electronics

Article

Secure Sensor Prototype Using Hardware Security Modules and Trusted Execution Environments in a Blockchain Application: Wine Logistic Use Case

Antonio J. Cabrera-Gutiérrez [1,2], Encarnación Castillo [2], Antonio Escobar-Molero [1], Juan Cruz-Cozar [1,2], Diego P. Morales [2] and Luis Parrilla [2,*]

1 Infineon Technologies AG, Am Campeon 1-15, 85579 Neubiberg, Germany
2 Department of Electronics and Computer Technology, University of Granada, Avda. de Fuente Nueva s/n, 18071 Granada, Spain
* Correspondence: luis@ugr.es

Abstract: The security of Industrial Internet of Things (IIoT) systems is a challenge that needs to be addressed immediately, as the increasing use of new communication paradigms and the abundant use of sensors opens up new opportunities to compromise these types of systems. In this sense, technologies such as Trusted Execution Environments (TEEs) and Hardware Security Modules (HSMs) become crucial for adding new layers of security to IIoT systems, especially to edge nodes that incorporate sensors and perform continuous measurements. These technologies, coupled with new communication paradigms such as Blockchain, offer a high reliability, robustness and good interoperability between them. This paper proposes the design of a secure sensor incorporating the above mentioned technologies—HSMs and a TEE—in a hardware device based on a dual-core architecture. Through this combination of technologies, one of the cores collects the data extracted by the sensors and implements the security mechanisms to guarantee the integrity of these data, while the remaining core is responsible for sending these data through the appropriate communication protocol. This proposed approach fits into the Blockchain networks, which act as an Oracle. Finally, to illustrate the application of this concept, a use case applied to wine logistics is described, where this secure sensor is integrated into a Blockchain that collects data from the storage and transport of barrels, and a performance evaluation of the implemented prototype is provided.

Keywords: Blockchain Oracle; hardware security modules; Hyperledger Fabric; secure sensor; trusted execution environment; Trusted Firmware-M; wine logistic

check for updates

Citation: Cabrera-Gutiérrez, A.J.; Castillo, E.; Escobar-Molero, A.; Cruz-Cozar, J.; Morales, D.P.; Parrilla, L. Secure Sensor Prototype Using Hardware Security Modules and Trusted Execution Environments in a Blockchain Application: Wine Logistic Use Case. *Electronics* **2023**, *12*, 2987. https://doi.org/10.3390/electronics12132987

Academic Editors: Alexander Barkalov, Larysa Titarenko, Dariusz Kania and Remigiusz Wiśniewski

Received: 13 June 2023
Revised: 4 July 2023
Accepted: 5 July 2023
Published: 7 July 2023

1. Introduction

In Industrial IoT (IIoT) networks, the extraction of data from the physical to the digital world via sensors is becoming increasingly important [1]. This data collection takes place at nodes distributed within a specific network topology at the edge that are equipped with various sensors that convert physical measurements into digital information. In Industry 4.0, these edge devices are the most important part of the IIoT system because, in addition to extracting data, they can also perform computations at the edge without having to delegate decisions and data flows to a centralised entity [2].

Figure 1 shows the typical IIoT architecture, where different layers have been considered. The edge/perception layer is responsible for collecting data from the outside world through different types of sensors. These sensors are usually attached to an IoT device, which has a communication system to send these data or the necessary information to higher layers of the system. The network layer is responsible for sending these data to the various technologies that process the information. Technologies such as Wi-Fi and Bluetooth Low Energy (BLE) are included in this layer, as well as the hardware devices that perform this task: routers, gateways, etc. The processing layer includes the technologies that allow information to be stored and processed, such as databases, data analysis systems or control software. An even higher layer can be defined in terms of system application, where data and information are delivered to the client in different ways depending on the use case [3].

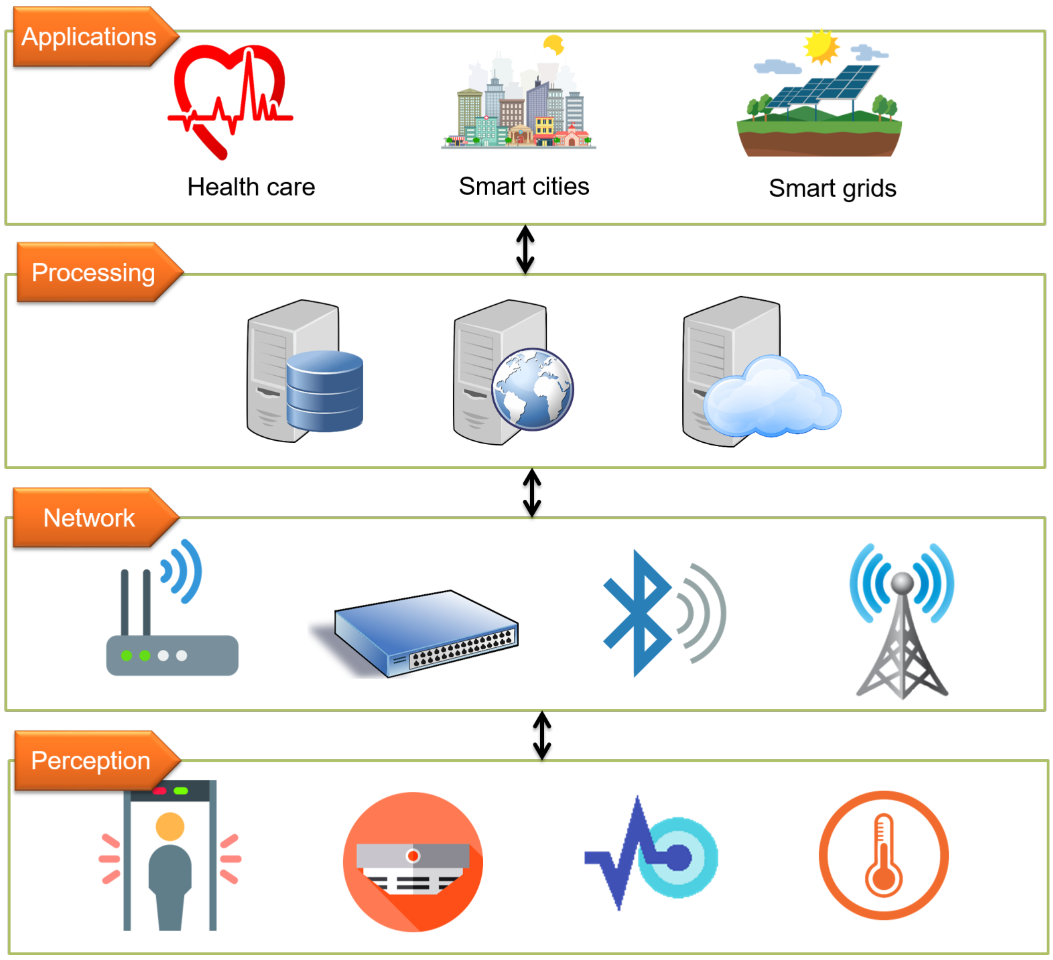

Figure 1. IIoT architecture.

Nowadays, providing security for IIoT devices and the overall system that they are a part of is critical [4]. At first glance, it might seem that, by providing communications security, the system would be protected, but nothing could be further from the truth. Although one of the main aspects is to protect communications, as these provide the largest attackable area of the system [5], this is not enough. In this type of system, paradigms such as traditional client–server-based protocols and implement access control through a public key infrastructure (PKI) are no longer enough [6].

New Decentralised Ledger Technologies (DLTs) [7]-based paradigms such as Blockchain are becoming a fundamental scheme for ensuring the security of these IIoT systems [8]. Through decentralisation and the underlying cryptography, Blockchain offers secure machine-to-machine communication in which every piece of information or data entered into the network has to be validated by different entities that make up this network [9].

The underlying cryptography in this type of network is normally based on asymmetric cryptography, where Rivest–Shamir–Adleman (RSA) [10] or Elliptic Cryptographic Curve (ECC) [11] algorithms are used intensively. To protect the cryptographic keys, it is essential that the devices that make up these networks have additional protection at the hardware level.

To protect IIoT devices, technologies such as Trusted Execution Environments (TEEs) [12] and Hardware Security Modules (HSMs) [13] provide firmware and hardware security capable of repelling attacks such as software and physical attacks [14,15]. HSMs offer rock-solid and certified protection against side-channel attacks, such as fault injection [16]. However, they do not typically support flexible high-level functionality, such as programmable device drivers or secure peripherals. The proposed idea is to combine a non-programmable HSM with a flexible TEE, (i.e., Trusted Firmware-M (TF-M) [17] running in the ARM Cortex-M0 core of a PSoC64 [18]), obtaining the advantages of both solutions.

Designing IIoT devices using TEEs and HSMs is not a trivial task; furthermore, the integration of these devices with Blockchain networks has not yet reached a desired level of maturity [19]. Blockchain networks are increasingly being used and the fields of application are numerous [20–23]. However, there is still much room for improvement in the application of these technologies, as many of them only focus on the deployment and interconnection of devices, creating a decentralised ledger where information can be shared. A typical problem to be addressed is the introduction of data from the off-chain world to the Blockchain world through the so-called Blockchain Oracles [24].

In addition to the design that combines an HSM with a TEE in an IIoT device, this work proposes the integration of the IIoT node within a Blockchain network working as an Oracle. Furthermore, as an application example, the use of this device in a goods logistics use case, such as the transport and storage of wine, is described.

The solution proposed in this article is based on the union of a TEE and an HSM in an IIoT node that fits in Blockchain networks working as an Oracle, providing data securely to the Blockchain. Based on this, this work contributes to create a secure sensor concept in a dual-core platform that extracts data from the physical world (off-chain world) in a secure way. The feasibility of this concept is demonstrated in the application of a use case on wine logistics.

The rest of the article is organised as follows: Section 2 describes the different approaches and categorisations regarding security in IIoT nodes. Section 3 presents the proposed design of the IIoT node describing the HSM and the TEE used in the prototype and how they work together in order to achieve the desire security level. In Section 4, the integration of this IIoT node with the Blockchain is described, with special emphasis on the functioning as a Blockchain Oracle. Section 5 presents the use case oriented to the wine logistics, and the entire system is described. Finally, Section 6 summarises the conclusion, emphasising the benefits and the applicability of this proposal.

