Innovative methods of cyber protection of information system for a smart house *

Valerii Nedilenko[1,†], Elena Nyemkova[2*,†], Mikolaj Karpinski[3,4,†], Yuriy Lakh[2,*,†] and Pawel Sawicki [5,†]

[1] *Heat Ceram, Private enterprise "Rozumniy dim", Lviv, 79019, Ukraine*
[2] *Lviv Polytechnic National University, Lviv, 79005, Ukraine*
[3] *Department of Software Engineering, Institute of Security and Computer Science, University of the National Education Commission, Krakow, 30-08, Poland*
[4] *Department of Cybersecurity, Ternopil Ivan Puluj National Technical University, Ternopil, 46001, Ukraine*
[5] *SunsetPicnic UG Ltd., Berlin, 13597, Germany*

Abstract

A project was developed and implemented for an industrial Internet of Things device, interacting with AWS IoT Core, using IoT as an example for water leakage protection. The project is dedicated to enhancing the security of data storage and transferring by using modern encryption methods and protection against hacking. Aspects of cybersecurity of IoT devices were investigated and innovative solutions were implemented for effective protection against potential threats. During the work were developed: an implementation of a method for secure data storage in Flash ESP32; a method for client authorization for the open BLE protocol, a stream-safe in the context of FreeRTOS method for implementing the MQTT data exchange protocol allowing Over-the-Air updating. The results include the development of a complex, combining hardware and software solutions increasing the security and functionality of IoT devices in the AWS cloud service.

Keywords

data security, AWS IoT Core, encryption methods, cyber security of IoT-devices, hardware and software solutions, ESP32, FreeRTOS.

1. Introduction

Taking into account constant rapid development of modern technologies and the growing demand for innovative solutions, protection against unpredictable situations and threats is becoming a critically important task. Now one may observe a sharp increase in the popularity of smart devices and the Internet of Things (IoT), providing convenience and efficiency in managing various devices and systems. In this context, the research work is dedicated to the development of a water leakage protection device using Amazon Web Services (AWS) IoT Core infrastructure.

The main research goal is to develop an effective and reliable protection mechanism that ensuring the security of data storage and transferring from the device to the AWS IoT Core cloud service. Special attention is dedicated to the use of modern encryption and anti-hacking methods to ensure the confidentiality, integrity and availability of information. IoT devices can contain various security-related binary data enabling the use of the appropriate protocols, such as certificates for

* *SmartIndustry 2025: 2nd International Conference on Smart Automation & Robotics for Future Industry, April 03-05, 2025, Lviv, Ukraine*
† These authors contributed equally.
valeranedilenko@gmail.com (V. Nedilenko); olena.a.niemkova@lpnu.ua (E. Nyemkova); mikolaj.karpinski@uken.krakow.pl (M. Karpinski); yurii.v.lakh@lpnu.ua (Y. Lakh); pwlsawicki@gmail.com (P. Sawicki)
0009-0009-7941-9616 (V. Nedilenko); 0000-0003-0690-2657 (E. Nyemkova); 0000-0002-8846-332X (M. Karpinski); 0000-0003-4153-8125 (Y. Lakh); 0000-0002-8798-7279 (P. Sawicki)

secure communication with remote servers. (mqtt, http etc). Malefactor, having physical access to the device, can easily get binary images from device storage and search for such text fragments. A potential leak of security certificate could escalate attacks to other parts of the infrastructure, especially if the relevant permissions are incorrect or sufficiently weakened. To avoid this, one need to encrypt the device's memory so that no sensitive information from the firmware image or elsewhere can be read, even with physical access to the device.
The research includes not only the development of a hardware and software complex, but also the study and analysis of the relevant cybersecurity aspects, in particular in the field of protecting IoT devices from potential threats and attacks. Further improving and expanding of the security and interoperability capabilities of AWS IoT Core are the key aspects of this research.
The aim of the work is to study available solutions and methods for developing a protected information system for a smart house as well as to develop one of the elements of such a system using the example of a water leak monitoring device. To achieve this goal, the following tasks have been performed:
Analysis has been performed of modern tools using for the development of IoT devices, based on ESP32 (ESP-IDF) working in the AWS environment.
The project structure was presented, and available solutions were analyzed.
A device for monitoring water leakage has been implemented in the context of a secure information system for a smart house.
The work is aimed at solving current cybersecurity problems and developing innovative technologies having the potential to significant enhancing the quality and safety while using IoT devices in real-world operating conditions.

