Digital Representations in IoT: Cryptographic Tools for Improved Security

Saltanat Adilzhanova [1], Aigerim Rakhysh [1,*], Murat Kunelbayev [2], Gulshat Amirkhanova [2], and Dana Sybanova [1]

[1] Department of Cybersecurity and Cryptology, Faculty of Information Technology, Al Farabi Kazakh National University, Almaty, Kazakhstan
[2] Department of Artificial intelligence and Big Data, Faculty of Information Technology, Al Farabi Kazakh National University, Almaty, Kazakhstan
Email: asaltanat81@gmail.com (S.A.); rakhysh_aigerim3@live.kaznu.kz (A.R.); murat7508@yandex.kz (M.K.); gulshat.aa@gmail.com (G.A.); sybanova_dana2@live.kaznu.kz (D.S.)
*Corresponding author

***Abstract*—This study addresses the pressing challenge of developing efficient and secure cryptographic solutions for Internet of Things (IoT) systems, where devices operate under constraints of limited computing power, memory, and energy consumption. Against the backdrop of the rapid proliferation of IoT devices based on Embedded Systems (ESP32) microcontrollers and the growing cybersecurity threats, a systematic analysis of hardware and software encryption methods is conducted. The paper investigates the implementation of cryptographic protection at the microcontroller level in IoT networks, with particular emphasis on execution time, energy efficiency, and hardware resource utilization. The scientific novelty of this work lies in the integration of machine learning techniques for the automatic selection of the most efficient cryptographic algorithm Advanced Encryprion Standard (AES), Secure Hash Algorithm (SHA) or Hash-based Message Authentication Code (HMAC), based on data volume, execution time, and power consumption. Unlike prior studies that focus on isolated implementations, this research proposes a universal adaptive encryption selection system tailored for resource-constrained microcontroller environments. A key contribution is the implementation and testing of this system on the ESP32 and ESP32-S3 platforms. The results demonstrate that hardware-based encryption significantly outperforms software-based methods in terms of execution speed and energy efficiency. The developed logistic regression model achieved 100% classification accuracy on the ESP32 platform, outperforming alternative algorithms and confirming its applicability for energy-efficient real-time IoT applications. Finally, the paper identifies promising avenues for future work, including the development of secure digital twins for healthcare systems and autonomous devices.**

***Keywords*—hardware encryptor, cryptographic libraries, Embedded Systems (ESP32), cryptosecurity, Internet of Things (IoT) devices, machine learning**

I. Introduction

The Internet of Things has become one of the key technological paradigms in recent years, providing ubiquitous connectivity between sensors, embedded devices, and cloud services. The scale of the ecosystem is growing rapidly: according to forecasts, by 2030 the number of connected devices will reach about 40 billion [1], forming new automation scenarios in industry, energy, healthcare and the "smart" urban environment. At the same time, the task of protecting data and trusted computing at the very "edge" of the network becomes critically important—in conditions of limited computing power, memory and energy consumption of nodes.

Simultaneously with the expansion of the Internet of Things (IoT), cybersecurity issues are becoming more acute. Heterogeneity of the hardware base, simplified protocol stacks, incomplete authentication and access control make devices a target for a wide range of attacks: from unauthorized access and spoofing of telemetry to denial of service and attacks through third-party channels. According to research, about 70% of IoT devices exhibit vulnerabilities [2], which is confirmed by the practice of identifying threats in real-world operating conditions [3]. Among the main challenges are the protection of data confidentiality, ensuring their integrity, and the development of cryptographically stable but lightweight solutions compatible with the resource constraints of microcontrollers [4–6].

Research gap and work contribution. While many papers analyze individual algorithms or hardware acceleration methods, there are no integrated studies that simultaneously:

- Software and hardware-accelerated implementations on mass-produced ESP32/ESP32-S3 class microcontrollers are compared.
- They take into account execution time and power consumption on real amounts of data.

Manuscript received July 17, 2025; revised August 29, 2025; accepted October 10, 2025; published February 23, 2026.

- They provide automatic selection of a cryptographic primitive for the conditions of the task.

In this paper, we close this gap: we perform a systematic experimental evaluation of Advanced Encryprion Standard (AES), Secure Hash Algorithm (SHA), and Hash-based Message Authentication Code (HMAC) (software and hardware implementations), build and implement a machine learning model for dynamically selecting the optimal algorithm for load parameters, and discuss measures to increase resistance to side-channel attacks in the context of resource-limited Embedded Systems (ESP32) nodes. It is shown that hardware acceleration in combination with ML-selection provides time and power consumption gains relative to purely software implementations and static approaches.

II. Literature Review

Recent work has also addressed the importance of using hardware acceleration to improve cryptographic efficiency in microcontrollers. Ullah *et al.* [7] enhanced the wolfSSL library to use hardware-accelerated cryptographic functions on Espressif Reduced Instruction Set Computer (RISC-V)-based ESP32 platforms, demonstrating significant performance gains. Djenna *et al.* [8] conducted a detailed comparison of cryptographic algorithms on the ESP32 architecture, showing how hardware acceleration can significantly reduce execution time and energy consumption. Wu *et al.* [9] investigated the integration of hardware-based and software-based cryptographic libraries to optimize the security of IoT systems.

Machine learning (ML)-based approaches for dynamic cryptographic optimization are emerging as a promising direction. Nishinaga and Mambo [10] evaluated several lightweight cryptographic algorithms using ML models in the context of IoT-based healthcare systems, thereby demonstrating improved energy efficiency. Thakor *et al.* [11] proposed a privacy-preserving ML framework (Priv-IoT) that enables secure data processing in IoT networks while preserving user privacy.

The application of post-quantum cryptography to microcontroller platforms has also attracted considerable attention.

Amrita *et al.* [12] developed an efficient implementation of the CRYSTALS-KYBER key encapsulation mechanism for the ESP32, demonstrating its suitability for IoT applications requiring post-quantum security.

Chinbat *et al.* [13] introduced a lightweight, Physical Unclonable Function based (PUF-based) authentication-and-key-exchange protocol for low-power IoT devices.

However, the work presented by the authors provides an integrated approach that combines a systematic comparison of hardware-accelerated and software-based cryptographic methods on ESP32/ESP32-S3 platforms with dynamic algorithm selection based on machine learning. This comprehensive framework enables optimized, real-time cryptographic performance tailored to the unique constraints of resource-limited IoT environments.

III. Materials and Methods

The research is based on a systematically organized experimental methodological framework aimed at a detailed assessment and comparative analysis of the performance, energy efficiency, and resource optimality of hardware-accelerated cryptographic mechanisms and software-based solutions on modern microcontroller platforms used in Internet of Things (IoT) systems. The choice of ESP32 and ESP32-S3 microcontrollers is motivated by their wide applicability in IoT devices, support for embedded cryptographic accelerators, and compliance with low-power consumption requirements and operation under constrained computational resources. Although the ESP32 lacks support for 5 GHz Wi-Fi and may not be suitable for applications requiring ultra-high throughput or low latency, this limitation is not critical for the majority of IoT use cases. Most real-world deployments (e.g., smart meters, wearables, home automation systems, industrial sensors) rely on the 2.4 GHz band due to its broader coverage and lower power demand. Moreover, the ESP32 integrates hardware accelerators for AES, SHA, and Rivest, Shamir, Adleman (RSA), offers dual-core processing at up to 240 MHz, and remains one of the most cost-effective and widely adopted IoT platforms. These characteristics make it a representative and practical choice for benchmarking lightweight cryptographic solutions in resource-constrained environments.

The research follows a reproducible experimental protocol, which documents in detail the hardware configurations, software environments, and measurement procedures used, thereby ensuring the transparency of the experiment and enabling independent verification. Software development was conducted using the Arduino Integrated Development Environment (IDE) and the Espressif IoT Development Framework (ESP-IDF), integrated with Visual Studio Code extensions. The implementation of cryptographic algorithms employed well-established libraries—Embedded Transport Layer Security (mbed TLS), TinyAES, and AESLib—representing contemporary approaches to software cryptography in systems with limited resources.

Three widely used cryptographic algorithms were selected for analysis: the Advanced Encryption Standard (AES), the Secure Hash Algorithm (SHA), and the Hash-Based Message Authentication Code (HMAC). Their performance was evaluated across varying input sizes (ranging from 64 to 4096 bytes), enabling an assessment of algorithmic scalability and variations in computational load.

Built-in high-resolution timers within the ESP-IDF framework were used to measure the execution time of cryptographic operations. Real-time current consumption was measured using the ACS712 current sensor (±20 A range) connected to an Arduino Uno board. The operating voltage was maintained at a constant 3.3 V to eliminate the influence of power instability on measurement accuracy. Energy consumption was calculated as the product of the measured current, voltage, and execution time, providing

precise estimates of the energy efficiency of cryptographic operations.

To enhance result reliability, each experiment was repeated at least ten times, and the median values of execution time and energy consumption were calculated, thereby minimizing the influence of random noise and outliers. Additional measures were taken to reduce electromagnetic interference during the measurement process.

