UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS

PhD Thesis Summary

Securing the IoT infrastructure through blockchain solutions

Florin Răstoceanu

Thesis advisor:
Prof. PhD. eng. Răzvan-Victor Rughiniș

BUCHAREST

2023

Contents

Abstract

The Internet of Things (IoT) is constantly progressing, continuously influencing the quality of our lifes through a multitude of useful and easy-to-use applications, but at the same time, exposing our personal data to significant security threats. The specificity of IoT architectures, which foresee the interconnection between classic computer networks and living beings, requires the commitment to an increased degree of security. This is more difficult to achieve due to IoT architectures' specific aspects, such as the heterogeneity of devices or the limited resources they have available. The need for security is noteworthy, as cyber-attacks targeting IoT devices increased exponentially latterly.

In this thesis I propose a series of solutions aiming at improving security in IoT infrastructures. Basic security services such as confidentiality, integrity, authentication and non-repudiation can be implemented using cryptographic protocols and mechanisms. They can provide the desired degree of security only if they use cryptographic keys and random input parameters. Considering these aspects, but also the specificity of the IoT environment, I approach the issue of security on several levels.

First, I identified ways to integrate blockchain technology with an IoT architecture to provide support for the implementation of security services. The use of IoT nodes, that also provide specific blockchain functionalities, was analyzed from the point of view of resources and costs. In this sense, I propose the usage of IoT nodes within a fog computing architecture as blockchain nodes with functionalities adapted to the available resources and the implementation of IoT nodes using two types of FPGA architectures.

The blockchain ledger is used as a source of trust in implementing a simple and secure session key negotiation protocol. The solution provides increased security by using cryptographically secure primitives. Taking into account the simplicity of the solution and the selection of appropriate cryptographic functions, optimized power consumption is provided for resource-constrained IoT devices.

The effectiveness of the cryptographic algorithms is closely related to the cryptographic keys used, which must be random and cannot be deduced by potential attackers. The keys can only be generated using properly evaluated random generators, with enhanced security and efficiency features. Considering these aspects, I propose a random number generator solution that uses a lightweight encryption algorithm and fulfills high security properties by re-initializing the inputs with fresh entropy at each call.

The desired degree of unpredictability for the inputs of random number generator can be achieved by using sources capable of providing an appropriate level of entropy. In this respect, the entropy source must be stable, resistant to attacks and efficient to be used in IoT environments. The solution proposed in this thesis uses randomness generated by motion sensors. In this way, resources already existing in IoT platforms are used to ensure efficiency. Using an original and comprehensive analysis methodology for the source of entropy in terms of noise acquisition method, stability and attack resistance, I have validated the entropy source to be applied in IoT applications using motion sensors.

The solutions proposed in this thesis can ensure the protection of data transmitted in IoT environments by using the blockchain ledger as a trust anchor within a key establishment protocol and secure mechanisms to generate random numbers and entropy. Proven power efficiency through platform-specific implementations is qualifying these solutions as fitting in IoT applications.

1. Introduction

The Internet of Things (IoT) is spreading fast and is increasingly present in our lives. A thing can be a smart device (smart watch, printer, fridge, washing machine, car, drone, smart home, smart lock, etc), an implant that monitors and regulates a person's heartbeat or blood sugar levels, or a smart chip implanted in an animal on a farm. Something can be considered part of the IoT if it is connected to a network and has the ability to exchange data with other components in the system [1]. Through the IoT infrastructure, using sensors and actuators a connection is made between the Internet, seen as a global network of computers, and other devices with computer addresses, the natural environment, represented by people, animals or elements in nature [2].

IoT infrastructure involves interconnecting computer networks with living things or elements in nature. Thus, risks in the cyber environment are transferred to the latter, bringing much more serious and harder damages. Ensuring security in such systems is all the more important as the adoption of this technology in our lives is taking place at an increasing pace. According to www.statista.com [3], in 2020 there were approximately 9.7 billion active IoT devices and the number is expected to triple in ten years to 29.4 billion. On the other hand, cyber-attacks targeting IoT devices have increased alarmingly recently. For example, Symantec reported a 600% increase in attacks from 2016 to 2017 [4], and in the first half of 2021, Kaspersky reported 1.5 billion attacks deployed against IoT devices.

1.1. Motivation

Ensuring security in such a diverse and complex environment is fraught with many issues specific to IoT devices. First and foremost, this system is highly heterogeneous. There are a multitude of devices on the market today that differ in operating systems, network interfaces, protocols used, security mechanisms and functions implemented. To address these issues, new technologies must be identified that build on IoT security architectures. As specified in the National Institute of Standards and Technology (NIST) report on cybersecurity standardization for IoT [1], blockchain (BC) technology has significant potential in this area. Providing security in a decentralised way offers certain advantages to the IoT environment compared to traditional solutions relying on Public Key Infrastructure (PKI). Using blockchain technology can turn certain disadvantages into advantages. Thus, the large number of IoT devices can be beneficial to ensure better decentralisation and increase trust in the deployed solution, but it can also provide high availability by ensuring a sufficient number of entities to validate transactions. The immutability feature can provide the possibility to store data that cannot be modified, which can be very useful for event auditing functions. Even the heterogeneous nature of IoT infrastructures can be assimilated, as blockchain technology only needs an address and the ability to communicate in peer-to-peer networks.

On the other hand, with few exceptions, developers are focused on ensuring the interconnectivity of devices and much less on ensuring their security. This is difficult to achieve considering that many of them have low processing and storage capacity or no permanent power supply, running on battery power. In this case, the implementation of classical cryptography is almost impossible to achieve and solutions for cryptographic algorithms, protocols and mechanisms adapted to resource-constrained devices need to be found. Cryptography is used to provide basic security services such as confidentiality, integrity, authentication and non-repudiation. To ensure their robustness and resistance to different types of attacks, the whole security infrastructure must be considered. Figure 1.1 shows all its elements.

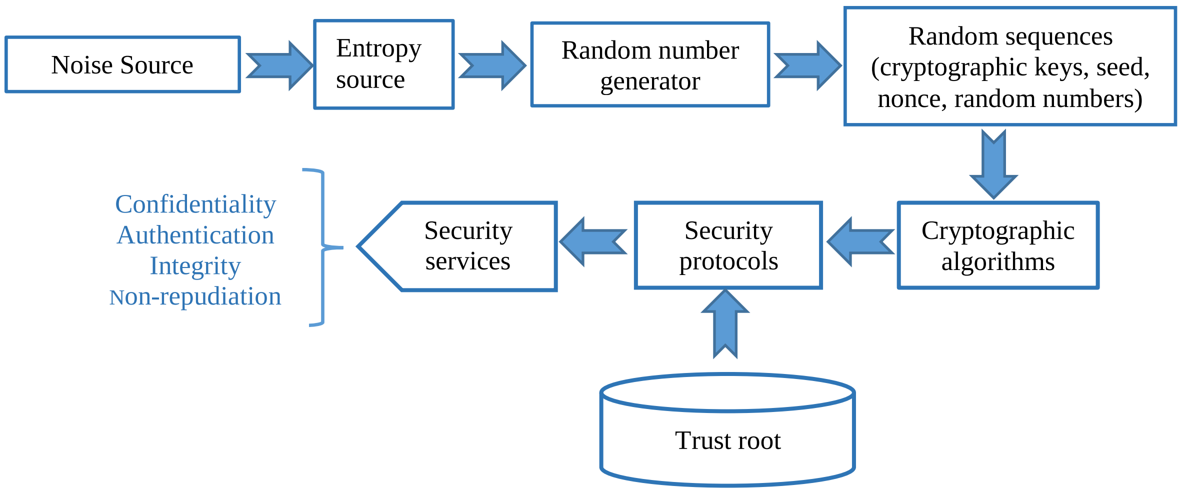

Figure 1.1 Security services implementation flow

1.2. Objectives

The aim of the thesis is to identify solutions for providing security in IoT infrastructures. This objective can be achieved on several levels, given the security service implementation flow as shown in Figure 1.1 and the constraints specific to the IoT environment.

The first objective is to identify solutions for integrating blockchain technology into IoT infrastructures, with the aim of providing secure and efficient methods to implement mutual authentication and privacy services for communications between different IoT devices. This objective can be achieved by:

- Identifying the optimal ways in which blockchain technology can support an IoT infrastructure to bring benefits superior to classical solutions using PKI;
- Proposing a common IoT - BC architecture and optimising it in terms of resources and costs;
- Identify optimal architectural solutions for hardware platforms used to deploy IoT nodes, but which also provide functionality specific to BC nodes.

The second objective is to identify a simple and secure solution for a protocol used to establish the encryption and authentication keys needed to secure communications between IoT nodes. This objective can be achieved by:

- Ensuring compatibility with blockchain technology;
- Optimal integration with resource-constrained IoT devices;
- Optimising power consumption using appropriate cryptographic mechanisms and functions;

- Ensuring a level of security that does not allow the compromise of the data being transported.

The third objective is to identify a random number generator that provides the highest degree of security but can be deployed on IoT devices with limited resources. This objective can be achieved by:

- Identifying a solution that provides the highest degree of security, given the randomness of the generated data and the cryptographic strength of the deterministic component of the generator;
- Optimising the efficiency of the random number generator in terms of resources consumed relative to the speed and volume of data generated.

The fourth objective is to identify an entropy source that can be deployed with minimal resources on IoT devices, but which provides a enough entropy to be used in a cryptographic context. This objective can be achieved by:

- Identifying an entropy source that uses as few resources as possible, possibly using from existing resources on IoT nodes;
- Estimating the level of entropy generated using standardised and reliable methodologies;
- Optimize the efficiency of the entropy source by optimal parameterization in different operating cases;
- Applying an entropy source testing and evaluation methodology to analyze the behavior of the entropy source in different use cases, over the long term, and its resistance to different types of attacks.

2. IoT infrastructure security

In this chapter I have presented key issues related to the security of IoT infrastructures. I have treated the issue from the point of view of the state of the art, presenting also theoretical notions that define the concepts developed in the following chapters of the thesis.

2.1. Cybersecurity areas in IoT

To ensure security in an IoT ecosystem, several aspects need to be considered according to the NIST report on standardizing the IoT cybersecurity environment [1]. Among the most important are the following:

- Cryptographic techniques implemented to ensure the protection of stored or transmitted sensitive data. The biggest challenge here is the limited resources specific to many IoT devices;
- Security assessment which aims to ensure the following: implementation of security mechanisms in the IT system or product, performing security tests to validate a certain level of security, applying universal metrics to measure the strength of the implemented cryptographic mechanisms and functions;

- Physical security which aims to protect IoT devices against passive or intrusive attacks designed to extract cryptographic keys or sensitive data. To prevent such attacks, filters can be applied to the power supply or enclosures can be built that do not allow radiation to be emitted outside. In [5] we conducted a study that simulates and models the electromagnetic field of a sensor node enclosure.
- Security of hardware/software components by ensuring that they have no known vulnerabilities;
- Identity and access management by ensuring discretionary access to data of different entities represented by people, organizations, hardware devices, software applications;
- Network security which ensures the secure management, operation and use of data;
- Risk management of product development and delivery, which addresses issues related to how to ensure that products are delivered to specification.

2.2. Security architectures specific to IoT infrastructures

Most scientific papers address security in close relation to system architecture. Architectures presented in the literature are composed of 3, 4 or 5 levels.

The simpler architectures are presented on three basic levels: perception, network and application. The perception level is present in any architecture and represents the connection to the environment. It is made up of sensors and actuators embedded in the sensor node, which has the ability to transmit the accumulated data to the next transport or network level. The transport/network layers provide connectivity and message transmission to the other layers. On the third layer run various applications that aggregate the accumulated sensor data and give it a clear and precise purpose. Application maintenance, access control and software security updates are performed at this level.

Additional layers have been introduced to better integrate the heterogeneous and complex nature of IoT systems. Thus, after the transport layer, an additional layer has been added to act as an intermediate processing layer for data before it is sent to the application layer. This can be identified in the literature under different names: middleware or services. This layer provides the interoperability and scalability needed to offer services to users without the hardware component. Service management and database access are provided at this level.

The third type of IoT system architecture is the five-layer architecture. In such an architecture, an additional layer is introduced above the application layer. In most papers it can be identified by the name of business layer. This layer is responsible for managing the IoT system as a whole, providing business models, graphs, structured data tables based on information from the application layer. In another five-layer approach, the last layer is the user interface.