2. Analysis of literary sources

The Internet of Things is developing rapidly, however with it come increasing risks related to security, privacy, and trust. Because IoT relies heavily on the Internet infrastructure, it becomes vulnerable to various attacks. The increase in the number of connected devices only exacerbates these problems, making security a priority. To better understand the threats facing IoT, the OWASP community regularly compiles a list of the top ten vulnerabilities [1], based on analysis of real-world attacks and threat scenarios.
Recently, researchers have been actively working on new approaches ensuring privacy in IoT. Some methods focus on protecting the data itself, others on user anonymity. Thus, in [2], the application of post-quantum cryptography in IoT networks is considered. The researchers proposed a special certificate-free ring signature scheme that helps protect user data and guarantees anonymity without requiring centralized control. Additionally, this scheme has the advantages of irrefutability, traceability, and resistance to future quantum computer attacks. This is especially important considering that quantum technologies may fundamentally change approaches to IoT security in the future. Researchers [3] note that standard cryptographic methods do not provide complete data protection during wireless transmission in IoT. A promising approach is signal obfuscation, a controlled distortion of data that makes it difficult for unauthorized parties to intercept it.
The study [4] proposes a new approach to authentication in the Industrial Internet of Things (Industrial IoT) – Lightweight Industrial IoT Authentication. This method aims to solve security problems in critical infrastructures such as smart cities and industrial systems. It provides mutual authentication between devices, integrity verification of transmitted data, key negotiation, and resistance to various attacks, including man-in-the-middle attacks, spoofing, and eavesdropping. At the same time, the method places minimal load on devices, making it effective even in resource-intensive environments. This approach also takes into account the security of cyber-physical systems, which is critical for their stable operation. The method allows to verify the reliability of communication, detect attempts to unauthorized modification of messages, establish secure

session keys between users and IIoT devices, and regularly update keys to maintain a high level of security. Another interesting idea is presented in [5], which proposes a method for simulating identifiers for IoT devices. This approach allows one to determine which group an object belongs to and at the same time identify each device within that group. The main advantage of the method is that the identifiers are constantly updated, which makes them harder to track and increases the level of confidentiality. In addition, the method is quite easy to implement, allowing it to use even in the low-power IoT devices.

In the study [6], the authors analyze in detail the security threats of IoT from the perspective of a three-layer architecture: the perception layer, consisting of sensors and data collection devices; the network layer, which is responsible for data transmission via wired or wireless channels; and the application layer, where data processing, analysis, and visualization are performed. At the perception level, the main issues are the protection of sensor data and the physical security of devices, at the network level, attention is paid to encryption, authentication, and resistance to man-in-the-middle attacks, and at the application level, to protecting the confidentiality of user data and access control.

In [7], the authors analyze current approaches to intrusion detection, keys agreement, and anomaly detection systems, discuss their strengths and weaknesses, and identify gaps requiring further research. In addition, the authors propose innovative strategies to improve IoT security and present a roadmap for the future research in this area. A separate research area concerns authentication and keys agreement protocols in low-power wireless networks. In [8], secure authentication mechanisms for IPv6 in a 6LoWPAN environment are considered, which is a key for IoT deployment in low-power networks. However, the research focuses exclusively on 6LoWPAN and does not cover broader aspects of integrating these mechanisms into general IoT ecosystems.

Another important aspect of IoT security concerns privacy, interoperability, and deployment risks. In [9], the authors review the main IoT applications and the associated privacy and security challenges. The study highlights the challenges of integrating these aspects into real IoT systems, but does not provide a detailed architecture for their practical implementation. Significant attention is also paid to data security issues in IoT. The article [10] presents a critical review of the main challenges in this area, including ensuring the integrity and confidentiality of information. The authors analyze data protection strategies and propose methods for their improvement, but the study covers aspects of anomaly detection and secure communication protocols to a lesser extent.

Recent research highlights the importance of a comprehensive approach to securing IoT systems, particularly in industrial networks. One key aspect is examined in [11] – certificate-based authentication of industrial components. The research focuses on certificate management throughout its entire lifecycle, from creation to disposal. The authors analyze existing certificate management methods, compare proposed architectures, and consider the functionality of different authentication control models in industrial IoT environments. Particular attention is paid to the role of stakeholders in the process of managing trust relationships and ensuring the continuity of device security throughout their entire life cycle.

Therefore, current IoT security research focuses on various aspects, from authentication to threat analysis and data protection, but there remains a need to integrate these approaches into comprehensive solutions for real IoT systems.

3. Methods and means

Taking into account the scale of production and distribution of IoT systems in the modern world, as well as the high level of technical training of attackers, the choice of appropriate tools and technologies when developing new smart devices is critically important. This section examines key modern technologies that have been used to ensure an adequate level of protection for the IoT system from potential threats.

