Secure IoT Seed-based Matrix Key Generator

A Novel Algorithm for Steganographic Security application

Youssef NOUR-EL AINE, Cherkaoui LEGHRIS
Laboratory of Mathematics, Computer Science and Applications-Sciences and Technologies Faculty of Mohammedia,
Hassan II University, Mohammedia, Morocco

***Abstract*—The rapid evolution of the Internet of Things (IoT) has significantly transformed various aspects of both personal and professional spheres, offering innovative solutions in fields from home automation to industrial manufacturing. This progression is driven by the integration of physical devices with digital networks, facilitating efficient communication and data processing. However, such advancements bring forth critical security challenges, especially regarding data privacy and network integrity. Conventional cryptographic methods often fall short in addressing the unique requirements of IoT environments, such as limited device computational power and the need for efficient energy consumption. This paper introduces a novel approach to IoT security, inspired by the principles of steganography – the art of concealing information within other non-secret data. This method enhances security by embedding secret information within the payload or communication protocols, aligning with the low-power and minimal processing capabilities of IoT devices. We propose a steganographic key generation algorithm, adapted from the Diffie-Hellman key exchange model, tailored for IoT. This approach eliminates the need for explicit parameter exchange, thereby reducing vulnerability to key interception and unauthorized access, prevalent in IoT networks. The algorithm utilizes a pre-shared 2D matrix and a synchronized seed-based approach for covert communication without explicit data exchange. Furthermore, we have rigorously tested our algorithm using the NIST Statistical Test Suite (STS), comparing its execution time with other algorithms. The results underscore our algorithm's superior performance and suitability for IoT applications, highlighting its potential to secure IoT networks effectively without compromising on efficiency and device resource constraints. This paper presents the design, implementation, and potential implications of this algorithm for enhancing IoT security, ensuring the full realization of IoT benefits without compromising user security and privacy.**

Keywords—Security; IoT; steganography; key exchange; cryptography

I. Introduction

The remarkable rise of the Internet of Things (IoT) has revolutionized numerous aspects of our daily and professional lives, bringing significant innovations across diverse fields ranging from home automation to industrial manufacturing [1]. This transformation is largely fueled by the seamless integration of physical devices with digital networks, enabling them to communicate, analyze, and act upon data with unprecedented efficiency and scale [24]. However, this rapidly expanding network of interconnected devices presents substantial security challenges, particularly in the realm of data privacy and network integrity [2].

In this context, the development of robust and innovative security protocols becomes paramount [25]. Traditional cryptographic methods, while effective in many scenarios, often struggle to meet the unique demands of IoT environments [3]. These challenges stem from factors such as limited computational power of IoT devices, the need for efficient energy consumption, and the requirement for seamless, continuous communication among a vast array of devices [26]. Therefore, there is a pressing need for tailored security solutions that address these specific constraints while providing robust protection against evolving cyber threats [4].

To secure network communications, cryptographic algorithms are employed, with the Diffie-Hellman key exchange algorithm being widely used in a public communication channel considered insecure [5]. While cryptography-based algorithms are robust in ensuring secure exchanges, they come with a significant cost in terms of energy consumption, memory usage, and resources [6]. This makes them unsuitable for IoT devices, which have constraints related to processing power, storage, and battery life autonomy.

Research efforts have focused on developing lightweight versions of these cryptographic algorithms. [7], in his article, categorizes lightweight cryptography primitives into four groups: lightweight block ciphers, lightweight stream ciphers, lightweight hash functions, and lightweight elliptic curve cryptography. Lightweight block ciphers use smaller block sizes, smaller security keys, and simpler round designs, as well as key schedules. Lightweight stream ciphers aim to reduce chip area, key length, and minimize internal state. Lightweight hash functions focus on reducing output size and message size. Lastly, lightweight ECC works on reducing memory requirements, optimizing public functions, group arithmetic, and improving speed.

In Dhana's study [7], 54 of these various lightweight cryptography algorithms were compared. The results of the comparison show that lightweight cryptographic algorithms have significant potential for adapting traditional cryptography to IoT device constraints. However, the techniques employed can still be costly in terms of processing power and memory usage. Additionally, reducing key size or length can expose the system to various attacks targeting IoT architectures, potentially weakening security.

This paper introduces an innovative approach to IoT security, drawing inspiration from the principles of steganography — the art of concealing information within other non-secret text or data. Unlike conventional cryptographic methods that primarily focus on encrypting data, steganography involves embedding secret information within the payload or within the communication protocols themselves, thus camouflaging the existence of the sensitive data. This technique not only enhances security by obfuscating the data transfer but also aligns well with the low-power and minimal processing capabilities of many IoT devices.

The core of this research lies in the development of an algorithm akin to the Diffie-Hellman key exchange, a cornerstone of modern cryptography, but adapted for the unique landscape of IoT. This steganographic key generation algorithm leverages the principles of key exchange while embedding the security within the data transmission process itself. The noteworthy advantage of this approach is the elimination of explicit parameters exchange, thereby substantially reducing the vulnerability to key interception and unauthorized access, which are prevalent security challenges in IoT networks.

II. Motivation and Key Challenges

Designing a security algorithm to secure communication between IoT devices poses a significant challenge. The constraints associated with these devices and the complexity of cryptographic algorithms have led us to consider alternative, less computationally intensive techniques. Steganography, as a message concealment technique, has appeared highly promising, as the effort required to hide a message is less costly than encrypting and decrypting the same message.