2.3. Cryptographic key management

The security analysis of a cryptographic system assumes that the cryptographic algorithms are known and that its strength lies in its ability to protect the cryptographic keys used. In this sense, the use and protection of cryptographic keys throughout their lifecycle is as important as the protection of sensitive data. The life cycle of cryptographic keys, shown in Figure 2.1, comprises, depending on the use of the keys, the following stages: generation, storage, transport, import, export, use and destruction or zeroization.

The ways in which key management systems can be implemented vary from case to case. Symmetric keys must be in the possession of the correspondents. They can be distributed electronically or manually using other methods of protection during transport. Distribution by electronic methods requires the provision of confidentiality, integrity and authentication services, which can be achieved using other cryptographic keys, which are often asymmetric keys. These must also be distributed. In this case, distribution is achieved using a Public Key Infrastructure (PKI). A PKI system is a centralised digital certificate management structure.

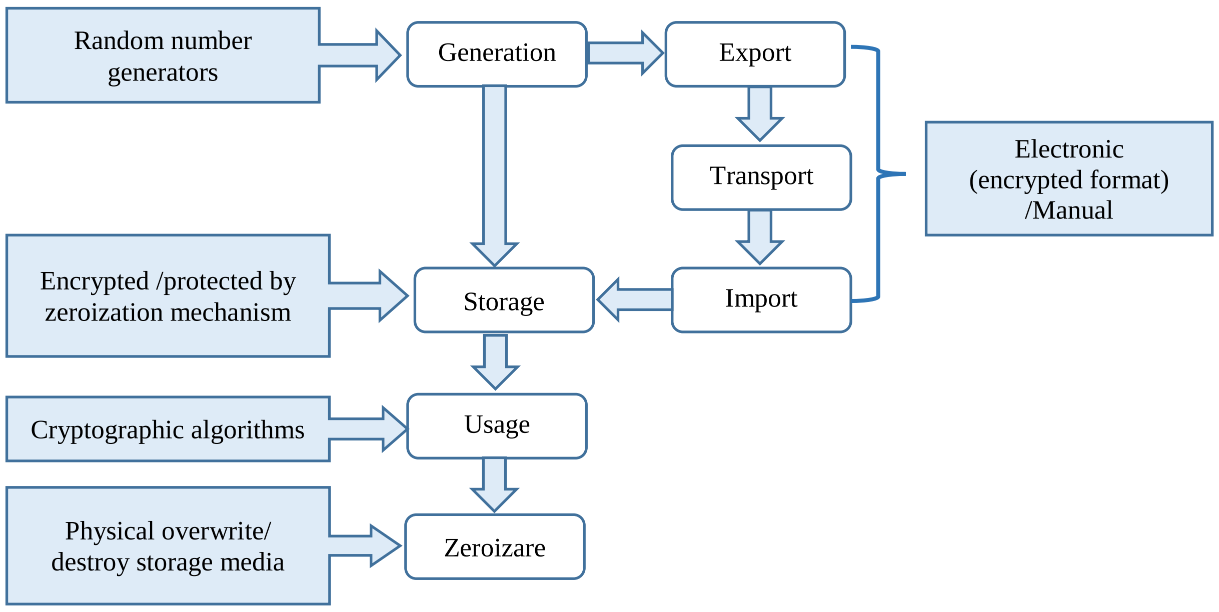
Random number generators
Generation
Export
Transport
Electronic (encrypted format) /Manual
Encrypted /protected by zeroization mechanism
Storage
Import
Cryptographic algorithms
Usage
Physical overwrite/ destroy storage media
Zeroizare

Figure 2.1 Cryptographic keys life cycle

2.4. Cryptographic key generation methods - random number generators

According to NIST [6], a random number generator is a device or algorithm capable of generating random sequences of bits that have the properties of being statistically independent and uniformly distributed. Random number generators can be of two types: deterministic and non-deterministic. Deterministic Random Number Generators (DRNGs) are built on an algorithm that uses a secret initial random value, also called seed, to generate longer sequences of random numbers. They are called pseudo-random number generators. Non-deterministic random number generators (TRNGs) use entropy sources, based on noise sources obtained from random physical phenomena or random events.

While non-deterministic sources can generate random numbers with maximum entropy, they have a low generation speed relative to the needs of most cryptographic applications. For this reason, deterministic generators are used because they can generate at much higher speeds. They use inputs from noise sources, which can be entropy sources or non-deterministic generators.

Entropy sources consist of a noise source, a conditional function, which may be optional, and a suite of health tests. The noise source is the element that generates the randomness of the entropy source. It contains the elements that give non-deterministic character to the data generated by the entropy source. Noise sources can be classified into two types: software and hardware. Software noise derives its randomness from the randomness of various processes and events specific to operating systems. These types of noise sources require operating systems where many processes are running or where operator intervention is frequent. Since operator activity on IoT devices is low, processes are limited due to power consumption or there are no operating systems. These solutions are not suitable for use in IoT. Typically, noise sources are based on physical phenomena that occur randomly. Existing solutions use diodes, FPGA circuits or the randomness of some IoT sensors [7][8].

2.5. Lightweight cryptographic algorithms

The IoT infrastructure contains a multitude of devices that have limited resources, such as RFID devices or different types of sensors. These devices allocate very few resources to ensuring data security. This has created a need to develop cryptographic algorithms that maintain a sufficient level of security but do not use too many resources. In conclusion, an implementation is a trade-off between security, performance and cost. An algorithm with good performance and low cost will be exposed to side-channel attacks. If measures are implemented to prevent these types of attacks, costs increase and performance decreases. The role of lightweight algorithms is to find the optimal solution to achieve all these goals in a satisfactory way. Depending on the implementation mode, software or hardware, efficiency can be evaluated differently.

For hardware implementations, memory consumption and the size of the implementation, expressed in the number of ports used, matter. This should be as small as possible. Other important parameters are:

- processing speed, expressed as the number of bytes processed per second;
- latency, which measures the time elapsed from setting up the circuit to obtaining the output sequences;
- power consumption measured in Watts.

In the case of software implementations, important parameters are: RAM consumption, which is the amount of memory required for an algorithm run, source code size, processing speed and power consumption.

2.6. Cryptographic management for security protocols used in IoT

In this chapter, specific issues related to cryptographic key management for the following protocols used in IoT infrastructures are presented:

- Bluetooth Low Energy (BLE) protocol;
- IEEE 802.15.4 protocol;
- Zigbee protocol;
- LoRaWAN protocol;
- Z-Wave protocol;
- IEEE 802.11 protocol - Wi-Fi Protected Access;
- TLS protocols - Transport Layer Security.

2.7. Access and identity management

IoT has introduced the concept that entities are interconnected. In order to achieve a secure context in which they can communicate, a mechanism is first needed to identify them. Identity refers to a set of information used to uniquely identify an entity in a given context. For example, a person can be identified at work by a set of attributes such as name, job title, job type and in an online shop by attributes such as name and bank account.

An Identity and Access Management (IAM) system looks at the lifecycle of identities, which includes operations to register, update and revoke them. Within a system, the IAM must provide three security services: authentication, authorisation and audit. For example, in the case of an operator seeking access to a service, the authentication operation consists of entering the credentials for the claimed identity and the authorisation operation, which verifies the credentials and makes the decision whether or not to grant access. All these operations are monitored and recorded by the audit service.

2.7.1. Authentication methods

Authentication methods can be classified according to the credentials used. These can be of several types, as follows:

- username or ID and a password;
- credentials that refer to "something" that is owned. In the case of individuals, that "something" can be a unique password generator, a card, a token or a smartphone, . In the case of IoT devices, that "something" refers to an internally stored secret on the basis of which, using an algorithm, the authenticity of the invoked identity can be proven;
- credentials that refer to "what you are". In the case of individuals, it is biometric data and in the case of IoT devices it is PUFs (Physical Unclonable Function). These are physical objects (semiconductor devices, microprocessors) that provide unique responses that can be assimilated with fingerprints;
- context-related credentials. These are usually used in a complementary role. For example, for people it can represent GPS location combined with time information and

for IoT devices, it can represent features related to geographical location and communication technology.

2.7.2. Authorisation methods

Different types of authorisation methods are used in IoT systems, each with their advantages and disadvantages. These can be classified according to the access control model.

- DAC (Discretionary Access Control) - the owner of the IoT device decides the access rules that can restrict the time period access is granted, the operations available and the entities that have access;
- MAC (Mandatory Access Control) - authorisation is granted gradually depending on the type of access owned;
- RBAC (Role Based Access Control) - there are multiple roles to which permissions are granted and each user is assigned one or more roles depending on their responsibilities;
- ABAC (Attribute-Based Access Control) - more flexibility in that instead of defining a static role, it uses a set of policies to grant access;
- Cap-BAC (Capability-Based Access Control) is a token-based access control model that stores access rights for users who own them.

Other conventional authorization methods are presented in: Lattice-Based Access Control, Context-Based Access Control, Chinese Wall Lattice Model, Identity-Based Access Control.

2.7.3. Classification of identity management systems

Identity Management Systems (IMS) have evolved with technology. Five types of such systems can now be identified [9]:

- Isolated;
- Centralized;
- Federated;
- User-centric;
- Self-sovereign.

2.8. Applicability of blockchain technology in IoT

2.8.1. Security aspects of blockchain technology

Blockchain technology has at its core a ledger database made up of chained blocks (see Figure 2.2). The link between blocks is made using hash functions, in the sense that one block contains the hash of the previous block in addition to time, transaction or date information. Thanks to this design pattern, the immutability property is ensured, which means that in order to change the information in one block, all blocks succeeding it will have to be changed. This is not easy to achieve, as the register is stored in a distributed way by network members, who continuously update it according to the rules of a consensus protocol. To be successful, however, it requires the consensus of more than 50% of the network members.

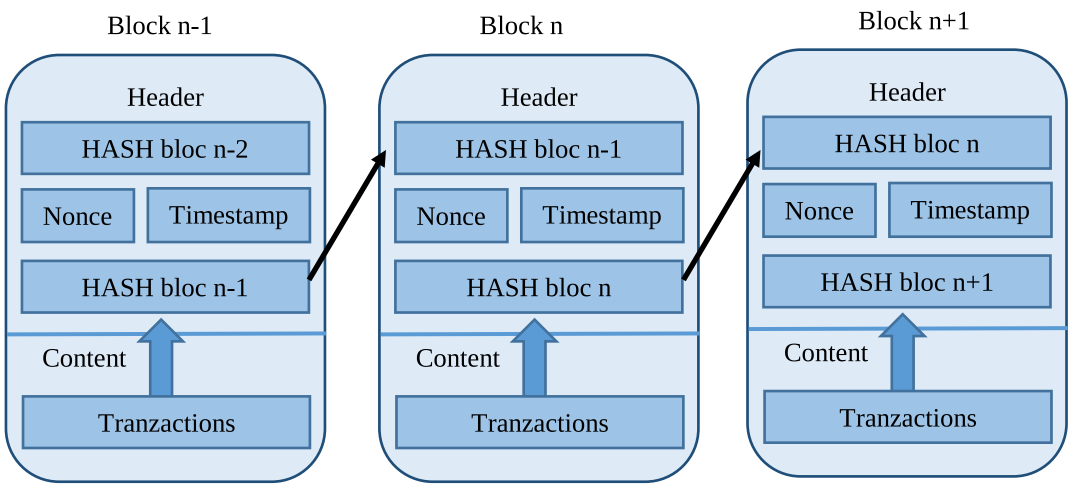

Figure 2.2 Blockchain arhitecture

The consensus protocol is one of the key elements ensuring security in blockchain. Based on it, network members validate the insertion of a block into the existing chain. Currently, several types of such protocols have been proposed, each with its advantages and disadvantages. They have the following characteristics:

- Proof of Work (PoW) - the most popular being used in Bitcoin and Ethereum. The main idea of this protocol is to use computing power to validate the block that is to be introduced. To do this, miners who are part of the users of the network try to identify a number that, together with the actual data in the block, has a hash with certain characteristics. This can be achieved by successive attempts. Computational power is required to compute the respective hash function. Even though it offers a high degree of security, this type of protocol is energy intensive;
- Proof of Stake (PoS). In this case, miners are chosen from among the network members, who make available some of the virtual coins they hold. If they do not validate correctly, they will lose the coins made available. This type of protocol will soon be implemented in Ethereum and similar variants are used in other blockchains such as Elrond;
- Proof of Capacity (PoC), where miners make a certain amount of storage capacity available to the network, which proves a certain degree of interest in order for the system to work correctly;
- Proof of Authority (PoA), which is based on the reputation of the miners selected for validation.