3.1. A toolkit of modern methods and technologies for constructing secure IoT systems

AWS IoT Core [12] (cloud gateway for IoT devices) is a message broker that working with protocols MQTT [13], HTTP and WebSockets. It supports long-term bidirectional communication, providing a continuous and reliable method of transmitting messages from endpoints to the broker. Another significant advantage of using cloud solutions is the ability to automatically scale to support more than a billion devices without the need to maintain infrastructure. The AWS IoT Core platform provides mutual authentication and encryption at all connection points. These methods are the following: AWS authentication (also known as SigV4), X.509 certificate-based authentication, and custom token-based authentication. AWS IoT Core supports certificates generated by the service itself, as well as certificates signed by a certificate authority of the client's choice. Policies for granting access to devices or applications can be attached to each certificate.

AWS IoT Core Shadow. With AWS IoT Core Shadow, one can create a persistent virtual version of each device that contains its latest state and allows applications or other devices to read messages from and interact with that device. Shadow devices preserve the last recorded state and desired future state of each device, even when it is offline.

AWS Cognito makes it easy to add user registration and authentication to mobile and web applications. Amazon Cognito enables user authentication through external providers and provides temporary data to access the application's server resources on the AWS platform or to any service through Amazon API Gateway. The technology features include the ability to integrate own implementation of the authentication service and data synchronization across multiple devices.

AWS IAM Identity Center allows to centrally manage access to multiple AWS accounts and business applications. It provides a single point of access to all assigned accounts and applications. User permissions can be set based on general job responsibilities and then further customized to meet specific security requirements. IAM Identity Center enables you to enforce multi-factor authentication (MFA) for all users, including requiring users to configure MFA devices when they sign in. With IAM Identity Center, you can use strong, standards-based authentication for all users.

Amazon S3 Bucket used to store and retrieve any amount of data from any point on the network. The storage service provides reliability, availability, performance and security, as well as virtually unlimited scalability at very low costs.

AWS Lambda allows to run code for virtually any type of application or service without provisioning or maintaining servers. It is an easy-to-use and fault-tolerant endpoint management tool.

With **AWS IoT Remote Jobs** service one can define tasks for a set of IoT devices. Tasks allow to download and install applications, run firmware updates, reboot, change certificates, or perform remote troubleshooting operations.

AWS Signer enables code signing to help ensure code integrity and trustworthiness. With IoT Code Signing, you can sign code for any AWS-supported IoT device.

Real time Operational System **FreeRTOS** [14] with open source microcontroller platform simplifies programming, deployment, security, connectivity, and management when working with small, low-power peripherals in the AWS Cloud or to more powerful peripherals running AWS IoT Greengrass

Over-the-Air allows to deploy firmware updates to one or more devices. OTA updates can be used to send any files to one or more devices registered with AWS IoT. Updates must be digitally signed to prevent man-in-the-middle attacks.

3.2. Using ESP-IDF platform as a mean for constructing secure IoT systems

The project uses the ESP32 [15] microcontroller, which combines built-in Wi-Fi and Bluetooth controllers, low power consumption, and affordable cost. The official ESP-IDF open platform designed for the ESP32, ESP32-S, ESP32-C, and ESP32-H SoC series was used to develop

applications. The connection to the router was made using the Bluetooth Low Energy [16] (BLE) protocol.
Secure storage of parameters in the device's flash memory is ensured by the encryption mechanism of the non-volatile storage (NVS) library. The project implements Encrypted NVS through pre-generation of keys with their subsequent overwriting during device initialization. Hardware support for encryption in the ESP32 microcontroller makes it impossible to recover most of the SPI flash content through physical reading.
ESP32 contains two main types of memory partitions: applications, which store software binaries, including device firmware, and data, which contains arbitrary user information. Partitions identified as data are not automatically processed by the bootloader and require a separate encryption mechanism to ensure their confidentiality and integrity.
Non-volatile storage (NVS) in ESP32 – that's a specialized memory section that uses a portion of the main flash memory via SPI. It is designed to store key-value pairs, security certificates, unique device identifiers, etc. NVS supports two encryption modes: run-time encryption, when the program generates the key and encrypts/decrypts data as it runs, and build-time encryption, where the NVS volume is pre-encrypted and written to the device. In both cases, a separate memory partition is used to store the encryption key.
The ESP-IDF platform supports this mechanism by providing a partition subtype (type data, subtype nvs_keys) and automatically handling its encryption through the loader.
The device control controller processes signals from water flow sensors, controls the actuator (electrically actuated ball valve), and provides light and sound notification of emergency situations. It can be connected to the user's Wi-Fi network and supports integration with a mobile application for remote monitoring and control.

4. Practical implementation

4.1. Practical implementation of a secure IoT system to prevent water leakage using AWS and ESP-IDF

This section presents the project structure and discusses cyber security aspects in detail. The project is organized into separate directories, each containing configuration files, program code, and necessary software components (Figure 1).