In our initial work [8], we introduced a basic method to ensure the confidentiality of exchanges between an IoT sensor and a Fog server in a Smart Greenhouse. The results obtained clearly demonstrate that the use of steganography in network communication security can significantly reduce various costs without compromising communication security. These results motivated us to propose a new security algorithm based on steganography, enabling the generation of security keys between an IoT client and an IoT server. To achieve this, we established the following objectives:

- The algorithm must be adapted to steganographic uses while adhering to the Kerckhoffs's principle [9].
- It should generate as many security keys as needed, on-demand, without requiring explicit communication.
- The generation of a new security key should occur simultaneously on both the client and server sides without any explicit exchange.
- Each new exchange between the client and server must utilize a different security key from the previous one, or even different from all previously used keys.
- The entire solution must be suitable for IoT devices, accommodating their various constraints.

III. Related Works

In study [10] the authors proposed an anonymous authenticated key agreement protocol utilizing pairing-based cryptography. The suggested scheme comprises three steps: initialization, anonymous registration, and anonymous authentication key agreement. In the initialization step, the Home Server (HS) generates public and private parameters, which are employed to compute and generate private keys for end-users and IoT devices. During the registration step, users and IoT devices generate U and A values using a random number generator, communicate these values to the HS, and receive their private keys from it. To transmit the private key securely to users and IoT devices via a public channel, an additional session key is computed for encrypting the private key. The authentication and key agreement step enables mobile users and IoT devices to authenticate with the server, employing an authentication process based on timestamp verification. The presented solution elicits concerns, particularly due to the extensive exchange of parameters between the server and IoT devices/mobile users. This heightened parameter exchange raises apprehensions regarding its potential adverse effects on the performance of IoT devices. Additionally, the imperative to encrypt the authentication key for transmission over a public channel introduces a non-trivial layer of complexity. Another notable issue pertains to the non-mutual nature of the proposed authentication, rendering the system susceptible to Man-In-The-Middle (MITM) attacks. The vulnerability exposed by this authentication approach underscores the necessity for a more robust security framework to safeguard communication channels. Regarding the use of timestamps in the final step, the critique underscores the potential pitfalls associated with the system's reliance on current time values. The intricacies of effectively controlling and synchronizing timestamps pose a significant challenge, thereby compromising the overall system integrity.

In study [11] Shahwar Ali et al. present a comprehensive cryptographic approach designed for the unique challenges of wireless sensor networks (WSNs). This approach combines an enhanced key exchange protocol with a secure routing mechanism, ensuring both communication security and network efficiency. Key Generation Phase (Phase 1): The process initiates with the sender and receiver nodes selecting specific values based on prime numbers. The sender then selects two random prime numbers, P and G, and computes a complex hash function of these values. This step significantly strengthens the security of the key exchange process by adding an extra layer of complexity. Encryption Algorithm (Phase 2): Subsequently, the parties involved compute values P1 and P2 using the base G and modulus P, and then hash these values. These hashed values are exchanged and rehashed upon receipt. Each party then computes the final key as P2A mod P and P1B mod P for parties A and B, respectively. The encryption of the plaintext is performed by converting it and the key into binary values and then applying the XNOR operation, ensuring secure data transmission. LEACH Protocol (Phase 3): The third phase introduces the LEACH protocol for secure data routing in WSNs. This phase involves using a clustering approach where cluster heads are selected randomly. The sender sends data or key parameters to its closest cluster head, which then forwards the data to the sink node. This protocol is instrumental in delivering secure communication between sensor nodes, protecting against various attacks, and efficiently managing network energy consumption. This methodology stands out for its amalgamation of enhanced security features, computational efficiency, and an adaptive routing mechanism, making it highly suitable for WSNs where resource constraints and security are critical considerations. The integration of hashing in the key exchange, binary operations for encryption, and the implementation of the LEACH protocol exemplify a sophisticated and holistic approach to securing communications in WSN environments.

Another work [12] based on steganography proposes a new method for key exchange that combines the Diffie-Hellman protocol with image registration techniques. The method involves concealing a key within a set of transformed images. This is done by using the Diffie-Hellman protocol along with image registration using Fast Fourier Transform (FFT), The approach consists of finding transformations between images, which then become a tool for the receiver to recover the key. This process uses image registration techniques that calculate a spatial transformation function between images to superimpose them optimally. The key exchange procedure consists of following steps: Secret keys are divided into blocks of 2 bytes each, for each block, a set of translated images is generated and sent to receiver, the recipient uses image registration to align the received images with a source image, determining the transformations (Tx, Ty) for each block, these transformations represent the data blocks of the secret key. To increase the security level, the authors suggest synchronizing the sending of translated images by a permutation of pseudo-random numbers generated by chaos theory. This ensures that the transformed images (TxiTyi) are sent in a specific order that can only be deciphered using the same pseudo-random number sequence. The method is robust to noise and provides additional security layers through image transformation and registration and the use of Diffie-Hellman but it’s application in the context of Internet of Things can face various difficulty. The Complexity of Implementation: IoT devices often have limited computational resources and power. The proposed method, involving image registration and transformation using the Fast Fourier Transform (FFT), may be computationally intensive for many IoT devices, especially those with limited processing capabilities. Data Transmission Overhead: IoT environments typically involve large networks of devices communicating frequently. The method's reliance on transmitting sets of transformed images for each key exchange could lead to significant data transmission overhead, impacting network performance, especially in bandwidth-constrained IoT scenarios. Real-time Processing Limitations: Many IoT applications require real-time or near-real-time data processing and decision-making. The additional time required for image registration and key recovery might introduce latency that could be detrimental in time-sensitive IoT applications.