2. Related Works

IIoT node design determines the basis on which a system can be built upon. Just as in the construction of buildings, without a good foundation, the building will fall. It is significant that, despite the importance of these devices, there are hardly any specific designs or mentions of the safety of these devices in the literature. In [25], it is concluded that an unsecured hardware platform would lead to an unsecured software stack, highlighting the importance of hardware security. Other studies such as [26] conclude that hardware security is not treated with the same relevance as software security in this kind of system.

Although talking about hardware security is not very generic, there are some studies, such as [27]; in this work, the authors proposed a multi-core intrusion detection for embedded systems. They implemented a secure monitor that continuously monitors the execution behaviour of the controller, which works with an on-chip hardware unit called a timing trace module. In [28], the authors implemented a host-based intrusion detection system in a Field Programmable Gate Array (FPGA) hardware platform that is in charge of detecting attacks in real-time. In the work [29], the authors developed a hardware monitor that operates in parallel to the embedded processor and detects any attack that causes a wrong behaviour. In the recent work [30], the proposed framework (called C4IIoT) helps to detect and mitigate attacks on a complete IIoT system through offloading mechanisms and machine learning techniques. In particular, this work mentions the protection of nodes with HSMs. In [14], an IoT node solution is proposed that incorporates a special type of HSM, a Trusted Platform Module (TPM), for hardware security and integration with the Blockchain.

Apart from C4IIoT, recent studies have emerged, such as: building a complete identity verification framework based on Physical Unclonable Functions (PUFs) [31], FPGA hardware security for data centres [32] and a re-configurable hardware-based isolation and protection mechanism for IoT devices [33].

While there does not seem to be much literature on IoT devices capable of providing device-level security using HSMs, there exists an extensive literature on IoT devices design using TEEs. In this case, works such as [34] propose a solution combining Intel SGX [35] with Bluetooth security. Another solution proposed in [36] offers a system that protects IoT nodes and integrates them to the Blockchain through a TEE. In the work [37], the authors introduced a solution that enables security through ARM TrustZone by encrypting memory and managing access control, protecting sensitive data. The authors of [38] proposed a system for IoT devices that offers security-enhanced attestation for remote terminals based on Intel SGX. In [39], the authors used a TEE (ARM TrustZone [40]) in an IoT device to create a secure enclave. In [41], the authors introduced a system developed using ARM TrustZone in which the critical applications are running securely and communicate with other parts of the system.

Works such as those cited above highlight several factors: the low cost of the application of TEEs compared to HSMs, the ease of the integration of those in IoT devices since they do not involve introducing hardware changes and, finally, a greater familiarity and acceptance of these technologies by the scientific community, since they are easier to understand, manipulate and modify. However, a promising idea that this work exploits is the combination of these two technologies, which, as will be seen below, offers substantial advantages for creating a robust and secure system. Works combining these two technologies are not very abundant, although some cases can be found.

This is the case of [42], which introduces a design of an IoT node that incorporates ARM TrustZone for attestation; in addition, attested data are secured storage in a TPM. In the proposal [43], a secure logging system from end-to-end between embedded constraint devices and a remote database is implemented, and the architecture design combines a TEE with a TPM.

These proposals that integrate these two technologies are close to the work proposed in this article, but the use of both software and hardware protection is not oriented towards the general purpose of an IoT node but rather focuses on specific mechanisms such as attestation or to protect a logging system. The main innovation of this work on the state of the art described above is the incorporation of a TEE and an HSM in an IoT device to guarantee the reliability and integrity of the data read by the sensors that the device incorporates, creating a root of trust from the source. In addition, integrating these nodes with Blockchain technologies makes the data trustworthy even after they have left the device. To illustrate this, the use case of wine logistics is described.

3. Design Proposed

Traditionally, IoT device architectures that incorporate hardware security attach an HSM to the main board that is used to perform the necessary cryptographic operations: encryption of communications, data signing, public–private key generation, etc. In this type of architecture, the microcontroller is in charge of scheduling the necessary cryptographic tasks, delegating them to the HSM and receiving the response from it. On the other hand, the communication is normally carried out through standardised buses such as Inter-Integrated Circuit (I2C) and Serial Peripheral Interface (SPI). In the event that it is necessary to sign the data received by a sensor to ensure their integrity, the microcontroller will receive them from the sensor and then delegate the signing operation to the HSM. This approach, which can be seen in Figure 2, shows a serious drawback: when the data arrive to the microcontroller, an attacker can modify the data without being detected. In the case where the modification is made after the data are signed, this manipulation can be detected at the verification process and the data will be discarded.

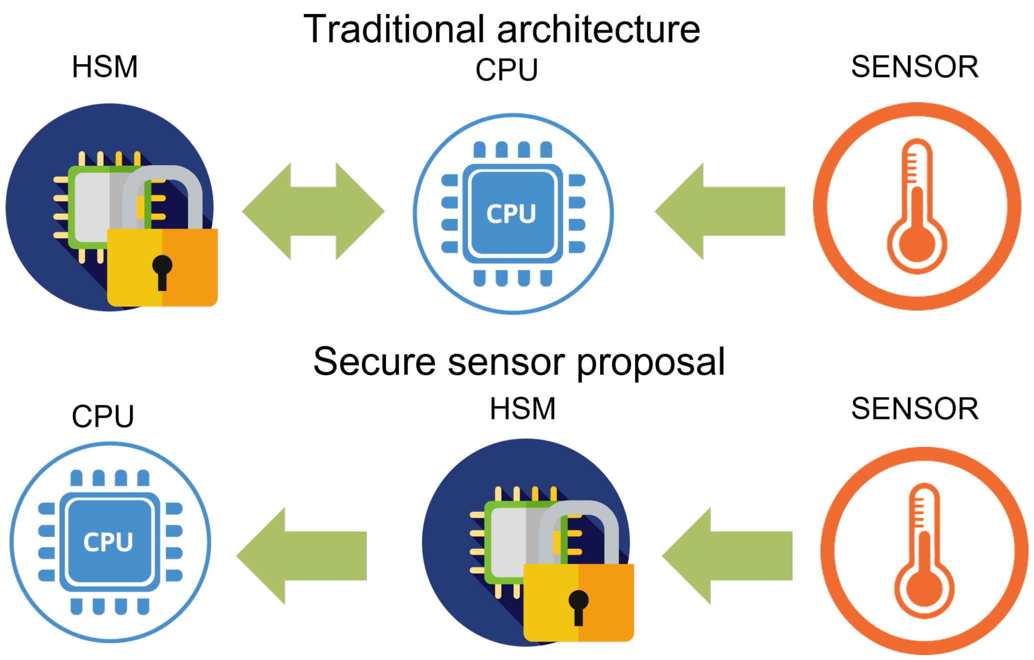

Figure 2. Design proposed.

In this sense, the secure sensor proposed in this article establishes that the data are sent directly to the HSM and, there, the HSM is in charge of sending it to the microcontroller once it is signed. In this way, if any change is made, it is automatically detected in the verification process, since the data have been previously signed within the secure environment that an HSM offers. Both an HSM and a TEE can play this secure environment role within this proposal.

The design of a hardware-secure IoT node following this concept relies on a careful choice of the platform on which it will be developed. One of the most straightforward choices would be to choose an ARM platform with OS support in order to be able to successfully and easily install a TEE such as the ones discussed above (ARM TrustZone or Intel SGX). However, these types of platforms have some disadvantages that can become a problem in an IoT environment. These disadvantages are related to consumption, since the platforms that support OS, such as Raspberry Pi, usually have a high consumption of energy, and this is a serious drawback when applied to a real environment.

Another requirement to take into account in the design is the ability to modularise the software running on the platform since, on one side, we have to consider the TEE, and on the other side, the rest of the platform firmware. Being able to create isolated execution environments becomes crucial for providing security in this type of architecture where there are two execution environments: a secure processing environment (SPE), represented by the TEE, and a non-secure processing environment (NSPE), where the rest of the application is executed.

This requirement translates into the use of a platform with more than one core. In this way, a core would be in charge of running the SPE and communicating with the NSPE that is running in the other core.

The platform chosen for this proposal was the PSoC64, which contains two cores: one core is an ARM Cortex-M4 and the other one is an ARM Cortex-M0+. In the Cortex-M4 core, the main application of the device is executed. This has the firmware necessary to carry out the communication with the outside world through a communications protocol that will vary depending on the application (BLE, Wi-Fi, Ethernet, etc.), as well as the libraries and kernel of the embedded OS. The ARM Cortex-M0+ is in charge of running the TEE, which is the TF-M, and incorporates support for the ARM-M microcontrollers family. The communication between the two cores is via an inter-process communication bus. The TF-M isolates the different execution environments (NSPE and SPE) through APIs that are called through the processes that form the system. In addition, the TF-M offers different services such as cryptography and attestation and allows for the execution of secure processes through secure partitions (this will be explained in detail in the following sections). These secure processes allow for the execution of firmware such as sensor drivers or communication protocols. Figure 3 illustrates this design.

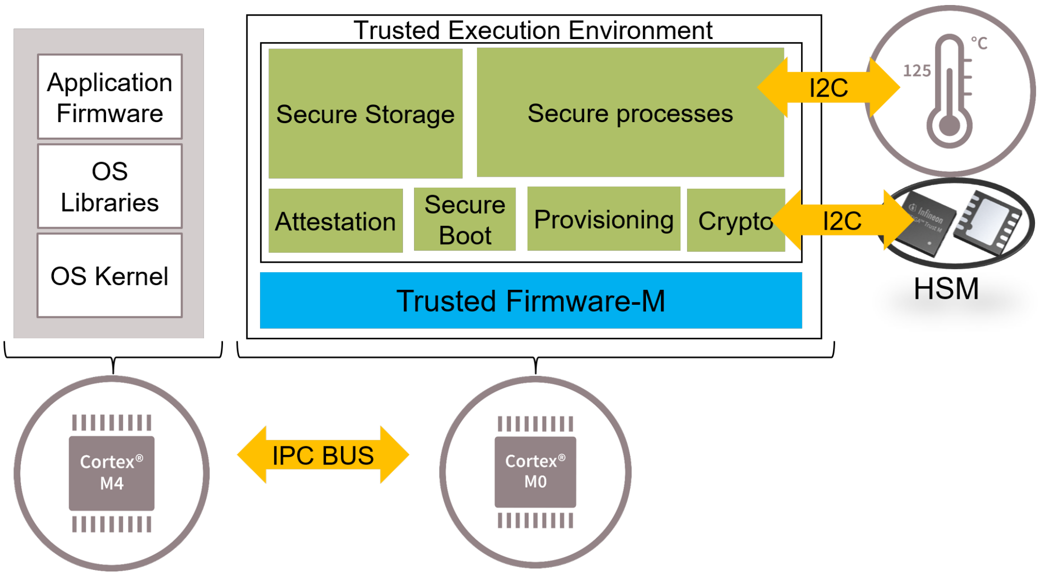

Figure 3. Secure sensor concept.

