Enhanced Sensor Communication through Trusted Computing

Anestis Papakotoulas[1], Theodore Milonas[2], Kakia Panagidi[1], and Stathes Hadjiefthymiades[1]

apapakot@di.uoa.gr, theo_milon@yahoo.gr, kakiap@di.uoa.gr, shadj@di.uoa.gr

[1]Department of Informatics & Telecommunications, National and Kapodistrian University of Athens

[2]Department of Science & Technology, Hellenic Open University

Abstract

Wireless Sensor Networks (WSNs) are an essential part of the Internet of Things (IoT), providing connectivity to a wide range of applications, such as environmental monitoring, smart homes, healthcare, and industrial automation. WSNs consist of a large number of small, low-cost, and power-constrained sensor nodes that communicate wirelessly to a base station or sink node. However, the inherent characteristics of WSNs, such as limited resources, ad-hoc networking, and open communication channels, make them vulnerable to various security threats. Therefore, secure and efficient communication is essential for WSNs to provide reliable and trustworthy services. One approach to enhancing the security of WSNs is by incorporating the Trusted Platform Module (TPM), which is a hardware-based security solution that provides secure storage and processing of sensitive information. The TPM can be used to secure the communication between sensor nodes and the base station, as well as to authenticate the nodes and prevent unauthorized access. Moreover, the TPM can be used to implement secure group communication, which is essential for many WSN applications. In this paper, we propose an enhanced sensor communication scheme for WSNs using TPM.

Our approach provides a secure and efficient communication mechanism for WSNs by integrating the TPM with the communication protocol. The proposed scheme supports secure end-to-end communication, group communication, and authentication, and it is designed to work with different communication protocols. The evaluation results demonstrate the effectiveness of the proposed scheme in enhancing the security of WSNs providing significant outcomes for the management of computing resources and memory. Therefore, this research can contribute to the development of secure and trustworthy WSNs for IoT applications.

1 Introduction

Wireless sensor networks (WSNs) have emerged as a promising technology in various domains, including healthcare, environmental monitoring, and industrial automation. With the advancement of sensor technology, WSNs have become more powerful and pervasive. However, the security and privacy of sensor data remain a significant challenge. Hence, the communication among sensors and the data collected from them must be secured to avoid potential attacks from malicious nodes or eavesdroppers.

Therefore, a trusted link (based on security services) is essential to provide trusted services between the aforementioned technical sectors and to establish a trustworthy IoT ecosystem for everybody. Trust is an essential component of security that may be earned or granted but should never be assumed. Many applications are constructed with uncontrollable server and client ends. The Trusted Computing Group (TCG) is developing and promoting trusted computing as an emerging technology. The Trusted Computing Group (TCG) is an industry collaboration that creates trusted computing-based [5]. Implementation of the Trusted Platform Module (TPM) is one of TCG's specifications [3]. TPM allows secure bootstrapping and enables remote parties to verify that only authorized code is running on a system [20]. In recent years, several approaches have been proposed to enhance the security of WSNs, including the use of the Trusted Platform Module (TPM) to provide secure communication and protect sensitive data.

In this paper, we propose a solution to enhance communication in WSNs using a TPM. The TPM provides secure storage and cryptographic operations, which can be used to authenticate and encrypt the communication between sensors. The proposed solution enhances the confidentiality, integrity, and authenticity of sensor data, which are critical requirements for WSNs. Our contribution provides a novel approach to secure WSNs and can be useful in various applications that require secure communication.

This paper focuses on the transmitted data that is generated from the sensors in a typical swarm design where a server serves as the resource controller. The supervision of nodes and the administration of sensors might be two of the most important roles of a resource controller. This architecture consists of smart embedded devices (called "nodes") networked to a server that orchestrates the nodes. Figure 1 depicts a node in its generic structure, which comprises mul-

tiple sensors that collect environmental data. At this node, a multitude of cloud-based apps that communicate with one another are operating. The server and other nodes connect with each other over a wireless interface. A Trusted Platform Module has been added to the node to bolster security. TPM functions as a Root of Trust for every node, enhancing its security.

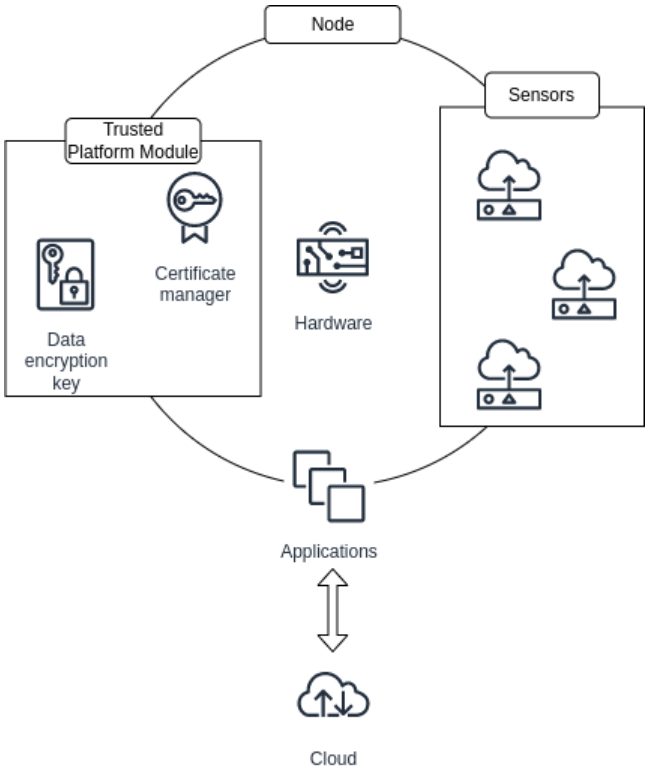

Figure 1. IoT architecture in Trusted Computing.

Each application and sensor is able to leverage the TPM, which employs cryptographic techniques to encrypt its data. TPM might be a hardware or software component. Also deployed on the server is a trusted component.

1.1 Contribution

In a flying ad-hoc network, UAVs can act as both sensor nodes and routers, collecting and forwarding data to a central location or to other nodes in the network. This enables the network to cover a large area and provide a comprehensive view of the environment being monitored. Lets assume that they are using sensing systems to operate road traffic checks. All of this transmitted data is extremely sensitive and must be secured accordingly.

The security of a distributed Internet of Things (IoT) system should consist of multiple components. As shown in Figure 2, some of these stages include infrastructure integrity, node authentication, and data confidentiality.

The first component relates to the infrastructure's own dependability. To ensure the integrity of the infrastructure, operating systems, and running programs, it is necessary to confirm who runs the application. In distributed systems, machines or programs running on those machines prove their authenticity to other machines by authenticating themselves.

Figure 2. Stages of the security of an IoT system.