In addition to the primary experiments, a Machine Learning (ML) component was implemented to model the relationship between cryptographic algorithm performance characteristics and the selection of the optimal algorithm under specific conditions. Training datasets were generated for each platform, incorporating input parameters, execution time, and power consumption. Based on these datasets, three classification models—logistic regression, random forest, and K-Nearest Neighbors (KNN)—were trained using the scikit-learn library in Python. Stratified cross-validation was used for model evaluation, and classification performance was assessed using standard metrics: precision, recall, F1-score, and overall accuracy.

The developed methodological framework offers a reproducible and scalable approach for studying the performance of cryptographic algorithms in resource-limited IoT environments and demonstrates the potential of machine learning techniques for dynamic optimization of cryptographic solutions on embedded devices.

This section presents a set of measurement algorithms developed to evaluate the performance and energy efficiency of cryptographic operations on embedded platforms, particularly ESP32-based microcontrollers. Three key procedures are illustrated using flowcharts.

- AES-256 Decryption Time Measurement—measures the execution time required to decrypt a data stream using the AES algorithm.
- Current Measurement and Energy Estimation—calculates real-time current and total energy consumption during algorithm execution using a current sensor.
- Combined Analysis for Efficiency Evaluation—integrates time and power metrics to assess an algorithm's suitability for low-power applications.

Together, these procedures constitute a practical framework for benchmarking the computational and energy characteristics of encryption systems in IoT environments.

Fig. 1 shows the process of measuring data decryption time using the AES-256 algorithm on an ESP32 microcontroller. The algorithm starts with initializing the AES context and loading a 256-bit key. The input data stream is then divided into blocks of 16 bytes, which are sequentially decrypted and placed in the output buffer. Upon completion of the decryption process, the total execution time is recorded using the built-in system timer. The results obtained make it possible to evaluate the effectiveness of the algorithm in real-world applications.

The measurement process begins with connecting a current sensor and performing multiple analog voltage readings. The average value is then calculated and converted into current strength using the known sensitivity of the sensor. Finally, energy consumption is estimated as the product of current, voltage, and execution time. The unified flowchart of this process is shown in Fig. 2.

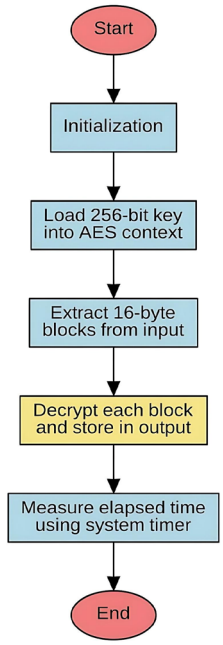

Fig. 1. The process of measuring data decryption time using the AES-256 algorithm on an ESP32 microcontroller.

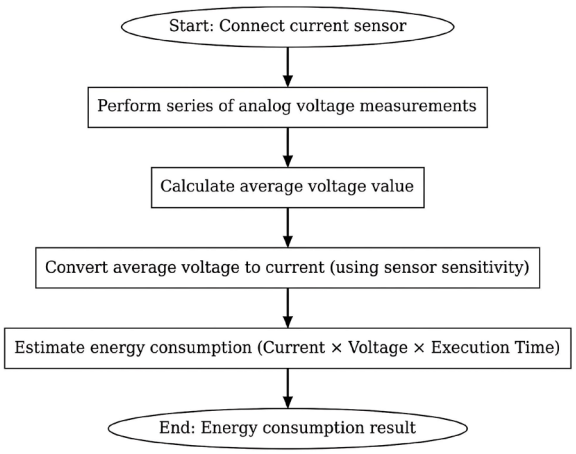

Fig. 2. Unified flowchart for measuring current and estimating energy consumption during cryptographic operations.

A. Microcontrollers in IoT

The development of artificial intelligence has heightened the demand for smart devices that collect and process data in Internet of Things (IoT) systems. In such systems, microcontrollers play a fundamental role, operating alongside sensors, actuators, and users to enable more efficient service delivery. When designing a smart microcontroller, 3 core principles must be considered. First, it must be flexible, meaning it should support adaptation in both software and hardware for diverse application domains. Second, the microcontroller must be intelligent—it should be capable of analyzing data, making

decisions, and executing logical control functions. Third, scalability is essential: multiple microcontrollers should be able to cooperate in performing complex tasks [14]. In IoT systems, microcontrollers are responsible for key functions such as sensor control, device monitoring, communication, and data processing. Given these critical roles, ensuring the security of IoT devices is paramount.

The primary rationale for encryption in IoT environments is to protect the confidentiality, integrity, and availability of data. As microcontrollers often process and store significant amounts of sensitive information, they may become targets for cyberattacks if not adequately protected. The application of modern cryptographic techniques—both hardware-based and software-based—ensures reliable data protection across all levels of the IoT architecture [15].

Although hardware-based encryption provides superior performance and enhanced security, not all microcontrollers are equipped with integrated cryptographic accelerators. In such cases, software-based cryptographic libraries that implement required cryptographic algorithms and functions serve as viable alternatives for securing data.

Software-based cryptographic libraries play a crucial role in protecting data within microcontroller-based systems, particularly when hardware encryption resources are unavailable or insufficient. These libraries implement a range of cryptographic algorithms and functions, offering developers flexible and effective solutions.

The implementation of encryption begins with the installation of appropriate cryptographic libraries on the microcontroller. During this stage, the cryptographic algorithm is selected, encryption keys are generated, and operational modes (encryption or decryption) are configured. Subsequently, data preparation is carried out, for block cipher algorithms, the data is divided into fixed-size blocks, and additional parameters, such as an Initialization Vector (IV), are introduced to enhance encryption security.

The following section analyzes cryptographic libraries suitable for microcontrollers:

- Networking and Cryptography Library (NaCl) enables high-speed operation while minimizing code size on Architecture Reference Manual (ARM) Cortex-M0 microcontrollers, making it particularly well-suited for small, resource-constrained IoT devices.
- Ultra-Lightweight Cryptographic Library (ULCL) is tailored for embedded systems, allowing optimization of performance by selecting only the required encryption algorithms. This facilitates robust security without incurring unnecessary resource overhead [16].
- Fair Evaluation of Lightweight Cryptographic Systems (FELICS) is a dedicated platform for evaluating the efficiency of cryptographic solutions. It assesses algorithm performance on microcontrollers in terms of memory usage, power consumption, and computational demands [17].
- mbedTLS, TomCrypt, and wolfSSL have been evaluated on ARM Cortex-M platforms. Each library demonstrates distinct characteristics regarding memory usage, processing speed, and energy efficiency, enabling developers to strike an appropriate balance between security and performance.
- Elliptic Curve Cryptography (ECC) is used in devices that require a high level of security. ECC algorithms, specially optimized for MSP430 and 8-bit Alf-Egil Bogen + Vegard Wollen + RISC (AVR) microcontrollers, not only work efficiently, but also help increase the level of protection against indirect attacks [18].

Table I shows various cryptographic libraries, but each has its own advantages and limitations.

TABLE I. CRYPTOGRAPHIC LIBRARIES COMPARISON

Library	Supported Algorithms	Advantages	Compatibility
Crypto	AES, RSA, SHA, others	Wide range of functions, flexibility	All well-known microcontrollers
AESLib	AES	Low resource consumption, simplicity	AVR, ESP8266
TinyAES	AES	Low memory usage, high speed	AVR, ARM Cortex-M, ESP8266, ESP32
uECC	ECC	Low memory usage, high security	STM32, AVR, ESP32
Mbedtls	AES, RSA, ECC, Secure Sockets Layer (SSL)/TLS	Easy integration, high performance	STM32, ESP32, ARM Cortex-M
mbed TLS	AES, RSA, ECC, others	High performance, flexibility	STM32, ESP32, NXP Low Pin Coun (LPC)

If flexibility and a wide selection of algorithms are important, then the Crypto, mbedtls, and mbed TLS libraries are excellent options. They support AES, RSA, ECC, and other algorithms and provide high performance. If the main goal is to save resources, then AESLib and TinyAES are suitable, as they are based on AES encryption and consume less memory, work quickly.

If elliptic curve cryptography is required, uECC is the best option, as it uses less memory and provides high security. While ECC and ChaCha20 are indeed recognized as lightweight and robust cryptographic technologies for IoT, their exclusion in this study was deliberate and based on 3 main considerations:

1. Native Hardware Support on ESP32. The ESP32 platform provides built-in hardware acceleration for AES and SHA, but not for ECC or ChaCha20. Including non-accelerated primitives would have created an unfair comparison, since the research goal was to measure the realistic performance trade-offs of hardware vs. software cryptography on widely deployed microcontrollers.
2. Consistency with Research Objectives. The central aim of this paper is not to exhaustively benchmark every lightweight algorithm, but to propose and validate an adaptive framework that dynamically selects between supported cryptographic primitives (AES, SHA,

HMAC) on ESP32/ESP32-S3. Adding ChaCha20 or ECC without hardware support would dilute the focus and introduce uncontrolled variability in execution-time vs. energy measurements.