In this paper [13] the authors present a novel approach to secure communication using a combination of steganography and cryptography, the method integrates image steganography with the One-Time-Pad (OTP) encryption algorithm. Steganography is the practice of concealing a message within another medium, in this case, an image, it uses Discrete Haar Wavelet Transform (DHWT) to transform the cover images into sub-bands. This transformation helps in embedding the encrypted data into the image with minimal impact on the image quality. The encrypted data is embedded into the image using the LSB method. This technique involves modifying the least significant bits of the pixel values of the image to encode the data, then the secret message is encrypted using the OTP encryption algorithm, known for its theoretical security. This encryption is applied before embedding the data into the image. After embedding the data using the LSB method, Optimal Pixel Adjustment Process (OPAP) is used to minimize the errors in the stego-image (the image containing the hidden message), enhancing the method's undetectability. The encryption key for the OTP algorithm is not directly shared between the sender and receiver. Instead, a shared pool of keys is maintained at both ends, from which keys are randomly selected. This approach avoids the need for a separate secure channel for key exchange, addressing a common weakness in OTP encryption. while the proposed method offers an intriguing combination of steganography and cryptography for secure communication, its practical application in the IoT context raises significant concerns regarding resource and energy efficiency, real-time processing capabilities, scalability, and seamless integration with existing IoT protocols and infrastructures, in particular the computational complexity involved in executing Discrete Haar Wavelet Transform (DHWT), Least Significant Bit (LSB) embedding, and One-Time-Pad (OTP) encryption might be too demanding for such devices, and for memory constraints the proposed method requires a pool of keys to be maintained at both the sender and receiver ends for the One-Time-Pad (OTP) encryption. Given that OTP requires keys as long as the message itself for true security, this could demand substantial memory, particularly in scenarios where large or numerous messages are being transmitted.

IV. Proposed Algorithm

This section aims to provide a comprehensive overview of this novel algorithm, detailing its design, implementation, and potential implications for IoT security. It represents a significant step forward in the ongoing effort to secure the ever-expanding universe of interconnected devices, ensuring that the benefits of IoT can be fully realized without compromising the security and privacy of users.

The algorithm (see Fig. 1) leverages a pre-shared 2D matrix and a synchronized seed-based approach to establish shared pairs of elements for steganographic purposes, thus enabling covert communication without explicit data exchange.

A. *Matrix Selection Phase*

The foundation of our steganographic algorithm is a pre-shared 2D matrix. This matrix, known to both device A and device B, serves as the source for generating the keys that constitutes the shared secret used for steganographic purposes.

The pre-shared matrix M (see Fig. 2) is represented as a 2D array, where:

- M[i][j] denotes the value at row i and column j.
- N*N is the length of the matrix.

This matrix is shared between device A and device B to generate keys for steganographic purposes. The values in the matrix can be chosen according to the specific application of the algorithm. In our case, the values are integer numbers used as positions in the cover media where the message can be hidden. The length of the matrix is linked to the length of the cover media, the seize of transmitted data (message), the length of the key, and the degree of robustness required, in general:

- The length of the matrix L_M must be at least greater than or equal to the length of the Cover-Media L_{CM}
- The length of the selected key LK must equal the length of the Message LMSG
- The length of the Cover-Media LCM must be greater than length of the message LMSG:

$$L_M \geq L_{CM} \tag{1}$$

$$L_K = L_{MSG} \tag{2}$$

$$L_{CM} = \frac{L_{MSG}}{R}, \text{ with } 0 < R \leq 1 \tag{3}$$

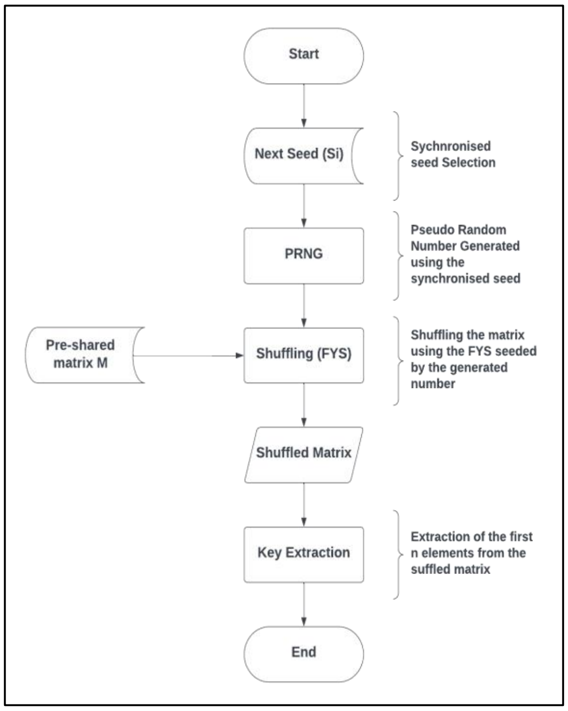
Start
Next Seed (Si)
Sychnronised seed Selection
PRNG
Pseudo Random Number Generated using the synchronised seed
Pre-shared matrix M
Shuffling (FYS)
Shuffling the matrix using the FYS seeded by the generated number
Shuffled Matrix
Key Extraction
Extraction of the first n elements from the suffled matrix
End

Fig. 1. Secure IoT seed-based matrix key generator algorithm.

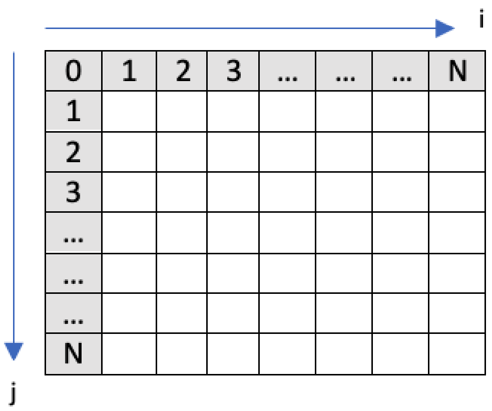

Fig. 2. Pre-shared 2D matrix with a length of N*N.

From Eq. (1), (2), and (3) we can conclude:

$$L_M \geq \frac{L_{MSG}}{R} \tag{4}$$

R is a coefficient; the robustness increases as the coefficient R decreases.

B. Seed-Based Synchronization and PRNG Phase

The essential element of our steganographic algorithm's success is the use of a synchronized seed value. This seed serves as an input to the pseudo-random number generator used by both device A and device B. The synchronization achieved through the seed ensures that both devices generate the same random each time. This shared randomness forms the basis for the establishment of identical keys.