The second component is node authentication, which is the process of verifying the identity of a node (device or sensor) before allowing it to access a wireless network. It is a security mechanism that ensures that only authorized nodes are allowed to communicate with the network and helps prevent unauthorized nodes from accessing sensitive data or disrupting the network. Node authentication typically involves the exchange of authentication credentials between the node and the network, such as digital certificates, passwords, or shared secrets. These credentials are used to verify the node's identity and determine whether it should be granted access to the network. Node authentication is an important aspect of wireless network security and is often used in combination with other security mechanisms, such as encryption and access control, to provide a comprehensive security solution.

An Internet of Things device manages and generates data that is transmitted to other nodes. This type of data should be protected to prevent vulnerabilities that allow intruders to damage topology components. Encryption can be used to protect this information, along with the use of TPM. The aforementioned is examined in detail in this paper, and its measurements are provided in Section V.

Given the importance of data security, an improved sensor communication approach for wireless sensor networks based on a TPM is proposed. To extract useful outcomes, the performance and security of the proposed solution are evaluated using simulations in a real-world scenario. Specifically, authors contribute to the field of WSNs by:

- Introducing a novel communication protocol that uses TPM to secure the communication between sensors in the network. Our approach provides end-to-end encryption and integrity protection for sensor data, thus ensuring the confidentiality and authenticity of the data.
- Proposing a lightweight key management scheme that enables efficient key distribution and management among sensors. The scheme uses the TPM's unique identity to generate session keys, which are used to encrypt and authenticate data packets.
- Conducting extensive experiments to evaluate the performance of our proposed approach in terms of communication overhead, energy consumption, and security.

Overall, our proposed approach enhances the security and efficiency of sensor communication in WSNs and can be applied to various applications, such as smart cities, industrial control, and environmental monitoring. It employs TPM to establish a secure and authenticated communication channel between the nodes and the gateway and yields substantial results for the management of computational resources.

1.2 Structure of the Paper

The rest of this paper is organized as follows: Section II discusses the related work on security in WSNs using TPM and defines the main concepts and components studied in this work. Section III presents our proposed enhanced sensor communication scheme, which is followed by the evaluation of the proposed scheme in Section IV. Finally, Section V concludes the paper and presents future research directions.

2 Related Work & Background

2.1 Related work - State of the art

Security is regarded as one of the primary prerequisites to the widespread adoption and implementation of the Internet of Things and wireless sensor networks. In fact, IoT and WSN security are popular discussion topics [18], [4]. The paper [8] provides an overview of the most significant security issues in WSN, with a particular emphasis on the security vulnerabilities in the various protocol layers.

2.1.1 Security in WSN

In [16], the authors provide a comprehensive survey of security issues in WSNs. They review the major security threats and attacks that can occur in WSNs and present an overview of the security mechanisms that have been proposed to address these issues. The paper also discusses the challenges associated with implementing security mechanisms in WSNs, such as limited resources and the need for energy-efficient solutions. Finally, the paper presents a discussion of the future research directions in WSN security, including the need for new security protocols and techniques that can be adapted to the unique characteristics of WSNs.

The [17] survey discusses the security challenges and potential threats in WSNs. The paper highlights different types of attacks that WSNs are vulnerable to, such as jamming, spoofing, eavesdropping, and denial-of-service attacks. Additionally, the paper also reviews the countermeasures proposed by various researchers to mitigate these attacks. The study provides insights into the current state of security in WSNs and identifies future research directions to enhance the security of WSNs.

In [13], an overview of the various security issues that arise in WSNs is provided. The authors highlight the unique characteristics of WSNs, such as resource constraints, scalability, and ad-hoc network topology, that make them vulnerable to security attacks. They discuss the various types of attacks that WSNs face, including routing attacks, denial-of-service attacks, and eavesdropping attacks, and the potential consequences of these attacks. The paper also reviews the existing security solutions for WSNs, including cryptographic algorithms, intrusion detection systems, and key management techniques. The authors conclude by identifying open research issues and potential directions for future work in the field of WSN security.

This research [14] presents a comprehensive survey of security threats and countermeasures in the IoT domain. The authors highlight the increasing number of IoT devices and their growing importance in various fields such as healthcare, transportation, and smart cities. However, the authors also point out the security challenges that arise due to the heterogeneity and complexity of IoT systems. The survey covers various security threats such as unauthorized access, data tampering, and denial-of-service attacks. Additionally, the authors also discuss several countermeasures to mitigate these threats, including authentication, encryption, and intrusion detection systems. The paper concludes with a discussion of the limitations of existing security solutions and the need for further research in this area.

2.1.2 TPM in WSN

The paper [7] proposes the use of TPM-based architectures to enhance the physical security of WSN. The authors argue that current security solutions for WSNs have limitations and do not address physical security adequately. They propose the use of TPMs, which are hardware-based security solutions that can securely store cryptographic keys and provide secure boot and remote attestation. The paper discusses the design and implementation of a TPM-based architecture for WSNs and presents the results of experiments conducted to evaluate its performance.

The paper [10] proposes a framework for the integration of security measures in the IoT to create a secure domain of sensor nodes, which is essential in critical areas like the military. The lack of robust, secure, and trusted measures to ensure the confidentiality, availability, and integrity of information throughout its lifecycle limits the application of IoT in such areas. The proposed framework uses TPM in wireless sensor nodes to generate credentials, build local trust structures, and establish trust relationships between domain nodes.

In [21], a TPM-based conditional privacy-preserving authentication protocol (T-CPPA) is proposed to ensure the integrity and authenticity of messages and instructions in vehicle ad-hoc networks (VANETs) based on the Internet of Vehicles (IoV). The T-CPPA protocol embeds the system master private key into the TPM to generate pseudonyms and signature keys, protecting the privacy of the vehicle. The authenticity of the message content is ensured by calculating message similarity in a cluster-based model. The scheme is designed in symmetric bilinear groups and includes a batch validation algorithm to improve efficiency.

The paper [12] explores the need for robust authentication protocols in Internet of Drones (IoD) networks to mitigate security and privacy threats. The paper discusses the challenges of designing efficient and lightweight authentication solutions due to the limitations of Unmanned Aerial Vehicles (UAVs) in terms of energy, computational, and memory resources. The paper presents up-to-date research studies on authentication mechanisms for IoD networks, including conventional technologies and methods such as hash functions, Public Key Infrastructure (PKI), and Elliptic-Curve Cryptography (ECC), as well as emerging technologies such as Mobile Edge Computing (MEC), Machine Learning (ML), and Blockchain. The paper also reviews effective hardware-based solutions for identifying and authenticating network nodes within the IoD based on TPM, Hardware Security Modules (HSMs), and Physically Unclonable Functions (PUFs). Finally, the paper provides future directions on these relevant research topics, stimulating further work.

