Hybrid Lightweight Encryption Algorithms for Internet of Things (IoT)

Rusul H. Altaie[1], Rihab Habeeb Sahib[2], Aseel Hamoud Hamza[3] and Ghadeer Abbas Hussein[4]
[1]*Department of Arabic Language, College of Arts, University of Babylon, 51001 Hilla, Iraq*
[2]*Department of Archaeology, College of Arts, University of Babylon, 51001 Hilla, Iraq*
[3]*Department of Public Law, College of Law, University of Babylon, 51001 Hilla, Iraq*
[4]*College of Arts, University of Babylon, 51001 Hilla, Iraq*
rusul.jasem@uobabylon.edu.iq, art.rehab.habeeb@uobabylon.edu.iq, assel.hamod@uobabylon.edu.iq, art.ghadeer.abbas@uobabylon.edu.iq

Keywords: Present Encryption (PRES.), Internet of Things (IoTs), Sensor Devices, Arduino UNO, Security Mechanism.

Abstract: IoT security has become increasingly critical in this situation, along with the widespread usage. Therefore, a method must be created to protect these devices and data in buildings and institutions from hackers. In this article, we mentioned hybrid algorithms to solve this problem. To guarantee that the data it transmits and receives is secure and unaltered in any manner, Arduino and Raspberry Pi, as IoT hardware platforms, also need security advancements. For embedded systems like Arduino with limited memory and processing capacity, employing a standard block cipher to secure transmission is computationally expensive. This paper offers a safety system to protect IoT devices in a set of rooms inside a building and enterprises. This study offers a safe approach to protect the hardware equipment from intruders and accidents in buildings. In this article, a various group of sensors are associated with Arduino to collect data. PRESENT and SPECK algorithms are the basic encryption algorithms used for encrypting data collected by sensors in several rooms within the building. In this study, PRESENT and SPECK are used in the Interleaved method and a smaller number of rounds. The data of encrypted data is again encoded in Raspberry Pi by the SPECK algorithm in a nested manner and then encrypted by PRESENT and sent to the computer through Message Queuing Telemetry Transport protocol to distribute data, and then sent to the cloud to increase security.

1 INTRODUCTION

Internet of Things (IoT) is called as a system of large numbers of physical things. These things are set in with many sensors, network connections, and electronics. These things are allowed to gather and interchange data. The Internet of Things (IoT) makes these objects sensible, controlled, and monitored remotely across current network infrastructures, it will then create more direct integrations between computerized systems and the physical world to improve efficiency, accuracy, and economic (Rondon L. P.) [1]-[3].

Currently, many Internet of Things devices, in addition to transmitting data, are vulnerable to hacking and theft. Therefore, a way must be created to protect these devices and data in buildings and institutions from hackers. In this paper, we will mention a method or algorithm to solve this problem. There is no standard defined working architecture across the board. The complexity of architectural layers depends on the specific tasks. The four layers of architecture are the standard and most widely accepted form; there are the perception, network, processing, and application layers [2]-[4] as shown in Figure 1.

Figures 2-6 are used to collect data from sensors, Figure 2 displays temperature and humidity that are implemented to check certain temperatures and humidity inside a room.

Figure 3 shows flame sensor is applied to check whether a room has a fire or not. The system checks if there fire is detected, and an alert is sent to the administrator.

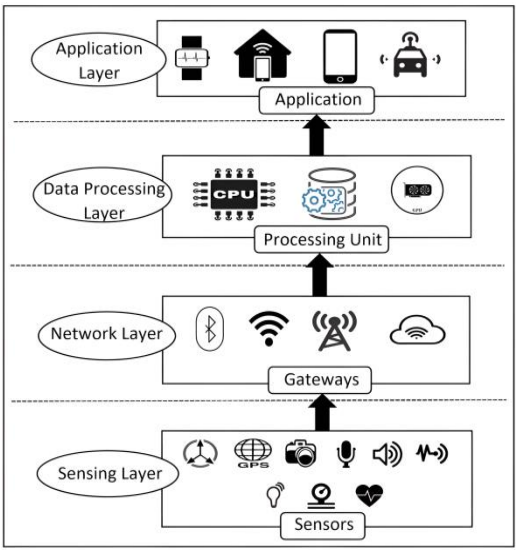


Figure 1: IoT architecture layers.