3. Future Extension Path. We acknowledge that ECC is preferable for public-key operations and ChaCha20 is often faster than AES on pure software platforms. However, these algorithms were beyond the present scope, which prioritized reproducibility and hardware-accelerated benchmarks. Future research will extend the proposed machine learning–based selection system to include ChaCha20 and ECC (via uECC or similar libraries), enabling comparative evaluation against asymmetric and stream-cipher alternatives once consistent hardware/software support is available.

In addition, compatibility with microcontrollers must be taken into account. For example, the Crypto library is compatible with all major microcontrollers, while mbedTLS is often optimized for the ESP32 platform.

To evaluate real-time encryption speed, the number of encryption operations performed per second was calculated. The number of encryption operations completed within a fixed time interval was used as a performance indicator. This approach allows for an assessment of encryption speed and enables comparison of the efficiency of different encryption methods. To implement software encryption on the ESP32-WROOM-32 microcontroller, the mbedTLS cryptographic library and the Espressif Visual Studio Code development environment were used. Programs were developed to encrypt 16-byte data blocks using the AES algorithm. For this purpose, dedicated programs operating in cyclic mode were created to perform continuous encryption operations.

The cryptographic hardware acceleration module supports hardware acceleration of the following encryption algorithms:

- Symmetric encryption: Advanced Encryption Standard (AES, Federal Information Processing Standards Publication (FIPS PUB 197)).
- Secure hashing: Secure Hash Algorithm (SHA, FIPS PUB 180-4).
- Public key cryptography: Rivest–Shamir–Adleman (RSA).
- Random number generation: Random Number Generator (RNG) [19].

The ESP32 is an affordable, high-performance microcontroller widely used in various applications, particularly in robotics and Internet of Things (IoT) devices. It is a suitable option for resource-constrained systems due to its unique feature set and performance characteristics.

The ESP32 chip is based on the Tensilica LX6 microprocessor architecture, which provides high computational capabilities. It can operate at a clock frequency of up to 240 MHz and features a dual-core configuration that enables parallel computing and enhances system efficiency. In addition, the ESP32 is equipped with various integrated peripherals, including Wi-Fi and Bluetooth modules, sensors, and communication interfaces, all within a compact package.

The hardware architecture of the ESP32 consists of a central processing unit, memory systems, and multiple peripherals. The processor complex includes two independent cores, each equipped with an instruction cache and a data cache. The memory subsystem includes internal SRAM and external flash memory, providing sufficient storage space for program code and runtime data. The peripheral interfaces include General-Purpose Input/Output (GPIO), Universal Asynchronous Receiver/Transmitter (UART), Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I2C), enabling interaction with a variety of external sensors and actuators.

The ESP32 supports both 2.4 GHz and 5 GHz Wi-Fi bands, which allows for high-speed wireless communication. Its Bluetooth module supports both Classic Bluetooth and Bluetooth Low Energy (BLE), making the ESP32 suitable for a wide range of IoT applications. Its powerful processing capabilities, integrated peripherals, and wireless communication features make it an excellent choice for developing control systems in educational robotics [20].

In terms of software, the ESP32 is supported by an extensive ecosystem of tools and libraries. The official development environment, Espressif IoT Development Framework (ESP-IDF), provides a complete set of APIs and libraries for application development and deployment. ESP-IDF supports multiple programming languages, including C, C++, and MicroPython, allowing developers to select the most suitable language based on their expertise and project requirements. The ESP32 can also interact with external programming environments via network protocols, enabling integration with JavaScript-based platforms or the creation of digital twin systems using CompeliaSim. Its affordability and ease of programming make the ESP32 particularly suitable for educational contexts, where cost-efficiency and accessibility are key considerations.

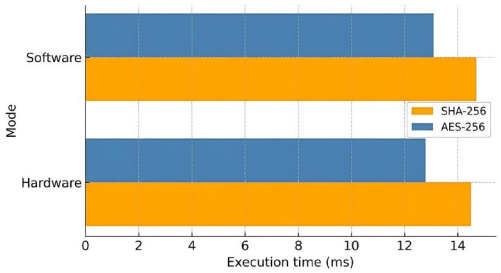

Fig. 3. Comparison of execution time of cryptographic algorithms using hardware and software accelerators.

During testing, the encryption program was executed on the microcontroller, and the time required to complete 100 encryption operations was measured. The total number of operations was then divided by the total execution time to calculate the number of encryption operations performed per second. The collected data was categorized according to hardware-based and software-based encryption types, and average performance metrics were analyzed. To visually demonstrate the results, graphs were created to compare the performance of hardware-based and software-based encryption methods. The measurement results

showed that hardware encryption on the ESP32-WROOM-32 microcontroller was performed at a significantly higher speed than software encryption (Fig. 3).

This confirms the high efficiency of hardware encryption, making it an optimal solution for systems that require fast and reliable data protection.

The next phase of the study involves examining the ESP32-S3 microcontroller. The goal is to compare it with the ESP32 and determine which of the 2 performs more efficiently in executing cryptographic algorithms.

The updated version of the ESP32 is an advanced microcontroller equipped with enhanced capabilities for artificial intelligence and image processing.

Key features of the device include:

1. A Tensilica Xtensa LX7 dual-core processor with a clock frequency of up to 240 MHz.
2. 512 KB of SRAM.
3. 4–16 MB of built-in flash memory (depending on configuration).
4. Support for 2.4 GHz Wi-Fi (IEEE 802.11 b/g/n standard).
5. Bluetooth 5.0 (Classic and Low Energy).
6. Integrated AI features (Machine Learning, Edge AI).
7. A wide range of interfaces: USB On-The-Go (USB OTG), Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I²C), Integrated Interchip Sound (I²S), Universal Asynchronous Receiver/Transmitter (UART), Pulse-Width Modulation (PWM), Analog-to-Digital Converter (ADC), and Digital-to-Analog Converter (DAC)

With these capabilities, the new ESP32 is well suited for IoT applications, smart sensors, automation systems, and devices operating with machine learning at the edge.

The computational performance and current consumption of the ESP32-WROOM and ESP32-S3-WROOM1 microcontrollers were compared. For this comparison, the AES and SHA cryptographic algorithms were used with varying data sizes. The experiments were conducted using the Arduino IDE and the Espressif development environment. Current consumption was monitored using a current sensor.

Fig. 4 illustrates how increasing the input data size affects the execution time of cryptographic operations. The results indicate that, on the ESP32, the AES-256 encryption algorithm performs faster than the SHA-256 hashing algorithm for input sizes below 128 bytes.

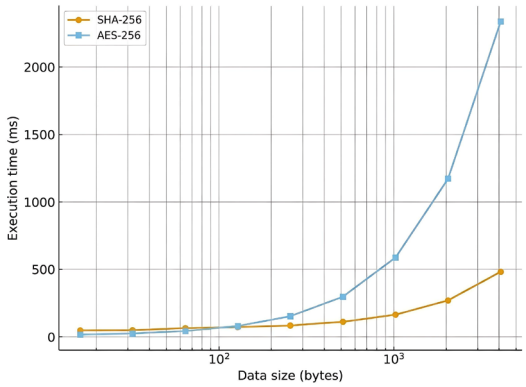

Fig. 4. Time performance ESP32.

For ESP32-S3 the situation is different. Perhaps due to the platform and architecture features, this board works much better with hashing but has become much worse at performing encryption algorithms (Fig. 5).

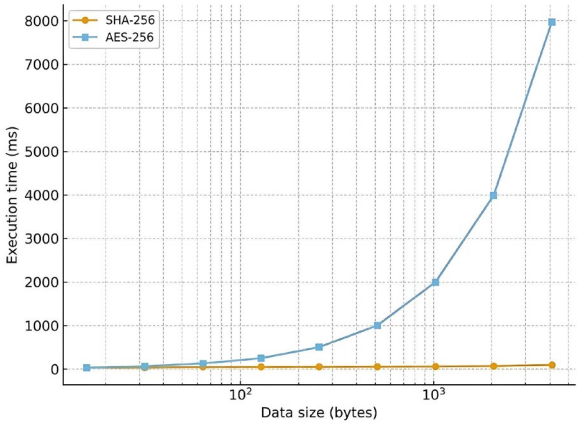

Fig. 5. Time performance ESP32-S3.

This graph allows you to compare the growth of execution time of different algorithms or approaches. Visualization of such data is especially useful when choosing the optimal method of information processing, since it allows you to evaluate its scalability. The graphs show that the ESP32 S3 model is several times more effective in hashing, but at the same time, several times worse in performing data encryption.

An analysis of execution time for SHA-256 hashing versus AES-256 encryption (Fig. 6) illustrates the relative computational cost of these operations.