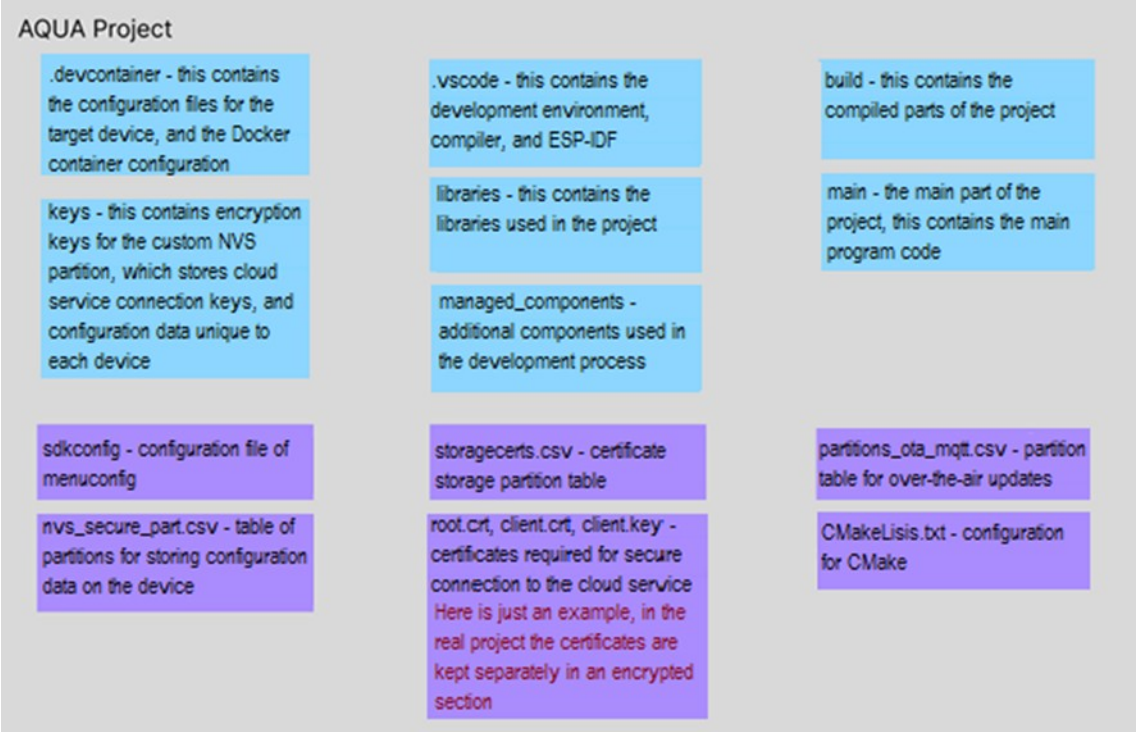

Figure 1: Project directories description.

The project is structured into five main modules (Figure 2), each of which is responsible for a specific aspect of its functionality:

1. **The Main Programme** – is responsible for controlling motorized faucets, processing signals from water flow sensors and meters, and initializing and integrating all the modules. This block implements the launch of FreeRTOS tasks, as well as all operations related to memory management.
2. **Interaction with BLE** – contains all the functions required for data exchange via the Bluetooth Low Energy (BLE) protocol. It describes the structure of the GATT-server, including services, descriptors, and characteristics. In addition, it implements the processing of input data from the mobile application and the generation of the appropriate responses.
3. **AWS IoT Core** – implements mechanisms for connecting and interacting with the AWS MQTTS-broker. This module manages certificates, keys, and authorization data, as well as processes JSON objects required for communication with the cloud infrastructure.
4. **OTA Updates** – responsible for integration with the "Over-the-Air" (OTA) update server. The functionality includes checking for new versions, downloading the binary code, temporarily storing it in the OTA1 or OTA2 memory sections, checking data integrity, and modifying the bootloader to correctly launch the updated software.
5. **Interaction with the radio transmitter** – contains functions for controlling the NRF 868MHz module via the SPI interface. It implements the initialization of the radio module, writing and reading data to FIFO buffers, controlling transmission modes, and its own data exchange protocol, which provides verification of the integrity of received packets, determining the signal level (RSSI), and adaptive selection of the frequency channel in case of overloading. It also provides mechanisms for handling errors and critical failures.