The synchronization process can be a simple incrementation of the seed value:

- $S_{i+1} = S_i + 1$ for E_{i+1}
- Where:
- i denotes the sequence number of the exchange at a given moment,
- S_{i+1} is the next seed to be used.
- E_{i+1} is the next exchange to be performed.

Pseudo Random Number Generators (PRNGs) are algorithms used to produce a sequence of numbers that approximates the properties of random numbers [14], these numbers aren't truly random but are pseudo-random, meaning they are generated in a predictable fashion using a mathematical formula. When a seed value is provided to a PRNG, it uses this value as the initial state, the PRNG applies a mathematical operation to this seed value to produce a new number, which then becomes the input for the next iteration and so on. Seeding a PRNG with a specific number makes its output reproducible [15], if not seeded the PRNG will usually use a value derived from the system clock called timestamp as a seed, (see Fig. 3).

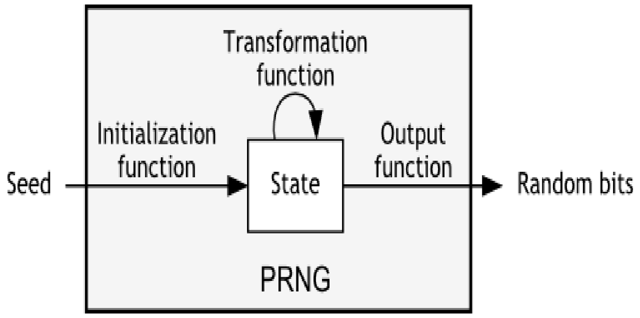

Fig. 3. PNRG initialization with seed value.

C. Shuffling the Matrix Phase

The Fisher-Yates shuffle, also known as the Knuth shuffle, is an algorithm for generating a random permutation of a finite sequence—in other words, for shuffling the sequence [16]. The algorithm effectively shuffles the array or sequence in place, meaning it requires only a small, fixed amount of memory space regardless of the size of the array [17]. The Fisher-Yates shuffle a sequence of random generated numbers, by default if no sequence is giving the FYS use a default sequence derived from a system-related source. Given an array A with n elements (indexed from *0* to *n-1*), the Fisher-Yates Shuffle algorithm proceeds as follows:

```
 = length (A)
for i from n-1 down to 1 do:
    j = random integer in range [0, i]
    swap A[i] with A[j]
```

The algorithm effectively shuffles the array A in place, ensuring each element has an equal probability of ending up in any position, the Fisher-Yates Shuffle algorithm can be modified to use a seeded random number generator as follows:

```
seededPRNG(seed)
n = length(A)
for i from n-1 down to 1 do:
    j = random integer in range [0, i] from seededPRNG
    swap A[i] with A[j]
```

In our case the FYS will be state initialized using the Pseudo random number generated in the previous phase, the pre-shared matrix will be shuffled using the FYS initialized to the state fixed by the pseudo random generated number, using the same number will always produce the same shuffled order of the matrix, in this way each time devices A and B will have the same shuffled matrix.

Considering that each unique arrangement of the matrix elements represents a distinct shuffle, the total number of elements in the matrix is N^2 (since there are *N* elements in each of the *N* rows). The number of different ways to arrange these N^2 elements is giving by the factorial of N^2 (denotes as $(N^2)!$). So the number of different shuffled matrices we can have from the original 2D matrix is $(N^2)!$

D. Key Extraction

After shuffling the matrix within the Fisher-Yates shuffle algorithm, the first n elements are extracted from the shuffled matrix M and are concatenated in the order of extraction to form the key:

$$Key \; = \|_{i=0}^{n} \; [k_i]$$

In this formula:

- k_i represents the i-th element of the matrix M
- **||** is the symbol of concatenation
- *Key* is the resultant key formed by concatenating the first n elements of M.

In consideration of a two-dimensional matrix *M* with dimensions *N*N* it follows that the total number of distinct elements within the matrix is N^2. Consequently, when determining the total number of potential keys that can be derived from this matrix where each unique permutation of the matrix's elements constitutes an individual key, the combinatorial function is expressed as $P(N^2, n)$.

$$P(N^2, n) = \frac{(N^2)!}{(N^2 - n)!}$$

Here, P represents the permutation of N^2 elements considered n at a time, thus providing the count of all possible ordered arrangements that can be constructed from the matrix elements to form keys of length *n*.

V. Results and Discussion

A. Robustness of the Algorithm

In order to empirically validate the robustness of our proposed key generation algorithm, we conducted a comprehensive experiment across a heterogeneous array of Internet of Things (IoT) devices. The algorithm was implemented on five Raspberry pi Pico microcontroller units (see Fig. 4) each initiated with a different matrix M (from M1 to M5), and tasked with the generation of one million unique keys. This extensive production of keys served to simulate a real-world application scenario and to stress test the algorithm's scalability and adaptability across devices with varying workloads and operating conditions. Subsequently, the generated keys from each device were subjected to an empirical study [18] using the NIST Statistical Test Suite (STS).

The NIST Test Suite is a statistical package consisting of 15 tests that were developed to test the randomness of (arbitrarily long) binary sequences produced by either hardware or software based cryptographic random or pseudorandom number generators. These tests focus on a variety of different types of non-randomness that could exist in a sequence. Some tests are decomposable into a variety of subtests. The most commonly used tests in the NIST Statistical Test Suite for evaluating random number generators include [23]:

- Frequency (Monobit) Test: Checks if the number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence. To pass, the p-value must typically be greater than 0.01.