In this way, the driver of the HSM is implemented within the TEE, improving the reliability of the accesses to the HSM functionality. The functionality of the HSM can only be used from the NSPE (ARM Cortex-M4) through a controlled and predefined API by the TF-M. Additional mechanisms can be integrated in the TEE, such as a sensor driver through a secured I2C peripheral, enabling high-level and high-value functionality, such as an API request to obtain the value of the sensor directly signed by the HSM from the NSPE, greatly limiting the attack vectors and tampering with possibilities of the sensor value from the NSPE.

The design of this architecture was implemented in the CY8CKIT-064S0S2-4343W evaluation board; specifically, the MCU used was the CYS0644ABZI-S2D44, which has two cores: an ARM Cortex M4 working at 150 MHz and an ARM Cortex M0+ at 100 MHz. Regarding the sensor, in this proposal, we attached an S2GO pressure DPS310 board by Infineon [44] to the PSoC64. This board is equipped with one DPS310 barometric pressure sensor, offering a high pressure resolution (±0.005 hPa) and temperature accuracy of approximately ±0.5 °C. For communication, the board provides an I2C interface that connects the DPS310 sensor with PSoC64. The HSM used was the Infineon OPTIGA Trust M [45], which has the functionalities for private and public key generation, signing and encryption. It also has different power saving modes, which makes it perfect for this kind of IoT application.

3.1. Trusted Firmware-M

TF-M is the TEE implemented in the Cortex-M0+ of the PSoC64. The implementation is aligned with Platform Security Architecture (PSA) certified guidelines and offers isolation between NSPE and SPE. TF-M consists of several modules: secure boot to authenticate the integrity of NSPE and SPE images, isolation between environments, communication between SPE and NSPE and modules that enable cryptography, internal secure storage, attestation and secure services.

Figure 4 shows how these modules are distributed as well as the isolation level that TF-M provides. The first level of isolation, represented in Figure 4 with a black solid line, separates the NSPE from the SPE. This limit means that applications running on the NSPE do not have access to the secure core where the TF-M runs, thus only through a defined API (PSA API) can these applications make calls to the services offered by the TF-M. The applications provided by the TF-M are divided into two types: PSA Root of Trust (RoT) and applications RoT. Between these two types, there is another level of isolation established by the TF-M. This level 2 isolates the services running in the application RoTs from those running in the PSA RoT, making these services run independently of each other, unless communications are established between them through certain APIs (service API and PSA secure partition API). The reason for why this division exists is that the services provided by PSA RoT are already pre-established by the TF-M and the services provided by the application RoT are programmable and modified by the programmer through the creation of secure partitions. The services predefined by the RoT PSAs are as follows:

- Cryptography: This service implements cryptographic operations, including symmetric ciphers, asymmetric algorithms, hash algorithms, key derivation and random number generation.
- Attestation: This service reports the immutable device identity and boot state.
- Internal trusted storage: This service provides secure storage of small data items such as cryptographic keys.
- Firmware update: This provides firmware image update functionality by retrieving image information about the firmware images stored on the device memory. This service validates the image and trigger a reboot to restart the platform.

This last PSA RoT service is strongly related to the secure boot module that TF-M implements. This module is essential for security to enforce firmware authenticity to protect it against malware execution. This is achieved by building a chain of trust where each step in the execution chain authenticates the next step. The chain of trust is based on an RoT that is implemented using asymmetric cryptography. The RoT is a combination of an immutable bootloader and a public key. The bootloader is started when the CPU is restarted, running in secure mode and authenticating the firmware image by means of hash validation and an RSA signature. The public key, with which the checks are performed, can be embedded into the bootloader image or can be provided to the chip during manufacturing. In case of successful authentication, the bootloader passes the execution to the secure image.

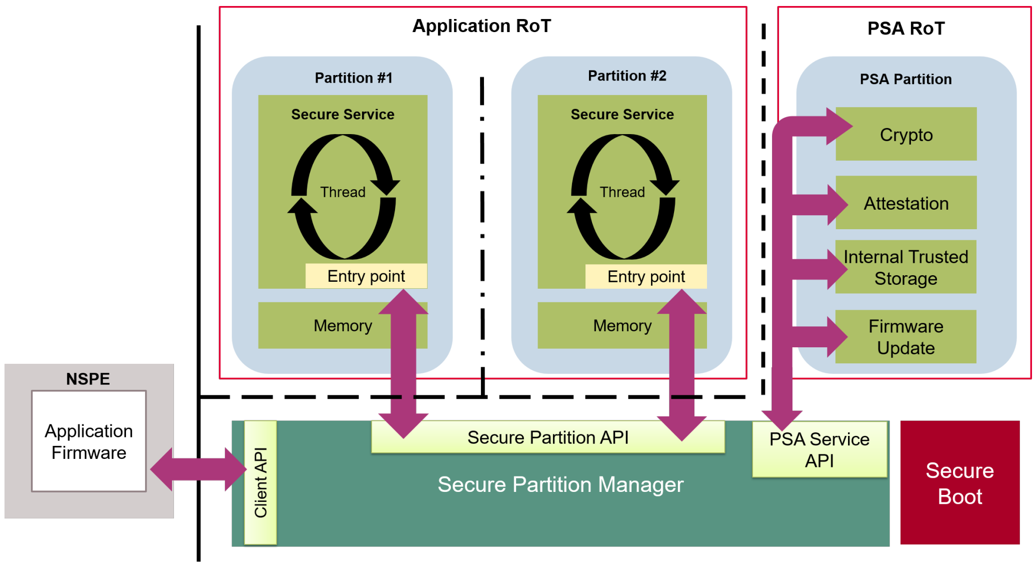

Figure 4. Trusted Firmware-M.

RoT applications are developer-defined applications that are installed on secure partitions. Each secure partition is a single thread of execution and is the smallest unit of isolation. If the isolation level 3 is implemented, every secure partition is isolated from each other. Each secure partition can host one or more application RoT service, which typically shares functionality or data. Secure partitions have a persistent identifier, called a partition ID, that can be used for access control within the system. These identifiers must be unique within the device, protected by the Secure Partition Manager (SPM) at runtime, and unchanged when firmware is rebooted or updated.

The SPM is the most privileged firmware, providing the fundamental security services to secure the PSA root of trust and enabling isolated firmware components to communicate between them through APIs (client API, secure partition API and PSA RoT services API). In addition, SPM implements the scheduling logic to ensure that the request is delivered and processed correctly. Finally, one of the most important tasks of the SPM is to access secure peripherals and handle interrupts.

Secure peripherals are critical in this proposal since two hardware devices (atmospheric pressure sensor DPS310 and OPTIGA Trust M) are being used through secure services implemented in the TF-M. These devices are connected to the Cortex M0+ using an I2C interface and while the data are served to this core, the TF-M must implement certain mechanisms to make the communication bus blocked or inaccessible to the other core. In addition, each peripheral has an associated driver and one or more entries in the device's interrupt table. These interrupts must be managed by the TF-M in order to avoid other cores managing them.

For addressing these issues and creating a secure peripheral, the TF-M creates secure Memory Mapped I/O regions (MMIOs) that exclude access to these regions for other secure or non-secure processes. Thus, when a secure process occupies the peripheral, it excludes any other process, secure or not, from accessing it at the same time.

Thus, the implementation of these secure services is based on a driver in which interrupts are handled through the SPM and a memory area (MMIO) from which the memory registers used by the driver are mapped. This driver is implemented in a thread that is in an infinite loop waiting for a call through an entry point that communicates through the secure partition API. The partition name, the interrupts, the MMIO and the secure services on which a partition depends are registered in a partition manifest to which the SPM associates a service ID (SID). The SPM associates an SID to the secure service implementing the HSM driver as well as to the secure service implementing the pressure sensor driver. In Figure 5, the flowchart is shown from when the non-secure core requests data to the core where the TF-M is running. This diagram shows how the program jumps between different levels of isolation by means of the diverse APIs available, until it executes the driver peripheral through the entry point. This peripheral can be the HSM or a sensor.

In this way and through these mechanisms, the drivers of the two hardware devices connected to the Cortex M0+ are implemented in a secure environment isolated from other processes that may interfere with the data flow and the I2C communication protocol, creating a secure peripheral in which the data received are reliable.