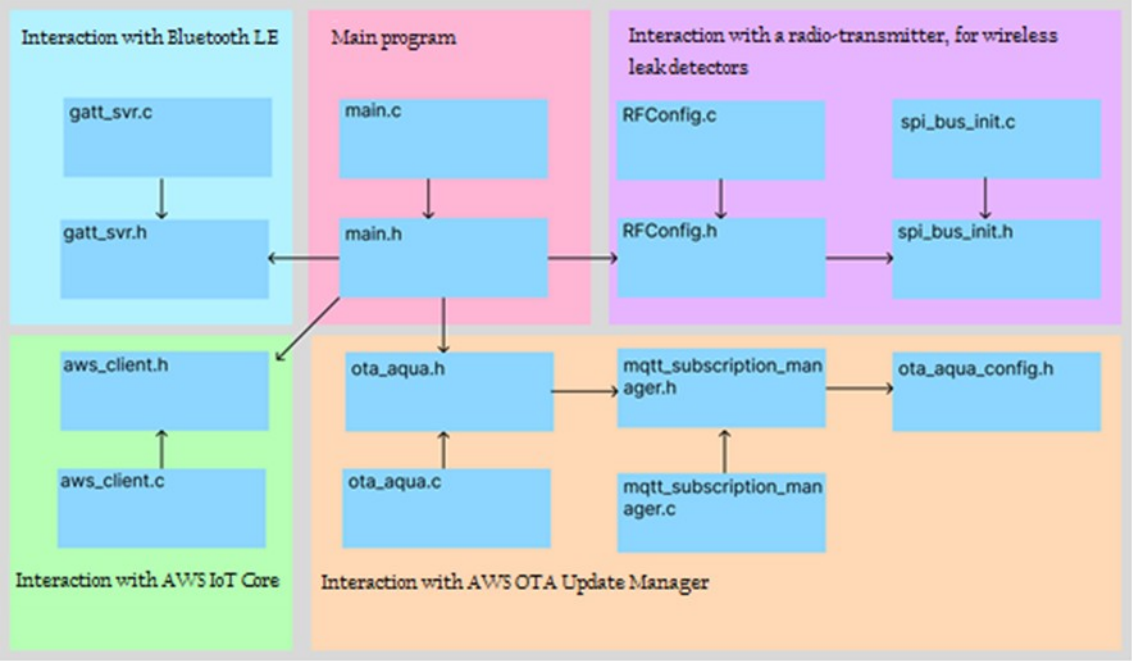

Figure 2: Content of main directory.

4.2. Establishing a secure connection with the AWS IoT Core cloud broker

The client verification procedure has been performed within a specialized function interacting with the corresponding device characteristic. For successful authentication, the client must record a unique key, generated during the initial initialization of the device and stored in a cloud database, which ensures its availability to the mobile application. Such an approach prevents unauthorized access to devices by attackers and makes impossible to intercept control.

Before starting to work with the other features of the device, the client must go through the authentication process, otherwise its connection will be forcibly disconnected. To protect against "brute-force" attacks, a blocking mechanism is provided: after entering an incorrect key, the client is deprived of the ability to reconnect for 5 minutes. Such a mechanism provides an additional security level, as far as the key generation and initialization process is fully automated and does not involve any human intervention.

Upon successful authentication, the device grants the client access to its features, including the ability to read, write, and subscribe to updates. The user also has the ability to view the device's current status, a list of available networks with their parameters, and enter connection details.

To establish a secure connection with the AWS IoT Core cloud broker, manage certificates, and exchange data, the appropriate modules should be connected, which is done in the file aws_client.h.

Since an accurate timestamp is required for a successful TLS connection, it is necessary to initialize the Simple Network Time Protocol (SNTP) at this stage. The SNTP–stack initialization function configures four SNTP–server addresses, which provides automatic switching in the event of a failure of one of them, guaranteeing constant access to the current time via the Internet (Figure 3).

```
°·°·sntp_setservername(0,·"time.google.com");¶
°·°·sntp_setservername(1,·"pool.ntp.org");¶
°·°·sntp_setservername(2,·"time.nist.gov");¶
°·°·sntp_setservername(3,·"time.cloudflare.com");¶
```

Figure 3: Settings for connecting to accurate time servers using Simple Network Time Protocol.

The main task of **FreeRTOS**, which is responsible for connecting and transferring data to AWS IoT Core, is the function aws_iot_main. It initializes the MQTT client, establishes a secure connection to the server, and ensures guaranteed message delivery, as the project uses MQTT with QoS1 level. The function then subscribes to cloud broker topics, resubscribes in case of connection loss, and publishes to the device status indication topic. Upon reconnection, it clears the MQTT, TLS, CertMgr cache, and repeats the entire process, starting with MQTT client initialization.
When connecting directly to the broker, the certificate chain is verified, followed by the server fingerprint. If successful, the process of decrypting the device certificate and private key, which were saved in the previous steps to the encrypted volume on the device, begins. Then, the connection parameters (unique device identifiers stored in the NVS secure partition) are initialized, and an attempt is made to establish a secure connection to the server using this data.
To set up a new device, a series of sequential actions were performed, which consisted of connecting it to the AWS cloud infrastructure. AWS IoT Core contains a device gateway and a message broker providing data processing and transmission between the IoT device and the cloud. Device access to AWS IoT Core is managed through the AWS IAM Center. Firstly, a security policy was created that defined the IoT device's access rights to the broker. During device registration, its unique name and configuration parameters were specified. The final step was to generate the certificates and keys required to establish a secure connection to the cloud broker. This project uses automatic generation of certificates, which can be viewed and downloaded only once when creating a device. After completing the settings, the device is considered successfully registered. The generated certificates can be saved in the microcontroller's memory, after which the prepared software can be launched.

