result. On average, the number of bits input to NIST test suite are 14000000. For Algorithm 2 only data from the 07082020 dataset was tested.

Dataset	Data stream	Frequency	Block Frequency	Cumulative Sums	Runs	Longest Run	Rank	FFT	Non Overlapping	Overlapping	Universal	Approx. Entropy	Random Excursions	Ramd. Excu. Variant	Serial	Linear Complexity	Random
e	Acc	✗	✗	✗	✗	✗	✗	✗	0%	✗	✗	✗	–	–	✗	✗	✗
	Acc_x	✗	✗	✗	✗	✗	✗	✗	0%	✗	✗	✗	–	–	✗	✗	✗
	$Gyro_x$	✗	✗	✗	✗	✗	✗	✗	0%	✗	✗	✗	–	–	✗	✗	✗
	Gyro	✗	✗	✗	✗	✗	✗	✗	0%	✗	✗	✗	–	–	✗	✗	✗
	Q_w	✗	✗	✗	✗	✗	✗	✗	0%	✗	✗	✗	–	–	✗	✗	✗
	Quarts	✗	✗	✗	✗	✗	✗	✗	0%	✗	✗	✗	–	–	✗	✗	✗
	YPR	✗	✗	✗	✗	✗	✓	✗	0%	✗	✗	✗	–	–	✗	✓	✗
	YPR+Heading	✗	✗	✗	✗	✗	✓	✗	0%	✗	✗	✗	–	–	✗	✓	✗
	Heading	✗	✗	✗	✗	✗	✓	✗	1%	✗	✗	✗	–	–	✗	✓	✗
π	Acc	✗	✗	✗	✗	✗	✗	✗	0%	✗	✗	✗	–	–	✗	✗	✗
	Acc_x	✗	✗	✗	✗	✗	✗	✗	1%	✗	✗	✗	–	–	✗	✗	✗
	$Gyro_x$	✗	✗	✗	✗	✗	✗	✗	0%	✗	✗	✗	–	–	✗	✗	✗
	Gyro	✗	✗	✗	✗	✗	✗	✗	0%	✗	✗	✗	–	–	✗	✗	✗
	Q_w	✗	✗	✗	✗	✗	✗	✗	0%	✗	✗	✗	–	–	✗	✗	✗
	Quarts	✗	✗	✗	✗	✗	✗	✗	0%	✗	✗	✗	–	–	✗	✗	✗
	YPR	✗	✗	✗	✗	✗	✗	✗	0%	✗	✗	✗	–	–	✗	✓	✗
	YPR+Heading	✗	✗	✗	✗	✗	✓	✗	0%	✗	✗	✗	–	–	✗	✓	✗
	Heading	✗	✗	✗	✗	✗	✓	✗	0%	✗	✗	✗	–	–	✗	✓	✗

D NIST RANDOMNESS RESULTS: ALGORITHM 3

Table D.1: Results of randomness for Algorithm 3, all datapoints. The percentages depict percentage of sub tests passed. ✓ represents the NIST test is passed, while ✗ represents the NIST test is failed. A – means the NIST test suite gives no result. On average, the number of bits input to NIST test suite are 5000000.

Dataset	Data stream	Frequency	Block Frequency	Cumulative Sums	Runs	Longest Run	Rank	FFT	Non Overlapping	Overlapping	Universal	Approx. Entropy	Random Excursions	Ramd. Excu. Variant	Serial	Linear Complexity	Random
19072020	$Acc_x \oplus Acc_y \oplus Acc_z$	✓	✓	✓	✓	✓	✓	✓	100%	✓	✓	✓	100%	100%	✓	✓	✓
	$Gyro_x \oplus Gyro_y \oplus Gyro_z$	✓	✓	✓	✗	✗	✓	✓	97%	✗	✓	✗	100%	100%	✓	✓	✓
	$Mag_x \oplus Mag_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	98%	✓	✓	✓	88%	100%	✓	✓	✓
	$Acc_x \oplus Gyro_y \oplus Mag_z$	✓	✓	✓	✗	✓	✓	✓	99%	✓	✓	✗	100%	100%	50%	✓	✓
	$Q_w \oplus Q_x \oplus Q_y \oplus Q_z$	✓	✓	✓	✓	✓	✓	✓	100%	✓	✓	✓	100%	100%	✓	✓	✓
	YPR⊕Heading	✓	✓	✓	✓	✓	✓	✓	99%	✓	✓	✓	100%	100%	✓	✓	✓
	$Q_w \oplus$Yaw⊕Heading	✓	✓	✓	✓	✓	✓	✓	100%	✓	✓	✓	100%	100%	✓	✓	✓
	$Q_w \oplus$Yaw⊕Heading$\oplus Acc_x \oplus Gyro_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	99%	✓	✓	✓	100%	100%	✓	✓	✓
07082020	$Acc_x \oplus Acc_y \oplus Acc_z$	✓	✓	✓	✓	✓	✓	✓	100%	✓	✓	✓	–	–	✓	✓	✓
	$Gyro_x \oplus Gyro_y \oplus Gyro_z$	✗	✓	50%	✓	✓	✓	✓	99%	✓	✓	✓	–	–	✓	✓	✓
	$Mag_x \oplus Mag_y \oplus Mag_z$	✗	✓	✗	✗	✓	✓	✓	97%	✓	✓	✓	100%	100%	✓	✓	✓
	$Acc_x \oplus Gyro_y \oplus Mag_z$	✓	✓	✓	✗	✓	✓	✓	98%	✓	✓	✓	100%	100%	✓	✓	✓
	$Q_w \oplus Q_x \oplus Q_y \oplus Q_z$	✓	✓	✓	✓	✓	✓	✓	100%	✓	✓	✓	100%	100%	✓	✓	✓
	YPR⊕Heading	✓	✓	✓	✓	✓	✓	✓	99%	✓	✓	✓	–	–	✓	✓	✓
	$Q_w \oplus$Yaw⊕Heading	✓	✓	✓	✓	✓	✓	✓	98%	✓	✓	✓	100%	100%	✓	✓	✓
	$Q_w \oplus$Yaw⊕Heading$\oplus Acc_x \oplus Gyro_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	99%	✓	✓	✓	–	–	✓	✓	✓