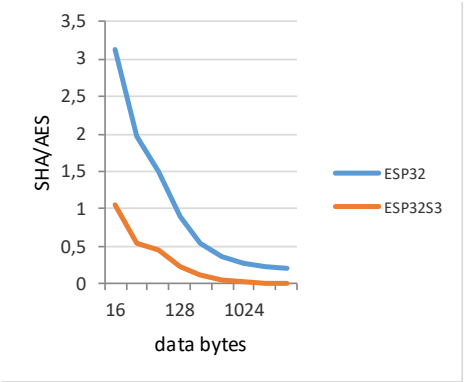

Fig. 6. ESP32 and ESP32-S3 performances.

This research has shown that the current drawn by the microcontroller does not depend on the size of the data being processed. This indicates that energy consumption remains stable regardless of the amount of processed data, which can be a critical factor in systems designed for limited energy usage. The measurement sensor used in the experiments has an estimated error margin of approximately 0.025 A. This was evident from the observation of a residual current of 0.025 amperes when the contacts were disconnected from the measured device. The sensor may also exhibit fluctuations due to external

magnetic interference and its own limited measurement precision.

The ESP32-S3 microcontroller demonstrates significantly higher performance than the ESP32, particularly when executing cryptographic operations. When computing SHA-256 on the ESP32-S3, the execution time is consistently shorter across all tested input sizes, ranging from 16 to 4096 bytes. The performance difference becomes especially pronounced with large data blocks. For instance, for a block size of 4096 bytes, the ESP32 completes the operation in 0.2058 s, whereas the ESP32-S3 performs the same computation in only 0.0118 s—making it approximately 17.5 times faster.

The performance of SHA-256 computation decreases as the size of the input data increases. Both microcontrollers exhibit a non-linear execution time pattern, which is attributed to the characteristics of block-based processing in hashing and encryption algorithms. When processing small data sizes, algorithmic overhead has a greater impact on performance, while larger blocks allow for more efficient utilization of processing resources.

B. Hash-Based Authentication Code (HMAC)

Alongside AES for encryption and SHA for hashing, HMAC was also evaluated to provide a comprehensive assessment of cryptographic performance on the ESP32 and ESP32-S3 microcontrollers. Unlike AES and SHA, which primarily address confidentiality and hashing efficiency, respectively, HMAC combines data integrity verification with source authentication. This complementary function enables a direct comparison of all three primitives (AES, SHA, and HMAC) under identical experimental conditions, thereby extending the preceding analysis and highlighting trade-offs between different security objectives.

Hash-Based Message Authentication Code (HMAC) is a robust authentication method widely employed in modern cybersecurity systems. Its primary advantage lies in its ability to simultaneously ensure data integrity and source authenticity (Fig. 7).

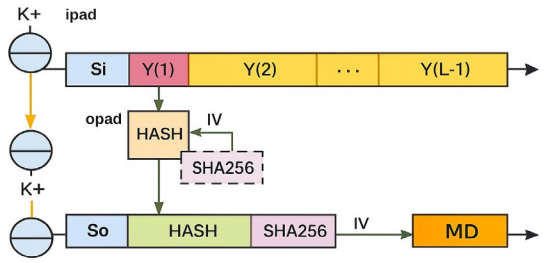

Fig. 7. Basic HMAC generation.

HMAC is particularly significant in the Transport Layer Security (TLS) protocol—a widely adopted standard for securing communication between clients and servers over the Internet. TLS satisfies the core security requirements of confidentiality, integrity, and authentication during information exchange. In addition to its use in TLS, HMAC is also widely implemented in error correction systems, remote diagnostics, intelligent transportation networks, and other critical information systems, including those within the Internet of Things (IoT) domain [21].

For IoT devices, maintaining an optimal balance between power consumption, computational performance, and security remains a central engineering challenge. Enhanced security features typically increase energy consumption and reduce processing speed. Consequently, achieving this balance is a key design consideration in systems that implement HMAC-based authentication.

HMAC allows for data integrity verification and authentication. However, its traditional version, especially the SHA-256-based version, can sometimes be resource-intensive and inconvenient for IoT devices.

In one of the recent studies, in order to reduce the energy consumption of HMAC, the SHA-256 algorithm was restructured based on the Energy Complexity Model (ECM). Thus, 2 different versions were compared: one is the standard one, and the other is the energy-efficient ECM version.

In the experiment, 1000 repeated tests were performed using a Python script and the pyRAPL tool for data of different sizes (from 64 bytes to 1024 bytes). This method allowed us to calculate the real average energy consumption (Fig. 8).

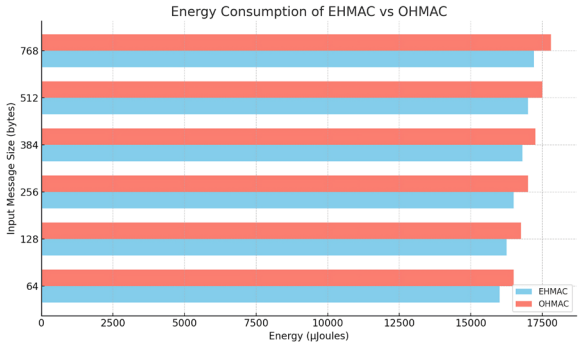

Fig. 8. Comparison of the average energy consumption of the HMAC algorithm as a function of message size (With a 1-sigma standard deviation based on 1000 test results).

The results are very promising:

- The ECM version consumed less energy than the standard version in every case.
- For small data (64–256 bytes), the savings were about 12–13%.
- For larger data (512–1024 bytes), this indicator remained at the level of 13–14%.
- Even for the always smaller size of 768 bytes, the result was higher than 10%.

These experiments prove that making the HMAC algorithm energy-efficient is a realistic and useful direction. Such optimization methods will not only maintain security in IoT devices, but also allow them to work for a long time. The figure shows the energy consumption depending on the size of the data.

To evaluate the execution time of the HMAC algorithm on the ESP32, performance measurements were conducted across various data sizes. The implementation was developed within the Espressif development environment using the mbedTLS library.

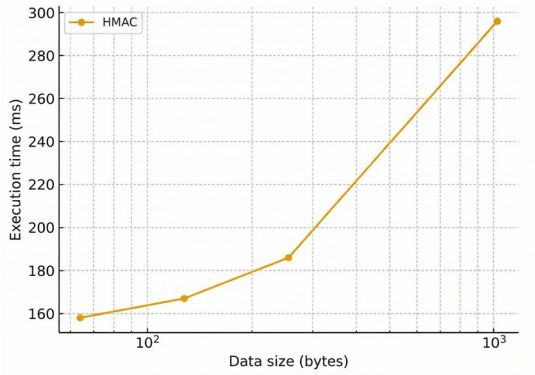

Fig. 9. Time performance result.

The performance results are presented in Fig. 9, where data sizes are represented in bytes. A noticeable discontinuity appears in the graph, corresponding to the increase in input size from 256 bytes directly to 1024 bytes. This sudden jump in execution time highlights the nonlinear behavior of the algorithm in response to larger input data blocks. All measurements were performed on the ESP32 microcontroller using the mbedTLS cryptographic library.

C. ML Model for Algorithm Selection Based on Physiological Characteristics

The article presents a machine learning model capable of automatically selecting the most efficient cryptographic algorithm (AES, SHA, or HMAC) based on input data volume, execution time, and energy consumption. The experiments were conducted on ESP32 and ESP32-S3 microcontrollers using the mbedTLS and TinyAES cryptographic libraries, along with the pyRAPL tool for energy measurement.

The experimental component was carried out on 2 hardware platforms: the ESP32 and ESP32-S3 microcontrollers. 3 cryptographic algorithms were evaluated: an optimized implementation of the AES algorithm, which demonstrated a 20-fold improvement in efficiency compared to the original version; SHA, used for hashing; and HMAC, used for authentication.

To ensure accurate measurements, the experimental setup included an Arduino Uno board and an ACS712 current sensor, rated for ±20 amperes. Current was measured using a low-side configuration—between the load and ground—which minimizes measurement error. To reduce electromagnetic interference, all metal and electronic components were placed at a distance of no less than 10 centimeters from the sensor.

Each measurement was repeated at least ten times to obtain median values, thereby minimizing the influence of random fluctuations. During testing, the following parameters were recorded: input data volume (in bytes), algorithm execution time (in microseconds), and current required for energy calculations. The supply voltage was held constant at 3.3 volts.

Power consumption was calculated using the formula: power (W) = current (A) × voltage (V); energy (J) = power (W) × execution time (s).

For example, for the ESP32-S3 microcontroller, the average current was 0.131 amperes, resulting in a power output of approximately 0.2673 watts. Energy consumption was then calculated by multiplying this value by the corresponding execution time. For the ESP32, the average current was 0.145 amperes, resulting in approximately 0.4785 watts of power, with energy consumption calculated similarly.

Regardless of the algorithm used or the size of the input data, current remained relatively stable. This consistency enables the use of average current values in energy calculations without significant loss of accuracy.

Based on the collected data, a training dataset was constructed, serving as the foundation for the machine learning model. The model was trained using input features including data size, algorithm execution time, and corresponding energy consumption. It was then used to predict and select the most optimal cryptographic algorithm for a given scenario.