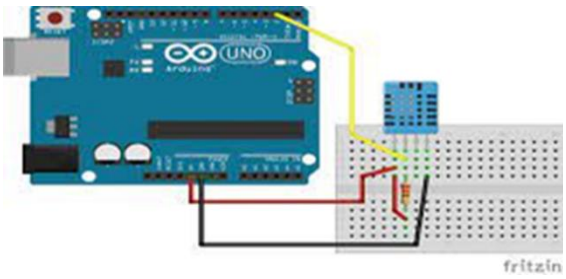


Figure 2: DHT11 sensor.

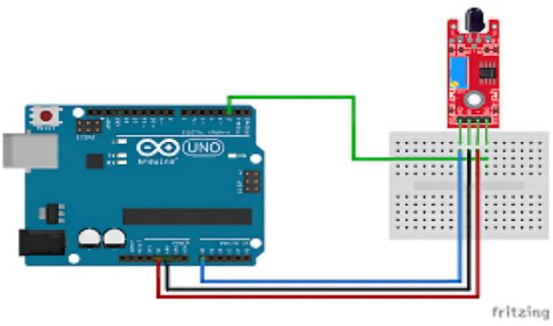


Figure 3: Flame sensor.

A light sensor is applied to read the light level inside a certain place in the room and send a signal of zero if the level is low and 1 if high, as shown in Figure 4.

A sound sensor is used to detect the intensity of sound in a room, as shown in Figure 5.

Figure 6 displays a door sensor is applied to sends a signal of 0 or 1 and turns on the alarm when the door of the area to which it is attached is opened or not.

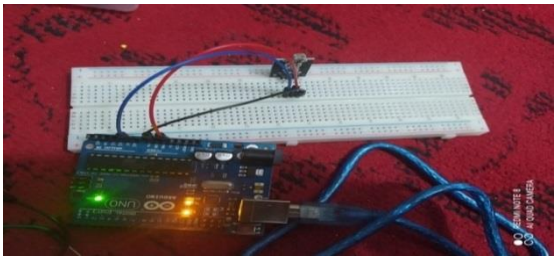


Figure 4: Light sensor.

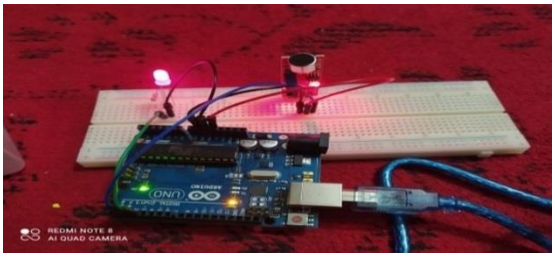


Figure 5: Sound sensor.

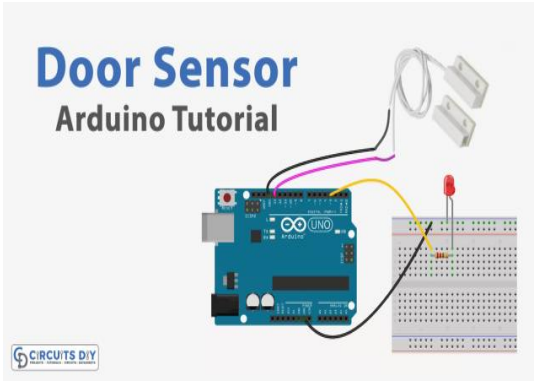


Figure 6: Door sensor.

The Ethernet chip (Wiznet W5100) serves as the foundation for the Arduino Shield. The board of Arduino is linked to the internet using this Ethernet shield. A network IP stack is provided by this chip, W5100. It can support both TCP and UDP, and it can handle four socket links simultaneously. The Arduino Shield of Ethernet and board of Arduino are connected using a lengthy wire wrap header. This preserves the pin configuration and enables the addition of a further shield on top. As explained in Figure 7, the Micro-SD slot is located on the extended shield of Ethernet and can be utilized as storing to keep records over the internet.