4.3. Performing Over-the-Air device firmware updates for IoT devices

The process of performing OTA (Over-the-Air) device firmware updates using AWS IoT Core Remote Jobs is provided by the mqtt_subscription_manager, which is responsible for processing incoming and outgoing MQTT packets containing update information and parts of the new firmware binary. The main functionality that implements OTA updates is located in the file **ota_aqua.c**. The upgrade process involved several key steps. First, semaphores were initialized for OTA buffers and for the transaction confirmation system that provides feedback. Then, a mutex was initialized to manage MQTT transactions. This was followed by a TLS connection to the server and an MQTT connection to the broker. The next step was to initialize the OTA client, after which a task was created to perform the update operations. Now, when the device expects a signal from the server that a new firmware version is available and the update becomes available, the device starts receiving data packets until the last confirmation packet is transmitted. After the download is complete, the received binary file is checked for integrity. If the check is successful, the new firmware version is activated and the device reboots with the updated software.
By default, the OTA protocol configuration in the **ota_config.h** file assumes the use of the MQTT protocol. To enable OTA device updates via MQTT, each device was registered as an object in AWS IoT and had an appropriate security policy. In particular, the iot:Connect permission was granted, enabling the device to connect to AWS IoT via MQTT. Additionally, the iot:Subscribe and iot:Publish permissions on the AWS IoT job topic (.../jobs/) gave the device the ability to receive notifications about new jobs, as well as publish information about their execution. For OTA updates to work correctly, it was needed to set the iot:Subscribe and iot:Publish permissions on the AWS IoT OTA streams topics (.../streams/), which allowed the device to receive firmware updates via MQTT.
As a part of the project, security policies were configured for the user responsible for performing OTA firmware updates for IoT devices. This ensured secure and controlled device updates via AWS IoT. First of all, access was provided to the S3 segment where the firmware update files were stored. Access certificates stored in AWS Certificate Manager were used to provide authentication

and secure connectivity. Permissions were also configured to use the AWS IoT MQTT-based file delivery feature and the FreeRTOS OTA update mechanism.
Additionally, the user received rights to manage AWS IoT tasks, access to the Identity and Access Management (IAM) system, the ability to sign code for AWS IoT, and administer the list of FreeRTOS hardware platforms. The last stage of configuration was adding permissions to tag and untag (status) AWS IoT resources, which allows you to effectively manage the device update process and track their current status in the system. The implemented settings provide a secure firmware update process with the ability to centrally manage access rights.
Within the framework of this project, a mechanism was implemented to ensure the integrity of new firmware versions by using a digital signature. For this purpose, a code signing certificate and the corresponding private key were used. During the testing phase, a self-signed certificate and private key were used, however, for industrial applications, the possibility of integrating certificates obtained through a trusted certification authority (CA) is provided. Since the project used Espressif ESP32 microcontrollers, it was decided to use the supported SHA-256 digital signature algorithm in combination with an ECDSA certificate. To implement this process, OpenSSL was installed in the working environment. Next, a configuration file cert_config.txt was created, containing the necessary parameters for generating keys and certificates. The next step was to generate a private key for signing the code using ECDSA, after which a digital certificate was generated for signing the firmware. Finally, all the resulting data – the code signing certificate, private key, and certificate chain – were imported into AWS Certificate Manager. This enabled the secure use of digital signatures in the firmware update process via AWS IoT, ensuring its authenticity and integrity.

5. Discussion of study results

The presented project is large and all its aspects cannot be covered in the same detail in the article. Nevertheless, we will present the data path from the water leak sensor to the user (smart house owner's mobile phone, industrial operation), (Figure 4).