Another important aspect, which also influences security, is the type of blockchain. Each blockchain has rules that can allow anyone to participate in the network or limit participation only by permission. Depending on the application these two types can offer certain advantages. The first type allows unlimited access to the network, thus ensuring full decentralisation and transparency. In this case there is no central authority and the anonymity of participants is ensured. The second model is suitable for use in organisations where anonymity of members is not required. This model offers only partial decentralisation which may provide some security in case of an attack from outside. Due to the limited number of participants this architecture offers higher speeds and increased scalability. It can also provide better protection of the data stored in the registry since access is restricted.

2.8.2. Blockchain applications in IoT

The advantages of blockchain technology has allowed it to be used in applications in many fields. In addition to the financial field, established through cryptocurrency applications, other areas of interest are mentioned in [10], such as:

- cyber security;
- government applications;
- registration and management of property (houses, land) or valuables (cars, phones);
- identity management;
- reputation management system;
- intellectual property;
- fundraising;
- energy systems;
- IoT applications.

IoT applications are classified into several types: providing cyber protection in energy systems, providing cyber protection in transportation systems, providing cyber protection in aviation systems, food safety systems, smart homes, military applications such as IoBT (Internet of Battle Things), access management system or public key management system..

3. Blockchain solutions for IoT security

This chapter presents some solutions for using blockchain technology to ensure IoT security. After analysing the advantages and disadvantages of using blockchain technology in IoT, two solutions for IoT integration with BC are proposed. The first solution presents a fog computing architecture model, which integrates a blockchain to use its security properties in order to achieve a trust relationship between network members. Using the blockchain for storing identity keys, a simple and secure protocol for establishing session keys and authentication is proposed. The solution is evaluated in terms of computational power and cost and compared to a classical solution with TLS and PKI. The second solution proposes the use of FPGA technology for IoT integration with BC. Thus, two FPGA architectures are proposed to implement sensor nodes with the dual role of sensor nodes and BC nodes. The proposed solutions are implemented on several FPGA circuit families with different power consumption and resources.

3.1. Analysis of the possible use of blockchain technology in IoT

Ensuring security in IoT infrastructures requires appropriate technologies. Blockchain technology has certain advantages that can qualify it for this purpose. The following are some of its features that prove its integration into IoT infrastructure:

- *Decentralisation* offers the possibility that transactions are not validated by a central entity, which could be overloaded in case of a large number of transactions or in case of Denial of Service (DoS) attacks;
- *Immutability* - transactions stored in the BC ledger, cannot be modified. This allows IoT devices to easily and securely verify stored data;
- *Data resilience* is ensured by the fact that nodes are in possession of a copy of the blockchain database;
- *Cryptographic support*, as blockchain technology relies on functions that can provide privacy, integrity and authentication services;
- *Trust* - blockchain can provide trust between network members without the need for a central authority;
- *Audit* - blockchain technology offers the possibility to record transactions in the ledger in a secure and immutable way, which can be viewed by all members of the network.

At the same time, the features of an IoT infrastructure are compatible with blockchain technology. A large number of nodes are needed to ensure the best possible decentralisation, which IoT infrastructures fulfil. Since a large number of nodes can be active in an IoT infrastructure at any given point in time, this is an advantage for a blockchain infrastructure, which needs entities to validate transactions.

On the other hand, integrating the two technologies also comes with certain challenges. The blockchain technology requires certain resources that not all IoT nodes have (low computing capacity, storage capacity and limited number of transactions).

At present, not all the problems that could arise when integrating the two technologies are solved. The challenge is to find architectures that integrate the two technologies to take advantage of the benefits that blockchain technology brings to security, but that mitigate as far as possible the shortcomings of implementing blockchain in IoT.

3.2. IoT- BC integration solution

3.2.1. Architectural features of fog computing technology

IoT applications use data acquired from sensors. Most of the time the amount of data collected by sensors is very large. Much of this data is not used directly and needs to be processed. Since the devices through which the data is collected do not have enough computing power, it was decided to collect the data in the cloud, where it is processed and distributed to specific applications. Often, sensor nodes are distributed over a large geographical area or are located in places where communication networks are not available to support the transfer of large volumes of data. In these cases, long delays in data transfer or significant loss of information can occur, which can adversely affect the quality of service. One of the solutions that can solve these problems is to create an intermediate layer of devices between the sensor nodes and the Cloud. This architecture is called fog computing. Fog computing is a decentralised architecture, specific to IoT architectures, which carries part of the services available in the Cloud to the edge of the network. This type of architecture has the advantage of bringing specific computing resources from the Cloud close to where the data is acquired. In this way, complex data processing algorithms or artificial intelligence can be used to efficiently sort data on devices located close to where it is created. This improves efficiency by reducing the amount of data transferred to the Cloud.

In [11], NIST described this architecture as a cloud-based ecosystem in which fog computing serves the final IoT devices. The architecture is built on four tiers. The first tier consists of end IoT devices, represented by sensors and actuators. Data are collected by devices in the next level, called mist. On this layer are various specialised sensor nodes that are implemented using mainly microcontrollers or microcomputers. These have low computing power. This layer is the link to the next layer - fog computing, which is made up of nodes with much higher computing power, storage capacity and network resources than the devices on the mist layer. These can be powerful network devices such as gateways, routers, or computers with high computing power such as servers and mini data centres - cloudlets.

3.2.2. Securing a fog computing architecture using BC

Blockchain technology has the resources to provide security services. Because it is built on a decentralised infrastructure, it has the potential to provide security in a fog computing architecture. Nodes that decide, based on a consensus protocol, which data is recorded in the BC ledger, provide security in the BC. The larger the number of nodes and the more geographically distributed they are over a larger area, the stronger the security provided by the network. Such features can be provided by a fog computing architecture.

In current architectures, security is provided centrally through public key infrastructures. In this way, a trusted entity issues - signs - digital certificates for all members of the network. These digital certificates contain a pair of asymmetric keys that provide cryptographic protection in different security services. There is uncertainty, however, whether these systems are suitable for IoT infrastructures. A first argument would be that the very large number of IoT devices could not be served promptly and efficiently by such an infrastructure. Another argument is the processing capacity of some IoT devices, which do not support such technology. This uncertainty is also fuelled by the fact that the PKI infrastructure is centralised and therefore insufficiently transparent. Blockchain has the ability to solve these problems.

3.2.3. Description of the proposed architecture

The proposed solution integrating blockchain technology into a fog computing architecture aims to provide a secure context for ensuring mutual authentication, confidentiality services and integrity. The challenge here is to identify ways in which the two technologies can be integrated in such a way as to bring benefits superior to traditional solutions using PKI. The proposed architecture and the experimental results obtained have been published in the paper [12].

The use of BC in a PKI infrastructure may encounter certain problems. The first problem concerns the resources needed to provide the basic functions of the BC. Consensus protocols such as PoW require significant computing power. Solutions to replace this protocol with less power consuming ones do not fully solve the problem. Considerable power is also required to provide communication capabilities between nodes. In the solution proposed in Figure 3.1, the functions of the BC nodes are implemented on nodes specific to the fog computing architecture, which have sufficient computing power. Thus, devices in the Cloud or on the FOG layer are assigned the BC nodes that perform basic functions such as mining and register storage. Resource-constrained FOG layer devices or MIST layer devices are assigned functions such as validating or querying the BC ledger. Other nodes can participate in the BC network as users, who have direct access to the blockchain or via a node with which it has already established a trusted connection.

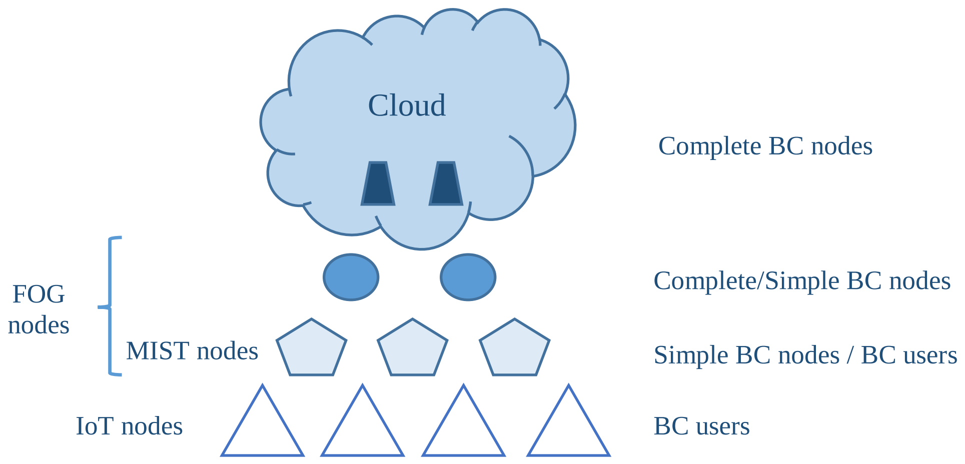

Figure 3.1 Fog computing - blockchain architecture

To validate the proposed solution, we have implemented a smart contract through which an IoT node can interact with the BC. The implementation and testing was performed using the Remix IDE and Ganache environment, which simulates the Ethereum blockchain. The functions implemented in the smart contract were as follows:

- IoT node registration - records information about an IoT node : device ID, identity key and IP address;
- Modify IoT node - change IP address or identity key;
- Read IoT node information - read IP address or identity key;

Table 3.1 shows the costs of implementing the smart contract on the Ethereum blockchain. From the results presented in the table, it can be seen that the highest costs are required to create the contract in BC. The cost of registering a node is 10 cents and for modification, the cost is reduced to half. For costs reading data from the BC are zero. The price estimates shown in the table are based on the average price over the last year (16.05.2022 - 16.05.2023).

Table 3.1 Smartcontract costs

Transaction type	Ethereum gas	Gas (gwei) price	ETH	ETH price ($)	Price ($)
Contract creation	267.177	31,42	0,000267177	1.535,31	0,41
IoT node registration	65.672	31,42	0,000065672	1.535,31	0,10
IoT node modification	29.472	31,42	0,000029472	1.535,31	0,05

The costs required to implement and maintain such a solution have been calculated for 1000 IoT devices and compared to traditional solutions with PKI certificates. For comparison, we used the information available on the clickSSL.net website [13]. Table 3.2 shows the total costs for the two solutions. In the case of the solution with BC, the costs of registering IoT devices and the annual costs of replacing the encryption key are included. In the case of the PKI solution, the costs of issuing digital certificates for IoT devices for different periods of time are included.

Table 3.2 Costs comparison

Period	Price 1 year ($)	Price 3 years ($)	Price 5 years ($)
Proposed solution	101,23	191,73	1236,03
PKI solution	14.000	36.000	50.000

3.3. Cryptographic key negotiation protocol for an IoT-BC architecture

3.3.1. Proposed solution

The proposed solution uses distributed blockchain technology as a security anchor to establish trust relationships between members of an IoT network and then, through message exchanges, establishes encryption and authentication keys. Figure 3.2 shows the architecture of the solution. An IoT node wishing to initiate a communication session with another node needs its identity key. This identity key can be found in a blockchain where nodes have been registered beforehand. The architecture presents two types of nodes. F-type IoT nodes, which have enough resources to participate in the blockchain, will query the BC registry to find out the identity key of the correspondent. These can be FOG or MIST type nodes with roles as nodes with active functions in the BC or just users of the BC. The second type of nodes are N nodes, which do not have sufficient resources to access the BC but use an F node to access the BC ledger.