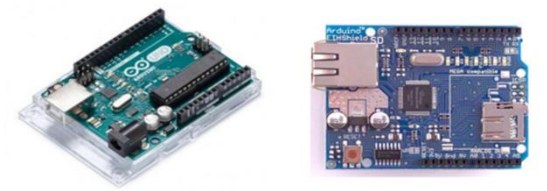


Figure 7: Ethernet defend Arduino UNO.

2 PRESENT ALGORITHM

PRESENT algorithm (PRE) is an instance of an ultra-lightweight block cipher that has a fixed block size. The main goal of designing the present cipher was simplicity. There are many characteristics of a PRESENT algorithm. This cipher is implemented in hardware; the applications of this cipher only require moderate levels of security, and the implementations of the PRESENT algorithm might be optimal for performance or space without real influence [1], [3]-[6].

Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christ of Paar, Axel Poschmann, Mathew J. B. Robhaw, Yanick Seurin, and C. Vikelsoe designed the PRESENT algorithm. Encryption key consists of two keys are 80-bit (10 hex characters) and 128-bit (16 hex characters) [7]-[8]. It uses a 64-bit block size and has a 16-value S-box, which translates 16 4-bit values. It includes 31 rounds and each round involves an XOR operand to create round key Ki, K32 uses for postwhitening, a linear permutation, and a nonlinear substitution layer. In each round, nonlinear substitution uses Sbox, which is run 16 times in parallel [9],[10],[11] as shown in the foolowing (1):

$$\text{AddRoundKey involves } bj = bj \oplus kj, \qquad (1)$$

The sBoxlayer comprises 16 words of 4 bits each (w0…w15). The output corresponds to the updated value s[wi]. The table shows the substitution values indexed by the hexadecimal input (0–F). Elements C, 5, 6, B, 9, 0, A, D, 3, E, F, 8, 4, 7, 1, 2 represent the mapping of input values to output sBox entries..

v	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

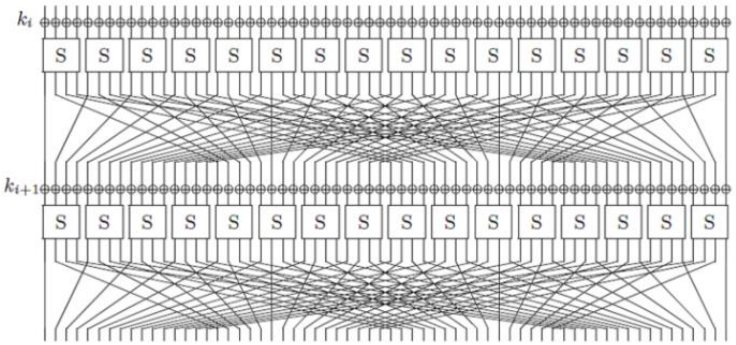
Figure 8: Structure of the sBoxlayer.

A player is the permutation layer of the present algorithm.

Figure 8 shows that the key schedule of the PRESENT algorithm can accept keys with 80 or 128 bits. The user-supplied key is represented as k79k78... k0 and is kept in a key register K. According to (1), in round (i), the 64-bit round key Ki = k63k62... k0 is made up of the 64 leftmost bits of register T's current contents as in Figure 8. As a result, at round (i) we have [12]-[15]:

$$T_i = T_{63}T_{62\ldots}T_0 = T_{79}T_{78}\ldots T_{16} \qquad (2)$$

The PRESENT algorithm utilized a round key procedure. Each of the 31 rounds uses a bitwise XOR process. The S-Box procedure is utilized in nonlinear layers. Equivalent operations are performed 16 times for each round. Likewise, the PRESENT algorithm cipher will be merged with another encryption algorithm to increase the safety of hybrid encryption [16]-[18]. Figure 9 displays a block diagram of the PRESENT cipher.



i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Figure 9: S/P network of PRES. Algorithm.