Table D.2: Results of randomness for Algorithm 3, 500 datapoints. The percentages depict percentage of sub tests passed. ✓ represents the NIST test is passed, while ✗ represents the NIST test is failed. A – means the NIST test suite gives no result. On average, the number of bits input to NIST test suite are 4000000.

Dataset	Data stream	Frequency	Block Frequency	Cumulative Sums	Runs	Longest Run	Rank	FFT	Non Overlapping	Overlapping	Universal	Approx. Entropy	Random Excursions	Ramd. Excu. Variant	Serial	Linear Complexity	Random
19072020	$Acc_x \oplus Acc_y \oplus Acc_z$	✓	✓	✓	✓	✓	✓	✓	97%	✓	✗	✓	–	–	✓	✓	✓
	$Gyro_x \oplus Gyro_y \oplus Gyro_z$	✓	✓	✓	✓	✓	✓	✓	96%	✓	✗	✓	–	–	✓	✓	✓
	$Mag_x \oplus Mag_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	100%	✓	✗	✗	–	–	✓	✓	✓
	$Acc_x \oplus Gyro_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	98%	✓	✗	✗	–	–	✓	✓	✓
	$Q_w \oplus Q_x \oplus Q_y \oplus Q_z$	✓	✓	✓	✓	✓	✓	✓	97%	✓	✗	✗	–	–	✓	✓	✓
	YPR⊕Heading	✓	✓	✓	✓	✗	✓	✓	98%	✓	✗	✓	–	–	✓	✓	✓
	$Q_w \oplus$Yaw⊕Heading	✓	✓	✓	✓	✓	✓	✓	95%	✓	✗	✓	–	–	✓	✓	✓
	$Q_w \oplus$Yaw⊕Heading$\oplus Acc_x \oplus Gyro_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	97%	✓	✗	✗	–	–	✓	✗	✓
07082020	$Acc_x \oplus Acc_y \oplus Acc_z$	✓	✓	✓	✓	✓	✓	✓	99%	✓	✗	✓	–	–	✓	✓	✓
	$Gyro_x \oplus Gyro_y \oplus Gyro_z$	✓	✗	✓	✓	✓	✓	✓	97%	✓	✗	✗	–	–	✓	✓	✓
	$Mag_x \oplus Mag_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	97%	✓	✗	✓	–	–	✓	✓	✓
	$Acc_x \oplus Gyro_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	96%	✓	✗	✗	–	–	✓	✓	✓
	$Q_w \oplus Q_x \oplus Q_y \oplus Q_z$	✓	✓	✓	✓	✓	✓	✓	100%	✓	✗	✓	–	–	✓	✓	✓
	YPR⊕Heading	✓	✓	✓	✓	✓	✓	✓	97%	✓	✗	✓	–	–	✓	✓	✓
	$Q_w \oplus$Yaw⊕Heading	✓	✓	✓	✓	✓	✓	✓	99%	✓	✗	✗	–	–	✓	✓	✓
	$Q_w \oplus$Yaw⊕Heading$\oplus Acc_x \oplus Gyro_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	97%	✓	✗	✗	–	–	✓	✓	✓

Table D.3: Results of randomness for Algorithm 3, 100 datapoints. The percentages depict percentage of sub tests passed. ✓ represents the NIST test is passed, while ✗ represents the NIST test is failed. A – means the NIST test suite gives no result. On average, the number of bits input to NIST test suite are 3700000.