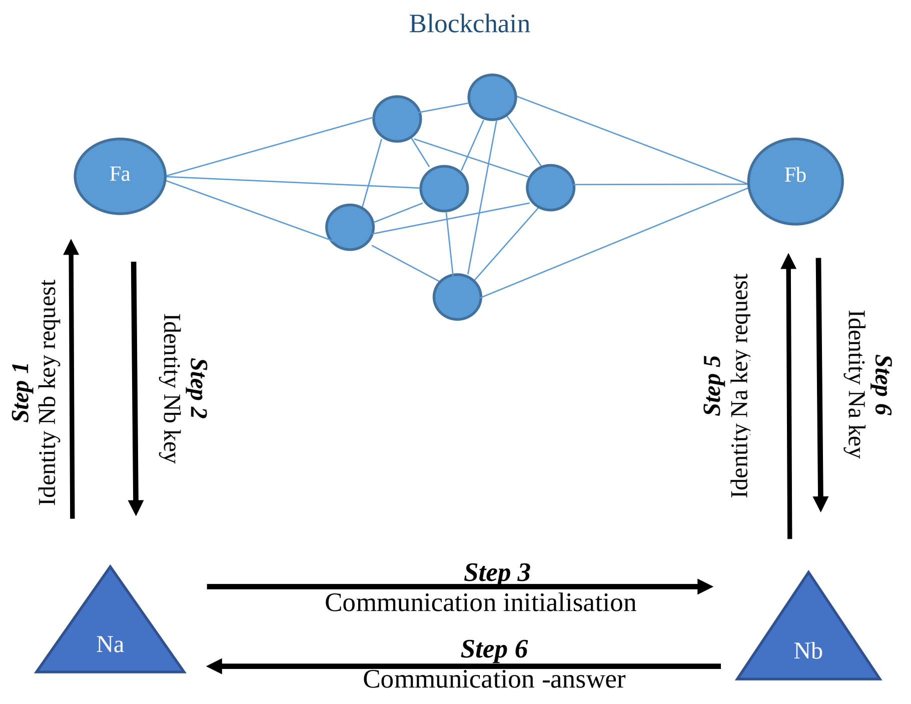

Figure 3.2 Architecture of the key negotiation protocol

In order to ensure that the protocol generates session keys in the desired security context, it is necessary to perform some operations beforehand. First the IoT nodes must be registered in the blockchain. This can be done using a smart contract presented earlier. It must contain the node's identity key and some information necessary to identify it: id, IP address, etc. The steps required to register a node in the BC are as follows:

- The IoT node generates a pair of identity keys using the elliptic curve x25519;
- The IoT node calls a smart contract through which it registers the public part of the identity key together with the necessary identification information in the BC. This operation can also be performed by the owner of the IoT node as a BC user;
- Each node will be identified by a unique Id, which is the hash of the public identity key, calculated with the SHA256 function;
- Each node of type N will receive by a secure method the identity key of the node F to which it is assigned.

Depending on the type of nodes, the protocol has a different number of steps. In the case of type N nodes, two more steps are added, in which it requests and receives from the type F node, to which it is affiliated, the identity key of the correspondent. The cryptographic functions used in the protocol are as follows:

- generation of authentication and encryption keys - HKDF function (key, salt). The key represents the secret generated using the elliptic curve x25519 from the private part of the sender's identity key and the public part of the receiver's key. The salt represents a random number. To ensure that a different key is generated, each time the salt will be different;

- message encryption is performed with the AES cryptographic algorithm in CBC mode with a key length of 32 bytes and an initialisation vector of 16 bytes;
- message authentication is performed with HMAC SHA 256 algorithm. Each message is authenticated. The recipient will verify the authenticity of the message upon receipt.

3.3.2. Solution analysis

3.3.2.1. Security analysis

All sensitive information that could provide opportunities for attacks is encrypted and all messages are authenticated. The encryption and authentication keys used are different for each individual message due to the use of a different and random salt when generating them. Cryptographic operations are performed using strong and secure algorithms: AES 256 for symmetric encryption, HMAC SHA 256 for integrity and authentication, HKDF for key and secret generation and elliptic curve x25519 for asymmetric encryption.

3.3.2.2. Performance analysis

In order to highlight the low power consumption required to negotiate a session key using the presented protocol, a comparison with TLS 1.2 and TLS 1.3 protocols has been made. The implementation hasbeen performed on the B-L475E-IOT01A platform, developed by STMicroelectronics..

Table 3.3 Energy consumption of cryptographic functions

Cryptographic function	Energy consumed (Wh)
ECDH signing (32 bytes)	206,955
ECDH signature verification (32 bytes)	131,5872
x25519 secret generation	15,33168
x25519 key generation	15,22872
HMAC SHA256 (32 bytes)	0,0575316
SHA 256 (334 bytes)	0,074646
HKDF SHA256	0,177021
ENC AES CBC 128 (16 bytes)	0,01340352
DEC AES CBC 128 (16 bytes)	0,0132561
ENC AES GCM 128 (16 bytes)	0,0946737
DEC AES GCM 128 (16 bytes)	0,0815346
ENC AES CBC 256 (32 bytes)	0,0231984
DEC AES CBC 256 (32 bytes)	0,0229086

In my calculations I have only taken into account the cryptographic functions used. Their implementation was done using the wolfSSL software library. The energy consumed was calculated by multiplying the current, measured during the execution of each function, by the time required to execute that function and by the supply voltage (5V). The microcontroller current was measured on pin JP5 of the electronic board.

Table 3.3 shows the results obtained from the implementation of the cryptographic functions used by the compared protocols. As can be seen from the table, the energy required to execute the signature and signature verification functions is much higher than that required for the encryption or hash calculation operations.

The energy required for the entire key generation process, corresponding to each participating node, is shown in Table 3.4.

Table 3.4 Energy consumed during session key negotiation

Protocol	Node	Consumed energy (Wh)
TLS 1.2	Client/Server	647,82
TLS 1.3	Client/Server	740,88
Proposed solution	Na	46,80
	Nb	46,98
	Fa	15,84
	Fb	15,84

As can be seen, the proposed solution consumes much less energy than TLS protocols. This is because it is much simpler than the TLS solution, which has been designed to cater for much more diverse and complex situations. For low complexity IoT applications, the proposed solution can provide a high degree of security using significantly less resources.

3.4. Integration of IoT nodes into BC using FPGA

3.4.1. Sensor node implementation solution using FPGA for integration with BC

In the study published in [14] we described a solution to implement a sensor node using FPGA circuitry to be integrated into the BC. A node implemented with FGPA circuits can provide communication interfaces with sensors but also with interfaces specific to a blockchain node, thus fulfilling both roles.

Depending on the hardware resources of the FPGA circuit, the node can perform various roles in the BC. The following will present two specific FPGA circuit architectures and the specific functions they can perform.

- ***Classic FPGA architecture for an IoT sensor node***

In the proposed architecture (Figure 3.3) all component blocks are implemented using the logic gates provided by the FPGA. This architecture has the advantage that it is very flexible and can adapt to the type and number of sensors it needs to serve, but can also have blockchain-specific data processing blocks implemented. In this case, the best energy consumption can be achieved for the implementation of these functionalities.

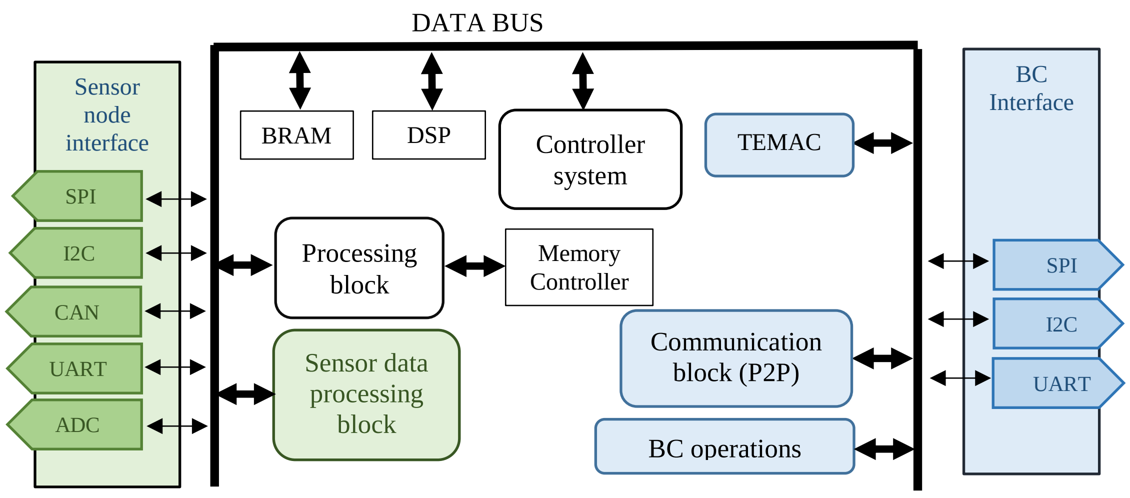

Figure 3.3 Classic FPGA architecture for a sensor node

- ***FPGA SoC architecture for an IoT sensor node***

This type of FPGA architecture has dedicated processing modules included in the hardware-processing block. Only advanced circuit families such as Zynq-700, Zynq Ultra Scale+, Cyclone V have architectures of this type. These circuits are part of the System on a Chip (SoC) category that integrates processing units together with several types of controllers and can be used to implement monitoring, memory and dedicated encryption modules. In order to function properly, this block requires external RAM, which is energy intensive. These advantages of SoC architectures greatly increase capacity and processing speed, recommending them for applications requiring high throughput, high processing power and data acquisition from many sensor nodes. Taking these aspects into account, it can be stated that this architecture can be used for hybrid nodes integrating both BC node and IoT sensor node functionality.

In the proposed architecture (Figure 3.3) the sensor interface is implemented as in the previous version, using logic gates. The BC interface is implemented using dedicated blocks in the hardware processing block. In this way, the interface is provided with the resources it needs.

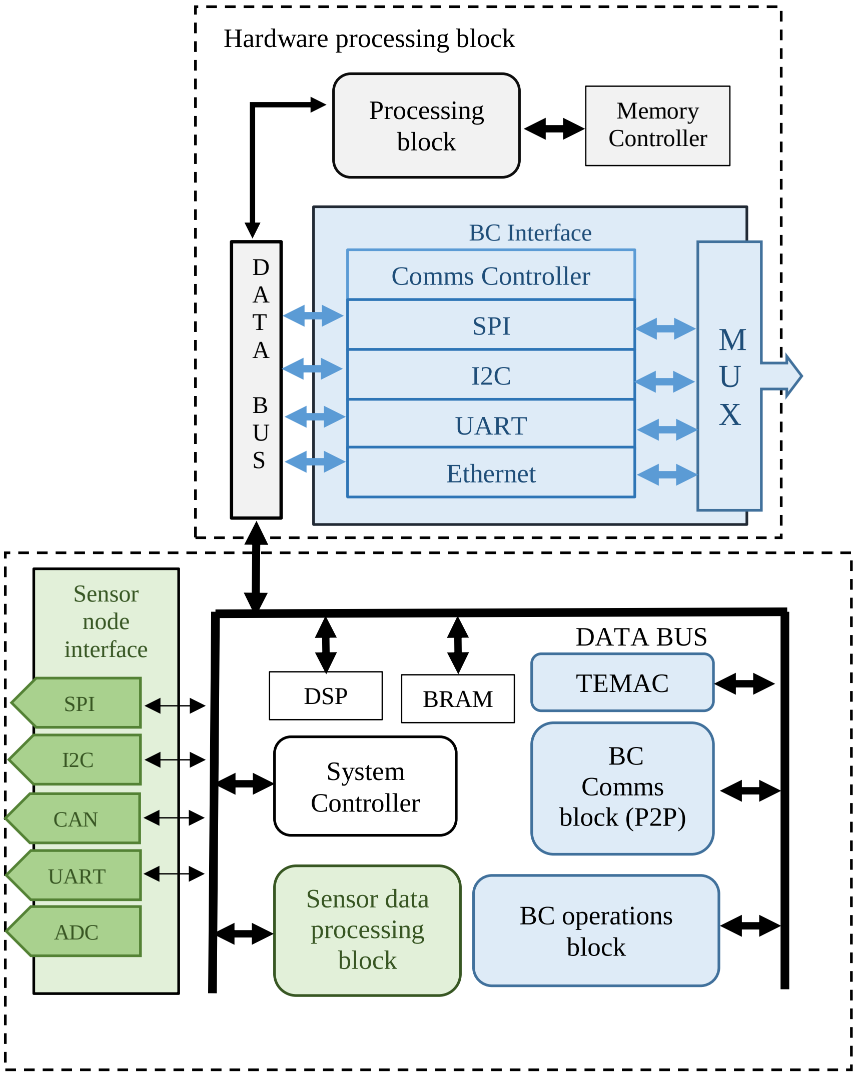

Figure 3.4 FPGA SoC architecture for a sensor node

3.4.2. Experiments and results

Solutions for implementing IoT nodes using FPGA circuits were analyzed in the study we published in [14]. The experiments were conducted in two stages. In the first stage I implemented three of the most widely used communication blocks: SPI, I2C and TEMAC. The aim of the experiment was to identify the resource requirements used to implement these blocks needed for any IoT node. In this way, the resources available for implementing the other BC specific functions can be calculated. In the second stage of the experiment, I have realized an optimized implementation of the SHA256 function, intensively used for mining operations in blockchain architectures using the PoW consensus protocol.