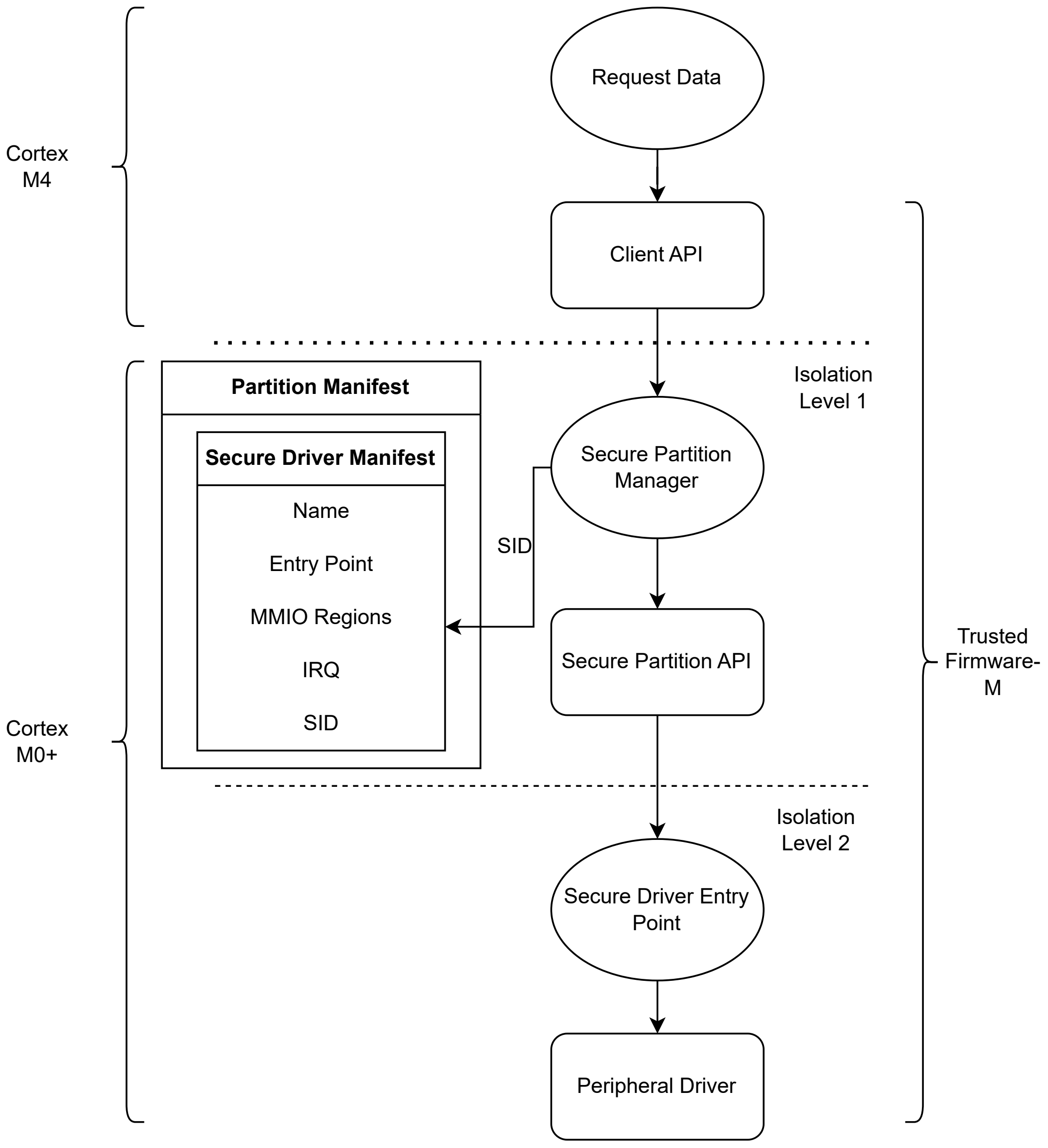

Figure 5. Driver implementation in TF-M.

3.2. Hardware Security Module

In order to maintain the integrity of the data once they leave the secure core, they must be signed. Although signing can be performed through the PSA RoT cryptographic service, this entails certain risks as discussed above. Thus, including an HSM in this way offers extra protection to the system by adding a layer of hardware security. OPTIGA Trust M performs the functions of a cryptographic service implemented in a secure partition, where it performs key generation, public key export and hash and signing operations.

Figure 6 shows the order of operations within the execution flow of the secure core processes. Key generation is performed through a True Random Number Generator (TRNG) that incorporates the HSM, which generates true random numbers from high-entropy microscopic physical events. The signature mechanism is based on an Elliptic Curve Digital Signature Algorithm (ECDSA) [46], which uses ECC keys for the signature; in particular, in this proposal, the curve is the secp256r1, also known as NIST P-256. Key generation is carried out in the startup and the private key is stored permanently in the HSM. During the commissioning process, the public key is exported from the secure sensor to the entity, which verifies the data signatures made during the operational mode of the device.

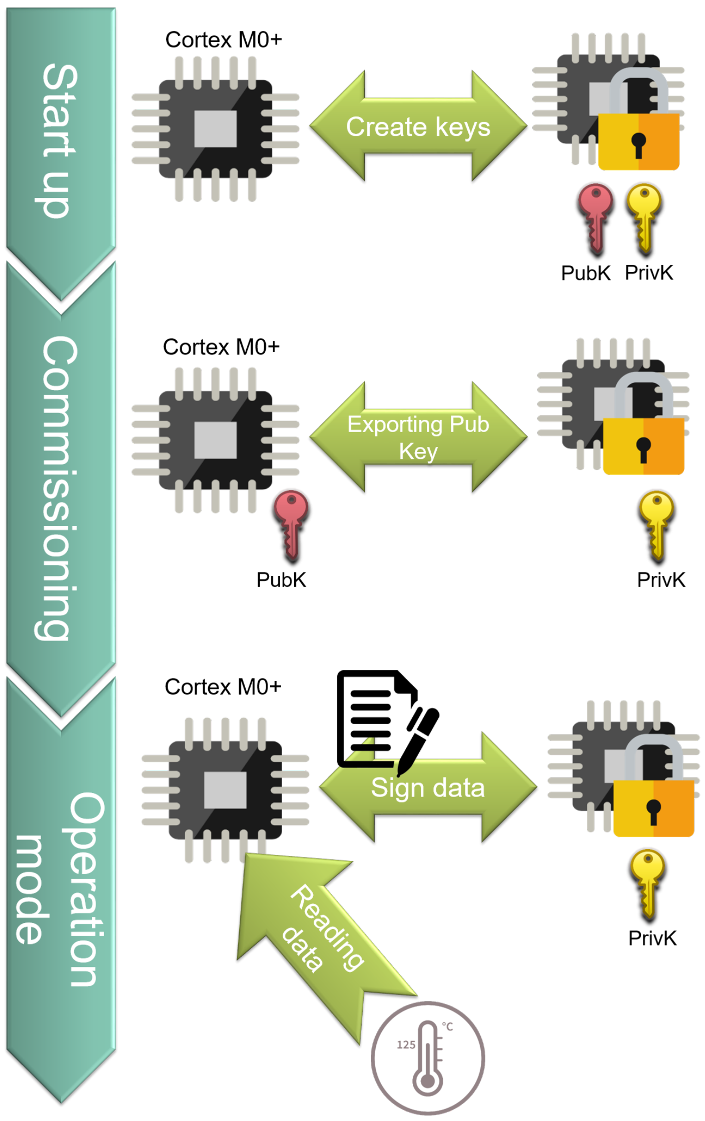

Figure 6. Operations carried out by the HSM.

In this way, through the generation and secure storage of the private key within the HSM, the secure sensor cannot be impersonated by extracting the private key or duplicating it.

This secure sensor concept fits into various IoT architectures and can be integrated with technologies used in this field, but there is one type of technology where this proposal has a fist-in-glove fit. This is the case of Blockchain technologies.

4. Blockchain Application

The application of the secure sensor concept in an IoT node offers, as mentioned above, several advantages. In any system where it is applied, the secure sensor is able to provide secure data by signing and verifying the data, ensuring data integrity.

Systems where the secure sensor is implemented must have a PKI system where an IoT node implementing the secure sensor approach must be enrolled. In this case, the node that provides data to the system and therefore signs it must be registered in the same PKI as the node that receives those data and therefore verifies them. This process of signature and verification must take place in all environments and systems where this concept is applied. However, as stated in Section 1, PKI paradigms are no longer enough in IIoT environments [6]. New paradigms such as Blockchain have emerged, providing further security benefits to IIoT networks [47,48]. In this situation, the integration of this type of technology with the concept of a secure sensor becomes fundamental [49]. In this case, it depends on the architecture and topology of the Blockchain network to incorporate a PKI-based access control system. In any case, Blockchain networks have mechanisms to decentralise PKI through different entities and based on smart contracts [50].

Traditionally, private and public keys have been used to enrol the users and/or devices into a network by using a PKI combined with a symmetric cipher. In the case of systems implementing Blockchain, these keys are also used for signing and verifying transactions and, additionally, to register and to provide the permissions required prior to authentication in private Blockchain networks. Due to the involving of all these operations, the protection of cryptographic keys in an HSM becomes especially relevant in Blockchain networks since the extraction of these keys can lead to the impersonation of certain nodes and the fraudulent sending of transactions that, if they are subsequently verified, are impossible to undo.

In addition to the integration of the secure sensor in a PKI implemented in Blockchain, this concept can also provide reliable data to the Blockchain functioning as an Oracle.

Working as an Oracle

In Blockchain terminology, an Oracle is the entity that provides reliable data to the Blockchain network. These data are by definition reliable and network participants are obliged to use them without doubting their veracity as they have been provided by the Oracle.

This component of Blockchain networks, although certainly neglected in the literature [51,52], is of crucial importance for the reliability of this type of network. The vast majority of Blockchain networks use data coming from the off-chain world, and the untruthfulness of these data can cause important failures in these systems as well as erroneous ledger states.

There are different types of Oracles [53] that, depending of the information source, can be classified into three groups: software Oracles, where data come from online sources, hardware Oracles, which obtain the data from the physical world—for example, from sensors—and human Oracles, for which the data come through a human, who enters and verifies information, e.g., financial data. A classification can also be established depending on the trust model [54]:

- Centralised trust model: This Oracle model is based on an entity that certifies the reliability of provided data.
- Decentralised trust model: Decentralised Oracles avoid the single point of failure but they add an extra overhead since, being based on several entities, the latency increases as a consensus protocol between them is required.

According to these established classifications, in the proposal of this work, the Oracle would correspond to a hardware type, as it would collect data from sensors, and a centralised trust model, as it is a single entity that provides the data to the Blockchain.

The signature of the data is performed at the extraction of the data in the secure core so that the data can hardly be modified before the signature. The keys are stored in the HSM, which offers tamper-resistant protection, and the communication with the sensor is carried out through the secure core. Once the data have been signed, and therefore any modification to them is detected in a subsequent verification, they leave the secure core to be collected in the NSPE. The NSPE performs the communication functions with the Blockchain, delivering the data to the entity that has requested them, which performs this verification using the public key before introducing the data into the Blockchain through a transaction. The public key is available through a certificate that has every entity of the network and has been issued by a Certification Authority (CA) that belongs to the Blockchain or is implemented through the Blockchain.