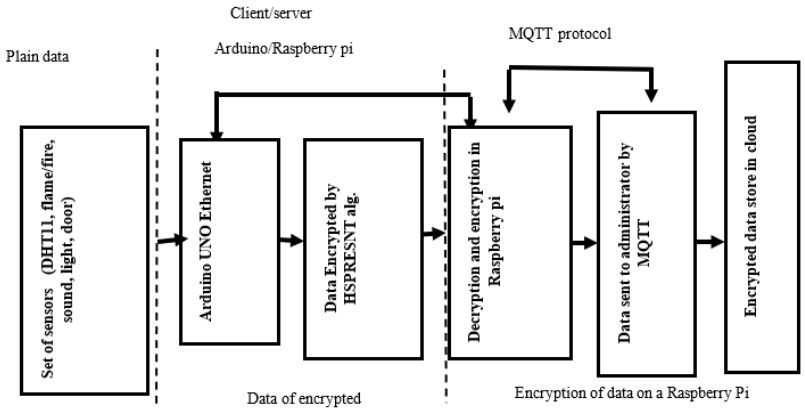


Figure 10: Structural of system design.

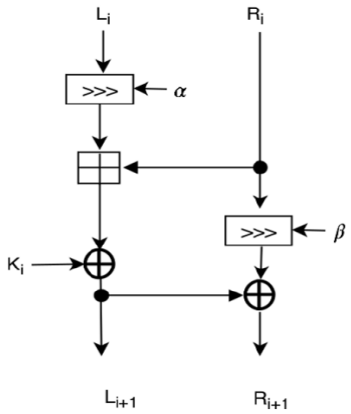


Figure 81: A representation for the general round of SPECK cipher.

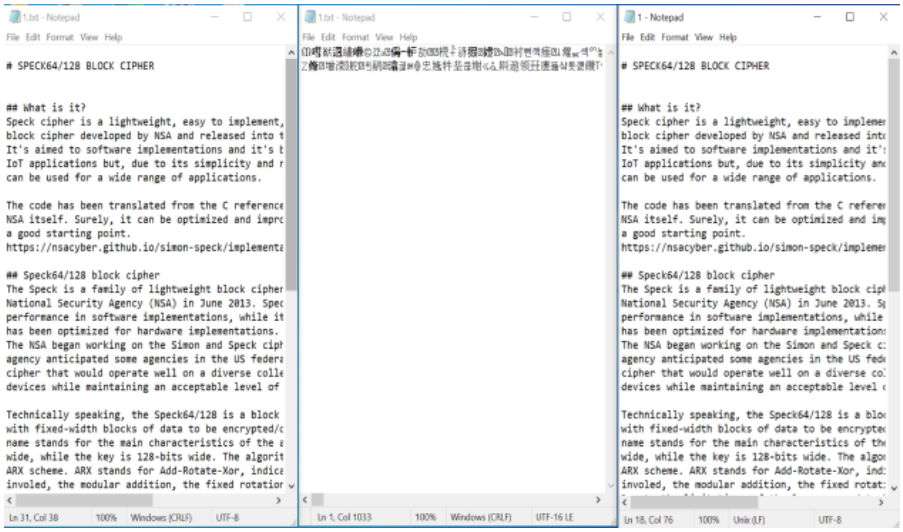


Figure 12: Data of encryption and decryption.

3 RESEARCH METHOD

In this paper, we proposed a hybrid PRESENT & SPECK cipher algorithm. The proposed algorithms were used mostly due to their ultra-lightweight encryption and rapid speed. Block cipher would be the primary structure, and it would primarily be based on the current structure of the cipher. The 20-round keys of the PRESENT algorithm replace the 31-round keys. Figure 10 clarifies the proposed work of this paper. The data of the sensors used are humidity (humid) and temperature (temp.) (DHT11), light(lit.), door(dor.), sound(snd.), and flame(flm.) sensors data. These data are encrypted in Arduino using Hybrid SPECK-PRESENT (HSPRESENT) encryption. To safeguard data transmission over network channels, the encryption process involves transforming plaintext into a cryptographically secure form. First, the data is encrypted by PRES. Algorithm. Then, the SPECK algorithm is used to encrypt the data again. Then, encrypted data is encrypted by PRESENT algorithm later. The proposed system for sending safe data across the network incorporates the compact protocol MQTT. Encrypted information was sent to the administrator by protocol and then sent to the cloud. This action prevents illegal users from editing and altering the original data that the sender sent over the network.