Table 3.5 shows the resources required to implement the most commonly used communication modules expressed in Look-Up-Tables (LUTs). This information is very useful in the process of designing an IoT node and selecting the type of FPGA circuit that will meet the communication needs of the node.

Table 3.5 Communication block resources needed for implementation

Module	LUTs
TEMAC	1256
SPI	164
I2C	216

In the second part of the experiments, we have made a runtime-optimized implementation of the SHA 256 function. The SHA 256 module was implemented in 64 clocks for a 512-bit input block, using only 1152 LUTs.

This implementation of the SHA 256 function was realized using clocks from several FPGA families. Implementations were performed on both resource-constrained circuits and circuits with multiple available resources within the same family.

From the results presented in Table 3.6 it can be deduced that circuits with very few resources and low power consumption can be used for the implementation of the SHA 256 function at an acceptable speed. FPGA platforms containing Virtex 7 type circuits can be intensively used for validation. In these cases, speeds close to 1 Tbit/s have been achieved. If power consumption is important, an optimal variant can be chosen in terms of speed/power ratio. In this case, the best results were obtained with the Artix 7 FPGA XC7A12T and the Zynq 7000 FPGA SoC XC7Z007S.

Table 3.6 Results of SHA 256 implementation on different FPGA circuits

FPGA family	Architecture type	Available LUTs	SHA256 cores	Maxim frequency (MHz)	Speed (Gbit/s)	Curent Iccq(mA)
Spartan 6 – XC6SLX9	FPGA	5720	3	69,46	1,66	4,9
Spartan 6 – XC6SLX150T	FPGA	92152	78	69,46	43,44	63
Artix 7 – XC7A12T	FPGA	8000	5	138,7	5,5	51
Artix 7 – XC7A200T	FPGA	134600	115	138,7	127,6	268
Kintex 7 – XC7K70T	FPGA	41000	34	151,5	41,2	208
Kintex 7 – XC7K480T	FPGA	298600	257	151,5	311,4	840
Virtex 7 – XC7V585T	FPGA	364200	314	196,1	492,6	1597
Virtex 7 – XC7VX1140T	FPGA	712000	473	196,1	966,3	3698
Zynq 7000 – XC7Z007S	FPGA SoC	14400	12	138,7	13,3	172
Zynq 7000 – XC7Z020	FPGA SoC	53200	46	138,7	51,3	437
Zynq 7000 – XC7Z030	FPGA SoC	76600	66	151,5	79,9	437
Zynq 7000 – XC7Z100	FPGA SoC	277400	240	151,5	290,8	1095

4. Entropy source for use in IoT applications

In this chapter, I propose a model of an entropy source specific to the IoT environment, which extracts its data from motion sensors. The proposed solution is subjected to a complex testing and evaluation methodology, with the aim of identifying the level of entropy generated in different usage scenarios, analyzing the stability of the source in the long term and in case of passive and active attacks. It also sought to identify parameters for optimising the performance of the entropy source in terms of power consumption and generation speed.

4.1. Entropy source with sensor data

One solution that can meet the specific requirements of an IoT platform is an entropy source that extracts its noise from data collected from sensors. This solution can be implemented on any sensor node. The only issues that need to be taken into account are the sensor types and characteristics. Since it uses sensor data, it can be said that it mostly uses resources already existing on the platform. Finally yet importantly, this solution provides full access to the noise source that can be analysed in detail.

Figure 4.1 shows the sensor entropy source that I proposed in the paper [7]. In this architecture, the noise source can retrieve data from the following motion sensors: accelerometer, magnetometer and gyroscope. Following the analysis of the emitted entropy, the least significant 8 bits of the sensor-generated sequences were chosen. These data are converted to digital format and then processed using the SHA256 function. This is used to concentrate the entropy and to smooth the output data. The entropy source is continuously health-checked for errors.

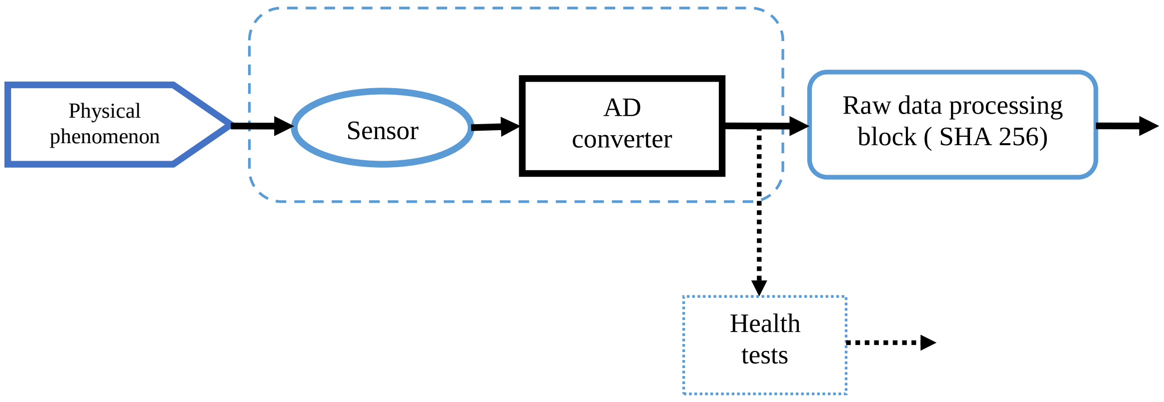

Figure 4.1 Entropy source architecture with sensors

4.2. Entropy source analysis, testing and validation

4.2.1. Entropy estimation methodology

The entropy estimation methodology of the proposed solution was analyzed based on the NIST recommendations in [15]. It involves an initial estimate of the entropy of the noise source and then an estimate of the output of the entropy source, using data collected immediately after its restart. If the estimates obtained at restart are not less than half of the originally estimated entropy value, the final entropy of the source will be the minimum of the previously estimated values, i.e. Hinit - the initial entropy, Hline and Hcol the entropies estimated from the data collected at restart. Entropy is estimated using minimum entropy.

To estimate the initial Hinit entropy, 1,000,000 sequences collected directly from the noise source are used. The length of the sequences are 8 bits. In the case of a motion sensor the least significant bits provide based source the highest unpredictability.

Depending on the type of data the source can generate, entropy is evaluated differently. For non-independent or uniformly distributed data the method with estimators is applied. This method is also used for sensor-based entropy sources. Thus, the entropy of the collected data is calculated using a number of 10 different estimators, which analyse different statistical properties. Finally, the lowest entropy value obtained by the estimators is considered.

If the final entropy is not maximum, a conditioning function can be used (see Figure 4.2). For this purpose we used the SHA 256 function which computes a summary from a number of sequences containing 512 bits of entropy. In this case the SHA 256 function acts as an entropy concentrator. The number of sequences used at the input of the function is calculated according to the number of entropy bits contained in a sequence.

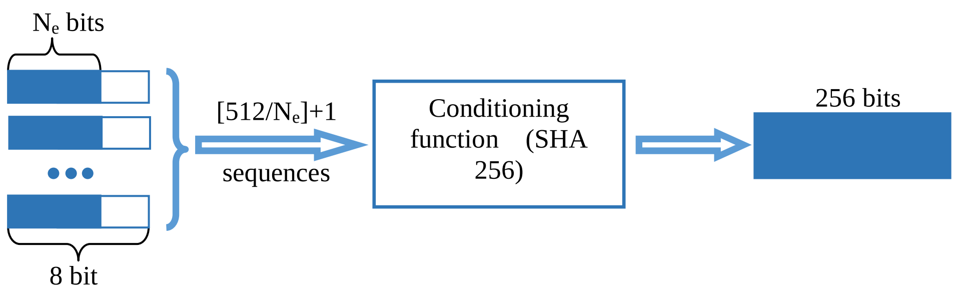

Figure 4.2 Entropy uniformization method

4.2.2 Entropy source analysis, testing and validation methodology

The entropy source must be capable of providing a certain level of entropy under all conditions of use. For this reason, the analysis of an entropy source has to take into account multiple aspects related to the underlying physical phenomenon of the noise source and the stability of the source over time and under different conditions of use. In the following, I will present an original and comprehensive methodology for testing and validating sensor-based entropy sources, based on the NIST recommendations in [15]. In Figure 4.3 a summary of the methodology for motion sensor-based entropy source analysis and evaluation is presented.

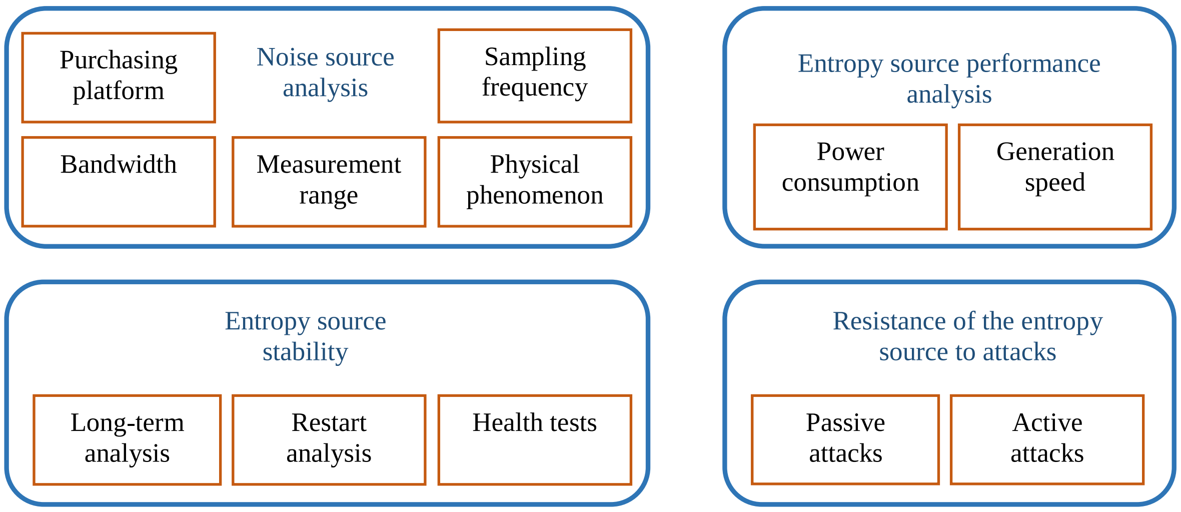

Figure 4.3 Entropy source analysis and assessment methodology

4.2.2.1. Noise source analysis

In the analysis of the noise source, we started from its construction model, presented in Figure 4.4, trying to identify the influence of each element in the entropy value of the output data.

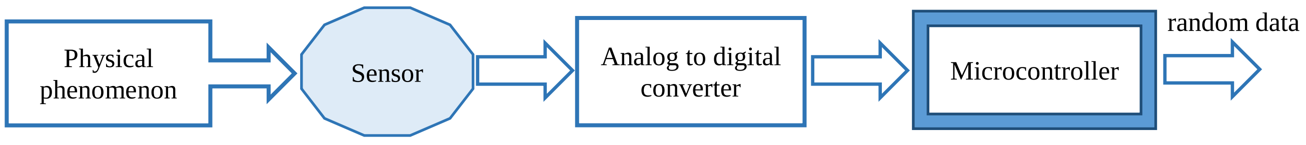

Figure 4.4 Noise source model

The experiments were carried out using three different types of platforms to capture five types of movement.

- Platform 1 is a B-L475E-IOT01A electronic board used as a sensor node for IoT applications;
- Platform 2 consists of an Arduino UNO acquisition board and an MPU 9250 motion sensor, which contains an accelerometer, a gyroscope and a magnetometer.
- Platform 3 uses the motion sensors from a mobile phone. In the experiments, we captured data from the sensors while the sensors were subjected to five types of motion.

The results of the experiments can be found in Table 4.1 for accelerometers in Table 4.2 for gyroscopes and in Table 4.3 for magnetometers. A few observations can be drawn from the analysis of these results:

- in all situations the sensors generated entropy. The larger or faster the motion applied to the accelerometer and gyroscope sensors, the higher the entropy value;
- entropy is achieved even when no additional motion is applied to the sensors;

- a repetitive motion does not negatively influence entropy, on the contrary it achieves the best results.