In this way, the Oracle that implements the secure sensor proposal is able to deliver data to the Blockchain, ensuring that any modification of the data is detected. While guaranteeing the integrity and authenticity of the data, there are possible attacks that must be considered in this concept, such as replay attacks, and how they can be mitigated using timestamps.

Timestamping

To prevent an attacker from obtaining the data and sending them as the data required by the receiver at that precise moment, a timestamping mechanism must be implemented.

Before the signed data leave the secure environment, a timestamp attached to the data is needed so that once the entire packet (data plus timestamp) is signed, the receiver can perform a verification that the timestamp corresponds to the time at which the request for the data was made.

The logical solution would be to synchronise the clocks of both the transmitter and the receiver. This solution, although the simplest, has different problems, which are that modifying a clock in a hardware device is very simple and that it is a very typical attack in this type of device [55].

The solution proposed in this work relies on obtaining the timestamp from a reliable external source; thus, two proposals are given below:

- Using Network Time Protocol (NTP) [56]: An NTP provides the basic protocol mechanisms necessary to synchronise the time of different systems to an accuracy of one nanosecond. It is easily scalable and, through handshaking between the NTP server and the client, the clocks are synchronised. In order to enhance the security of this protocol, it is proposed to use the Network Time Security (NTS) protocol, which enables NTP clients to be cryptographically identified against NTP servers to ensure the authenticity and integrity of exchanged time packets [57]. This solution, although efficient and independent of the secure sensor operation process with the Blockchain, requires an NTS server in the system where it is implemented.
- Using the timestamp from Blockchain's block: Blockchain networks, when committing a new block to the chain, include a timestamp in the block. This timestamp is a reliable source in order to be able to synchronise the two communicating devices. However, this timestamp is discrete, i.e., it does not count the time sequentially as a clock would, but seconds may have passed between the consulted block and the next one, as in the case of Ethereum [58], or even minutes, as in the case of Bitcoin [59]. To solve this, a mechanism is presented as shown in Figure 7. When the Blockchain client requests data from the secure sensor, it queries the timestamp of the last Blockchain block (T'). Before sending the signed data, the secure sensor asks for the timestamp of the last block of the Blockchain (T). When the data and the signed timestamp reach the client, the client queries the timestamp of the last block (T''). To decide whether these data are accepted or discarded, the following equation has to be fulfilled:

  $$T' \leq T \leq T'' - \varepsilon \tag{1}$$

  ε is a variable that sets the acceptance tolerance of the timestamp and is related to the average generation time of a block on the Blockchain; therefore, it depends on the Blockchain where the secure sensor is used and it will never be equal to or greater than this averaging time. This solution does not require any external infrastructure to obtain the timestamps as they come from the Blockchain.

This method, although obtaining timestamps from a more reliable source than the previous proposal, works better with a high average number of generated blocks than with a low one. It can be concluded that the selection of one of these two solutions is determined by the type of Blockchain that is applied: in the case of a Blockchain such as Ethereum, where the number of blocks per minute is high, the second method is more convenient, while in the case of a Blockchain such as Bitcoin, where the number of blocks per minute is low, using the second method would add nothing to the reliability of the data as even the three timestamps can be the same within a 10 min interval, for example; therefore, the first method would be more appropriate.

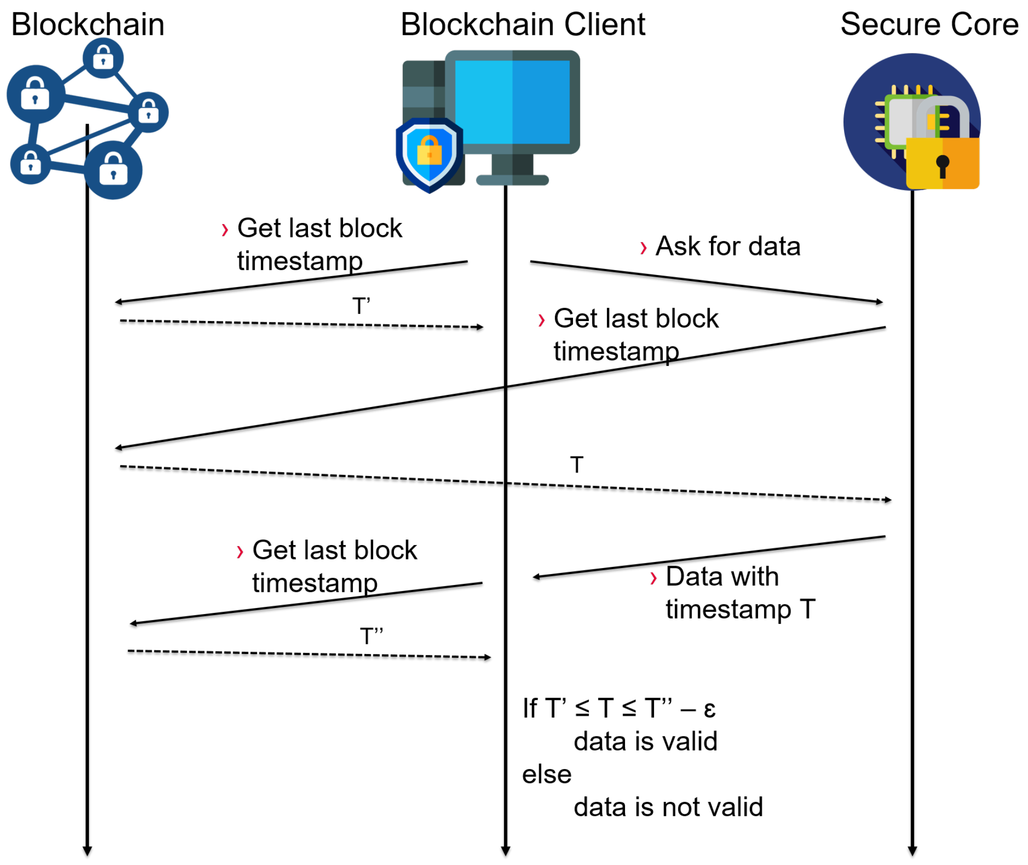

Figure 7. Timestamp mechanism.

Another alternative solution is to adopt a hybrid one that captures both the benefits of using the first approach (more granular timestamps) and the second (more reliable timestamps).

It should be noted that the processing and delay times within the network would be negligible since it is assumed that these times are constant and that there is no large abrupt variation during the timestamp acquisition mechanism as exemplified in Figure 7. In the case of a sharp increase in these times, the timestamp (T) would no longer be valid in the case where it does not fulfil Equation (1). In such a scenario, ε would mark the threshold of acceptance or rejection. This variable will be set depending on the environment where this mechanism is implemented. Applying the hybrid approach mentioned before, it is possible to solve this kind of problem by performing different timestamp verification: through the last block of the Blockchain and through an NTP server.

Finally, it is worth mentioning that these processes must be implemented in the secure core since an implementation in the NSPE can lead to malicious modifications of the timestamp by an attacker before signing.

Having described the design of the secure sensor and its integration in Blockchain networks working as an Oracle, it remains to show its applicability in a real and disruptive use case.

5. Wine Logistic Use Case

The quality of a wine can be noticed by its taste and flavour. From the cheapest to the most expensive wines, a failure in the production, transport or sales chain can ruin the product. Traditionally, wine is stored in a cellar with temperatures between 10 and 16 °C [60]. Exposure to temperature or humidity spikes can cause spoilage such as fungus or maturation interruption [61]. Despite the importance of the temperature and humidity for the quality of the wine, this information is not generally shared with the customer.

The use of Blockchain in this type of application, as well as the secure extraction of data through sensors, makes the concept proposed in this article fit perfectly in this use case. Blockchain works very well in logistics use cases [62], where data are regularly collected to create a record. Thanks to these data stored in a decentralised database, the records are immutable and, through smart contracts, it is possible to automate processes such as the devaluation of the price of wine if it exceeds a certain temperature for a certain period time. To achieve this, an architecture was built based on the secure sensor and Blockchain, incorporating several HSMs.

5.1. System Description

The system is compounded by an edge node sensor platform based on a PSoC64, as is shown in Figure 8. Using the secure sensor concept with its secure booting assures the integrity of the firmware running on both cores. One core is used as a security co-processor, where an HSM is used to store the key material, providing additional protection against physical attacks. The drivers of the HSM and the sensor are programmed in the secure core, which is in charge of both gathering and signing the sensor value, assuring its integrity after it leaves the secured environment. The value is then sent to the Amazon Web Service (AWS) cloud, where a Blockchain based on Hyperledger Fabric (HLF) [63] guarantees a trusted traceability: abnormal events can trigger alerts in smart contracts, automating business logic and dramatically reducing operating costs, delays and the possibility of fraud. After the wine is bottled, the customer can read the secure and unique ID of each bottle via an HSM-based Narrow Field Communication (NFC) sticker, which is then used to retrieve all its historical data from the Blockchain. Figure 9 shows the whole architecture.

The node consists of the DPS310 atmospheric pressure sensor, the HSM OPTIGA Trust M and an LTE communication shield providing an antenna. In this way, the data are sent through this channel to the cloud, where an instance of HLF runs. The node incorporates a shield to attach sensors such as the DPS310. In Figure 8, only two attachments are shown (the DPS310 and the HSM), as well as a free socket where one can attach the humidity sensor, a GPS or any other type of sensor.

This node is located in the barrels to monitor the temperature and humidity parameters during transport and storage as shown in Figure 10. Each edge node located in a barrel contains a digital ID stored in the HSM. This identifier is associated with the ID contained in the HSM-based NFC sticker of the bottles that are filled with wine from the barrel. By associating these IDs, it is possible to keep track of the information associated with wine bottles from purchase in the supermarket to direct bottling in barrels. As well as storing sensor data related to wine on the Blockchain, the different entities that make up this network can also store additional information related to wine production: type of grape variety, type of harvest, parcel number, etc.