The paper [19] addresses the growing concern over cy-

ber threats to Supervisory Control and Data Acquisition (SCADA) and automation systems in the Industrial Internet of Things (IIoT) and Industry 4.0 eras. The use of insecure communication protocols, such as Modbus TCP, DNP3, and S7, increases the vulnerabilities of these systems, exposing the data to outside networks and making them susceptible to attack. To respond to this security issue, the paper proposes integrating TPM to ensure the authenticity of transmitted data. The experimental results show that integrating TPM in automation/SCADA systems can enhance security and reduce the risk of intrusion. Two methods are proposed to assure the authenticity of transmitted messages, and the paper presents measurements related to the increased time latency introduced due to the proposed concept. Overall, the paper presents a perspective on addressing the security challenges of legacy structures in the context of the IIoT and Industry 4.0 era.

In [9], a proprietary solution for using the TPM to authenticate sensors in wireless sensor networks that create a sensor's domain is described. The authors present a model of the network, which includes three types of nodes: the M node, which is the authentication authority and data recipient; the S node, which is the source of sensor data; and the rM node, which acts as a backup for the M node. The paper also outlines the main operations available in the sensors' domain, including managing sensors, authentication, and regeneration of node credentials.

The paper [11] proposes a lightweight authentication and key agreement scheme based on TPM for WSN. The scheme is designed to provide secure communication and mutual authentication between the nodes of the WSNs without requiring a significant amount of computational resources. The proposed scheme utilizes the TPM to generate and manage the secret keys, which are used for authentication and encryption. It also employs ECC for efficient key exchange and message authentication.

All the above research discusses the use of TPM for various aspects of wireless sensor network security, including key management, data aggregation, trust management, and communication protocol design.

2.1.3 WSN & Security requirements for WSN

WSNs face a number of challenges, including limited energy and computing resources, high data traffic, network topology, node placement, security, and data reliability. Additionally, WSNs often require specialized protocols and algorithms to optimize network performance and address these challenges. Furthermore, the wireless nature of WSNs makes them vulnerable to security threats. To ensure the security of WSNs, several security requirements need to be considered[15] [8].

Firstly, authentication and authorization mechanisms should be employed to ensure that only authorized users or devices can access the network. Secondly, data confidentiality should be guaranteed through encryption techniques to prevent unauthorized access or eavesdropping. Thirdly, message integrity should be ensured to prevent tampering with the messages transmitted over the network. Fourthly, network availability should be maintained to prevent denial-of-service attacks. Additionally, intrusion detection and prevention mechanisms should be in place to detect and mitigate any potential security breaches. Finally, the limitations of the resources have to be considered. Sensors are typically low-power and low-cost devices with limited resources, which makes it challenging to implement strong security measures.

By fulfilling these security requirements, WSNs can be made more secure and reliable for various applications.

2.2 TPM

The TPM is a secure microcontroller that provides cryptographic functionality to a computer. The TPM can be used to securely store cryptographic keys and perform cryptographic operations, such as digital signatures, encrypting/decrypting data, and generating cryptographic hashes [6] [3].

Measurements using TPM involve securely measuring and storing the state of the system, including software and configuration information. This information is used to establish the trustworthiness of the system and ensure that it has not been altered or compromised. The TPM can measure various components of the system, including the BIOS, boot loader, kernel, and applications. The TPM also provides a secure environment for storing keys and other sensitive information, making it difficult for attackers to access this information.

Overall, the TPM provides a secure way to store and process sensitive information, and to measure the state of the system in order to establish trust.

3 The Proposed Scheme

To satisfy the WSN security standards, the Secure Hash Method (SHA), the AES symmetric key algorithm, and the RSA asymmetric key algorithm are utilized in conjunction with the hardware TPM. In addition to verifying digital signatures, the TPM is capable of encryption and decryption operations.

The novelty of this research is that it presents an efficient method for securely transmitting sensor data in a distributed network by using the capabilities of TPM 2.0.

The proposed model is presented in two parts, the first of which specifies how to ensure the communication between two nodes by explaining in detail the processes that take place in each. In the second section, two flow diagrams of the data sequence on the Client and Server are presented.

3.1 Enhanced Sensor Communication Scheme

The establishment of communication between Client and Server is carried out using the steps outlined below, where the first four steps are identical for both the RSA and AES encryption algorithms and the remaining steps vary depending on the encryption algorithm selected by the Client for the encrypted information being transmitted:

- The Client initiates communication by sending a connection request to the Server (Request Connection).
- Upon receiving the connection request, the Server generates a pair of RSA keys (public PUKs and private PRKs) and returns the client's public key (PUKs).
- Following receipt of the server's public key, the client generates its own RSA key pair (public PUKc and private PRKc), encrypts its public key with the server's

public key, and sends the server its public key encrypted PUKs (PUKc).

- The Server then decrypts PRKs(PUKs(PUKc)) using its own private key, which it generated at the beginning of the communication, after receiving the Client's public key.

3.1.1 RSA Encryption (a)

Figure 3 shows the key and data exchange between Client and Server using RSA algorithm. The steps is implemented as follows: i. The Server continues its operation by generating a pair of shared keys (public PUKcom and private PUKcom) RSA-1024, encrypting it with the client's public key that received the new public key PUKc(PUKcom), and sending it back to the client. ii. Upon receiving the new public shared key from the Server, the Client decrypts it with its private key PRKc(PUKc(PUKcom)) and, if the decryption is correct, encrypts a message with the PUKcom shared public key ("message"), which is returned to the Server. iii. Receiving the new encrypted message, the Server decrypts it with the shared private key PRKcom(PUKcom ("message")), and if the decryption was successful, it digitally signs with the private key a message for the successful communication SIGN(PRKcom ("message"), which is sent to the Client. The Server is now prepared to accept encrypted information from the Client encrypted information with the shared public key (PUKcom). iv. Lastly, the Client receives the digitally signed message, verifies it with the shared public key (PUKcom), and if it has a valid signature (VALIDATE (PUKcom(SIGN(PRKcom("message")))), establishes it as permanent.

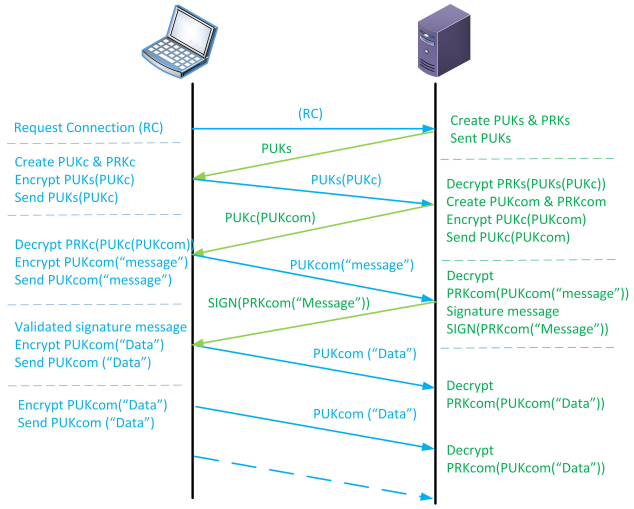

Figure 3. Cryptographic key and data exchange between nodes using RSA algorithm.