Dataset	Data stream	Frequency	Block Frequency	Cumulative Sums	Runs	Longest Run	Rank	FFT	Non Overlapping	Overlapping	Universal	Approx. Entropy	Random Excursions	Ramd. Excu. Variant	Serial	Linear Complexity	Random
19072020	$Acc_x \oplus Acc_y \oplus Acc_z$	✓	✓	✓	✓	✓	✓	✓	96%	✓	✗	✓	–	–	✓	✓	✓
	$Gyro_x \oplus Gyro_y \oplus Gyro_z$	✓	✓	✓	✓	✓	✓	✓	99%	✓	✗	✓	–	–	✓	✓	✓
	$Mag_x \oplus Mag_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	97%	✓	✗	✓	–	–	✓	✓	✓
	$Acc_x \oplus Gyro_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	97%	✓	✗	✓	–	–	✓	✗	✓
	$Q_w \oplus Q_x \oplus Q_y \oplus Q_z$	✓	✓	✓	✓	✓	✓	✓	99%	✓	✗	✗	–	–	✓	✗	✓
	YPR⊕Heading	✓	✓	✓	✓	✓	✓	✓	94%	✓	✗	✓	–	–	✓	✓	✓
	$Q_w \oplus$Yaw⊕Heading	✓	✓	✓	✓	✓	✓	✓	95%	✓	✗	✓	–	–	50%	✓	✓
	$Q_w \oplus$Yaw⊕Heading⊕$Acc_x \oplus Gyro_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	94%	✓	✗	✓	–	–	✓	✗	✓
07082020	$Acc_x \oplus Acc_y \oplus Acc_z$	✓	✓	✓	✓	✓	✓	✓	97%	✓	✗	✓	–	–	✓	✓	✓
	$Gyro_x \oplus Gyro_y \oplus Gyro_z$	✓	✓	✓	✓	✓	✓	✓	93%	✓	✗	✓	–	–	✓	✓	✓
	$Mag_x \oplus Mag_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	96%	✓	✗	✓	–	–	✓	✓	✓
	$Acc_x \oplus Gyro_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	99%	✓	✗	✓	–	–	✓	✓	✓
	$Q_w \oplus Q_x \oplus Q_y \oplus Q_z$	✓	✓	✓	✓	✓	✓	✓	96%	✓	✗	✗	–	–	✓	✓	✓
	YPR⊕Heading	✓	✓	✓	✓	✓	✓	✓	97%	✓	✗	✓	–	–	✓	✓	✓
	$Q_w \oplus$Yaw⊕Heading	✓	✓	✓	✓	✓	✓	✓	97%	✓	✗	✓	–	–	✓	✓	✓
	$Q_w \oplus$Yaw⊕Heading⊕$Acc_x \oplus Gyro_y \oplus Mag_z$	✓	✓	✓	✓	✓	✓	✓	97%	✓	✗	✓	–	–	✓	✓	✓

E DATASET DESCIPTION

Title of the dataset:
Data underlying the thesis: Towards Implicit Authentication in Smart Logistics: A Random Number Generator for Sensor PUFs in Resource Constrained IoT Devices

Author: Maaike Hillerstrom
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente

Contact information:
m.a.m.hillerstrom@student.utwente.nl
The Netherlands

Description:
This dataset contains raw data gathered with the 9DoF Razor IMU (Inertial Measurement Unit) M0 from Sparkfun.
This IMU contains an Accelerometer, Gyroscope and Magnetometer.
With the onboard SAMD21 processor the Quaternions, Yaw, Pitch, Roll and Heading are calculated.
The data was gathered while driving in a car, in two separate trips. One trip of 50 KM and one trip of 150 KM.
Majority of the trips were driven on highways and the remaining in urban areas. There was no off-roading.

The following settings are used for data gathering:
The data is gathered at 100Hz.
Accelerometer range of +- 2G.
Gyroscope range of +- 500 dps (degrees per second).
Magnetometer range of +- 4800 microTesla (default value).

The dataset contains the following sensor data:
Device up time,
Accelerometer X-, Y-, Z-axis
Gyrosope X-, Y-, Z-axis
Magnetometer X-, Y-, Z-axis
Quaternions W-, X-, Y-, Z-axis
Yaw,
Pitch,
Roll,
Heading.

This dataset contains two folders both containing text files with the sensor data as described above.
The data is stored in separate consecutive text files. This means that the files in one folder together form a full subset of sensor data gathered in one driving trip.
The folder 100Hz_2g_500dps_19072020 contains eight text files.
The folder 100Hz_2g_500dps_07082020 contains 17 text files.