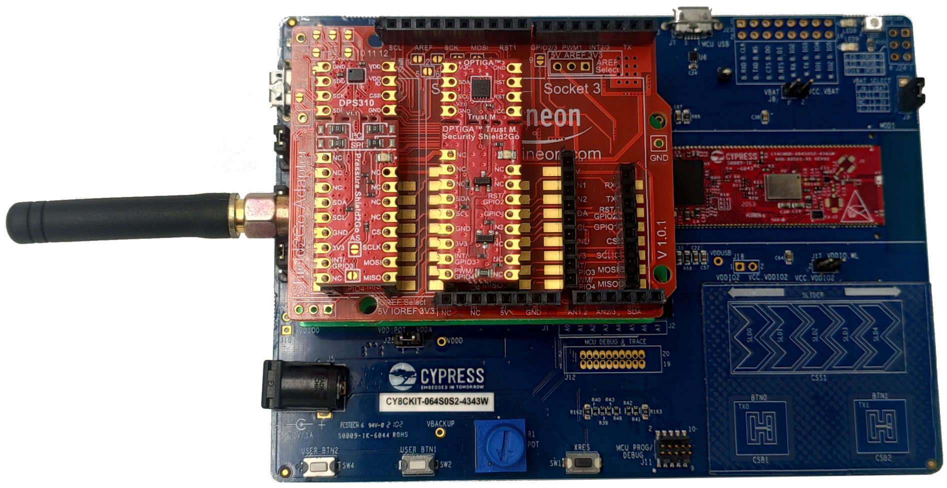

Figure 8. PSoC64-based edge node.

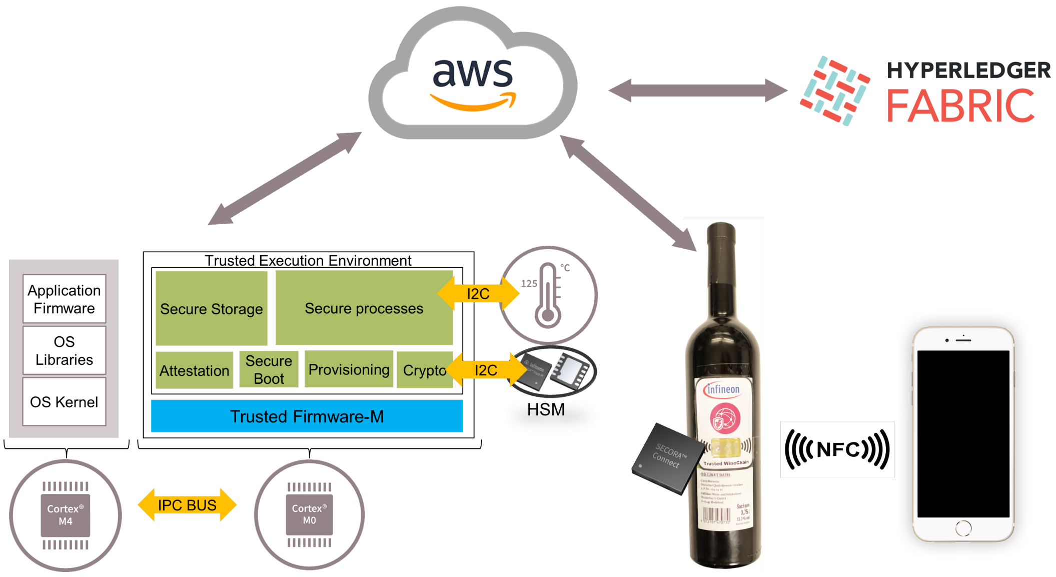

Figure 9. Architecture proposed in wine logistic use case.

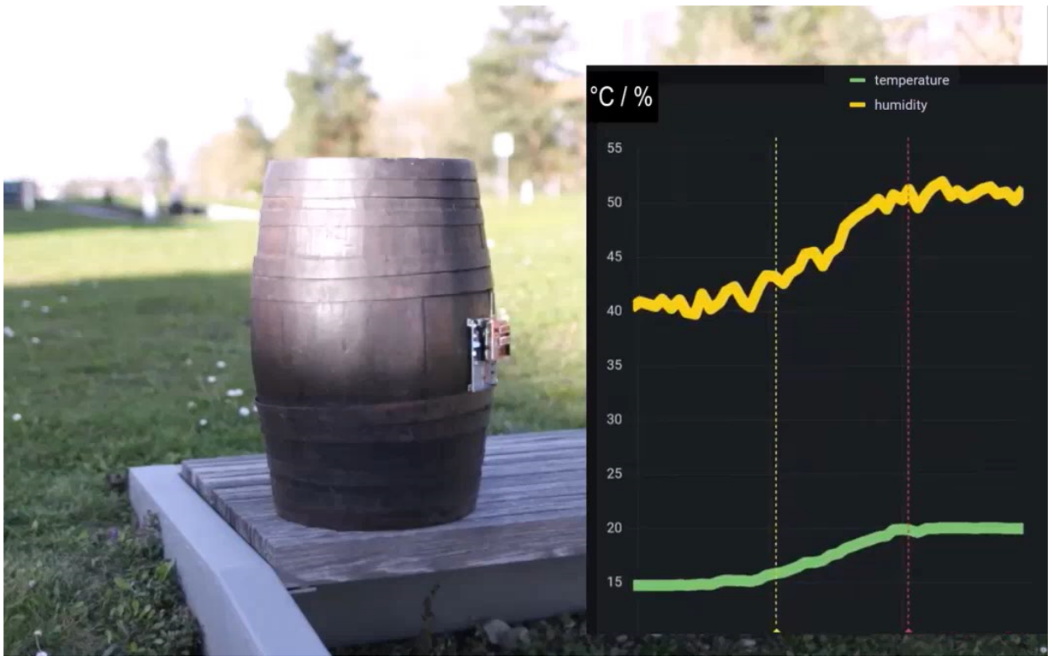

Figure 10. PSoC64 located in a barrel with the data records gathering by sensors.

When wine is stored in barrels, each barrel is assigned a unique digital ID that is generated through a private key stored in the HSM. During transport, the node takes periodic measurements of the environmental conditions through the node's sensors. These data are collected and stored in the ledger. When the barrel is uncorked and its contents are poured into different bottles, a cryptographic key derived from the digital ID of the barrel is imported into each bottle's HSM. Thus, the keys in the bottle are associated to one cryptographic key in the barrel as well as the data registry recorded by the sensors. Table 1 shows how the data structure is stored in the ledger.

Table 1. Blockchain ledger example.

BarrelID	Derived Key1	Derived Key2	Derived KeyN	Data[]: {Timestamp, Position, Temperature, Humidity}
ZHm...hZnZG	RmYXz...hc2Rm	MjMxN...3NmZm	ODc5...hkYWN	[{1687886326, 31.95162, −28.12310, 28.4, 12%}, . . .]
cnR...mRzZG	2Fmc2...0ZXJ3	M2Vyd...RjdGy	MjNb...HV6dc	[{1687886010, 1.95165, −10.23433, 10.1, 29%}, . . .]
ODc...NjU0y	aW9rm...a8O2a	NDMgY...Z2hc2	NDMy...zZnJi	[{1687800125, 7.35234, 20.45643, 35.9, 45%}, . . .]

The digital IDs of the barrels are 32-byte base64-encoded hashes, as well as the derived keys for the bottles. The sensor data are associated to the barrel digital ID, thus forming an array where in each position there is a measurement identified by a timestamp. The components of the Blockchain deployed depend on the number of organisations involved in the business model. If it considers a transport company plus different storage companies (supermarkets, warehouses, etc.), each of them will have a copy of the ledger that will be updated over time. Each update will be periodically triggered through a Blockchain transaction and validated by the different peer nodes of each organisation.

In this way, thanks to the Blockchain and embedded HSMs in the edge nodes and in the stickers' bottle, information is stored securely, providing a robust and reliable system, automating processes through smart contracts and delivering information more clearly and accurately to customers who, through a simple smart phone with built-in NFC, can access this information. The secure sensor acting as an Oracle sends the signed information to the Blockchain, where it is further validated before entering the ledger. Using this architecture ensures that the data entering the ledger are reliable from the moment they are extracted by the sensors, and that they have not been modified either during processing at the edge node or when they are stored in the ledger, as their veracity at that moment is approved by the ledger members.

5.2. Performance Evaluation

This section contains the experimental performance measurements carried out for the concept of secure sensor described above. In this sense, the performance evaluation was focused on the processing time required by the implementation of a secure sensor on the PSoC64, from the moment when the sensor data are read until they are sent to the non-secure core for subsequent transmission. For this reason, the tests performed on the communication system and the Blockchain were discarded, as they depend on the communication protocol used as well as the nodes deployed in the Blockchain network for AWS.

Measurements were carried out on the PSoC64 platform, specifically using the MCU CYS0644ABZI-S2D44, which has two cores: an ARM Cortex M4 working at 150 MHz and an ARM Cortex M0+ operating at 100 MHz. The TF-M is running in the M0+ and an HSM (OPTIGA Trust M) is attached to this core. The sensor used in this evaluation was the Infineon DPS310 atmospheric and temperature sensor. The measurements of this secure sensor concept were compared to the traditional concept shown in Figure 2. In this concept, the MCU requests the data from the sensor, and when they are returned to the main MCU, they are subsequently signed by the MCU using an HSM. This approach has a variant in which a software-simulated HSM is used, i.e., the signature and key creation operations are carried out using C libraries implemented in the PSoC. Both of these approaches are much less secure as the data arrive unsigned at the MCU and no use is made of the TEE to secure the peripherals. The times shown in Table 2 are the average of a sample of 20 measurements. Times are in milliseconds.

Table 2. Performance evaluation in milliseconds comparing three approaches.

	Traditional Approach with Software HSM	Traditional Approach with Hardware HSM	Secure Sensor Approach
Key Generation	9450	2900	2945
Signature Algorithm	5237	310	350
Sensor Read	43	43	85
Total Time	15.305	3810	4160
Total Time without Key Generation	5387	423	518

From Table 2, it is shown that the fastest option is the one using hardware HSM in the traditional architecture. This is mainly because the use of a TEE such as the TF-M is a communication time overhead as it incorporates more complex software mechanisms. Figure 4 shows all the modules and how they communicate with each other. This is a significant increase in communications and the TF-M protects the sensor driver and the HSM, creating a secure peripheral.The advantage of performing operations using an HSM is the speed compared to the software approach. In short, the secure sensor approach is 1.09 times slower than the traditional approach using HSM, but is 3.67 times faster than the traditional approach using a software HSM. Although the secure sensor approach is not the fastest, it is the most secure approach as the TEE implemented in the PSoC64 introduces security mechanisms such as the secure peripherals discussed earlier in the manuscript.