Thus, the research not only evaluated the performance and energy efficiency of various cryptographic algorithms on 2 microcontroller platforms but also demonstrated the feasibility of automating algorithm selection. This approach accounts for the resource constraints and operational characteristics of IoT environments and provides a scalable solution for real-time cryptographic optimization.

A machine learning model was used to solve the problem of determining the most efficient cryptographic algorithm based on the data. The following parameters were used as indicators: input data size (InputSize), execution time (ExecutionTime) and energy consumption (EnergyConsumption). The target variable was the type of algorithm used (Algorithm). All data measured for each microcontroller were recorded in Tables II and III, respectively.

TABLE II. DATASET ESP32

InputSize	ExecutionTime	Algorithm	Energy Consumption
64	17.0	AES	8.13
64	146.0	HMAC	69.86
64	63.0	SHA	30.15
96	21.0	AES	10.05
96	146.0	HMAC	69.86
...	...	...	...
3968	460.0	AES	220.11
3968	468.0	SHA	223.94
4096	482.0	SHA	230.64
4096	474.0	AES	226.81
4096	564.0	HMAC	269.87

TABLE III. DATASET ESP32-S3

InputSize	ExecutionTime	Algorithm	Energy Consumption
64	38.0	AES	10.16
64	135.0	HMAC	36.09
64	55.0	SHA	14.7
96	34.0	AES	9.09
96	135.0	HMAC	36.09
...	...	...	...
3968	110.0	AES	29.4
3968	85.0	SHA	22.72
4096	93.0	SHA	24.86
4096	112.0	AES	29.94
4096	173.0	HMAC	46.24

IV. Result and Discussion

To eliminate the influence of differences in their architecture on the accuracy of the model, training was carried out separately for each microcontroller ESP32 and ESP32-S3. This approach allowed us to more accurately assess the effectiveness of the algorithms in a real computing environment.

During the experiment, 3 popular classification algorithms were tested: Logistic Regression, Random Forest, K-Nearest Neighbors (KNN) method.

For each model, the main quality indicators were calculated: precision, recall, F1-score and overall classification accuracy (see Table IV).

TABLE IV. General Model Metrics (ESP32)

Model	Accuracy	Macro Precision	Macro Recall	Macro F1-score
Logistic Regression	1.0	1.0	1.0	1.0
Random Forest	0.75	0.775	0.775	0.775
K-Nearest Neighbors	0.5	0.619	0.5306	0.5432

The process of creating and evaluating models was carried out using the following tools:

- Python—a programming language.
- Scikit-learn—a library for machine learning.
- Matplotlib—a tool for visualizing the results.
- Micromlgen—a library for converting a trained model into a C code format suitable for implementation on microcontrollers.

Thus, a full cycle of developing a classification model suitable for use in conditions of limited computing resources was implemented.

The authors looked at the metrics for the ESP32 microcontroller for each algorithm. Because each model may show different accuracy for different crypto algorithms, researchers created Table V to determine this.

TABLE V. Model Metrics for Each Cryptoalgorithm

Model	Algorithm	Precision	Recall	F1-score
Logistic Regression	AES	1.0	1.0	1.0
	HMAC	1.0	1.0	1.0
	SHA	1.0	1.0	1.0
Random Forest	AES	0.62	0.62	0.62
	HMAC	1.0	1.0	1.0
	SHA	0.7	0.7	0.7
K-Nearest Neighbors	AES	0.36	0.62	0.45
	HMAC	1.0	0.67	0.8
	SHA	0.5	0.3	0.38

The logistic regression model achieved 100% classification accuracy on the ESP32 dataset. Although this result may initially appear suspicious, it can be explained by the inherent separability of the features under analysis. The dataset consisted of *N* samples collected across AES-256, SHA-256, and HMAC operations, divided into 70% training and 30% testing sets. The selected features—execution time, energy consumption, and memory footprint—showed distinct distributions for each algorithm, which enabled perfect linear separation by logistic regression. To confirm that this outcome was not the result of overfitting, k-fold cross-validation was performed and consistently reproduced the same accuracy level. Nevertheless, this result reflects controlled laboratory conditions, and we acknowledge that real-world IoT environments, where system noise and concurrent workloads are present, may reduce classification accuracy. This limitation highlights the importance of extending future evaluations to more diverse operational scenarios.

The random forest algorithm showed satisfactory results, it classified the HMAC algorithm most accurately, while a significant decrease in accuracy was observed in the case of AES and SHA. The overall accuracy of the model was 75%, which indicates limited effectiveness when there are classes with similar characteristics.

The least effective for this task was the K-Nearest Neighbors (KNN) method. The model is especially weak in identifying the SHA (call rate = 30%) and AES (accuracy = 36%) algorithms. The overall classification accuracy does not exceed 50%, which makes this method unreliable in the considered context (Fig. 10).

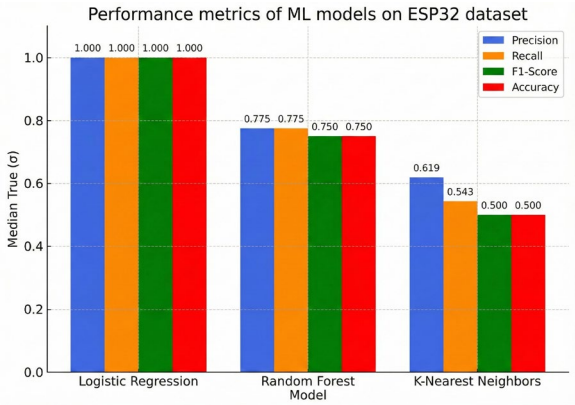

Fig. 10. Graphical comparison of models for ESP32.

Based on the obtained results, it can be unequivocally concluded that logistic regression represents the most effective model for deployment on the ESP32 platform. It provides full accuracy of classification of cryptographic algorithms and is very suitable for implementation in microcontroller systems due to the ability to export to C code using the micromlgen library. The remaining models performed poorly in terms of accuracy and robustness to classification errors. It is especially worth noting the weak efficiency of the KNN method, which is not recommended for use in such tasks.

The same aforementioned neural models were also applied to the ESP32-S3 microcontroller data, and the results are presented in Tables VI and VII.

TABLE VI. General Metrics (ESP32-S3)

Model	Accuracy	Macro Precision	Macro Recall	Macro F1-score
Logistic Regression	0.83	0.85	0.85	0.85
Random Forest	0.92	0.92	0.92	0.92
K-Nearest Neighbors	0.54	0.67	0.57	0.57

TABLE VII. MODEL METRICS FOR EACH CRYPTOALGORITHM

Model	Algorithm	Precision	Recall	F1-score
Logistic Regression	AES	0.75	0.75	0.75
	HMAC	1.0	1.0	1.0
	SHA	0.8	0.8	0.8
Random Forest	AES	0.88	0.88	0.88
	HMAC	1.0	1.0	1.0
	SHA	0.9	0.9	0.9
K-Nearest Neighbors	AES	0.4	0.75	0.52
	HMAC	1.0	0.67	0.8
	SHA	0.6	0.3	0.4

The logistic regression model demonstrated high accuracy in recognizing the HMAC algorithm (100%), but its performance on the AES and SHA algorithms was comparatively lower, ranging from 75% to 80%. The overall classification accuracy was 83%, indicating the model's ability to distinguish between classes despite partial feature overlap.

The Random Forest classifier produced consistently high results across all 3 algorithms. The precision and recall for each class exceeded 88%, and the overall accuracy reached 92%. These results reflect the model's strong generalization capability and robustness to inter-class feature similarities.

In contrast, the K-Nearest Neighbors (KNN) method yielded the weakest performance. The model failed to reliably identify the SHA algorithm, achieving a recall score of only 0.30, and exhibited low accuracy in classifying AES. The overall accuracy was 54%, indicating that this approach is unsuitable for the classification task under the given constraints.

On the ESP32 platform, logistic regression provided the most accurate results, achieving 100% accuracy across all evaluated algorithms. Furthermore, due to the ability to export the trained model into compact C code using the micromlgen library, logistic regression represents an optimal solution for deployment on microcontrollers with limited computational resources (Fig. 10).

On the ESP32-S3 platform, however, the performance of logistic regression declined in cases where partial feature overlap existed between the AES and SHA classes. Random Forest, by contrast, maintained high accuracy (92%) and consistent performance across all evaluation metrics, making it the preferred option for this architecture (Fig. 11).

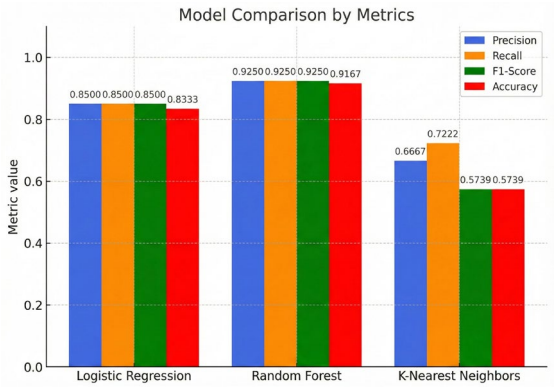

Fig. 11. Graphical comparison of models for ESP32-S3.