3.1.2 AES Encryption (b)

Figure 4 shows the key and data exchange between Client and Server using AES algorithm. The steps is implemented as follows: i. The Server continues its operation by generating an AES public key (AESKcom), encrypting it with the previously received public key of the Client, and sending the new public key PUKc(AESKcom) back to the Client. ii. Upon receiving the new public key from the Server, the Client decrypts it with its private key PRKc(PUKc (AESKcom)) and, if the decryption is correct, encrypts a message with the public key AESKcom(" message") before sending it back to the Server. iii.When the Server receives the new encrypted message, it decrypts it with the AESKcom public key(AESKcom("message")), and if the decryption was successful, it encrypts a message with the AESKcom("message") public key and sends it to the Client. o The Server is now prepared to accept encrypted information from the Client using the shared public key (AESKcom). iv. Finally, the Client receives the new encrypted message, decrypts it using the AESKcom shared key(AESKcom ("message")), and if the decryption was successful, establishes permanent communication with the Server and begins sending encrypted data using the AESKcom shared key ("Data").

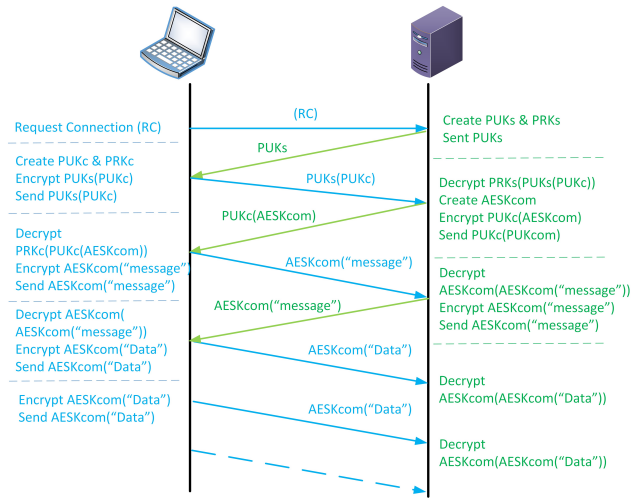

Figure 4. Cryptographic key and data exchange between nodes using AES algorithm.

In order to find the most efficient way to encrypt and decrypt information in terms of time and the use of IoT resources (memory, CPU usage, etc.), we will look at both the performance of assigning the encryption and decryption processes to the processor (CPU) and the performance of the system when the encryption and decryption processes are given to TPM 2.0.

3.1.3 Assumptions

Authentication, as described in Section 1.1, is the second security component of a distributed IoT system (Node authentication). The authentication of the client and server is therefore not relevant to this study; it is assumed that nodes are authenticated. Additionally, the proposed solution does not include an attestation service, which is the requirement of the first component (Integrity of the infrastructure). The TPM is utilized to encrypt sensor-generated data and decrypt it on the other end. Using a trusted component, such as TPM, to ensure the confidentiality of transmitted data is the objective of this research.

3.1.4 *Flow diagram of the proposed scheme*

Figure 5 shows the overall flow diagram for data exchange between the Server and the Client, and encryption and decryption using the CPU and TPM 2.0.

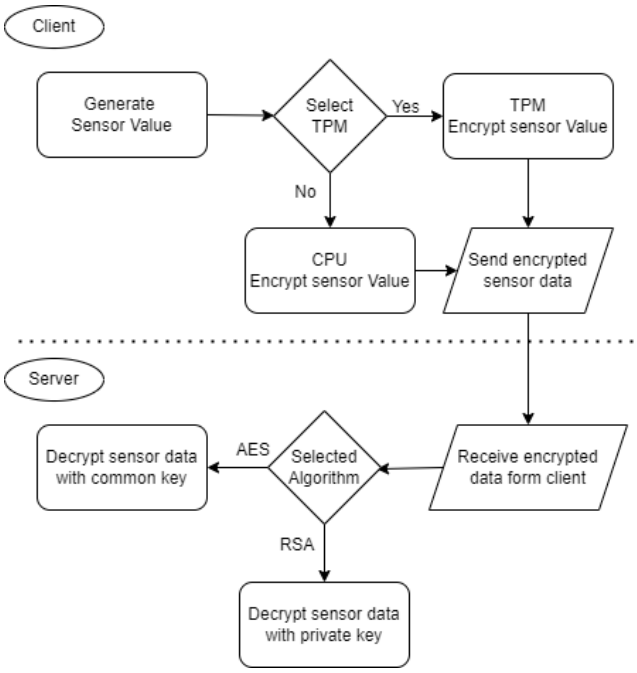

Figure 5. Flow diagram of Enhanced Sensor Communication Scheme

It is appropriate to mention that the server decrypts data with the setting (TPM or CPU) set during the authentication phase.

4 Experiments

4.1 Experimental Environment

The experimental part was conducted using two approaches: one with the Raspberry Pi as Client for data encryption and a PC as Server, and the other with the Raspberry Pi as Server for data decryption and a PC as Client. The PC used to send data to the RPi, either as a Client or a Server, has an i7 processor of the seventh generation, 8 cores, and a frequency of 2.8 GHz, which is significantly higher than the RPi's 1.2 GHz. As a result, it completes the tasks assigned to it more quickly. The RPi 3 Model B board was chosen to support this project's implementation since it has enough capacity to manage cryptographic procedures. The RPi Model 3 B is widely used in Unmanned and autonomous vehicles (UxVs) and is supported by various bootloaders. For the sensor communication scheme, the Infineon IRIDIUM SLI 9670 TPM2.0 board will be used, on which the Infineon OPTIGA™ SLI9670 TPM 2.0 circuit is installed. OPTIGA™ TPM SLI 9670 microcontroller adheres to the TCG 2.0 family's standards (Figure reffig5), is compatible with the RPi Model 3 B, and provides a variety of security functionalities, including those described in [2] [1].

For the evaluation of the system, a basic scenario with five sensors was created. As shown in Table 1, the selected sampling frequencies in the basic scenario are 1Hz (1sec), 0,5Hz (2sec), and 0,2Hz (5sec) for 60%, 20%, and 20% of the values, respectively.