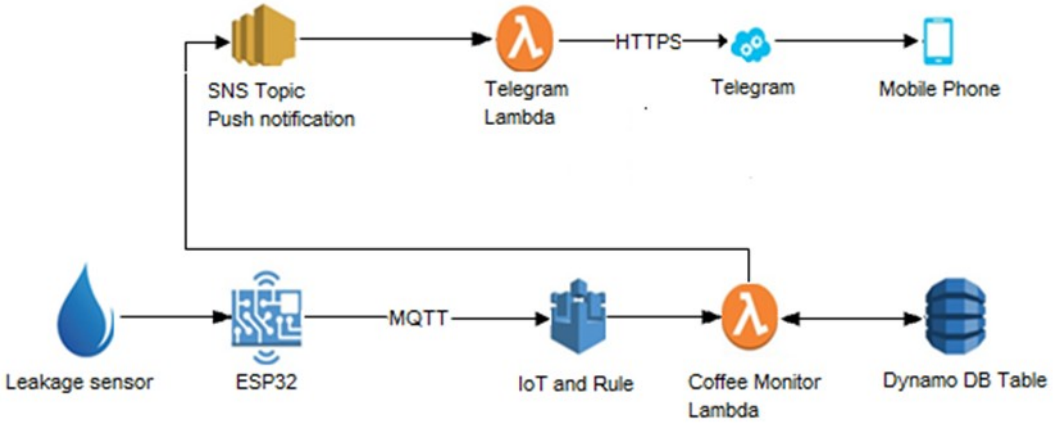

Figure 4: Scheme of the developed system.

All the elements of the scheme in their physical implementation were designed and manufactured, the leakage protection system passed bench tests, was presented at exhibitions and is on sale at retail outlets.

We tried to explain in the article the issues, related specifically to the developed elements of the system's cyber defense. Figure 5 presents the methods of protection against cyber attacks implemented in the project and which are responses to the challenges of the TOP-10 OWASP IoT Vulnerabilities (update 2024).

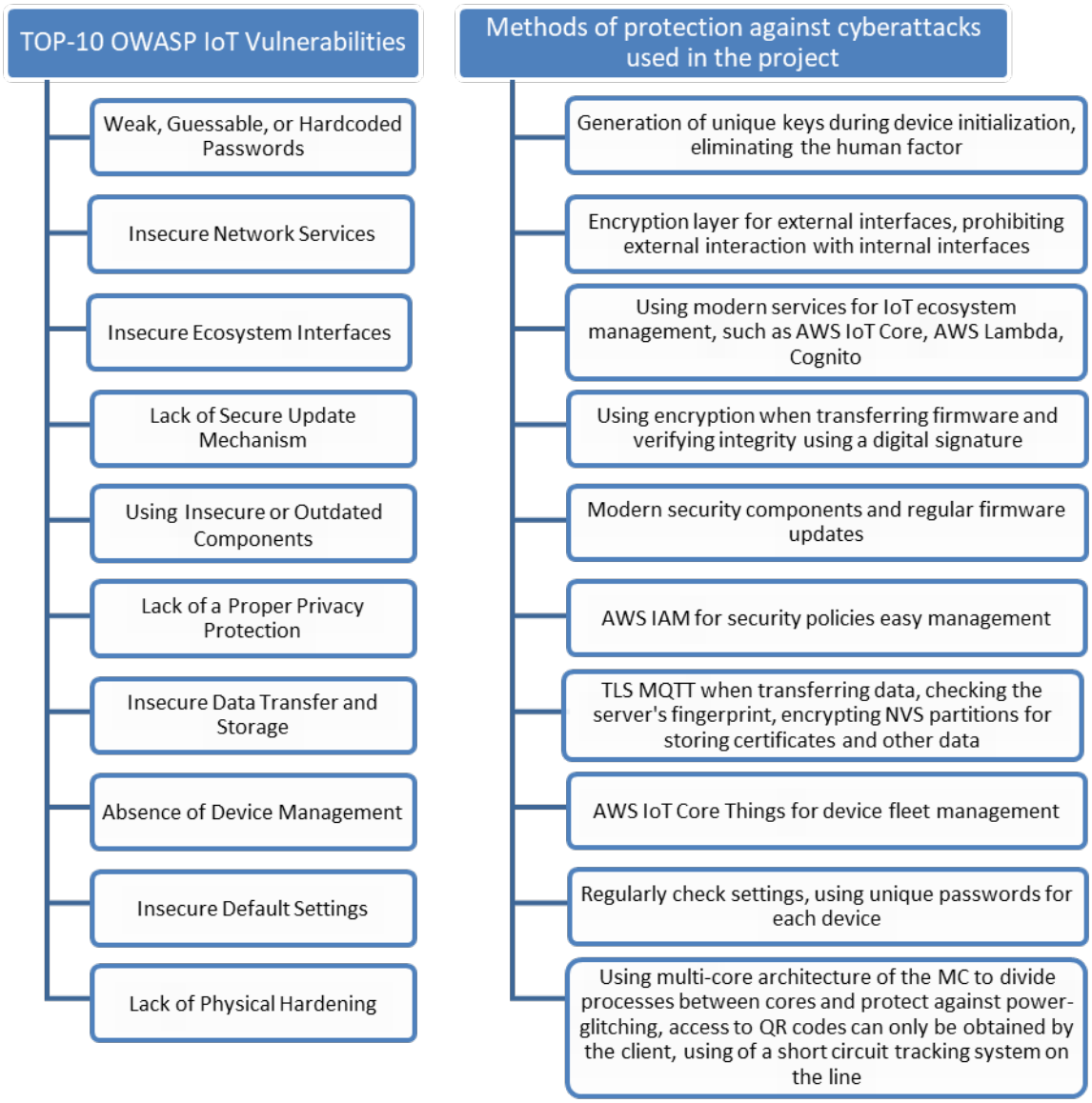

Figure 5: Cybersecurity threats and corresponding implemented methods for their elimination.