Decryption is the process of converting the received ciphertext back into the original plaintext. It employs inverse techniques like S-box and P-layer processes.

The encryption procedure is the conversion of the original text to a ciphertext to defend data transmission over the internet .The proposed encryption HSPRESENT algorithm is described as follows:

```
Algorithm 1: Proposed Encryption Process.
Input: original text.
Output: ciphertext.
Steps:
 1) generate RoundKeys();
 2) padding (stat, Ke);
 3) for k=1 to 15 do;
 4) XOR (stat, Ke);
 5) AddRoundKey(stat,ki);
 6) S_box layer1(stat);
 7) P_layer1 (stat);
 8) XOR (stat, Ke)
 9) AddRoundKey(stat,ki);
 10) S_box layer2(stat);
 11) P_layer2 (stat);
 12) XOR (stat, Ke);
 13) Apply SPECK algorithm with 8 rounds;
 14) End for;
 15) XOR (stat, Ke16);
 16) Update addroundkey(stat,31);
 17) End.
```

The original text length is 64 bits and is padded to 128 bits, and is shifted 16 bits to the left based on the first produced key. 128 bits is divided into two parts, each of which is 64 bits. Then the original text is encrypted by the XOR function. In a round, the decryption process is the translation of the ciphertext into the original text. It is the reverse process of encryption. The decryption PRESENT algorithm is defined as follows:

```
Algorithm 2: Decryption process.
Input: ciphertext.
Output: original text.
Steps:
 1) generate RoundKeys();
 2) padding (stat, Ke);
 3) XOR (stat, Ke16);
 4) for j= 15 to 1 do;
 5) Apply decryption SPECK algorithm with 8 rounds;
 6) XOR(stat, kj);
 7) Inverse S_box layer2(stat);
 8) Inverse P_layer2(stat);
 9) XOR (stat, Ke);
 10) Inverse S_box layer1(stat);
 11) Inverse P_layer1(stat);
 12) XOR (stat, Ke);
 13) End for;
 14) AddRoundKey(stat,Kj);
 15) End.
```

Where the S-box, Inverse of S-box, Layer P, and Inverse Layer P are generated using the S-box and random number generation. In the beginning, a novel key is produced and XOR process with the ciphertext. Then, the ciphertext is decrypted by the XOR function with a round key. After that, we custom Inverse S_box to substitute a binary string using the Inverse of S_box table before transmitting it to Inverse Layer P. Inverse Layer P is used based on the Inverse permutation table.

There is a loop of 15 rounds until the latter step. The latter state is in the procedure to generate the original text. The SPECK algorithm round was used in the HSPRESENT algorithm is as shown in Figure 11. After that, we use S_box to replace a binary string using the S_box table before transferring it to Layer P. Layer P is used according to a table of permutations. There are 15 rounds till the last step. The novel key is encoded with the

latter state of the cycle by XOR to generate the ciphertext.

4 RESULTS

The algorithm was implemented in C++ and Python. The C++ language is implemented in Arduino, and Python executes the work in the Raspberry Pi. The kind of data that is indicated for suggested method is data that is gathered by a collection of sensor devices. Table 1 shows the data which is collected from different types of sensor devices in the environment.

Table 1: Examples of data sensors for 3 rooms.

Room no.	Temp. Sensor	Humid. Sensor	Flm. Sensor	Snd. sensor	Dor. sensor	Lit .sensor
ARD1	27.15	0.66	0	0	1	0
ARD2	66.16	0.49	1	0	1	1
ARD3	26.18	0.66	1	0	0	0

The Python language is used to create and code the suggested algorithm. The type of data that is advised for this intended approach is the data collected by sensors. For analyzing the randomness characteristics of cryptography algorithms, many statistical tests are available. The suggested method's randomness was compared to the current encryption algorithm using the well-known NIST statistical test suite. We have tested with different keys and block sizes (n = 64) of 1000 bits long. The consequences of the NIST arithmetic assessments are exhibited in Table 2.