Table 1. Parameters of sensors

No. of Sensor	Frequency (Hz)	Data Type
$Sensor_1$	1.0	Signed Integer
$Sensor_2$	1.0	Unsigned Integer
$Sensor_3$	1.0	Boolean
$Sensor_4$	0.5	Float
$Sensor_5$	0.5	Unsigned Integer

The scenario initially applied the "Create & Send" method, the mass sending method with two time windows of 10 and 20 seconds, and finally the mass sent by packaged method with the LIFO (Last Inputs – Last Output) algorithm for both unencrypted information and encrypted information. For encryption, the algorithms AES-128, RSA-1024, and RSA-2048 were used both with the use of JAVA libraries, where the assignment of encrypting and decrypting is assigned to the CPU of the device, and with Microsoft's TSS libraries for JAVA, where the above processes are assigned and carried out in TPM 2.0.

With the start of the scripts, the clock skew between the devices operated by the Client and Server apps was initially calculated so that absolute times could be found with relative accuracy. The applications were executed for a period of at least 10 minutes so that there is a representative sample of information encryption and decryption measurements.

4.2 Implementation Assumptions

In this paragraph, the implementation assumptions of the code are stated. IPv4 is the protocol in use. Before beginning the authentication procedure, the time difference between the server and client clocks is determined. The Client sends an unencrypted packet of data to the Server along with its settings (TPM support, selected encryption algorithm, key size, and maximum encryption packet size) at the beginning of the communication, and the Server sends an identical packet to the Client. The application uses the encryption/decryption method (CPU or TPM) for which it is configured, regardless of the settings of the other; i.e., if the Client is configured to encrypt with the CPU, it will not change the method if the Server uses TPM; however, the Server will decrypt with TPM in this case. Multiple Clients can connect to the Server simultaneously (logical star topology). In addition, the PC's TPM operates at a higher frequency than that of the RPi. In order to obtain a representative sample of information encryption/decryption measurements, algorithms were executed for a minimum of ten minutes. There were twenty repetitions for each cryptographic method. The application uses for encryption and decryption the method (CPU or TPM) for which it is configured, regardless of the settings of the other; i.e., if the Client is set to encrypt with the CPU it will not change the method if the Server works with TPM, while in this case the Server will decrypt using TPM. The Client application algorithm for the "Create and Send" setting is implemented using a thread and goes into inaction mode until the data is produced by each sensor; instead, the other setting "Mass Sending" is implemented with a single

thread algorithm that never goes into rest as it collects sensor values and checks if the minimum payload size for sending the information to the Server has been met.

Considering that the process of authentication between Client and Server is complete, applications are ready to exchange encrypted information using either the common public key or the client's public key. Thus, it is not possible for third parties to know what information is exchanged, but at the same time, integrity is ensured as any modification to the information transmitted after sending makes it impossible to decrypt and thus reject it by the application that performs server transactions.

4.3 Experimental results

According to the times and the payload size of the information encrypted or decrypted, the processed statistics for the encryption or decryption rate (Bytes/msec) are given in relation to the payload size (bytes) of the data in Table 2.

Average CPU and memory usage values for each selected encryption/decryption algorithm are given in the Tables 3 and 4, as well as the recorded values of the resources when not applying an encryption algorithm (no encryption).

The maximum amount of RSA-1024 information that can be encrypted, according to the application settings, is 117 bytes; the remaining 11 bytes are used for the header, while bigger amounts of information are separated into packets of 117 bytes or less. The maximum amount of RSA-2048 information that can be encrypted using asymmetric encryption is 245 bytes; the remaining 11 bytes are used for the header, while larger amounts of information are broken up into packets of 245 bytes or lower. When RSA-1024 is used to encrypt data, it requires 128 bytes, whereas RSA-2048 requires 256 bytes. Before comparing and analyzing the encryption methods, it is necessary to make a few remarks and clarify how the tests were conducted. All provided times are recorded for the RPi. It is mentioned that the server produces keys twice, which is the most time-intensive operation. As the PC generates the two needed keys in a shorter amount of time, the time required to complete the authentication process while the RPi is acting as a client should be less than when the RPi is acting as a server.

4.4 Encryption/Decryption Outcomes - Data Encryption/Decryption Rates

Observing Figure 6 and examining each algorithm individually reveals the following during the Client application's encryption processes: For the AES-128 algorithm, we observe a steady increase in data encryption up to approximately 500 bytes, then an increase to approximately 3000 bytes, and then a stabilization of performance. For the RSA-1024 algorithm utilizing CPU and TPM, as well as the RSA-2048 algorithm utilizing CPU, it exhibits a linear performance without significant fluctuations in the data it encrypts over time, i.e. a steady performance. Furthermore, the performance of RSA-2048 using TPM improves as the number of bytes increases.

Table 2 demonstrates, based on the decryption of data on the RPi using the Server application, that, for the AES-128 algorithm, an improvement in performance is stabilized for a payload of 3000 bytes. The decryption performance of all other algorithms is consistent. It should be noted that there was no variation in RSA-2048 using TPM, as the minimum information necessary for decryption is 256 bytes, regardless of the payload size of the useful information (about 17). Overall and for all encryption algorithms, a conclusion that could be stated is that AES-128 has the best encryption times, followed closely by RSA-1024 and RSA-2048 when using a CPU, and then by RSA-2048 and RSA-1024 when employing a TPM.

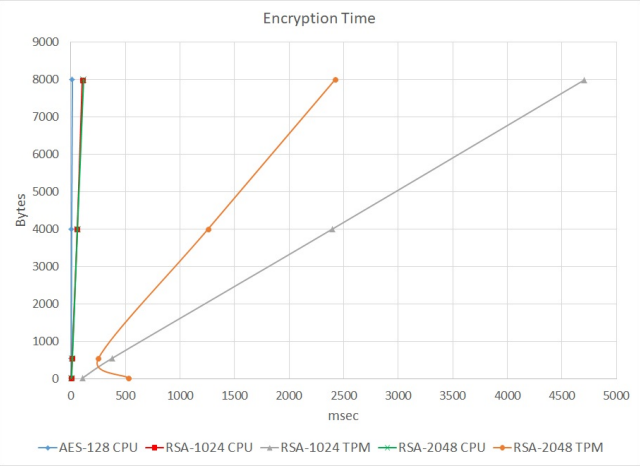

Figure 6. Average encryption time (msec) according to the payload size (bytes) of the information per cryptographic algorithm.

Comparing the decryption algorithms reveals that AES-128 is the most efficient, followed by RSA-1024 and RSA-2048 using CPU, and then RSE-1024 and RSA-2048 using TPM.

Figure 7 demonstrates the schematic evaluation of decryption in terms of time.