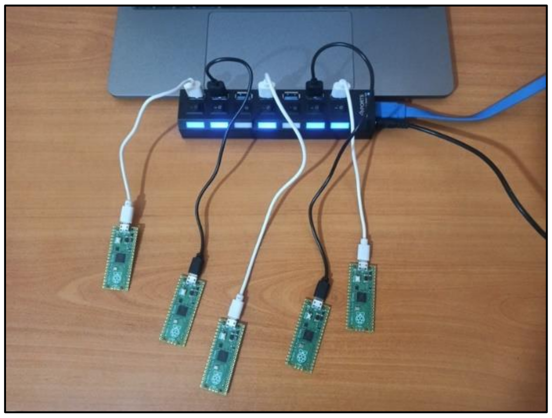

Fig. 4. Implementation scenario for key generation in Raspberry Pi Pico devices.

TABLE I. NIST STATISTICAL TEST RESULTS

Test	M1		M2		M3		M4		M5	
	P-Value	*Proportion*	*P-Value*	*Proportion*	*P-Value*	*Proportion*	*P-Value*	*Proportion*	*P-Value*	*Proportion*
Frequency	0,350485	10/10	0,911413	10/10	0,739918	9/10	0,739918	10/10	0,534146	10/10
BlockFrequency	0,534146	10/10	0,066882	10/10	0,739918	10/10	0,213309	9/10	0,122325	10/10
CumulativeSums	0,534146	10/10	0,911413	10/10	0,350485	9/10	0,122325	10/10	0,534146	10/10
Runs	0,350485	10/10	0,911413	10/10	0,739918	10/10	0,911413	10/10	0,534146	10/10
LonguestRun	0,350485	10/10	0,350485	10/10	0,066882	10/10	0,911413	10/10	0,066882	10/10
Rank	0,350485	10/10	0,004301	9/10	0,000199	10/10	0,066882	10/10	0,017912	10/10
FFT	0,017912	10/10	0,534146	10/10	0,534146	10/10	0,534146	10/10	0,122325	10/10
NonOverlappingTemplate	0,433723	9/10	0,429832	09/10	0,440727	10/10	0,428214	9/10	0,439083	10/10
OverlappingTemplate	0,534146	10/10	0,350485	10/10	0,739918	10/10	0,534146	10/10	0,017912	10/10
ApproximativeEntropy	0,213309	10/10	0,122325	10/10	0,350485	10/10	0,739918	10/10	0,122325	08/10
RandomExcursions	--	2/2	--	2/2	--	2/2	--	2/2	--	2/2
RandomExcursionsVariant	--	2/2	--	2/2	--	2/2	--	2/2	--	2/2
Serial	0,911413	10/10	0,534146	10/10	0,534146	10/10	0,213309	10/10	0,739918	10/10
LinearComplexity	0,534146	10/10	0,066882	10/10	0,066882	9/10	0,213309	10/10	0,739918	10/10

- Block Frequency Test: Determines whether the frequency of ones and zeros in an M-bit block is approximately half. Pass condition is similar, with p-values generally expected to exceed 0.01.
- Cumulative Sums (Cusum) Test: Assesses whether the cumulative sum of the binary sequence fluctuates as would be expected for a random sequence. Passing usually requires p-values above 0.01.
- Runs Test: Evaluates the sequence for runs of both ones and zeros of various lengths to determine if they appear too frequently or infrequently. To pass, the p-value should again be above 0.01.
- Longest Run of Ones in a Block Test: Looks at blocks of the binary sequence to determine if the longest run of ones within these blocks conforms to the expected distribution for a random sequence. The p-value must be above 0.01 for a pass.
- Binary Matrix Rank Test: Examines the rank of disjoint submatrices of the entire sequence. The proportion of matrices with full rank is compared against that expected for random matrices. The pass criterion is a p-value above 0.01.
- Discrete Fourier Transform (Spectral) Test: Checks for periodic features (peaks in the frequency domain) that would indicate a deviation from randomness. The p-value must be greater than 0.01 to pass.
- To pass the NIST STS test:
- the P-value of each test must be greater than 0,01
- The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately = 8 for a sample size = 10 binary sequences.
- The minimum pass rate for the random excursion (variant) test is approximately = 1 for a sample size = 2 binary sequences.
- The random excursion and the random excursion variant require a minimum of 1,000,000 bits for each sequence.

Our experimental analysis employed the NIST Statistical Test Suite (STS) as the evaluative benchmark to assess the efficacy of the key generation algorithm. The suite of tests applied included the Frequency, Block Frequency, Cumulative Sums, Runs, Longest Run of Ones in a Block, Matrix Rank, and the Fast Fourier Transform (FFT) tests. The compiled results are systematically presented in Table I. Notably, our algorithm demonstrated a robust performance, successfully passing all seven of the aforementioned NIST STS tests. For the purpose of this analysis, we standardized the bitstream length to 10 for the datasets procured from five distinct microcontrollers. The criterion for passing each individual test was established such that the P-Value must be equal to or greater than 0.01. The uniform success across this suite of tests underscores the reliability of our algorithm in generating keys that exhibit the requisite randomness and uniformity for security applications.

B. Performance Analysis

To assess the performance of our newly developed key generation algorithm, we conducted a series of meticulously planned experiments implementations across a range of IoT device types.

1) Execution time and memory usage: To assess the performance analyses, we conducted the first implementation using two ESP32 microcontrollers (see Fig. 5). The first microcontroller was programmed to execute a very basic version of the Diffie-Hellman key exchange with small primitive numbers, generating a set of 100 keys as a reference for traditional cryptographic key exchange mechanisms. Concurrently, the second microcontroller was tasked with generating an equivalent set of 100 keys, utilizing our proprietary algorithm. This parallel generation scheme was not only intended to provide a direct performance comparison between the traditional Diffie-Hellman approach and our novel solution but also to demonstrate the practical applicability and efficiency of our algorithm in a real-world IoT environment. Table II show the subsequent comparison and analysis between the two protocols and also the proposed schemes by [10], [19], [20], [21], [22].