The tests of NIST demonstrate whether or not a specific proportion of the arrangement is random based on the value of significance (α), with the default value (0.01. As a result, the sequence would be regarded as random if the R-value was less than 0.01; otherwise, an R-value greater than 0.01 would show that the structure is not random. As indicated in Table 2, both the suggested approach and the existing encryption technique pass every NIST test. The R-values of the suggested algorithm are greater than the R-values of the PRESENT method for the majority of the NIST tests, according to the findings. As a result, the sequence produced by the suggested method has a sizable amount of unpredictability. On the other hand, the suggested method performs effectively and executes quickly. The test was executed on a processor running Windows 10 by an Intel Core i7 CPU clocked at 2.4 GHz and 8 G RAM, Arduino, and Raspberry. Text is the preferred data type, and each block in a text file has 64 bits. The HSPRESENT algorithm took 8 milliseconds to perform with a similar dimension of data as the proposed technique did in 5 milliseconds. Consequently, the execution time of the suggested technique satisfies the lightweight algorithm's speed. The suggested approach increases complexity while maintaining a low computational speed. Figure 12 clarifies the data before and after encryption.

Table 1: Experiments of the NIST statistical.

NIST Assessments for efficiency	(PRES.) Algorithm	Proposed Algorithm
Frqeuency test	0.121	0.531
Block Frqeuency	0.796	0.864
Cumulative Sums	0.418	0.508
Runs	0.770	0.830
Longest Run	0.673	0.751
Rank	0.111	0.408
Non Overlapping	0.0051	0.210
Overlaping Template	0.221	0.538
Universal test	0.774	0.902
Approximate Entropy	0.654	0.731
Rondom Excusions	0.500	0.860
Random Excursions Variants	0.0018	0.208
Serial test	0.451	0.711
Linearr Complexity	0.021	0.198

The tests of NIST demonstrate whether or not a specific proportion of the arrangement is random based on the value of significance (α), with the default value (0.01. As a result, the sequence would be regarded as random if the R-value was less than 0.01; otherwise, an R-value greater than 0.01 would show that the structure is not random. As indicated in Table 2, both the suggested approach and the existing encryption technique pass every NIST test. The R-values of the suggested algorithm are greater than the R-values of the PRESENT method for the majority of the NIST tests, according to the findings. As a result, the sequence produced by the suggested method has a sizable amount of unpredictability. On the other hand, the suggested method performs effectively and executes quickly. The test was executed on a processor running Windows 10 by an Intel Core i7 CPU clocked at 2.4 GHz and 8 G RAM, Arduino, and Raspberry. Text is the preferred data type, and each block in a text file has 64 bits. The HSPRESENT algorithm took 8 milliseconds to perform with a similar dimension of data as the proposed technique did in 5

milliseconds. Consequently, the execution time of the suggested technique satisfies the lightweight algorithm's speed. The suggested approach increases complexity while maintaining a low computational speed. Figure 12 clarifies the data before and after encryption.

5 CONCLUSIONS

This paper introduces a hybrid lightweight cryptographic algorithm to preserve sensor data through transition across the network. This system is designed for monitoring and protecting Internet of Things devices and enterprise buildings from accidents. Lightweight algorithms of encryption algorithms are more useful and frequently used with IoT tenders. This study presented HSPRESENT lightweight algorithms which represent an effective method to improve the security of devices and data. Cryptography is the main means of communication in the field of information security. While interacting with numerous devices on the internet, cryptography is required. Contrarily, constraint devices have constrained resources, making standard cryptography inapplicable to them. Constraint devices usually use symmetric key cryptography, which encrypts and decrypts data using a single key. Contrarily, constraint devices can be solved using fast cryptographic techniques. It is possible to apply encryption algorithms in enterprise building security and protect data from intruders. This study can be applied to protect the hardware equipment from intruders and accidents in buildings as, hospitals, schools, and residential buildings.