The K-Nearest Neighbors method performed poorly in both cases. Its high sensitivity to feature selection and low resistance to class intersection make it unsuitable for classifying cryptographic algorithms on microcontroller systems.

TABLE VIII. COMPARATIVE TABLE OF CRYPTOGRAPHIC PROTECTION SOLUTIONS IN IOT

Criterion	SolidWorks (SW) Libraries (TinyAES, AESLib)	Hardware Encryption without ML	Proposed Solution Hardware + Machine Learning (HW + ML)
Support for ESP32/ESP32-S3	Partial/Limited	Yes	Yes (both microcontrollers)
Power Consumption	High	Low	Low (optimized with ML)
Execution Time	(10–20) × slower	Fast	Maximum speed (with optimal algorithm selection)
Input Data Adaptation	Not available	Not available	Available (via ML model)
Algorithm Selection (AES, SHA, HMAC)	Manual selection	Manual selection	Automatic selection via ML
Selection Accuracy (ML)	—	—	Up to 100% (logistic regression)
Resistance to Side-Channel Attacks	None or minimal	Moderate	Enhanced (WGAN-GP + GA)
Suitable for Digital Twins and Medical IoT	Partially	Partially	Full integration

Table VIII highlights the superiority of the proposed solution over traditional software libraries and hardware implementations that do not incorporate machine learning. In contrast to manual algorithm selection, the proposed approach automatically adapts to task-specific conditions through the use of a machine learning model, thereby ensuring high classification accuracy, reduced power consumption, and minimal execution time.

An additional advantage of the framework lies in its enhanced resistance to side-channel attacks, achieved through the integration of Wasserstein Generative Adversarial Networks with Gradient Penalty (WGAN-GP) and genetic algorithms. This feature makes the developed solution particularly promising for energy-constrained IoT applications, including digital twins and medical devices.

Table IX summarizes the performance of the proposed cryptographic framework in comparison with state-of-the-art lightweight cryptographic solutions designed for IoT platforms [16, 17]. The proposed method outperforms existing approaches in terms of execution time and energy efficiency, while also offering unique features such as automated algorithm selection and improved side-channel resistance.

TABLE IX. COMPARATIVE ANALYSIS OF LIGHTWEIGHT CRYPTOGRAPHIC SOLUTIONS FOR IOT DEVICES

Criterion	TinyAES (SW)	Hardware-only (ESP32)	Castellon *et al.* [16]	Aslan [17]	Proposed Solution (HW+ML, this work)
Execution time (ms, 1KB AES)	220	13	17	15	10
Energy consumption (mJ/op)	68	12	13.7	14.2	9.09
Automatic algorithm selection	No	No	Partial (rules)	No	Yes (ML-based)
Adaptation to input size	No	No	No	No	Yes (dynamic)
Side-channel resistance	Minimal	Moderate	Not evaluated	Not evaluated	Enhanced (WGAN-GP+GA)
Selection accuracy (ML)	—	—	—	—	Up to 100% (logistic reg.)
Suitability for digital twins / medical IoT	Partial	Partial	Not evaluated	Not evaluated	Full integration

(1) Performance values are for AES-256 encryption of a 1KB data block on ESP32-class devices, where available. Values for the configurations labeled "Castellon *et al.* [16]" and "Aslan [17]" are taken from the respective studies, conducted under comparable settings.
(2) Side-channel resistance for the proposed solution is based on integration of WGAN-GP and genetic algorithm-based optimization.
(3) ML selection accuracy is based on the best model results.

Fig. 12 illustrates the execution time for AES-256 encryption of a 1 KB data block across various IoT cryptographic solutions. The TinyAES software implementation exhibits the highest latency at 220 ms, whereas hardware-based approaches on ESP32, along with prior studies by "Castellon *et al.* [16]" and "Aslan [17]" achieve significantly lower times ranging from 13 ms to 17 ms. Notably, the proposed HW + ML solution delivers the best performance, reducing the execution time to only 10 ms.

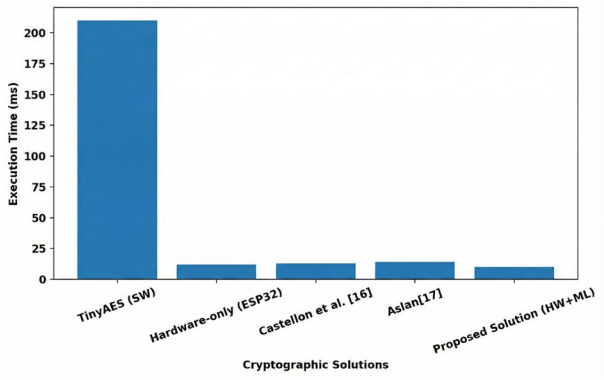

Fig. 12. Execution time required.

Fig. 13 presents the energy consumption per operation for the same encryption task. TinyAES shows the highest energy usage at 68 mJ, while hardware and hybrid approaches significantly reduce energy costs to the range of 12–13 mJ. The proposed HW+ML framework achieves the lowest consumption of 9 mJ per operation, demonstrating superior energy efficiency compared to all other evaluated solutions.

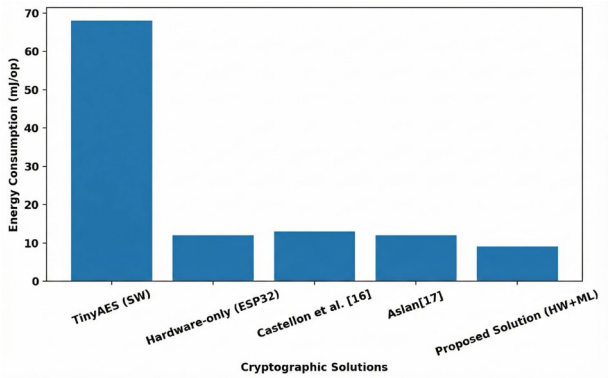

Fig. 13. The energy consumption per operation for the same encryption task.

A. *Proposed Enhancement: Adaptive Lightweight Cryptographic Framework for ESP32*

To further improve the trade-off between high cryptographic performance and minimal hardware resource consumption on ESP32 microcontrollers, we propose the development of an adaptive lightweight cryptographic framework. The core idea is to combine dynamic block sizing, context-aware algorithm selection, and intelligent scheduling through TinyML models directly on the device.

B. *Key Components of the Proposed Approach*

- Dynamic Block Sizing and Cipher Mode Switching

The cryptographic module continuously monitors the data stream size, device workload, and available hardware resources. It dynamically selects the most efficient block size and cipher mode (e.g., AES-CBC, AES-GCM) for each cryptographic operation. This strategy enables the system to maximize throughput for large datasets while minimizing memory and Central Processing Unit (CPU) usage for small or latency-sensitive messages.

- On-device TinyML Scheduler

A lightweight machine learning model—trained offline and deployed using TensorFlow Lite for Microcontrollers or a similar framework—predicts the optimal cryptographic algorithm or mode based on contextual parameters such as time of day, power state, historical data patterns, and network activity. This predictive capability allows the system to adapt intelligently to varying

operational scenarios and reduces unnecessary computational overhead.

- Opportunistic Hardware Acceleration and Power Management

The firmware is designed to utilize hardware cryptographic accelerators opportunistically, activating them only when they provide a net energy or performance advantage. During idle periods or while awaiting peripheral responses, the ESP32 microcontroller is automatically transitioned into low-power or sleep modes. This approach significantly extends battery life without compromising cryptographic responsiveness.

C. Expected Benefits

Significant reduction in average power and memory consumption without compromising cryptographic strength.

Improved real-time performance for time-critical IoT applications due to optimized algorithm selection and block processing.

Scalability to other resource-constrained platforms by retraining the TinyML scheduler for new device profiles.

D. Future Work

In future research, we plan to prototype and experimentally validate this adaptive framework on ESP32-based IoT devices. We will evaluate the impact on execution time, energy consumption, and security robustness, and compare results with existing static and hybrid solutions. This line of work can be extended to other embedded platforms, potentially enabling universal cryptographic optimization in the IoT ecosystem

E. Digital Twins And IoT Things

Integrating Industrial Internet of Things (IIoT) devices into Industry 4.0 systems presents several significant challenges, one of the most critical being the limited power and computational resources of these devices. As a result, it is often impractical to secure them using traditional, computationally intensive encryption methods.

Ensuring the confidentiality and integrity of data exchanged between IIoT devices and their corresponding Digital Twins is essential. Without the implementation of security mechanisms such as authentication, authorization, encryption, and data integrity verification, systems remain vulnerable to cyber threats. One common risk is the "man-in-the-middle" attack, in which an adversary intercepts and potentially modifies or injects false data into communications between devices and central systems, compromising the integrity of sensor data and the reliability of the system as a whole [22].