In the future research, the developed system can be adapted for use in the other areas, in particular, in the medical industry or for the protection of critical infrastructure facilities.

6. Summary and Conclusion

Due to the project, important results were achieved in the field of IoT device protection. The developed system successfully integrates with the AWS cloud service, ensuring a high level of security in data storage and transferring. The system architecture was described and also interaction between AWS and ESP-IDF, as well as encryption and anti-hacking methods to ensure data storage and transferring security. The technical aspects of the design, its innovative characteristics and relationships with selected AWS and ESP-IDF technological solutions have been analyzed in details.

The project takes into account high cybersecurity standards, using modern encryption and anti-hacking methods. The practical implementation of the system confirms its effectiveness and reliability in real operating conditions, which is confirmed by the successful implementation at two private enterprises in 2023.

This experience is important for further research in the field of IoT device protection and the development of secure systems using cloud technologies. All the obtained results emphasize the importance of improving the security and functionality of IoT systems to bring them to a new level of reliability and protection from potential threats.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] OWASP IoT Top 10 Vulnerabilities (2024 Updated). URL: https://www.wattlecorp.com/owasp-iot-top-10/.

[2] Y. Zhang, P. Duan, C. Li, H. Zhang, H. Ahmad. Preserving Privacy of Internet of Things Network with Certificateless Ring Signature Sensors (2025). doi:10.3390/s25051321.

[3] R. Cigno, F. Gringoli, S. Bartoletti, M. Cominelli, L. Ghiro, S. Zanini. Communication and Sensing: Wireless PHY-Layer Threats to Security and Privacy for IoT Systems and Possible Countermeasures. Information (2025). doi:10.3390/info16010031.

[4] A. Al Ghazo, M. Abu Mallouh, S. Alajlouni, I. Almalkawi. Securing Cyber Physical Systems: Lightweight Industrial Internet of Things Authentication (LI2A) for Critical Infrastructure and Manufacturing Appl. Syst. Innov. (2025). doi:10.3390/asi8010011.

[5] V. Maksymovych, E. Nyemkova, C. Justice, M. Shabatura, O. Harasymchuk, Y. Lakh, M. Rusynko. Simulation of Authentication in Information-Processing Electronic Devices Based on Poisson Pulse Sequence Generators. Electronics (2022). doi:10.3390/electronics11132039.

[6] M. Adam, M. Hammoudeh, R. Alrawashdeh, B. Alsulaimy. A Survey on Security, Privacy, Trust, and Architectural Challenges in IoT Systems. IEEE Access (2024) 57128–57149. doi:10.1109/ACCESS.2024.3382709.

[7] S. Szymoniak, J. Piątkowski, M. Kurkowski. Defense and Security Mechanisms in the Internet of Things: A Review. Appl. Sci. (2025). doi:10.3390/app15020499.

[8] F. Ashrif, E. Sundararajan, R. Ahmad, M. Hasan, E. Yadegaridehkordi. Survey on the authentication and key agreement of 6LoWPAN: Open issues and future direction. J. Netw. Comput. Appl. (2024). doi:10.1016/j.jnca.2023.103759.

[9] T. Magara, Y. Zhou. Internet of Things (IoT) of Smart Homes: Privacy and Security. J. Electr. Comput. Eng. (2024). doi:10.1155/2024/7716956.

[10] S.Z. Al-Otaibi. Data Security Challenges with its Defence Strategies of Internet of Things: Critical Review Study. Commun. Math. Appl. (2022), doi:10.26713/cma.v13i1.1980.

[11] J. Göppert, A. Walz, A. Sikora. A Survey on Life-Cycle-Oriented Certificate Management in Industrial Networking Environments. J. Sens. Actuator Netw. (2024). doi:10.3390/jsan13020026.

[12] Welcome to AWS Documentation. URL: https://docs.aws.amazon.com/.

[13] MQTT Documentation. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[14] FreeRTOS™ Real-time operating system for microcontrollers and small microprocessors. URL: https://www.freertos.org/.

[15] Espressif Documentation. URL: https://www.espressif.com/en/support/documents/technical-documents.

[16] Bluetooth. Specifications and Documents. URL: https://www.bluetooth.com/specifications/specs/.