REFERENCES

[1] L. P. Rondon, L. Babun, A. Aris, K. Akkaya, and A. S. Uluagac, "Survey on enterprise Internet-of-Things systems (E-IoT): A security perspective," Ad Hoc Netw., vol. 12, pp. 102-111, 2022.

[2] P. S. Velmurugan, S. Sridhar, and E. Gotham, "An advanced and effective encryption methodology used for modern IoT security," Mater. Today: Proc., vol. 81, pp. 389-394, 2023.

[3] I. Lee, "Internet of Things (IoT) cybersecurity: literature review and IoT cyber risk management," Future Internet, vol. 12, no. 9, p. 157, 2020, doi: 10.3390/fi12090157.

[4] L. Tawalbeh, F. Muheidat, M. Tawalbeh, and M. Quwaider, "IoT privacy and security: challenges and solutions," Appl. Sci., vol. 10, no. 12, p. 4102, 2020, doi: 10.3390/app10124102.

[5] S. B. Sadkhan and Z. Salam, "Security and privacy in Internet of Things—status, challenges," in Proc. 4th Int. Iraqi Conf. Eng. Technol. Appl. (IICETA), 2021, doi: 10.1109/IICETA51758.2021.9717785.

[6] A. N. Lone, S. Mustajab, and M. Alam, "A comprehensive study on cybersecurity challenges and opportunities in the IoT world," Secur. Privacy, vol. 6, no. 6, 2023.

[7] S. Sadek and A. Rowayda, "Hybrid energy aware clustered protocol for IoT heterogeneous network," Future Comput. Informatics J., vol. 3, no. 2, pp. 166-177, 2018.

[8] R. H. Altaie and H. K. Hoomod, "Artificial intelligent management for Internet of Things: a review," in Proc. 4th Int. Conf. Current Res. Eng. Sci. Appl. (ICCRESA), pp. 179-184, 2022, doi: 10.1109/ICCRESA.2022.00035.

[9] M. Sruthi and R. Rajasekaran, "Hybrid lightweight signcryption scheme for IoT," Open Comput. Sci., vol. 11, no. 1, pp. 391-398, 2021.

[10] H. K. Hoomod and H. Haider, "Hybrid SPECK encryption algorithm for Internet of Things (IoT)," in Lecture Notes on Data Engineering and Communications Technologies, pp. 317-326, 2024, doi: 10.1007/978-3-031-59711-4_27.

[11] A. Heidari and M. A. J. Jamali, "Internet of Things intrusion detection systems: a comprehensive review and future directions," Cluster Comput., vol. 26, no. 6, pp. 3753-3780, 2023.

[12] I. Kuzminykh, M. Yevdokymenko, and V. Y. Sokolov, "Encryption algorithms in IoT: security vs lifetime," Soc. Sci. Res. Netw., pp. 1-21, 2023.

[13] M. Rana, Q. Mamun, and R. Islam, "Lightweight cryptography in IoT networks: a survey," Future Gener. Comput. Syst., vol. 129, pp. 77-89, 2022.

[14] A. Abdullah and A. Shapina, "IoT security: data encryption for Arduino-based IoT devices," J. Positive Sch. Psychol., vol. 6, no. 3, pp. 8508-8516, 2022.

[15] M. El-Hajj, H. Mousawi, and A. Fadlallah, "Analysis of lightweight cryptographic algorithms on IoT hardware platform," Future Internet, vol. 15, no. 2, 2023.

[16] R. H. Altaie and H. K. Hoomod, "An intrusion detection system using a hybrid lightweight deep learning algorithm," Eng., Technol. Appl. Sci. Res., vol. 14, no. 5, pp. 16740-16743, 2024.

[17] H. I. Mhaibes, M. H. Abood, and A. K. Farhan, "Simple lightweight cryptographic algorithm to secure embedded IoT devices," Int. J. Interact. Mobile Technol., vol. 16, no. 20, 2022.

[18] S. Singh, P. Sharma, S. Y. Moon, and J. H. Park, "Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions," J. Ambient Intell. Humaniz. Comput., pp. 1-18, 2024.