The examination of the execution time, as delineated in Table II, reveals that our algorithm exhibits superior suitability for IoT devices that operate under stringent constraints of energy, memory, and performance. This efficacy positions our algorithm as an optimal choice for deployment in resource-limited environments where such considerations are paramount.

The second implementation was designed in two Raspberry Pi Zero devices (see Fig. 6) to analyze and compare the execution time Table III and the memory usage Table IV of our algorithm against a strong and secure version of Diffie-Hellman key exchange since the esp32 was crashed when trying to execute a strong version of Diffie-Hellman.

Our experimental evaluation showcases the performance of our algorithm compared to a Strong Diffie-Hellman implementation across different metrics: execution time and memory usage. These experiments were conducted to generate varying quantities of keys (10, 100, and 1000) with different key lengths (32, 64, 128, and 256 bits).

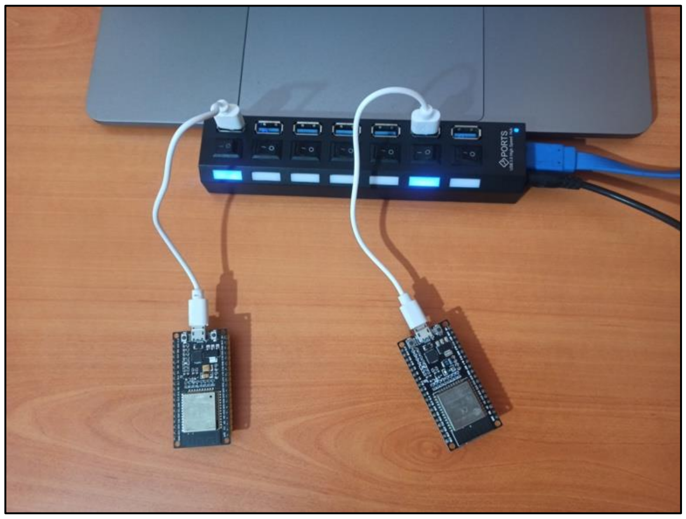
Fig. 5. Implementation scenario for execution time evaluation in ESP-WROOM-32 MCU devices.

TABLE II. EXECUTION TIME ANALYSIS

Algorithm	Execution time in seconde
Diffie-Hellman (basic version)	8,23
[10]	3,10
[19]	2,1
[20]	1,22
[21]	1,05
[22]	0,37
Our Proposed Algorithm	0,16

TABLE III. EXECUTION TIME RESULTS FOR GENERATING 10, 100, AND 1000 KEYS WITH VARYING LENGTHS

Key length in bit	Exec. time in ms for 10 keys generation		Exec. time in ms for 100 keys generation		Exec. time in for 1000 keys generation	
	Our algorithm	Strong DH	Our algorithm	Strong DH	Our algorithm	Strong DH
32 bits	52,614	93,740	369,399	647,025	3290,118	5128,065
64 bits	54,076	139,612	370,038	1772,165	3298,031	17316,150
128 bits	55,109	756,192	374,688	6754,179	3324,974	70802,648
256 bits	58,113	5136,845	391,013	40546,82	3441,168	Device bugs

TABLE IV. MEMORY USAGE (IN KILOBYTES) RESULTS FOR GENERATING 10, 100, AND 1000 KEYS WITH VARYING LENGTHS

Key length in bit	Mem. usage in KiB for 10 keys generation		Mem. usage in kiB for 100 keys generation		Mem, usage in KiB for 1000 keys generation	
	Our algorithm	Strong DH	Our algorithm	Strong DH	Our algorithm	Strong DH
32 bits	0,718	0,994	1,160	1,772	1,173	1,861
64 bits	0,855	0,951	1,186	1,815	1,195	1,904
128 bits	0,901	2,015	1,296	2,827	1,310	1,173
256 bits	1,128	3,590	1,570	4,954	1,583	Device bugs

a) Execution time: The execution time is critical in environments where quick key generation is essential for maintaining operational efficiency and responsiveness. Our algorithm demonstrates a consistently lower execution time across all tested scenarios, significantly outperforming the strong Diffie-Hellman algorithm version, especially as the number of keys and key lengths increase. Notably, for 1000 keys generation, our algorithm maintained a practical execution time even at higher key lengths (32 bits: 3290.118 ms, 64 bits: 3298.031ms, 128 bits: 3324.974 ms), whereas the strong Diffie-Hellman algorithm version showed a drastic increase in execution time, peaking at 70802.648 ms for 128 bits, a device bug occurred during 256-bit key generation for strong DH, limiting data availability. This discrepancy highlights the efficiency of our algorithm in scenarios requiring large volumes of key generations.

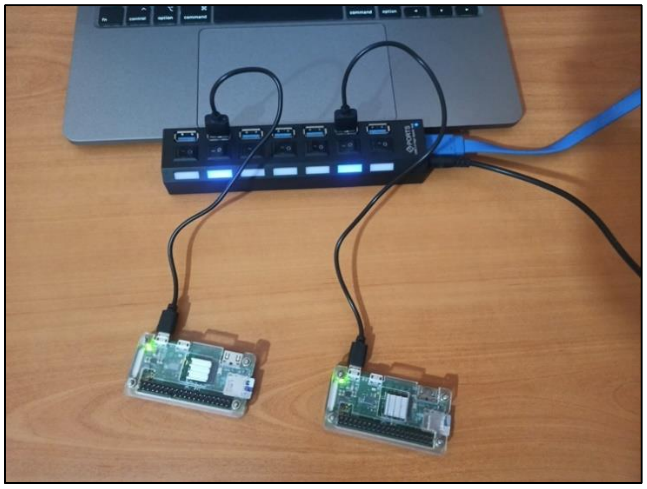
Fig. 6. Implementation scenario for execution time and memory usage evaluation in Raspberry Pi Zero devices.

b) Memory usage: Memory usage is another vital aspect, particularly for IoT devices with limited resources. Our algorithm exhibits significantly lower memory consumption across all tests. For instance, generating 100 keys of 256 bits only required 1,570 KiloBytes of memory for our algorithm, compared to 4,954 KiloBytes for Strong DH. This reduced memory footprint underscores our algorithm's suitability for resource-constrained environments, enabling secure communications without compromising device performance.