6. Conclusions

This paper proposes a secure sensor design and implementation using different mechanisms for providing a data root of trust. By combining a TEE and an HSM in a dual-core microcontroller with a secure sensor driver implemented in the secure core, the immutability and integrity of data are achieved. The main goal of the proposal is the use of the secure sensor concept, where a secure core is located between the main microcontroller and the sensor. In the secure core, both the HSM and TEE work together, obtaining the advantages of both solutions. HSMs offer hardware and certified protection against physical attacks. However, they do not typically support flexible high-level functionality, such as programmable device drivers or secure peripherals, features provided in this solution by the TEE. Combining an HSM with a flexible TEE, running in the secure processor, a secure environment is created where sensor drivers can be implemented, protecting the communication bus from software attacks as well as providing the ability to sign the data received by the HSM before the data leave the secure core.

The design of this secure sensor fits the role of a Blockchain Oracle, capable of providing reliable data to the ledger thanks to the incorporation of timestamping mechanisms to provide reliability against replay attacks. This fills a gap in the literature regarding Blockchain Oracles, which are rarely addressed.

The feasibility of this approach in a real use case is demonstrated in a wine logistics scenario, in which the secure sensor acts as an Oracle sending temperature and humidity data to the Blockchain. Thanks to these records stored in the Blockchain, customers can obtain accurate information about these products and, through smart contracts, automate business logic and reduce operating costs, delays and the possibility of fraud.

The incorporation of this secure sensor proposal into Blockchain technologies as an Oracle role makes this concept a promising proposition for the future, adding security and integrity to the data required by scenarios as a wine logistic chain. Whether in Blockchain or non-Blockchain networks, the secure sensor solution fills a gap in the security of systems that make intensive use of data from the outside world.

Author Contributions: Conceptualisation, A.J.C.-G.; methodology, A.J.C.-G.; software, A.J.C.-G.; validation, A.J.C.-G.; formal analysis, A.J.C.-G.; investigation, A.J.C.-G.; writing—original draft preparation, A.J.C.-G. and J.C.-C.; writing—review and editing, E.C., A.E.-M., J.C.-C., D.P.M. and L.P.; supervision, E.C. and L.P.; project administration, A.J.C.-G. and A.E.-M.; funding acquisition, E.C., A.E.-M., D.P.M. and L.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Infineon Technologies AG. Additional funding was provided by European Union's Horizon Europe research and innovation program through the funding project "Cognitive edge-cloud with serverless computing" (EDGELESS) under grant agreement number 101092950. The work has also been funded by FEDER/Junta de Andalucia-Consejeria de Transformacion Economica, Industria, Conocimiento y Universidades under Project B-TIC-588-UGR20.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AWS	Amazon Web Service
BLE	Bluetooth Low Energy
CA	Certification Authority
DLT	Decentralised Ledger Technologies
ECC	Elliptic Cryptographic Curve
ECDSA	Elliptic Curve Digital Signature Algorithm
FPGA	Field Programmable Gate Array
HLF	Hyperledger Fabric
HSM	Hardware Security Module
I2C	Inter-Integrated Circuit
IIoT	Industrial Internet of Things
MMIO	Memory Mapped Input Output
NFC	Narrow Field Communication
NSPE	Non-Secure Processing Environment
NTP	Network Time Protocol
NTS	Network Time Security
PKI	Public Key Infrastructure
PSA	Platform Security Architecture
PUF	Physical Unclonable Functions
RoT	Root of Trust
RSA	Rivest–Shamir–Adleman
SPE	Secure Processing Environment
SPI	Serial Peripheral Interface
SPM	Secure Partition Manager
TEE	Trusted Execution Environment
TF-M	Trusted Firmware-M
TPM	Trusted Platform Module

References

1. Barnaghi, P.; Bauer, M.; Biswas, A.R.; Botterman, M.; Cheng, B.; Cirillo, F.; Dillinger, M.; Graux, H.; Hoseinitabatabaie, S.A.; Kovacs, E.; et al. IoT Analytics: Collect, Process, Analyze, and Present Massive Amounts of Operational Data–Research and Innovation Challenges. In *Building the Hyperconnected Society-Internet of Things Research and Innovation Value Chains, Ecosystems and Markets*; River Publishers: London, UK, 2022; pp. 221–260.
2. Naveen, S.; Kounte, M.R. Key Technologies and challenges in IoT Edge Computing. In Proceedings of the 2019 Third International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 12–14 December 2019; pp. 61–65. [CrossRef]
3. Jabraeil Jamali, M.A.; Bahrami, B.; Heidari, A.; Allahverdizadeh, P.; Norouzi, F. IoT Architecture. In *Towards the Internet of Things*; EAI/Springer Innovations in Communication and Computing; Springer: Cham, Switzerland, 2020. [CrossRef]
4. Hassija, V.; Chamola, V.; Saxena, V.; Jain, D.; Goyal, P.; Sikdar, B. A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures. *IEEE Access* **2019**, *7*, 82721–82743. [CrossRef]

5. Mrabet, H.; Belguith, S.; Alhomoud, A.; Jemai, A. A Survey of IoT Security Based on a Layered Architecture of Sensing and Data Analysis. *Sensors* **2020**, *20*, 3625. [CrossRef] [PubMed]
6. Mathur, S.; Arora, A. Internet of things (IoT) and PKI-based security architecture. In *Industrial Internet of Things and Cyber-Physical Systems: Transforming the Conventional to Digital*; IGI Global: Hershey, PA, USA, 2020; pp. 25–46.
7. Antal, C.; Cioara, T.; Anghel, I.; Antal, M.; Salomie, I. Distributed Ledger Technology Review and Decentralized Applications Development Guidelines. *Future Internet* **2021**, *13*, 62. [CrossRef]
8. Alfandi, O.; Khanji, S.; Ahmad, L.; Khattak, A. A survey on boosting IoT security and privacy through blockchain. *Cluster Comput.* **2021**, *24*, 37–55. [CrossRef]
9. Monrat, A.A.; Schelén, O.; Andersson, K. A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities. *IEEE Access* **2019**, *7*, 117134–117151. [CrossRef]
10. Koç, Ç.K.; Özdemir, F.; Ödemiş Özger, Z. Rivest-Shamir-Adleman Algorithm. In *Partially Homomorphic Encryption*; Springer: Cham, Switzerland, 2021. [CrossRef]
11. Al Saadi, M.; Muscat, O.; Kumar, B. A Review on Elliptic Curve Cryptography. *Int. J. Future Gener. Commun. Netw.* **2020**, *13*, 1597–1601.
12. Sabt, M.; Achemlal, M.; Bouabdallah, A. Trusted Execution Environment: What It is, and What It is Not. In Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, 20–22 August 2015; pp. 57–64. [CrossRef]
13. Mavrovouniotis, S.; Ganley, M. Hardware Security Modules. In *Secure Smart Embedded Devices, Platforms and Applications*; Markantonakis, K., Mayes, K., Eds.; Springer: New York, NY, USA, 2014. [CrossRef]
14. Cabrera-Gutiérrez, A.J.; Castillo, E.; Escobar-Molero, A.; Álvarez-Bermejo, J.A.; Morales, D.P.; Parrilla, L. Integration of Hardware Security Modules and Permissioned Blockchain in Industrial IoT Networks. *IEEE Access* **2022**, *10*, 114331–114345. [CrossRef]
15. Jauernig, P.; Sadeghi, A.-R.; Stapf, E. Trusted Execution Environments: Properties, Applications, and Challenges. *IEEE Secur. Priv.* **2020**, *18*, 56–60. [CrossRef]
16. Potestad-Ordóñez, F.E.; Tena-Sánchez, E.; Acosta-Jiménez, A.J.; Jiménez-Fernández, C.J.; Chaves, R. Hardware Countermeasures Benchmarking against Fault Attacks. *Appl. Sci.* **2022**, *12*, 2443. [CrossRef]
17. TrustedFirmware.org. Trusted Firmware M. Available online: https://tf-m-user-guide.trustedfirmware.org/ (accessed on 18 July 2022).
18. Infineon.com. PSoC 64 User Manual. 2022. Available online: https://www.infineon.com/cms/en/product/microcontroller/32-bit-psoc-arm-cortex-microcontroller/psoc-6-32-bit-arm-cortex-m4-mcu/psoc-64// (accessed on 18 July 2022).
19. Iqbal, A.; Amir, M.; Kumar, V.; Alam, A.; Umair, M. Integration of next generation IIoT with Blockchain for the development of smart industries. *Emerg. Sci. J.* **2020**, *4*, 1–17. [CrossRef]
20. Dujak, D.; Sajter, D. Blockchain Applications in Supply Chain. In *SMART Supply Network*; Kawa, A., Maryniak, A., Eds.; EcoProduction; Springer: Cham, Switzerland, 2019. [CrossRef]
21. Allessie, D.; Sobolewski, M.; Vaccari, L.; Pignatelli, F. *Blockchain for Digital Government*; Publications Office of the European Union: Luxembourg, 2019; pp. 8–10.
22. Shahnaz, A.; Qamar, U.; Khalid, A. Using Blockchain for Electronic Health Records. *IEEE Access* **2019**, *7*, 147782–147795. [CrossRef]
23. Guo, Y.; Liang, C. Blockchain application and outlook in the banking industry. *Financ. Innov.* **2016**, *2*, 24. [CrossRef]
24. Caldarelli, G. Understanding the Blockchain Oracle Problem: A Call for Action. *Information* **2020**, *11*, 509. [CrossRef]
25. Arias, O.; Wurm, J.; Hoang, K.; Jin, Y. Privacy and Security in Internet of Things and Wearable Devices. *IEEE Trans. Multi-Scale Comput. Syst.* **2015**, *1*, 99–109. [CrossRef]
26. Kaur, J.; Agrawal, A.; Khan, R.A. Security Issues in Fog Environment: A Systematic Literature Review. *Int. J. Wirel. Inf. Netw.* **2020**, *27*, 467–483. [CrossRef]
27. Yoon, M.-K.; Mohan, S.; Choi, J.; Kim, J.-E.; Sha, L. SecureCore: A multicore-based intrusion detection architecture for real-time embedded systems. In Proceedings of the 2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), Philadelphia, PA, USA, 9–11 April 2013; pp. 21–32. [CrossRef]
28. Rahmatian, M.; Kooti, H.; Harris, I.G.; Bozorgzadeh, E. Hardware-Assisted Detection of Malicious Software in Embedded Systems. *IEEE Embed. Syst. Lett.* **2012**, *4*, 94–97. [CrossRef]
29. Mao, S.; Wolf, T. Hardware support for secure processing in embedded systems. In Proceedings of the 44th Annual Design Automation Conference (DAC '07), San Diego, CA, USA, 4–8 June 2007; Association for Computing Machinery: New York, NY, USA, 2007; pp. 483–488. [CrossRef]
30. Bravos, G.; Cabrera, A.J.; Correa, C.; Danilović, D.; Evangeliou, N.; Ezov, G.; Gajica, Z.; Jakovetić, D.; Kallipolitis, L.; Lukić, M.; et al. Cybersecurity for Industrial Internet of Things: Architecture, Models and Lessons Learned. *IEEE Access* **2022**, *10*, 124747–124765. [CrossRef]
31. Huang, Z.; Wang, Q. A PUF-based unified identity verification framework for secure IoT hardware via device authentication. *World Wide Web* **2020**, *23*, 1057–1088. [CrossRef]
32. Matas, K.; La, T.; Grunchevski, N.; Pham, K.; Koch, D. Invited tutorial: FPGA hardware security for datacenters and beyond. In Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Seaside, CA, USA, 23–25 February 2020; pp. 11–20. [CrossRef]
33. Hategekimana, F.; Whitaker, T.J.; Pantho, M.J.H.; Bobda, C. IoT Device security through dynamic hardware isolation with cloud-Based update. *J. Syst. Archit.* **2020**, *109*, 101827. [CrossRef]