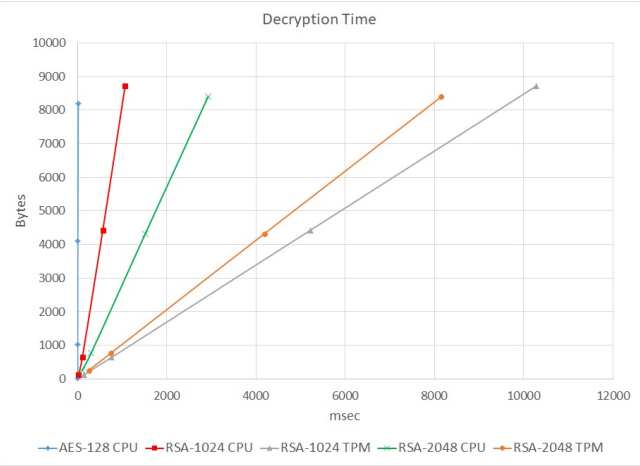

Figure 7. Average decryption time (msec) according to the payload size (bytes) of the information per cryptographic algorithm.

Comparing encryption and decryption separately for each algorithm reveals that decryption requires more time than en-

Table 2. Encryption & Decryption rates according to the payload size of the information

Algorithm	Encryption		Decryption	
	Payload size (bytes)	Rate (bytes/msec)	Payload size (bytes)	Rate (bytes/msec)
AES-128	17	8.27	17	15.00
	543	147.31	1024	597.89
	4003	609.58	4096	1121.83
	7994	593.60	8192	982.08
RSA-1024 CPU	17	4.97	128	5.73
	543	50.89	640	6.46
	4003	68.69	4429	7.87
	7981	76.50	8717	8.19
RSA-1024 TPM	17	0.16	128	0.84
	542	1.44	640	0.84
	4003	1.67	4422	0.85
	7981	1.70	8712	0.85
RSA-2048 CPU	17	2.97	256	2.62
	543	40.02	768	2.54
	4003	66.85	4318	2.85
	7988	68.74	8397	2.87
RSA-2048 TPM	17	0.04	256	1.03
	543	2.16	768	1.03
	4003	3.18	4322	1.03
	7992	3.30	8397	1.03

Table 3. Average percentage use of CPU & memory in encryption process

Algorithm	CPU	Memory
No encryption	23.65%	29.92%
AES-128 CPU	24.11%	30.38%
RSA-1024 CPU	24.29%	30.15%
RSA-1024 TPM	25.25%	31.35%
RSA-2048 CPU	26.41%	30.45%
RSA-2048 TPM	31.20%	30.70%

Table 4. Average percentage use of CPU & memory values in decryption process

Algorithm	CPU	Memory
No encryption	6.8%	29.46%
AES-128 CPU	6.73%	29.76%
RSA-1024 CPU	8.15%	29.67%
RSA-1024 TPM	8.65%	31.02%
RSA-2048 CPU	11.13%	28.73%
RSA-2048 TPM	8.18%	29.78%

coding for all RSA applications. In contrast to the AES algorithm, decryption times are shorter than encryption times, meaning it takes less time to decrypt the same data than it did to encrypt it. When we examine the AES algorithm closely, we observe that the greatest deviation is observed for small volumes of data, while as the volume of data increases, the times appear to converge. In contrast, in applications of the RSA algorithm, small data volumes are associated with the smallest time variations, whereas large data volumes are associated with significant time variations.

Regarding the client-side encryption rate for each algorithm, it can be seen that AES-128 increases the encryption rate (0-200 bytes/msec) geometrically from 0 to 550 bytes, then proportionally up to 5000 bytes of information to encrypt, reaching 820 byts/msec, before stabilizing. The encryption rate of RSA-1024 is marginally higher than that of RSA-2048 when using a CPU whose exponential increase reaches close to 500 bytes and is of the order of 60 bytes/msec for the first while for the second at 40 bytes/msec, then RSA- 1024 follows a numerical increase of the rate up to about 79 bytes/msec, until the end, while RSA 2048 follows a numerical rise of the pace up to about 70 bytes The RSA-1024 and RSA-2048 algorithms produce a similar picture of the TPM encryption rate, but with a significantly lower rate, which never exceeds 3.4 bytes/msec. Comparing all encryption algorithms, it is immediately apparent that AES-128 has the highest encryption rate, followed by RSA-1024 and RSA-2048 when using the CPU, and RSA 2048 and RSA-1024 when using the TPM. RSA 2048 and RSA-1024 have very low encryption rates, with a difference that exceeds twentyfold. Figure 8 provides a summary of the encryption rates for all examined algorithms.

AES-128 exhibits an exponential increase in information decryption rate up to 597 bytes/msec for information of 1024 bytes, then continues to rise and doubles to 1120 bytes/msec for information of 4096 bytes, and eventually reaches saturation and drops to 982 bytes/msec for information of 8192 bytes. The decryption rates are significantly lower for RSA-1024 (approximately 5.7 to 8 bytes/msec) and RSA-2048 (2.5 to 2.8 bytes/msec) when using the CPU; however, the initial variation observed in RSA-2048 between decryptions of 256 and 768 bytes, approximately 0.07 bytes per msec, is within the statistical error margin. RSA-2048, with a decryption rate just exceeding 1 byte/msec, is followed by RSA-

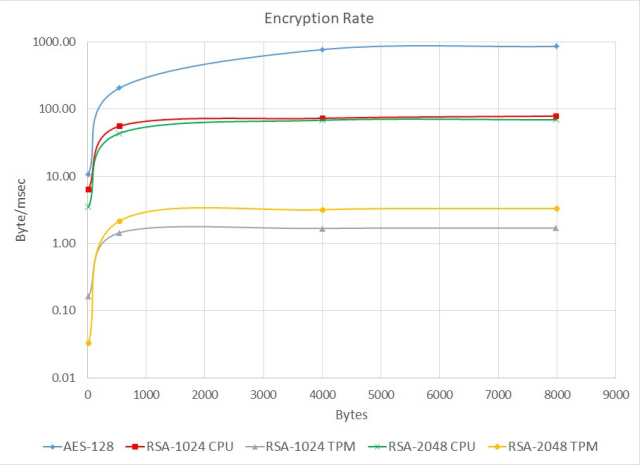

Figure 8. Data encryption rates (bytes/msec) per cryptographic algorithm.

1024, with a rate not exceeding 0.85 byte/msec. Figure 9 illustrates the decryption rates for each algorithm.