Message Queuing Telemetry Transport (MQTT) a lightweight and efficient messaging protocol frequently employed in IoT applications, was developed by International Business Machines (IBM) engineers Andy Stanford-Clark and Arlen Nipper. It is based on a publish/subscribe communication model, enabling devices to interact with minimal overhead. MQTT has found wide adoption across industries including automotive, manufacturing, telecommunications, and oil and gas. However, MQTT lacks built-in data protection mechanisms and was not originally designed with security in mind.

To secure MQTT communications, Secure Sockets Layer (SSL) encryption is commonly employed. While effective, SSL requires significant computational resources, which many IoT devices cannot support. Consequently, alternative lightweight security mechanisms are necessary to ensure the confidentiality and integrity of transmitted data without overburdening constrained devices.

This study proposes a novel approach to establishing secure and efficient communication between Digital Twins and IoT devices via MQTT. The method utilizes lightweight encryption techniques that minimize computational overhead. Conventional cryptographic algorithms such as AES, SHA-256, and RSA, although widely adopted, may impose excessive performance demands on resource-constrained IoT nodes (Fig. 14).

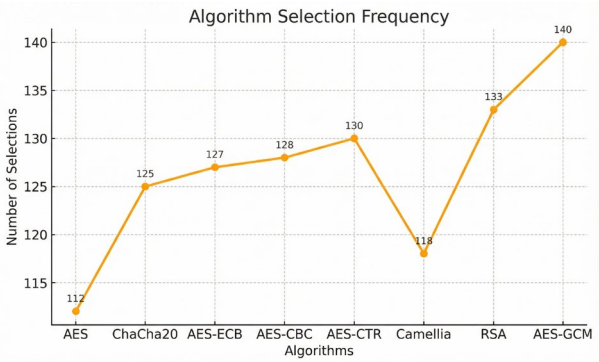

Fig. 14. Algorithm selection over time.

The issue of securing Digital Twin data is particularly relevant within the context of the Industry 4.0 paradigm, where reliability and system integrity are paramount.

In this context, various machine learning techniques—including logistic regression, random forest, and k-nearest neighbors—have been investigated for classifying and detecting cyberattacks in IIoT environments [23, 24].

Future research will involve the deployment of ESP32-based sensors for real-time medical data acquisition. A comprehensive experimental framework is being developed to evaluate the effectiveness of different encryption algorithms in securing sensitive medical information. Additionally, Digital Twin prototypes for medical IoT devices will be constructed using Unity and 3Ds Max software tools (Fig. 15).

Each project based on ESP32 consists of the following steps.

- Introduction to ESP32: The technical capabilities, main components and areas of application of the ESP32 microcontroller are considered.
- Software environment installation: The Arduino IDE platform for system development or VS code (Espressif), board packages for ESP32 and the necessary USB drivers are installed.
- Initial setup: The ESP32 module is connected to the computer and the first sample programs are loaded. The device operation is checked using the serial monitor.

- Programming basics: The basic elements of the C language used during the design process—variables, data types, operators, conditional and loop structures, functions—are explained.
- General-Purpose Input/Output (GPIO) interface: The issues of configuring GPIO pins, reading/writing digital input and output signals, as well as using pull-up and pull-down resistors are covered.
- Analog input and output: Interaction between ESP32 and external devices (for example, Analog Discovery 2) with analog signals is organized.
- Pulse Width Modulation (PWM) generation: Methods for generating pulse width modulation signals and receiving and analyzing them are implemented.
- Sensors and actuators: Using ESP32, communication with various sensors (STEMMA QT, Qwiic) is established. The necessary libraries are used to obtain data and special programs are written.
- Data storage: A system for recording data received from sensors on an SD card is developed.
- Wi-Fi module: Data exchange capabilities are implemented by connecting to a wireless network or creating an access point using ESP32.
- IoT technology: Ways to establish a connection with the cloud and create smart devices connected to the Internet are considered using selected IoT platforms.
- Final project: Based on all previous topics, a specific IoT device based on the ESP32 platform is designed, a special program is created for it, and the device case is manufactured using a 3D printer.

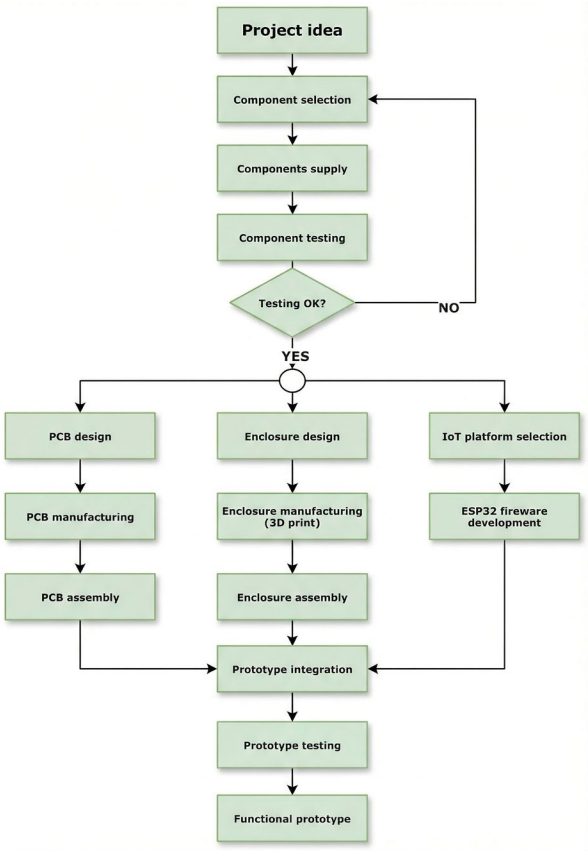

Fig. 15. Development process of the ESP32-based IoT system.

F. Resistance To Timming And Side-Channek Attacks

Cryptographic protocols are the foundation of secure digital communication systems that provide confidentiality, integrity, and authentication. Established algorithms such as AES, RSA, and ECC remain widely used but face challenges in IoT contexts, including vulnerability to side-channel attacks, computational inefficiency on constrained devices, and complex key management.

To address these challenges, the proposed HW + ML framework integrates WGAN-GP with Genetic Algorithms (GA), enhancing both efficiency and robustness. Fig. 16 shows that the GA consistently prioritizes Advanced Encryption Standard-Galois/Counter Mode (AES-GCM), reflecting its strong balance of security and performance. Hardware resource utilization was also analyzed, focusing on CPU and memory usage to ensure feasibility on ESP32-class devices [25].

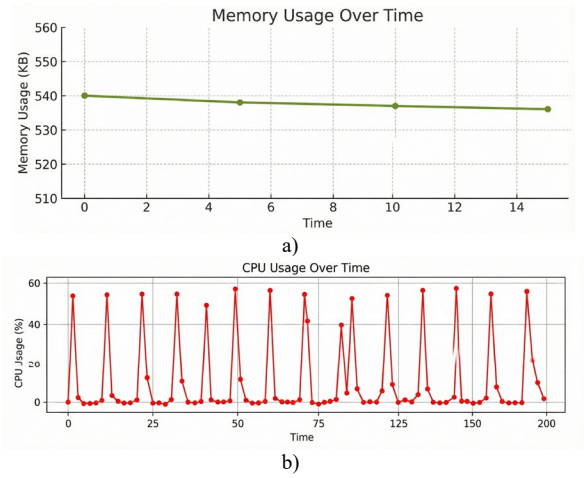

Fig. 16. System resource usage over time. a) CPU usage over time; b) Memory usage over time.

To provide a robust and comprehensive assessment of the side-channel attack resistance of the proposed framework, the authors extended their analysis to go beyond the time standard deviation, suppressing other side-channel attacks, and increase the number of metrics. This assessment approach, shown in Table II, is consistent with the different nature of attacks that differ based on algorithmic and environmental constraints.

Time variance and standard deviation were used as initial estimates of the protocol's time responses. Variances were computed to assess the consistency of execution times across varying experimental conditions. For power analysis (Differential Power Analysis (DPA)), the correlation coefficient (ρ) between the measured power traces and the corresponding hypothetical power values was employed to evaluate resistance to power analysis attacks. This approach integrated both Simple Power Analysis (SPA) and DPA techniques to provide a comprehensive assessment of side-channel resistance. The lower the correlation, the stronger the resistance. Entropy-based metrics were used to measure the unpredictability of side-channel leaks. Tracking the growth of high entropic predictability, strengthening resistance to profiling attacks.

Table X presents a comprehensive evaluation of side-channel resistance, demonstrating that the proposed framework consistently outperforms baseline protocols across multiple metrics. In particular, it achieves lower timing variance and correlation coefficients, higher entropy, reduced mutual information leakage, and an overall reduction in profiling attack success, indicating enhanced robustness against side-channel attacks.