34. Peters, T.; Lal, R.; Varadarajan, S.; Pappachan, P.; Kotz, D. BASTION-SGX: Bluetooth and architectural support for trusted I/O on SGX. In Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy, Los Angeles, CA, USA, 2 June 2018; pp. 1–9. [CrossRef]
35. McKeen, F.; Alexandrovich, I.; Anati, I.; Caspi, D.; Johnson, S.; Leslie-Hurd, R.; Rozas, C. Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave. In Proceedings of the Hardware and Architectural Support for Security and Privacy 2016, Seoul, Republic of Korea, 18 June 2016; pp. 1–9. [CrossRef]
36. Zhang, N.; Li, J.; Lou, W.; Hou, Y.T. PrivacyGuard: Enforcing private data usage with blockchain and attested execution. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology, Proceedings of the ESORICS 2018 International Workshops, DPM 2018 and CBT 2018, Barcelona, Spain, 6–7 September 2018*; Springer International Publishing: Cham, Switzerland, 2018; pp. 345–353.
37. Cao, C.; Guan, L.; Zhang, N.; Gao, N.; Lin, J.; Luo, B.; Liu, P.; Xiang, J.; Lou, W. CryptMe: Data Leakage Prevention for Unmodified Programs on ARM Devices. In *Research in Attacks, Intrusions, and Defenses. RAID 2018*; Bailey, M., Holz, T., Stamatogiannakis, M., Ioannidis, S., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018; Volume 11050. [CrossRef]
38. Wang, J.; Hong, Z.; Zhang, Y.; Jin, Y. Enabling Security-Enhanced Attestation With Intel SGX for Remote Terminal and IoT. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *37*, 88–96. [CrossRef]
39. Pettersen, R.; Johansen, H.D.; Johansen, D. Secure Edge Computing with ARM TrustZone. In Proceedings of the IoTBDS 2017, Porto, Portugal, 24–26 April 2017; pp. 102–109.
40. Alves, T. Trustzone: Integrated hardware and software security. *Inf. Q.* **2004**, *3*, 18–24.
41. Guan, L.; Liu, P.; Xing, X.; Ge, X.; Zhang, S.; Yu, M.; Jaeger, T. Trustshadow: Secure execution of unmodified applications with arm trustzone. In Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, Niagara Falls, NY, USA, 19–23 June 2017; pp. 488–501. [CrossRef]
42. Paulin, D.; Hennebert, C.; Franco-Rondisson, T.; Jayles, R.; Loubier, T.; Collado, R. HistoTrust: Ethereum-Based Attestation of a Data History Built with OP-TEE and TPM. In *Foundations and Practice of Security. FPS 2021*; Aïmeur, E., Laurent, M., Yaich, R., Dupont, B., Garcia-Alfaro, J., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2022; Volume 13291. [CrossRef]
43. Shepherd, C.; Akram, R.N.; Markantonakis, K. EmLog: Tamper-Resistant System Logging for Constrained Devices with TEEs. In *Information Security Theory and Practice. WISTP 2017*; Hancke, G., Damiani, E., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018; Volume 10741. [CrossRef]
44. S2GO Pressure DPS310, Infineon Technologies. Available online: https://www.infineon.com/cms/en/product/evaluation-boards/s2go-pressure-dps310/ (accessed on 3 February 2023).
45. Optiga™ Trust M SLS32AIA, Infineon Technologies. Available online: https://www.infineon.com/cms/en/product/security-smart-card-solutions/optiga-embedded-security-solutions/optiga-trust/optiga-trust-m-sls32aia/ (accessed on 3 February 2023).
46. Johnson, D.; Menezes, A.; Vanstone, S. The elliptic curve digital signature algorithm (ECDSA). *Int. J. Inf. Secur.* **2001**, *1*, 36–63. [CrossRef]
47. Papageorgiou, A.; Mygiakis, A.; Loupos, K.; Krousarlis, T. DPKI: A Blockchain-Based Decentralized Public Key Infrastructure System. In Proceedings of the 2020 Global Internet of Things Summit (GIoTS), Dublin, Ireland, 3 June 2020; pp. 1–5. [CrossRef]
48. Yan, J.; Hang, X.; Yang, B.; Su, L.; He, S. Blockchain Based PKI and Certificates Management in Mobile Networks. In Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 29 December 2020–1 January 2021; pp. 1764–1770. [CrossRef]
49. Vangala, A.; Das, A.K.; Kumar, N.; Alazab, M. Smart Secure Sensing for IoT-Based Agriculture: Blockchain Perspective. *IEEE Sens. J.* **2021**, *21*, 17591–17607. [CrossRef]
50. Yakubov, A.; Shbair, W.; Wallbom, A.; Sanda, D. A blockchain-based PKI management framework. In Proceedings of the First IEEE/IFIP International Workshop on Managing and Managed by Blockchain (Man2Block) Colocated with IEEE/IFIP NOMS 2018, Tapei, Tawain, 23–27 April 2018.
51. Ezzat, S.K.; Saleh, Y.N.M.; Abdel-Hamid, A.A. Blockchain Oracles: State-of-the-Art and Research Directions. *IEEE Access* **2022**, *10*, 67551–67572. [CrossRef]
52. Caldarelli, G. Overview of Blockchain Oracle Research. *Future Internet* **2022**, *14*, 175. [CrossRef]
53. Pasdar, A.; Dong, Z.; Lee, Y.C. Blockchain oracle design patterns. *arXiv* **2021**, arXiv:2106.09349. https://doi.org/10.48550/arXiv.2106.09349 .
54. Mühlberger, R.; Bachhofner, S.; Castelló Ferrer, E.; Di Ciccio, C.; Weber, I.; Wöhrer, M.; Zdun, U. Foundational oracle patterns: Connecting blockchain to the off-chain world. In *Business Process Management: Blockchain and Robotic Process Automation Forum, Proceedings of the BPM 2020 Blockchain and RPA Forum, Seville, Spain, 13–18 September 2020*; Springer International Publishing: Cham, Switzerland, 2020; pp. 35–51. [CrossRef]
55. Kazemi, Z.; Papadimitriou, A.; Souvatzoglou, I.; Aerabi, E.; Ahmed, M.M.; Hely, D.; Beroulle, V. On a Low Cost Fault Injection Framework for Security Assessment of Cyber-Physical Systems: Clock Glitch Attacks. In Proceedings of the 2019 IEEE 4th International Verification and Security Workshop (IVSW), Rhodes, Greece, 1–3 July 2019; pp. 7–12. [CrossRef]
56. Mills, D.; Martin, J.; Burbank, J.; Kasch, W. Network Time Protocol Version 4: Protocol and Algorithms Specification (No. rfc5905). 2010. Available online: https://www.rfc-editor.org/rfc/rfc5905.txt (accessed on 4 July 2023).
57. Franke, D.; Sibold, D.; Teichel, K.; Dansarie, M.; Sundblad, R. Network Time Security for the Network Time Protocol. RFC 8915. 2020. Available online: https://datatracker.ietf.org/doc/html/rfc8915 (accessed on 4 July 2023).

58. Estevam, G.; Palma, L.M.; Silva, L.R.; Martina, J.E.; Vigil, M. Accurate and decentralized timestamping using smart contracts on the Ethereum blockchain. *Inf. Process. Manag.* **2021**, *58*, 102471. [CrossRef]
59. Ma, G.; Ge, C.; Zhou, L. Achieving reliable timestamp in the bitcoin platform. *Peer-to-Peer Netw. Appl.* **2020**, *13*, 2251–2259. [CrossRef]
60. Echave, J.; Barral, M.; Fraga-Corral, M.; Prieto, M.A.; Simal-Gandara, J. Bottle aging and storage of wines: A review. *Molecules* **2021**, *26*, 713. [CrossRef]
61. Ough, C.S. Some effects of temperature and SO_2 on wine during simulated transport or storage. *Am. J. Enol. Vitic.* **1985**, *36*, 18–22. [CrossRef]
62. Tijan, E.; Aksentijević, S.; Ivanić, K.; Jardas, M. Blockchain technology implementation in logistics. *Sustainability* **2019**, *11*, 1185. [CrossRef]
63. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018; pp. 1–15. [CrossRef]