Table 4.1 Influence of motion type on accelerometer entropy

Movement type / Platform	Static (no movement)	Car on the move	Rotation	Running	Walking
Platform 1	0,1997	0,7424	-	-	-
Platform 2	0,2210	0,6846	0,8544		
Platform 3	-	0,5019	-	0,4021	0,2905

Table 4.2 Influence of motion type on gyroscope entropy

Movement type / Platform	Static (no movement)	Car on the move	Rotation	Running	Walking
Platform 1	0,0551	0,4147	-	-	-
Platform 2	0,1207	0,4348	0,8299	-	-
Platform 3	-	0,3598	-	0,2942	0,2335

Table 4.3 Influence of motion type on magnetometer entropy

Movement type / Platform	Static (no movement)	Car on the move	Rotation
Platform 1	0,4368	0,5690	-
Platform 2	0,0025	0,0049	0,1566

- *Measuring range and bandwidth*

The second factor that can influence the entropy value is the configuration of the sensor parameters used. For this purpose, the possibilities of parameterisation of motion sensors were studied in detail and it was found that some parameters influence the entropy level and others have no effect on it.

The results are presented in Table 4.4 for the accelerometer in Table 4.5 for the gyroscope and in Table 4.6 for the magnetometer.

Table 4.4 *Influence of measurement range and bandwidth on accelerometer entropy*

Bandwidth \ Measurement range	2 g	4 g	8 g
3330 Hz	0,6438	0,5133	0,4194
417 Hz	0,4957	0,3831	0,3382
53 Hz	0,3049	0,2549	0,1904

Table 4.5 *Influence of measurement range and bandwidth on gyroscope entropy*

Bandwidth \ Measurement range	245 dps	500 dps	1000 dps
1250 Hz	0,3931	0,4337	0,3356
312 Hz	0,3401	0,2386	0,2118
22 Hz	0,2094	0,1736	0,1083

Table 4.6 *Influence of measurement range on magnetometer entropy*

Measurement range	4 Gauss	8 Gauss	12 Gauss
	0,4414	0,4137	0,3850

From the results presented in Table 4.4, Table 4.5 and Table 4.6 it is obvious that the entropy level decreases directly proportional to the value of the measurement range and inversely proportional to the bandwidth.

- *Sampling frequency*

The sampling frequency could be an element influencing the entropy value. To determine this, sensor data were acquired at different sampling frequencies. The influence of other factors on entropy values was isolated and tests were performed with only one platform. No additional motion was applied to the sensors. The results are presented in Table 4.7 for the accelerometer and in Table 4.8 for the gyroscope. The conclusion that emerge from these results is that the sampling frequency does not influence the entropy value.

Table 4.7 *Influence of sampling frequency on accelerometer entropy*

Sampling frequency/ Bandwidth	Accelerometer
6660 Hz / 1666 Hz	0,6212
3330 Hz / 1666 Hz	0,6142

Table 4.8 Influence of sampling frequency on gyroscope entropy

Sampling frequency/ Bandwidth	Gyroscope
6660 Hz / 173 Hz	0,2278
3330 Hz / 172 Hz	0,2414

- *Acquisition platform*

Another element that could influence entropy is the acquisition platform. Even if the same type of platform is used, some elements of the platform could introduce additional noise or measurement errors, which can influence the entropy value one way or the other. To investigate this, we compared the entropy values extracted from two identical MPU9250 sensors connected to three Arduino UNO acquisition boards and one Arduino Atmega 256 platform. Tests were performed for two of the sensors: the accelerometer and the gyroscope. The results are presented in Table 4.9 for the accelerometer and in Table 4.10 for the gyroscope. Comparing the entropy values obtained in all the cases analysed, it can be seen that in the case of the accelerometer the entropy values are very close and in the case of the gyroscope there are small but insignificant variations.

Table 4.9 Influence of acquisition platform on accelerometer entropy

Acquisition platform	MPU9250 Sensor - 1	MPU9250 Sensor - 2
Arduino UNO - board 1	0,4618	0,4626
Arduino UNO - board 2	0,4660	0,4618
Arduino UNO - board 3	0,4671	0,4671
Atmega 256	0,4670	0,4571

Table 4.10 Influence of acquisition platform on gyroscope entropy

Acquisition platform	MPU9250 Sensor - 1	MPU9250 Sensor - 2
Arduino UNO - board 1	0,2690	0,2708
Arduino UNO - board 2	0,2371	0,2400
Arduino UNO - board 3	0,2513	0,2586
Arduino Atmega 256	0,2258	0,2387

4.2.2.2. Entropy source stability analysis

- *Long-term analysis*

More important than the amount of entropy source can generate is the ability to maintain this level of entropy over time. This requires analysing the level of entropy generated over time and counting the number of failures that result from health tests. In addition, the behaviour of the source in different operating modes can provide information on how the entropy source is implemented and used.

In Table 4.11 for the accelerometer, in Table 4.12 for the gyroscope and in Table 4.13 for the magnetometer the minimum, maximum and average entropy values calculated for each case are shown. The standard deviation is also calculated to show variations in entropy values over a long time. From the values presented, it can be seen that, the entropy values in all the analysed cases, do not vary very much. The standard deviation is of the order of 10^{-2}.

Table 4.11 Long-term entropy analysis for the accelerometer

Stability analysis parameter / **Platform/movement type**	**Average entropy value**	**Minimum value entropy**	**Maximum entropy value**	**Standard deviation**	**Number of sets**
Platform 1-no movement	0,6499	0,6379	0,6638	0,0058	100
Platform 1- moving car	0,8480	0,8076	0,8878	0,0263	10
Platform 2-no movement	0,4732	0,4503	0,5086	0,0139	30

Table 4.12 Long-term entropy analysis for the gyroscope

Stability analysis parameter / **Platform/movement type**	**Average entropy value**	**Minimum value entropy**	**Maximum entropy value**	**Standard deviation**	**Number of sets**
Platform 1-no movement	0,4154	0,3427	0,4619	0,0222	100
Platform 1- moving car	0,4295	0,3564	0,5132	0,0475	10
Platform 2-no movement	0,2347	0,1840	0,2745	0,0287	30

Table 4.13 Long-term entropy analysis for the magnetometer

Stability analysis parameter / **Platform/movement type**	**Average entropy value**	**Minimum value entropy**	**Maximum entropy value**	**Standard deviation**	**Number of sets**
Platform 1-no movement	0,4411	0,4189	0,4658	0,0108	100
Platform 1- moving car	0,5521	0,4964	0,5970	0,0268	10

- *Restart analysis*

Entropy sources may operate differently immediately after restart compared to operation in a normal regime.

In order to determine the behavior of the motion sensor-based entropy source on restart, we restarted it 1,000 times, each time collecting 1,000 sequences from each sensor. After each stage of data collection, the source was shut down for a 15-minute interval, the time required to simulate the platform returning to a resting state. From the collected data, a matrix of $M_{i,j}$

with 1,000 rows and 1,000 columns was created. Concatenating the data by rows and then by columns creates two data sets.

For the data collected, health tests were validated using the software library in [16]. In Table 4.14 the entropy values obtained for the two datasets are presented. It can be seen that the entropy values obtained at restart are higher than those obtained in normal operation, thus it can be stated that the entropy source is not negatively influenced by the values collected at restart.

Table 4.14 Restart analysis

Sensor type	**Entropy in normal operation**	**Entropy at restart - concatenated data set on lines**	**Entropy on restart - concatenated data set on columns**
Accelerometer	0,63	0,90	0,88
Gyroscope	0,34	0,47	0,50
Magnetometer	0,41	0,46	0,43

- *Health tests*

The solution presented in this thesis implements two continuous health tests presented in [15]: the repetition test and the adaptive proportions test.

To analyse the stability of the entropy source we estimated over a long period the number of errors reported by the two health tests mentioned above. The conclusions of this analysis are important to decide whether the proposed solution is suitable for use in real applications. If the number of errors were very frequent, the availability of the source would be reduced, as would the speed of generation. To support this analysis, we counted the errors reported by the health tests for different values of probability α. In order to obtain the best possible accuracy of the results, the experiments were performed using a large number of sequences. 100,000,000 sequences were extracted from each of the three sensors: accelerometer, gyroscope and magnetometer. The results are presented in Table 4.15 for the repetition test and in Table 4.16 for the adaptive proportions test.

Table 4.15 Repetition test

Sensor / α	**Accelerometer**		**Gyroscope**		**Magnetometer**	
	C	**Number of errors**	**C**	**Number of errors**	**C**	**Number of errors**
2^{-20}	4	10	8	0	7	0
2^{-15}	3	1075	6	2	5	10

Table 4.16 Adaptive proportions test

Sensor / α	**Accelerometer**		**Gyroscope**		**Magnetometer**	
	C	**Number of errors**	**C**	**Number of errors**	**C**	**Number of errors**
2^{-20}	38	0	120	0	89	0
2^{-10}	30	0	105	0	76	0
2^{-5}	24	0	94	0	67	0

From the analysis of the results presented in the above tables it can be seen that no errors are reported, with one exception. Only in the case of the accelerometer a number of ten errors were reported over the period analysed for the repetition test, which means that the entropy source could generate an error at an interval of 785 days.

4.2.2.3. Resistance to attacks

- *Passive attacks*

In order to analyze the resilience of the presented solution in case of a side-channel attack, I conducted an experiment where I collected data simultaneously with two identical acquisition platforms. In the experiment I tried, as far as possible, to have the data acquisition performed under the same conditions for the two platforms. The sequences extracted with the two platforms were analysed using two mathematical tools used to compare data strings: the Pearson correlation coefficient and the Hamming distance.

Data was extracted for each axis for all three-motion sensors: accelerometer, gyroscope and magnetometer. In order to analyse whether the correlation depends on the entropy level generated by the source, we collected data from the sensors in several situations. Different entropy levels were obtained by parameterizing the sensors.

Table 4.17 shows the Pearson coefficient values calculated for different values of entropy generated by the accelerometer, gyroscope and magnetometer. The analysis was performed on strings containing 512 bits of entropy, twice the size of a symmetric key. Depending on the entropy generated by the sensor, the analysed string has a different size (N_i). As can be seen from the values present in the table, the correlation between the two strings is almost non-existent. It also does not depend on the entropy value or the length of the data string.

Table 4.17 Correlation analysis using Pearson coefficient

Sensor	**Accelerometer**			**Gyroscope**			**Magnetometer**		
Entropy	0,64	0,38	0,19	0,43	0,23	0,10	0,44	0,41	0,38
N_i	100	168	337	148	269	591	145	155	167
P - Y-axis	0,0813	0,0695	0,0520	0,0626	0,0582	0,0807	0,0748	0,0812	0,0665
P - X-axis	0,0744	0,0634	0,0509	0,0636	0,0607	0,0524	0,0603	0,0616	0,0760
P - Z-axis	0,0836	0,0637	0,0649	0,0619	0,0593	0,0581	0,0682	0,0672	0,0733

Table 4.18 shows the values obtained for Hamming distances in all the cases presented above. The table also shows the mean deviations of these values for each sensor and entropy level analysed. It was chosen to present the results in this form because it can provide a better representation of the correlation between the two analysed strings. The calculation formula for the mean deviation is shown in (4.1).

$$\text{DH} = \frac{|DHx-4| + |DHy-4| + |DHz-4|}{3} \tag{4.1}$$

Table 4.18 Correlation analysis using Hamming distance

Sensor	**Accelerometer**			**Gyroscope**			**Magnetometer**		
Entropy	0,64	0,38	0,19	0,43	0,23	0,10	0,44	0,41	0,38
Mean deviation	0,03	0,38	0,73	0,02	0,84	1,54	0,13	0,34	0,47
P - Y-axis	4,05	4,13	4,98	3,96	3,22	2,44	4,04	3,74	5,10
P - X-axis	4,01	5,02	3,42	3,98	3,39	2,28	4,26	3,36	4,10
P - Z-axis	3,95	3,99	3,36	4,03	2,84	2,66	3,90	3,86	3,77

From the analysis of the Table 4.18 it can be seen that the Hamming distance values are around 4, which means that the data are not correlated. However, it can be seen from the analysis of the mean deviation values that the degree of correlation increases as the entropy level decreases.