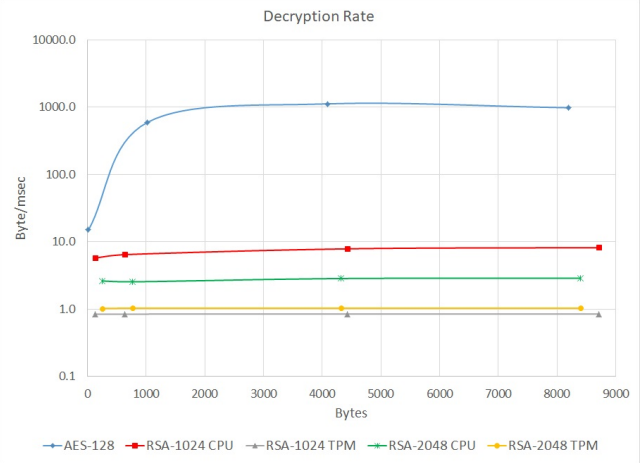

Figure 9. Data decryption rates (bytes/msec) per cryptographic algorithm.

Comparing the encryption and decryption rates for each algorithm reveals that for all RSA applications, the encryption rate is significantly higher than the decryption rate, with the exception of very small data volumes where the rates are nearly identical. In contrast to the use of a CPU, where the encrypting rate is 90 times greater than the decrypting rate, the difference between encryption and decryption rates is close to 50 percent when algorithms utilizing TPM are applied. Instead of the AES algorithm, the decryption rate is greater than the encryption rate, and while the two rates are identical for small data, they tend to converge for large data volumes.

4.5 Resources consumption outcomes

AES-128 appears identical to RSA-1024 in terms of processing power and the quantity of data it encrypts, as shown in Figure 10; the minimum greater requirement has RSA-1024 using TPM; and the highest requirement has RSA-2048 using CPU and TPM, where increasing the quantity of data to encrypt appears to identify. In general, we observe that the requirements for processing power are consistent, and the differences between them are in the order of 1–2%. It appears that the load of information encryption has no significant effect on CPU utilization. If we observe the use of non-encryption and encryption methods, the load is restricted to approximately 3% on average.

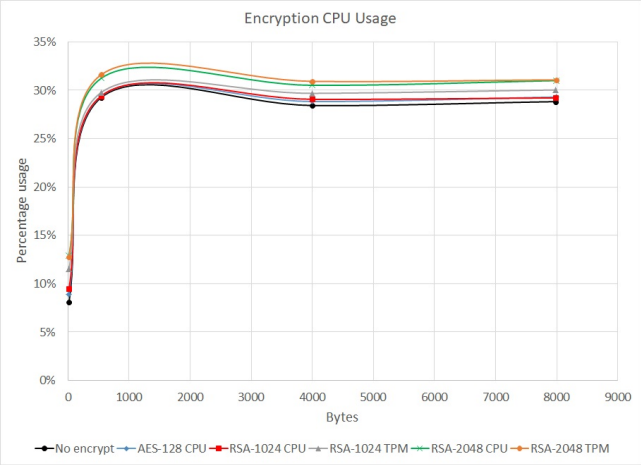

Figure 10. CPU usage in data encryption.

Examining the memory requirements during the encryption process for the encryption algorithms in Figure 11, we observe that there is little variation both within each algorithm and between them, with the average value being close to 30% and the deviation within each algorithm not exceeding 4%. The variance in memory usage for selecting and sending unencrypted information (NO Encrypt) to the client application relative to the applied algorithms is capped at approximately 3%.

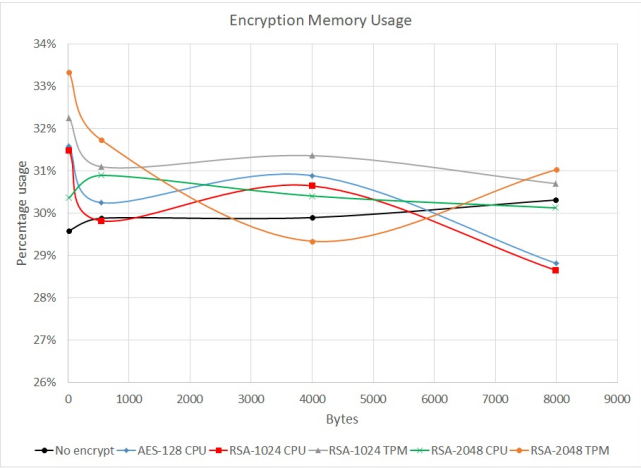

Figure 11. Memory usage in data encryption.

Figure 12 demonstrates that when decrypting data in terms of CPU usage, a small amount of data results in the highest CPU usage for all encryption algorithms and when receiving unencrypted data. This increased CPU utilization

is a result of the deluge of data packets that the Server application receives and the ensuing demands to manage these packets. As the payload size of the data packet increases, the processing power requirements decrease exponentially (between 500 and 1000 bytes), then increase linearly (up to about 4100 bytes), and for larger data packets, the CPU usage curves continue to rise at a very slow rate.

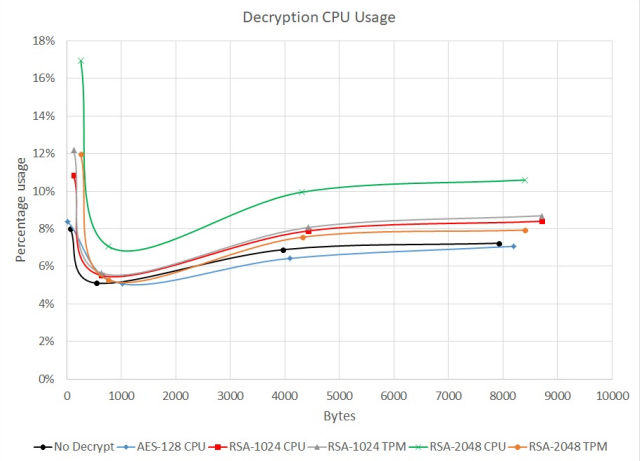

Figure 12. CPU usage in data decryption.

We observe, based on CPU utilization, that when downloading unencrypted data, we have slightly higher processing demands than when using AES-128, due to the simultaneous operations of receiving, displaying, and storing the data on the RPi. In contrast to AES-128 encryption, which has provided the best performance, the packet is decrypted, the results are displayed on the terminal, and then the data is saved, allowing the device to better distribute its functions and requirements.

Regarding the memory requirements for data decryption tasks in the Server application, the differences between algorithms are less than 2.5%, and they all adhere to a consistent pattern regardless of the payload size of the data.