2) Scalability: In our study, we prioritized the implementation of our algorithm on actual hardware over simulation to capture the nuances of real-world execution. This approach, leveraging physical devices, allowed us to obtain genuine performance metrics, reflecting the algorithm's operational efficiency in tangible IoT environments. While this methodology underscores the practical applicability and advantages of our solution under real operational conditions, it naturally constrains our ability to extensively evaluate the scalability of our algorithm across a vast network of devices. Recognizing this limitation, our future work will be dedicated to implementing our algorithm within a simulator. This strategic shift will enable us to rigorously assess the scalability of our solution, facilitating the evaluation over a considerably larger array of virtual devices. Such simulated environments will provide invaluable insights into the performance impacts and scalability potential of our algorithm, offering a comprehensive understanding of its efficacy in expansive IoT ecosystems. This dual approach grounding initial validation in physical implementations before extending evaluations through simulations strikes a balance between practical verification and extensive scalability testing, ensuring our solution is both robust and adaptable to the diverse needs of IoT infrastructures.

3) Interoperability with IoT systems and protocols: In addressing the interoperability of our algorithm with existing Internet of Things (IoT) systems and protocols, it's crucial to highlight that our solution is designed for seamless integration at the application layer. This strategic choice enables our algorithm to function effectively without necessitating alterations to the underlying layers of the network architecture. Implementing our key generation mechanism at this level offers several distinct advantages:

- Flexibility: By situating the algorithm at the application layer, it can be easily deployed across a wide range of IoT platforms and devices, regardless of their specific network configurations or protocols employed at lower layers.
- Ease of deployment: Application layer implementation allows for the introduction of our security solution without the need to modify existing network infrastructures. This significantly reduces the complexity and cost associated with deploying enhanced security measures.
- Versatility in application: Given the diverse nature of IoT applications, embedding our algorithm at the application layer ensures that it can be tailored to meet the unique security requirements of various use cases, from smart home devices to industrial IoT applications.
- Compatibility: This approach maintains compatibility with existing standards and protocols at the transport and network layers, ensuring that our algorithm can be integrated into existing IoT ecosystems without interoperability issues.

Recognizing the importance of widespread protocol support in enhancing the utility and adoption of our algorithm, our future research will focus on integrating our key generation mechanism with widely used IoT transport protocols, such as MQTT. MQTT (Message Queuing Telemetry Transport) is renowned for its lightweight and efficient communication capabilities, making it a staple in IoT deployments. By embedding our security solution within protocols like MQTT, we aim to provide end-to-end security in IoT communications, ensuring data integrity and confidentiality without sacrificing performance. This forward-looking approach not only broadens the applicability of our algorithm but also aligns with the evolving security needs of the IoT landscape, promising enhanced protection for devices and data in an increasingly connected world.

VI. Conclusion and Future Scope

In this article, we introduced a novel key generation algorithm designed to circumvent the need for explicit key exchange, a requirement inherent in many established protocols, such as the Diffie-Hellman technique and others referenced in the related work section. Our algorithm is particularly well-suited for IoT devices due to its non-reliant nature on explicit key exchanges and its foundation upon a pre-shared matrix. This matrix not only demands an insubstantial memory footprint but also possesses the capability to generate a vast array of unique keys, providing a new key for each individual exchange to maintain security integrity.

However, while our algorithm marks a significant stride towards optimizing key generation for IoT devices, it is not without areas necessitating refinement. A critical aspect that we aim to enhance in our future work is the seed synchronization process. Ensuring robust synchronization in the seed selection, which initiates the key generating sequence, is crucial to thwart attacks targeted at the desynchronization of this phase. Furthermore, we plan to deploy our algorithm within an authentic IoT environment to thoroughly evaluate its resilience against various security threats, including Man-In-The-Middle (MITM) attacks, and to accurately measure the energy consumption of IoT devices engaged in secure communication facilitated by our algorithm.

Our ambition is for this algorithm to stand as a viable alternative to the Diffie-Hellman protocol, particularly in applications of IoT devices where resource constraints are a critical consideration. This is especially pertinent in conjunction with steganographic algorithms, such as those we have proposed in our prior work. Through the continued development and rigorous testing of our algorithm, we anticipate contributing a robust and energy-efficient solution to the field of IoT security, enhancing the safe and private exchange of information in an increasingly interconnected world.

References

[1] Munirathinam, S. (2020). Chapter Six - Industry 4.0: Industrial Internet of Things (IIOT). *Adv. Comput.*, 117, 129-164. https://doi.org/10.1016/bs.adcom.2019.10.010.

[2] Venkatasubramanian, M., Lashkari, A., & Hakak, S. (2023). IoT Malware Analysis Using Federated Learning: A Comprehensive Survey. IEEE Access, 11, 5004-5018. https://doi.org/10.1109/ACCESS.2023.3235389.

[3] Sherali Zeadally, Ashok Kumar Das, Nicolas Sklavos, Cryptographic technologies and protocol standards for Internet of Things, Internet of Things,2021, vol. 14,2021, 100075, ISSN 2542-6605. https://doi.org/10.1016/j.iot.2019.100075.

[4] Alhajjar, E., & Lee, K. (2022). The U.S. Cyber Threat Landscape. European Conference on Cyber Warfare and Security. https://doi.org/10.34190/eccws.21.1.197.