- *Active attacks*

In the case of entropy sources based on data collected from sensors, passive attacks are performed to reduce the entropy level. In this study, we analyzed the source's resistance to four types of attacks:

- the repetitive rotational motion on the sensors was analysed in the noise source analysis. The results showed that the entropy value increased due to more motion being applied to the sensors. The conclusion would be that it is almost impossible to reproduce a perfect repetitive motion so as to cause identical values to be generated;
- sensor saturation - in this way the output values of the sensors could be maximum, reducing the entropy value to zero. To detect such attacks, the proposed solution

has implemented health tests capable of quickly identifying such abnormal behavior;

- sampling frequency could influence the entropy values generated by the source. This issue was analysed in the noise source analysis. The results presented in Table 4.7 and Table 4.8 shows that sampling frequency does not influence entropy;
- ambient temperature. To analyze the influence of this attack on entropy values we collected data from the sensor platform while it was operating at temperature extremes ranging from -18ºC to +82ºC. The estimated entropy values under these conditions were compared with values obtained at a typical operating temperature of +23ºC. Analysing the results presented in Table 4.19 it can be concluded that at low temperatures does not negatively affected the entropy level and high temperatures increase the entropy of the collected data.

Table 4.19 Analysis of the influence of temperature on entropy

Temperature / **Sensor type**	**-18°C**	**+23ºC**	**+82ºC**
Accelerometer	0,6313	0,6379	0,7596
Gyroscope	0,4393	0,4153	0,4637
Magnetometer	0,4500	0,4411	0,6365

4.2.2.4. Entropy source performance analysis

- *Generation speed*

The calculation of the entropy source generation rate was performed for each of the three sensors analysed: accelerometer, gyroscope and magnetometer. Execution times were estimated with the microcontroller's internal clock.

The following aspects have been taken into account in the calculation of the generation speed:

- the entropy source generates 256-bit sequences containing 256 bits of entropy. For this purpose, a sufficiently large number of sequences must be acquired from the sensor to contain 512 bits of entropy (twice as large as the source output, as recommended by NIST in [15]);

- the execution time considered is the time required to extract one sequence for each sensor axis: x-axis, y-axis and z-axis, summed with the time required to execute the SHA256 function.

In Table 4.20 the generation rates obtained for the three sensors are presented together with the extraction times for the sequences and the entropy values used in the calculations.

Table 4.20 Entropy source generation speed

Sensor type	Accelerometer	Gyroscope	Magnetometer
Sequence extraction time (μs)	2560	2560	1540
Entropies per bit sequence	0,63	0,34	0,42
Generation speed (Kb/s)	2,88	1,56	3,12

As can be seen from the results presented in Table 4.20, the highest speed was obtained for the magnetometer, although the accelerometer generates the most entropy per bit. This is because the extraction time for the magnetometer is smaller than that for the accelerometer.

- *Consumption analysis*

The entropy source current consumption analysis was performed for each of the three analysed zenograms: accelerometer, gyroscope and magnetometer. Execution times were estimated with the microcontroller's internal clock.

The following aspects were taken into account in the consumer analysis:

- the entropy source generates 256-bit sequences containing 256 bits of entropy, which means that the number of sequences used is dependent on the entropy value;

- the value of the current consumed by the platform takes into account the current consumed by the sensor I_{DD_S} (as reported in the datasheet), the current consumed by the microcontroller - $I_{DD\_MCU\_S}$ during data collection from the sensor and the current consumed by the microcontroller for the calculation of the SHA256 function (measured on pin JP5 of the microcontroller);

Table 4.21 shows the following information: the number of bits that can be generated with a 1000 mAh battery, the current values and the entropy values used in the calculations.

Table 4.21 Consumption analysis for entropy source

Sensor type	Accelerometer	Gyroscope	Magnetometer
I_{DD_S} (mA)	13,81	13,81	13,81
$I_{DD\_MCU\_S}$ (mA)	0,16	0,49	0,27
Entropies per bit sequence	0.63	0,34	0,42
Number of bits generated (Mb)	485,07	303,30	508,30

As can be seen from the results presented in Table 4.21, the solution using the magnetometer has the lowest current consumption and generate the highest number of bits using the current from a 1000 mAh battery.

5. Random number generator for use in IoT applications

In this chapter, I propose a secure random number generator for use on resource-constrained IoT devices. After analyzing the challenges of implementing random number generators in IoT applications and the security levels they can achieve, a solution that uses limited resources and provides the strongest level of security is proposed. The proposed solution is optimized by performing experiments through which an optimal solution for the noise source and deterministic algorithm is identified. Finally, a security and efficiency analysis is carried out and a comparison is made with a classical solution based on the AES algorithm in CTR mode.

5.1. The proposed solution

The proposed random number generator is designed to be implemented on IoT devices. The solution to be presented below takes into account the resource-constrained requirements specific to IoT devices while ensuring the highest degree of security required of random number generators. The general scheme of the random number generator solution is shown in Figure 5.1. To ensure the security properties while using limited resources, we selected a lightweight algorithm implemented in an Authenticated Encryption with Associated Data (AEAD) scheme. This scheme uses a 128-bit-long message (plaintext). The encryption key used by the algorithm is regenerated for each call of the random number generator function. Since it represents the output of an entropy source, its value cannot be estimated. At each iteration of the algorithm the nonce, a randomly generated number at the initialisation of the generator using the entropy source, is incremented.

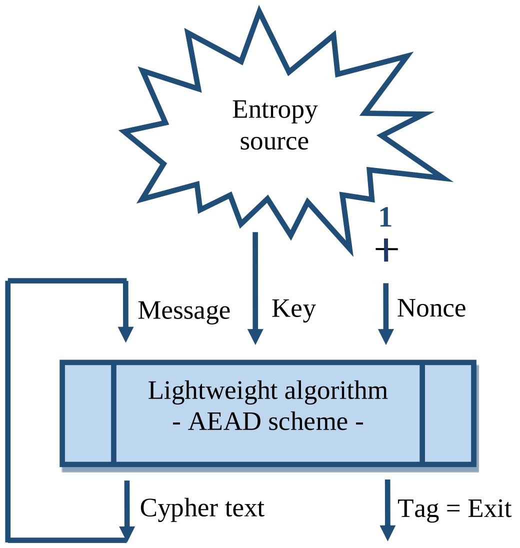

Figure 5.1 Implementation scheme of the random number generator

All inputs to the generator are generated using the entropy source in the initialization phase. The outputs of the generator are 128-bit blocks of data. To generate larger data sequences the algorithm runs in a loop, in the sense the ciphertext is used as message in the next iteration. The output data is represented by the tag value, which is an authentication code for the message. This scheme of using output data has the advantage that it allows variable length data to be generated. The fact that the use of the tag for the output data was chosen has the advantage that the internal parameters of the algorithm, such as the message and ciphertext, are not exposed at the output of the generator for use by a possible attacker.

The entropy source extracts its randomness from motion sensor data. This solution was chosen because it is very easy to implement in many IoT applications that use such sensors, and no additional hardware is needed to realize it. The MPU 9250 multi-chip module was used to implement the entropy source. It contains three types of MEMS (Micro Electro-Mechanical Systems) three-axis sensors: accelerometer, gyroscope and magnetometer.

The data extracted from the sensors is digitised on 16 bits, but not all of these bits are entropy carriers. From the analysis I conducted in the [8] on similar sensors, with the same settings, data being collected when the axes were not moving, it can be seen (Figure 5.2 for the accelerometer and Figure 5.3 for the gyroscope) that only the least significant bits on each axis can generate entropy.

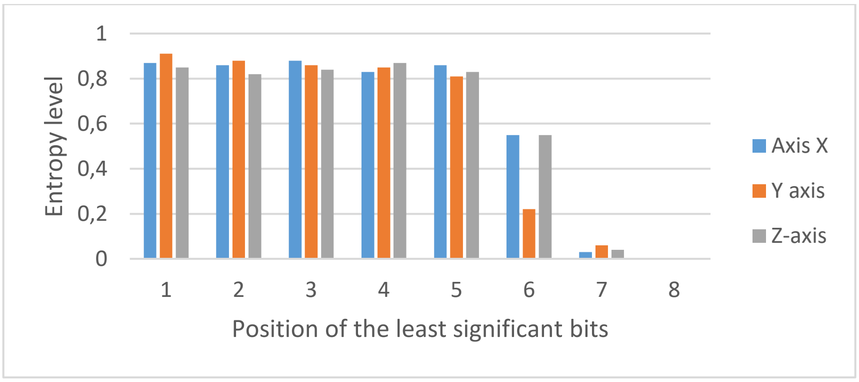

Figure 5.2 Estimated entropy for the bits on each axis for the accelerometer

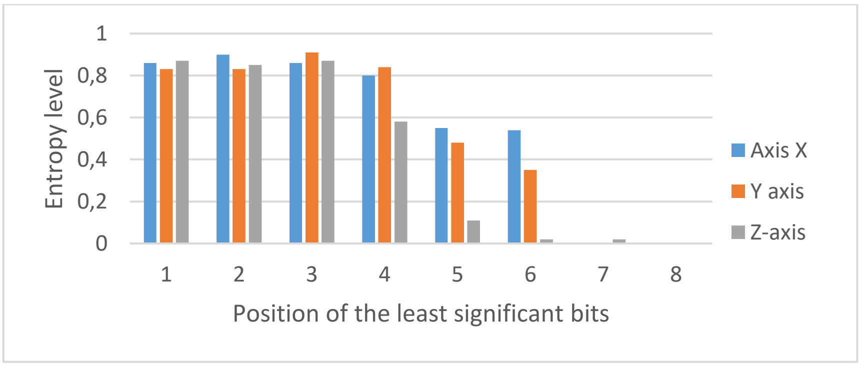

Figure 5.3 Estimated entropy for the bits on each axis for the gyroscope

Given this, the proposed entropy source solution extracts the least significant 4 bits from each sensor axis, which it concatenates to create 128-bit sequences needed for the DRBG inputs.

5.2. Analysis and evaluation of the proposed solution

5.2.1. Security analysis

In order to perform the security analysis of the proposed solution, I have taken into account the requirements proposed by the German Federal Office for Information Security in the methodology for the evaluation of random number generators AIS 20/ AIS 30, originally published in 2011 in [17] and updated in 2022 in [18]. This methodology supports the security assessment of random number generators using Common Criteria standard.

The solution proposed in this study complies with the functional requirements of PTG.3 class according to AIS 20/AIS 30 recommendations. This class recommends the most secure random number generator scheme with the following security properties:

- Backward secrecy;
- Forward secrecy;
- Enhanced backward secrecy;
- Enhanced forward secrecy;

The last, but most important, aspect concerns the randomness of the data generated. We evaluated the random number generator using the NIST_STS statistical test battery. The tests were performed on data sequences of 131,072 bytes. Since a statistical evaluation is more accurate the larger the volume of data tested, we applied the tests for a number of 1,000 distinct sequences for each case analyzed.

The test results are presented in Table 5 1. Analysing the results, it can be seen that for all cases, only one or two tests or subtests failed. Considering this, as well as the fact that all relevant tests passed, it can be considered that all the solutions analysed show very good randomness properties.

Table 5 1 Statistical test results

RNG type	Generated data output size	Past tests	Failed tests
RNG_Comet	128 bits	186 of 187	The Non-overlapping Template Matching Test - 1 subtest
	4096 bits	187 of 187	
	1 Mb	187 of 187	
RNG_Sparkle	128 bits	186 of 187	The Random Excursions Variant Test - 1 subtest
	4096 bits	187 of 187	
	1 Mb	185 of 187	The Non-overlapping Template Matching Test - 2 subtests
RNG_Romulus	128 bits	185 of 187	The Non-overlapping Template Matching Test - 2 subtests
	4096 bits	185 of 187	The Non-overlapping Template Matching Test - 2 subtests
	1 Mb	183 of 187	The Discrete Fourier Transform (Spectral) Test The Non-overlapping Template Matching Test - 3 subtests
RNG_Photon	128 bits	187 of 187	
	4096 bits	186 of 187	The Random Excursions Variant - Test 1 subtest
	1 Mb	186 of 187	The Discrete Fourier Transform (Spectral) Test

5.2.2. Efficiency analysis

In order to identify the solution that offers the best efficiency in terms of power consumed and resources required for implementation, I conducted a series of experiments. The elements we considered in the analysis were: making data collection from the entropy source more efficient, identifying a cryptographic algorithm for DRNG that would require the least resources, and implementing the solution on a platform that can also be used in IoT applications.

In order to identify the optimal entropy source I analysed the entropy generated by the accelerometer and gyroscope. In order to obtain the highest entropy level in the situation where the sensors are not moving I analysed in the paper [7] the possibility of parameterization of the sensors. Thus, I found that the bandwidth and the measurement range can influence the entropy level generated by the sensors. The tests showed that setting the measurement range to the lowest value (2g for accelerometer and 245 dps for gyroscope) and the bandwidth to the highest value (3330 Hz for accelerometer and 1250 Hz for gyroscope) gives the highest values for entropy.