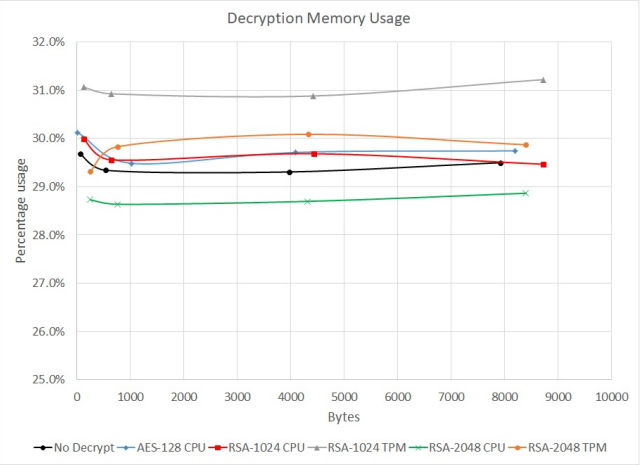

Figure 13. Memory usage in data decryption.

Figure 13 indicates that the optimal memory algorithm is RSA-2048, followed by RSA-1024 and AES-128 when using the CPU, and RSA-2048 and RSA-1024 when using the TPM.

It should be noted that based on CPU usage measurements, the difference between the encryption rate and the decryption rate approaches 90%, whereas in TPM, the variation between the rates is better, reaching 50%. However, the CPU encryption and decryption rates are multiple times faster than TPM encryption and decryption rates, which affects the total encryption and decryption times.

5 Conclusions

The paper presents a novel approach to enhancing sensor communication in WSNs using TPM. The proposed system offers several benefits, including secure communication, data integrity, and confidentiality. We conducted experiments to evaluate the effectiveness of the proposed system.

The results of our experiments indicate that the use of TPM in wireless sensor networks can significantly improve the security and reliability of data communication, which is crucial for applications such as healthcare, and environmental monitoring. The proposed system can be easily integrated into existing sensor networks and demonstrates adaptivity to different topologies and scalability.

In conclusion, the use of a TPM improves the security of an IoT system significantly, despite the fact that there are some time delays in comparison to the CPU-based setup. Considering energy efficiency, it is remarkable to mention the trade off between computational capabilities and energy consumption, in which TPM shows significant advantages. Essentially, our proposed communication scheme through TPM guarantees:

- The TPM secures and stores the keys so that they cannot be altered.
- Data can be encrypted in the secure environment of the TPM.
- No execution variations exist during encryption and decryption using TPM.
- The transmitted information is encrypted.
- The secure communication process is done using as few resources as possible.

In summary, the proposed system can effectively enhance sensor communication in WSNs using TPM. The system offers a promising solution for addressing security and communication issues in sensor networks and can be applied in various real-world scenarios. It is a state-of-the-art scheme utilizing TPM to establish a secure and authenticated communication channel between the nodes and the gateway. This protects against eavesdropping, man-in-the-middle attacks, and other network-based attacks.

While HW-based TPM is the most secure type of TPM, its performance must be improved to satisfy the increasing security need. Future work can investigate the scalability and performance of the proposed system in large-scale sensor networks.

6 Acknowledgments

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.

7 References

[1] IRIDIUM SLI 9670 TPM2.0. Available: https://www.infineon.com/cms/en/product/evaluation-boards/iridium-sli-9670-pm2.0/, 2020. [Online; accessed 23 4 2022].

[2] OPTIGATM TPM SLI9670. https://www.infineon.com/cms/en/product/security-smart-card-solutions/optiga-embedded-security-solutions/optiga-tpm/sli-9670/, 2020. [Online; accessed 23 4 2022].

[3] Trusted Computing. https://trustedcomputinggroup.org/trusted-computing/, 2020. [Online; accessed 23 4 2022].

[4] P. Aggarwal, C. Gonzalez, and V. Dutt. Hackit: A real-time simulation tool for studying real-world cyberattacks in the laboratory. In B. B. Gupta, G. M. Pérez, D. P. Agrawal, and D. Gupta, editors, *Handbook of Computer Networks and Cyber Security, Principles and Paradigms*, pages 949–959. Springer, 2020.

[5] D. G. K. Arthur, Will Challener. Using the new trusted platform module in the new age of security. In *A Practical Guide to TPM 2.0*, pages 285–288. Springer Nature, 2015.

[6] D. G. K. Arthur, Will Challener. Using the new trusted platform module in the new age of security. In *A Practical Guide to TPM 2.0*, pages 285–288. Springer Nature, 2015.

[7] M. Barbareschi, E. Battista, A. Mazzeo, and S. Venkatesan. Advancing wsn physical security adopting tpm-based architectures. *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration, IEEE IRI 2014*, pages 394–399, 02 2015.

[8] B. Bhushan and G. Sahoo. Recent advances in attacks, technical challenges, vulnerabilities and their countermeasures in wireless sensor networks. *Wirel. Pers. Commun.*, 98(2):2037–2077, jan 2018.

[9] J. Furtak and J. Chudzikiewicz. The concept of authentication in wsns using tpm. pages 183–190, 09 2014.

[10] J. Furtak, Z. Zieliński, and J. Chudzikiewicz. A framework for constructing a secure domain of sensor nodes. *Sensors*, 19(12), 2019.

[11] D.-H. Lee and I.-Y. Lee. A lightweight authentication and key agreement schemes for iot environments. *Sensors*, 20(18), 2020.

[12] E. T. Michailidis and D. Vouyioukas. A review on software-based and hardware-based authentication mechanisms for the internet of drones. *Drones*, 6(2), 2022.

[13] H. Modares, R. Salleh, and A. Moravejosharieh. Overview of security issues in wireless sensor networks. pages 308–311, 09 2011.

[14] M. Papaioannou, M. Karageorgou, G. Mantas, V. Sucasas, I. Essop, J. Rodriguez, and D. Lymberopoulos. A survey on security threats and countermeasures in internet of medical things (iomt). *Transactions on Emerging Telecommunications Technologies*, 33, 06 2022.

[15] R. Roman, J. Zhou, and J. Lopez. On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, 57:2266–2279, 07 2013.

[16] J. Sen. A survey on wireless sensor network security. 2010.

[17] G. Sharma, S. Vidalis, N. Anand, C. Menon, and S. Kumar. A survey on layer-wise security attacks in iot: Attacks, countermeasures, and open-issues. *Electronics*, 10(19), 2021.

[18] P. Singh, B. Bhargava, M. Paprzycki, N. Kaushal, and W.-C. Hong. *Handbook of Wireless Sensor Networks: Issues and Challenges in Current Scenario's*. 01 2020.

[19] A. Tidrea, A. Korodi, and I. Silea. Cryptographic considerations for automation and scada systems using trusted platform modules. *Sensors*, 19(19), 2019.

[20] R. Toegl, T. Winkler, M. Nauman, T. W. Hong, J. Winter, and M. Gissing. *Programming Interfaces for the TPM*, pages 3–32. Springer International Publishing, Cham, 2015.

[21] M. Zhang, B. Zhu, Y. Li, and Y. Wang. Tpm-based conditional privacy-preserving authentication protocol in vanets. *Symmetry*, 14(6), 2022.