[5] Rimani, R., said, N., Pacha, A., & Ozer, O. (2021). Key exchange based on Diffie-Hellman protocol and image registration. Indonesian Journal of Electrical Engineering and Computer Science, 21, 1751-1758. https://doi.org/10.11591/IJEECS.V21.I3.PP1751-1758.

[6] Hegde, S., Srivastav, S., & Ks, N. (2022). A Comparative study on state of art Cryptographic key distribution with quantum networks. 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT), 1-7. https://doi.org/10.1109/GCAT55367.2022.9971870.

[7] Dhanda, S.S., Singh, B. & Jindal, P. Lightweight Cryptography: A Solution to Secure IoT. Wireless Pers Commun 112, 1947–1980 (2020). https://doi.org/10.1007/s11277-020-07134-3.

[8] Nour-El Aine, Y., Leghris, C. (2021). Ensuring Smart Agriculture System Communication Confidentiality Using a New Network Steganography Method. In: Boumerdassi, S., Ghogho, M., Renault, É. (eds) Smart and Sustainable Agriculture. SSA 2021. Communications in Computer and Information Science, vol 1470. Springer, Cham. https://doi.org/10.1007/978-3-030-88259-4_2.

[9] Smart, N.P. (2016). Historical Stream Ciphers. In: Cryptography Made Simple. Information Security and Cryptography. Springer, Cham. https://doi.org/10.1007/978-3-319-21936-3_10.

[10] Yu B, Li H. Anonymous authentication key agreement scheme with pairing-based cryptography for home-based multi-sensor Internet of Things. International Journal of Distributed Sensor Networks. 2019;15(9). doi:10.1177/1550147719879379.

[11] Ali S, Humaria A, Ramzan MS, et al. An efficient cryptographic technique using modified Diffie–Hellman in wireless sensor networks. International Journal of Distributed Sensor Networks. 2020;16(6). doi:10.1177/1550147720925772.

[12] Rimani, R., said, N., Pacha, A., & Ozer, O. (2021). Key exchange based on Diffie-Hellman protocol and image registration. Indonesian Journal of Electrical Engineering and Computer Science, 21, 1751-1758. https://doi.org/10.11591/IJEECS.V21.I3.PP1751-1758.

[13] Takaoğlu, M., Özyavas, A., Ajlouni, N., & Takaoglu, F. (2023). Highly Secured Hybrid Image Steganography with an Improved Key Generation and Exchange for One-Time-Pad Encryption Method. Afyon Kocatepe University Journal of Sciences and Engineering. https://doi.org/10.35414/akufemubid.1128075.

[14] Park, S., Kim, K., Kim, K., & Nam, C. (2022). Dynamical Pseudo-Random Number Generator Using Reinforcement Learning. *Applied Sciences*. https://doi.org/10.3390/app12073377.

[15] Sathya, K., Premalatha, J., & Rajasekar, V. (2021). Investigation of Strength and Security of Pseudo Random Number Generators. IOP Conference Series: Materials Science and Engineering, 1055. https://doi.org/10.1088/1757-899X/1055/1/012076.

[16] Karawia, A. (2019). Image encryption based on Fisher-Yates shuffling and three dimensional chaotic economic map. *IET Image Process.*, 13, 2086-2097. https://doi.org/10.1049/IET-IPR.2018.5142.

[17] Servodio, S., & Li, X. (2021). An Efficient Shuffle-Light FFT Library. 2021 IEEE International Performance, Computing, and Communications Conference(IPCCC),1-10. https://doi.org/10.1109/IPCCC51483.2021.9679431.

[18] Aoun, O., & El Afia, A. (2018). Time-dependence in multi-Agent MDP applied to gate assignment Problem. Int. J. Adv. Comput. Sci. Appl, 9, 331-340, https://doi: 10.14569/IJACSA.2018.090247.

[19] Scott M. Authenticated ID-based key exchange and remote log-in with simple token and PIN number. Cryp- tology, 2002, pp.1–9, https://eprint.iacr.org/2002/164.pdf

[20] Wu F, Xu L, Kumari S, et al. A new and secure authenti- cation scheme for wireless sensor networks with formal proof. Peer Peer Netw Appl 2015; 10(1): 1–15.

[21] Xiong L and Peng DY. A lightweight anonymous authen- tication protocol with perfect forward secrecy for wireless sensor networks. Sensors 2017; 17(11): 2681.

[22] Srinivas J and Mukhopadhyay S. Secure and efficient user authentication scheme for multi-gateway wireless sensor networks. Ad Hoc Netw 2017; 54: 147–269.

[23] A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, National Institute of Standards and Technology Special Publication 800-22 Revision 1a, https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf

[24] Balfaqih M, Balfagih Z, Lytras MD, Alfawaz KM, Alshdadi AA, Alsolami E. A Blockchain-Enabled IoT Logistics System for Efficient Tracking and Management of High-Price Shipments: A Resilient, Scalable and Sustainable Approach to Smart Cities. Sustainability. 2023; 15(18):13971. https://doi.org/10.3390/su151813971.

[25] Almohammedi, A.A., Balfaqih, M., Nahas, S., Bokhari, A., Alqudsi, A. (2023). Design and Implementation of IoT-Enabled Intelligent Fire Detection System Using Neural Networks. In: Yang, Y., Wang, X., Zhang, LJ. (eds) Artificial Intelligence and Mobile Services – AIMS 2023 . AIMS 2023. Lecture Notes in Computer Science, vol 14202. Springer, Cham. https://doi.org/10.1007/978-3-031-45140-9_6.

[26] M. Balfaqih, Z. Balfagih, A. A. Almohammedi and K. M. Alfawaz, "A Smart and Privacy-Preserving Logistics System Based on IoT and Blockchain Technologies," *2023 1st International Conference on Advanced Innovations in Smart Cities (ICAISC)*, Jeddah, Saudi Arabia, 2023, pp. 1-5, doi: 10.1109/ICAISC56366.2023.10255090.