Since we noticed that the data extraction times from the sensors are different, we tried different combinations for the two sensors. In Table 5.2 the times required for data extraction in these cases are presented, together with the entropy level of the extracted data.

Table 5.2 Entropy source generation speed

Sensor type	Axe	Entropy	Extraction time for 1 entropy bit (ns)
Gyroscope	X	4,35	204,60
Gyroscope	Y	4,51	197,34
Gyroscope	Z	4,43	200,90
Gyroscope	*XYZ*	*4,48*	*96,73*
Accelerometer	X	2,71	697,42
Accelerometer	Y	3,02	625,83
Accelerometer	Z	2,92	647,26
Accelerometer	XYZ	3,3	232,32
Accelerometer / Gyroscope	XYZ	3,4	157,84

The following optimization was aimed at identifying an optimal lightweight algorithm variant for the DRNG implementation. An Arduino Mini board was used to implement the algorithms. Tests were performed for different lightweight algorithms. To demonstrate that the proposed solution brings performance improvements, we compared the results obtained with the AES_CTR solution proposed by NIST in [6].

First, I looked at the resources needed to implement the solution. Table 5.3 shows, for each implementation, the storage space for the source code and the dynamic memory required to run it. The values are given in number of bytes and as a percentage of total available memory. Analysing the results it can be seen that the solutions presented require a memory capacity comparable to the AES_CTR variant, with the exception of the Romulus algorithm. In contrast, AES CTR requires about 30% more dynamic memory.

Table 5.3 Resources needed to implement the RNGs

Type RNG	**Space stored program**		**Dynamic memory**	
	Byte count	**Percent**	**Byte count**	**Percent**
RNG_Photon	9322	28%	601	29%
RNG _Sparkle	7418	22%	523	25%
RNG _Romulus	15584	48%	523	25%
RNG _Comet	8226	25%	523	25%
RNG_AES_CTR	8494	26%	841	41%

Second, I analyzed the random number generation speeds for all five implementations. Tests were performed for generating the most common key sizes (128, 256, 512 , 1024, 2048, 4096 bits) but also longer sequences of 1Kb and 10 Kb. Analysing the results presented in

Figure 5.4 it can be seen that only the solutions implemented with the Comet and Sparkle algorithms manage to achieve better generation speeds than the AES_CTR variant.

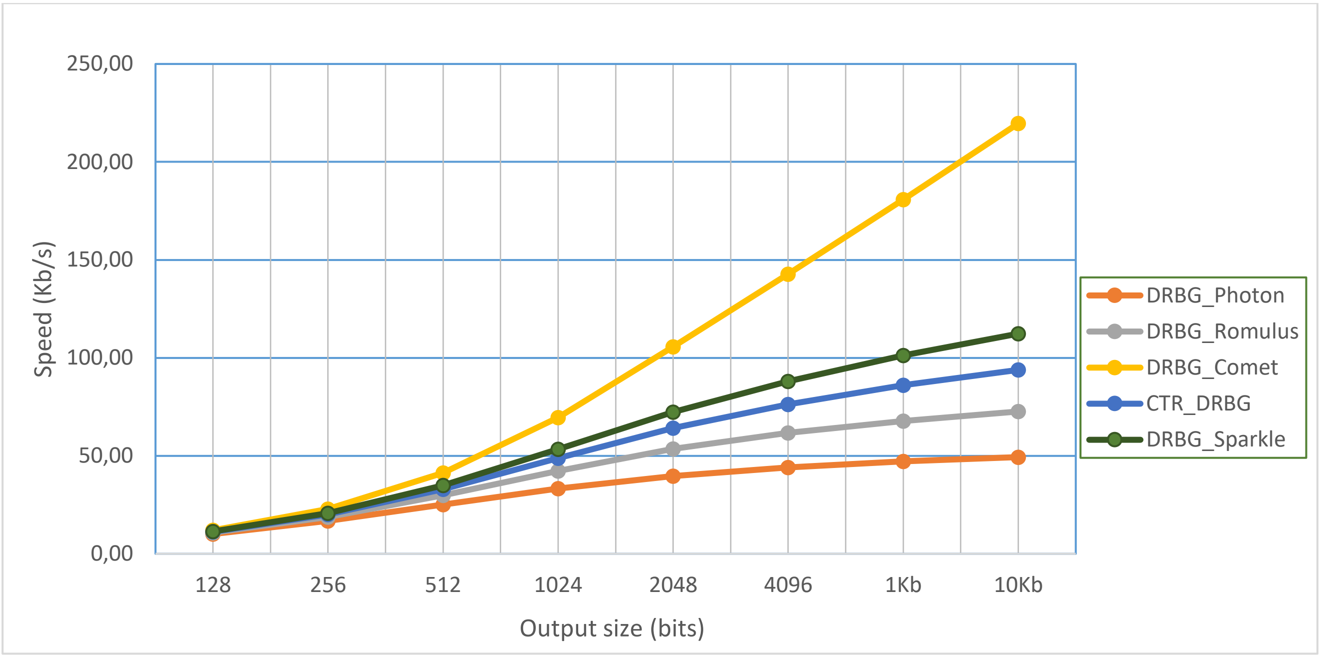

Figure 5.4 RNG generation speeds

We also looked at the DRNG variants in terms of power consumption. Measurements were made, using the FLUKE 8864A professional multimeter to measure the current on the power supply of the Arduino Mini board while running on it the program that generates 128-bit sequences, re-initialization being performed at each iteration. Since the MPU9250 sensor is powered from the Arduino Mini board, the measured current represents the total amount required to run the RNG. With this value available, it can be accurately estimated the battery capacity required to power this solution for generating a given amount of random data.

Table 5.4 shows the power consumed by the five solutions and the number of keys generated with a 1000 mAh battery. The power was calculated by multiplying the current consumed by the Arduino Mini board by its supply voltage of 5V. The current value is an average of the values recorded over a period of 10s. The current consumption of the microcontroller measured in the standby state is 0.257 mA, much lower than in the running state. Thus, it can be stated that the number of keys generated with a 1000 mAh battery can also be obtained under real conditions.

Table 5.4 Consumption analysis for RNG solutions

Type RNG	**Power consumption (mW)**	**Number of keys generated with a 1000 mAh battery**
RNG_Sparkle	87,0	$18.91*10^6$
RNG_Comet	86,5	$19.98*10^6$
RNG_Photon	88,0	$16.53*10^6$
RNG_Romulus	88,5	$17.59*10^6$
AES_CTR	86,5	$18.62*10^6$

Analysing the values in Table 5.4 it can be seen that the best performance can be obtained using the Comet algorithm, followed by Sparkle and the AES_CTR solution.

6. Conclusions

The analysis of the current security context in IoT infrastructures indicates that there are still enough problems to be solved in this area. In this thesis, several solutions are presented that bring improvements to ensure the security of data carried on IoT devices.

The thesis includes original theoretical and practical contributions in the field of ensuring data security in restrictive IoT environments. Thus, architectural solutions, integration of different technologies, analysis methodologies and security or efficiency analyses are proposed. The main contributions are described in detail below:

- Architectural solution for integrating blockchain technology into a fog computing IoT infrastructure. Following an analysis to identify the type of IoT architecture for which the implementation of BC functionalities is feasible, I proposed the optimal solution;
- Architectural solutions for implementing the functionalities of an IoT sensor node and a BC node on FPGA platforms. I proposed two solutions, taking into account power consumption. I have implemented essential components to provide the functionalities of a sensor node and a BC node on FPGA platforms with different resources;
- Simple, efficient and secure protocol for establishing session keys for deployment on IoT nodes. The protocol was implemented on a platform used in IoT, equipped with a microcontroller. I performed an analysis in terms of power consumption and execution speed. The solution uses the Ethereum blockchain platform as a trusted source, using a smart contract for this purpose. The proposed solution has been benchmarked in terms of efficiency and cost against the classical TLS - PKI protocol solution;
- Efficient entropy source solution with data extracted from sensors. The solution has been optimized by identifying sensor parameters to generate maximum entropy under different conditions of use. The generated entropy level was evaluated using NIST standard metrics. I realized a performance analysis of the source in terms of power consumption and generation rate;
- Original noise source analysis methodology that collects data from motion sensors. The analysis was performed considering the source components: physical phenomenon, sensor and acquisition platform. For this purpose, I realized a series of experiments highlighting the influence of each element on the entropy value;
- Methodology of entropy source stability analysis. The analysis was carried out on two levels. I performed a long-term analysis to identify behavior of the source in time and an analysis of the entropy level after restart;

- Methodology for analysing the attack resistance of an entropy source collecting sensor data. I have detailed and performed two types of attacks. The first type of attack was passive. It was performed on a side-channel, attempting data estimation using an identical acquisition platform. The second type of attacks were active. I constructed a series of attacks to detect source behaviour under cyclic motion, saturation, operation at extreme temperatures and changes in sampling frequency;
- Original, secure and efficient random number generator solution for use in restrictive IoT environments. The implementation of the solution was performed on a resource-constrained microcontroller. I performed an analysis from the point of view of consumed resources, power consumption and security, including the estimation of the randomness of the generated data, security properties and cryptographic strength of the deterministic component.

Bibliography

[1] M. Hogan, B. Piccaretta, "NISTIR 8200 - Interagency Report on the Status of International Cybersecurity Standardization for the Internet of Things (IoT)", November 2018

[2] https://ro.wikipedia.org/wiki/Internet, accesat la 16.03.2021

[3] https://www.statista.com/statistics/1194682/iot-connected-devices-vertically/ accesed on 16.03.2022

[4] *2018 Internet Security Threat Report*, Symantec Corporation, March 2018.

[5] A. Boteanu, F. Răstoceanu, I. Rădoi, C. Rusea, " Modeling and simulation of electromagnetic shielding for IoT sensor nodes case", 2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD), 10-12 October 2019

[6] E. Barker, J. Kelsey, "NIST Special Publication 800-90C (Second Draft)-Recommendation for Random Bit 5 Generator (RBG) Constructions", Aprilie 2016

[7] F. Rastoceanu, R. Rughinis, S.D. Ciocirlan, M. Enache, "Sensor-Based Entropy Source Analysis and Validation for Use in IoT Environments", Electronics , 10(10), 1173, 2021

[8] F. Rastoceanu, B.I. Ciubotaru, I. Radoi, C.V. Marian, "Extented Analysis Using NIST Methodology of Sensors Data Entropy, U.P.B. Sci. Bull., Series C, Vol. 83, Iss. 2, 2021

[9] X. Zhu, I. Badr, "A Survey on Blockchain-based Identity and Access Management Systems for Internet of Things", 2018 IEEE Confs on Internet of Things, Iulie 2018

[10] D. B. Rawat , V. Chaudhary, R. Doku, "Blockchain Technology: Emerging Applications and Use Cases for Secure and Trustworthy Smart Systems", *J. Cybersecur. Priv.* , *1*(1), 4-18, 2021

[11] Michaela Iorga, Larry Feldman, Robert Barton Michael, J. Martin, Nedim, Goren Charif Mahmoudi, NIST Special Publication 500-325 - Fog Computing Conceptual Model, March 2018, https://doi.org/10.6028/NIST.SP.500-325

[12] F. Rastoceanu and R. Rughinis, "Blockchain Solution for Securing Fog-Computing Communications in IoT Applications," 2022 14th International Conference on Communications (COMM), 2022, pp. 1-6, doi: 10.1109/COMM54429.2022.9817211

[13] https://www.clickssl.net/low-cost-rapidssl-certificate, accesat la 15.09.2023

[14] F. Rastoceanu, I Radoi, "FPGA based architecture for securing IoT with blockchain " 2*019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD),* 2019, pp

[15] M.S. Turan, E. Barker, J. Kelsey, K. McKay, “NIST Special Publication 800-90B-Recommendation for the Entropy Sources Used for Random Bit Generation”, Ianuarie 2018

[16] https://github.com/usnistgov/SP800-90B_EntropyAssessment, accesat la data de 05.10.2023.

[17] W. Killmann, W. Schindler, “A proposal for: Functionality classes for random number generators”, Septembrie 2011

[18] Matthias Peter, Werner Schindler, A Proposal for Functionality Classes for Random Number Generators Version 2.35 – DRAFT, September 2, 2022