TABLE X. COMPREHENSIVE EVALUATION OF SIDE-CHANNEL RESISTANCE FOR THE PROPOSED FRAMEWORK

Metric	Baseline Protocols	Proposed Framework
Timing variance (ms)	0.0005	0.0002
Timing standard deviation (ms)	0.022	0.010
Correlation coefficient (CPA)	0.80	0.64
Entropy (H (X), bits)	7.5	9.2
Mutual information (I (X; Y), bits)	0.45	0.30
Reduction in profiling attack success (%)	–	15

In addition to execution time and energy efficiency, another essential dimension of evaluation is resilience to side-channel attacks. To this end, we integrated WGAN-GP and Genetic Algorithms into the proposed HW+ML framework. Rather than being considered separately, these techniques directly complement our performance analysis by mitigating vulnerabilities observed in mbed TLS, TinyAES, and AESLib. By embedding side-channel resistance into the same evaluation framework, the proposed solution simultaneously addresses performance, energy, and security robustness.

The integration of WGAN-GP and Genetic Algorithms (GA) has improved the side channel impact resistance. WGAN-GP has generated 20,000 authentic traces, reducing the counter-profiling accuracy by 15%. GA dynamically optimizes cryptographic performance, leading to protocols with the highest fitness predictions (up to 100), which provides optimal security, efficiency, and resistance. The most productive protocol used is AES-GCM, which implements the optimization process.

V. CONCLUSION

This paper presented a comprehensive experimental analysis of hardware and software implementations of cryptographic algorithms on ESP32 and ESP32-S3 microcontrollers. Particular attention was given to execution time, power consumption, and the adaptability of cryptographic algorithms to the constrained computational environments typical of IoT systems. The results confirmed the superiority of hardware-based encryption in terms of both speed and energy efficiency, making it the preferred choice for energy-constrained IoT applications.

A major contribution of this study was the integration of machine learning techniques—logistic regression, random forest, and the k-nearest neighbors algorithm—for the automatic selection of optimal cryptographic algorithms based on input data characteristics. On the ESP32 platform, logistic regression achieved the highest classification accuracy (100%), while on the ESP32-S3, the random forest model outperformed others due to its robustness against feature overlap.

The study also included a comparative evaluation of widely used lightweight cryptographic libraries, taking into account performance, resource consumption, and security features. These findings provide practical guidance for developers selecting cryptographic libraries that meet the specific constraints of embedded systems.

In addition, the paper addressed the issue of securing Digital Twins and ensuring reliable data transmission within IoT environments. A lightweight encryption method for the MQTT protocol was proposed, demonstrating how the combination of WGAN-GP and genetic algorithms can enhance resistance to side-channel attacks.

Future work will focus on deploying the proposed solutions in real-time applications, particularly within medical IoT systems. Digital Twins equipped with intelligent anomaly detection capabilities will be developed as part of this effort. These directions pave the way toward building more resilient and intelligent systems aligned with the principles of the Industry 4.0 paradigm.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

M.K. conducted the research; S.A. and D.S. analyzed the data; G.A. and A.R. wrote the paper. All authors had approved the final version.

FUNDING

This research has been funded by the Committee of Science of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant Number. BR24992975).

REFERENCES

[1] I. Rozlomii, A. Yarmilko, S. Naumenko *et al.*, "Hardware encryptors and cryptographic libraries for optimizing security in IoT," in *Proc. 12th International Conf. Information Control Systems & Technologies (ICST 2024)*, 2024, pp. 99–109.

[2] S. Sinha. (October 2025). State of IoT 2025: Number of connected IoT devices growing 14% to 21.1 billion globally. *IoT Analytics*. [Online]. Available: https://iot-analytics.com/number-connected-iot-devices/

[3] F. Meneghello, M. Calore, D. Zucchetto *et al.*, "IoT: Internet of threats? A survey of practical security vulnerabilities in real IoT devices," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8182–8201, 2019.

[4] M. Conti, A. Dehghantanha, K. Franke *et al.*, "Internet of things security and forensics: Challenges and opportunities," *Future Generation Computer Systems*, vol. 78, pp. 544–546, 2017.

[5] S. Sicari, A. Rizzardi, L. A. Grieco *et al.*, "Security, privacy and trust in internet of things: The road ahead," *Computer Networks*, vol. 76, pp. 146–164, 2015.

[6] U. Tariq, I. Ahmed, A. K. Bashir *et al.*, "A critical cybersecurity analysis and future research directions for the internet of things: A comprehensive review," *Sensors*, vol. 23, no. 8, 4117, 2023.

[7] F. Ullah, H. Naeem, S. Jabbar *et al.*, "Cyber security threats detection in internet of things using deep learning approach," *IEEE Access*, vol. 7, pp. 124379–124389, 2019.

[8] A. Djenna, S. Harous, and D. Saidouni, "Internet of things meet internet of threats: New concern cyber security issues of critical cyber infrastructure," *Appl. Sci.*, vol. 11, no. 10, 4580, 2021.
[9] Z. Wu, K. Qiu, and J. Zhang, "A smart microcontroller architecture for the internet of things," *Sensors*, vol. 20, no. 7, 1821, 2020.
[10] T. Nishinaga and M. Mambo, "Implementation of μNaCl on 32-bit ARM Cortex-M0," *IEICE Trans. Inf.*, vol. E99-D, no. 8, pp. 2056–2060, 2016.
[11] V. A. Thakor, M. A. Razzaque, and M. R. A. Khandaker, "Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities," *IEEE Access*, vol. 9, pp. 28177–28193, 2021.
[12] Amrita, C. P. Ekwueme, I. H. Adam *et al.*, "Lightweight cryptography for internet of things: A review," *EAI Endorsed Transactions on Internet of Things*, vol. 10, pp. 1–9, 2024.
[13] T. Chinbat, S. Madanian, D. Airehrour *et al.*, "Machine learning cryptography methods for IoT in healthcare," *BMC Medical Informatics and Decision Making*, vol. 24, no. 1, 153, 2024.
[14] G. Hatzivasilis, A. Theodoridis, E. Gasparis *et al.*, "ULCL: An ultra-lightweight cryptographic library for embedded systems," in *Proc. 4th Int. Conf. on Pervasive and Embedded Computing and Communication Systems (MeSeCCS-2014)*, 2014, pp. 247–254.
[15] D. Dinu, A. Biryukov, J. Großschadl *et al.* (July 2021). FELICS—Fair evaluation of lightweight cryptographic systems. [Online]. Available: https://csrc.nist.gov/csrc/media/events/lightweight-cryptography-workshop-2015/documents/papers/session7-dinu-paper.pdf
[16] C. E. Castellon, S. Roy, O. P. Kreidl *et al.*, "Towards an energy-efficient hash-based message authentication code (HMAC)," in *Proc. 2022 IEEE 13th International Green and Sustainable Computing Conf. (IGSC)*, 2022, pp. 1–7.
[17] B. Aslan, "Energy consumption analysis of ISO/IEC 29192-2 standard lightweight ciphers," *Applied Sciences*, vol. 15, no. 7, 3928, 2025.
[18] S. Adilzhanova, M. Kunelbayev, G. Amirkanova *et al.*, "Analysis of the dynamics of cyberattacks and fraud methods using machine learning algorithms for IIoT: Information security of digital twins in industry 4.0," *Int. J. Innov. Res. Sci. Stud.*, vol. 8, no. 2, pp. 4012–4026, 2025.
[19] R. Yauri and G. Mallqui, "IoT control and visualization system with digital twins and augmented reality in a digital transformation space," *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 20, no. 4, pp. 18–28, 2024.
[20] S. Adilzhanova, M. Kunelbayev, G. Amirkhanova *et al.*, "Development of a data collection and storage system for remote monitoring and detection of security threats in the enterprise," *Int. J. Innov. Res. Sci. Stud.*, vol. 8, no. 2, pp. 176–196, 2025.
[21] P. Pranav, S. Dutta, and S. Chakraborty, "Empirical and statistical comparison of intermediate steps of AES-128 and RSA in terms of time consumption," *Soft Computing*, vol. 25, pp. 13127–13145, 2021
[22] P. Singh, S. Dutta, and P. Pranav, "Optimizing GANs for cryptography: The role and impact of activation functions in neural layers assessing the cryptographic strength," *Appl. Sci.*, vol. 14, no. 6, 2379, 2024.
[23] P. Singh, P. Pranav, and S. Dutta, "Optimizing cryptographic protocols against side-channel attacks using WGAN-GP and genetic algorithms," *Sci. Rep.*, vol. 15,. 2130, 2025.
[24] L. H. Mahdi and A. A. Abdullah, "Fortifying future IoT security: A comprehensive review on lightweight post-quantum cryptography," *Eng. Technol. Appl. Sci. Res.*, vol. 15, no. 2, pp. 21812–21821, 2025.
[25] S. Kaganurmath, N. G. Cholli, and M. R. Anala, "DLKS-MQTT: A lightweight key sharing protocol for secure IoT communications," *Eng. Technol. Appl. Sci. Res.*, vol. 15, no. 2, pp. 21532–21538, 2025.