International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.
© 2024 Scholar AI LLC.

Vol. 42 No. 2 January 2024, pp. 547-555

Securing the Internet of Battlefield Things with ChaCha20-Poly1305 Encryption Architecture for Resource-Constrained Devices

Vian Navalino[1], Achmad Farid Wadjdi[2], Yudistira Asnar[3], Rudy Agus Gemilang Gultom[4], Danang Rimbawa[5]

[1,2,4,5]Faculty of Defense Science and Technology, Cyber Defense Engineering
Republic of Indonesia Defense University
Jakarta, Indonesia
[1]viansatria57@gmail.com, [2]achm047@brin.go.id
[3]School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia

Abstract— Securing the ever-expanding Internet of Battlefield Things (IoBT) demands robust encryption solutions to safeguard sensitive data and system integrity. Unauthorized access here could result in personal data breaches and mission-critical system failures. This research evaluates the performance of the Chacha20-Poly1305 lightweight stream cipher on IoBT devices. with an average avalanche effect of 50.53%. Encryption time ranges from 261 ms for the shortest plaintext (16 bytes) to 17472 ms for the longest plaintext (8192 bytes). The decryption time varies from 266 ms to 17598 ms according to the plaintext length. The peak encryption throughput reaches about 468 Bps, and the decryption throughput is about 465 Bps for the longest plaintext. The results confirm that the Chacha20-Poly1305 algorithm operates with a high degree of speed and efficiency on the Raspberry Pi Pico RP2040, being a suitable solution for IoBT applications with fast response requirements and high security.

Keywords—ChaCha20-Poly1305, IoBT, Encryption

I. Introduction

The growth of digital technology has opened up new opportunities in building an adaptive and responsive defense system. One concept that can be applied in building such a defense system is NCW. According to Cebrowski in [1] NCW is a concept of war that focuses on network connectivity, which integrates strategies, tactics, techniques, procedures, and military organizational structures to achieve superiority in war. NCW combines elements such as sensors, shooters, and decision makers into an integrated network that increases situational awareness in the battle space.

The utilization of IoT has diversified into various fields, ranging from the implementation of Smart Cities for real-time fire monitoring using drones [2], to its use in medical imaging and medical emergency response [3][4]. In addition, the utilization of IoT devices in the context of military operations has opened up new opportunities to collect, transmit, and analyze crucial data in real-time. The use of IoT in the military context is known as the Internet of Battlefield Things (IoBT), which clarifies efforts to integrate IoT technology in the military defense aspect. The sensors used in IoBT, such as inertial sensors, accelerometer sensors, gyroscope sensors, temperature sensors, camera sensors, and microphone sensors, function as eyes and ears in the field [5]. IoBT builds upon the established framework of Command, Control, Communications, and Intelligence (C3I) [6], a pillar of modern military operations. C3I facilitates efficient decision-making through information technology, and its evolution into C6ISR (incorporating Cyber-defense and Combat systems) reflects the growing importance of cybersecurity in this information-driven domain [7]. This is important because modern military operations rely heavily on information technology, so cybersecurity is a very important factor. Thus, the integration of IoT and IoBT concepts in the context of NCW is an important element in efforts to build a strong and responsive defense system in the modern technological era.

However, strategic information, while invaluable, presents its own set of challenges. As military operations become increasingly data-driven, ensuring the data's integrity, confidentiality, and authenticity takes center stage. Unauthorized access in both IIoT and IoBT contexts can have devastating consequences: privacy breaches in IIoT can disrupt entire systems [8], while in IoBT, leaked battlefield secrets can cripple operations and endanger lives. Therefore, robust encryption measures are indispensable for data protection. From strategic communications and operation plans to location data and intelligence, all critical information must be shielded from prying eyes. Failure to do so can empower the enemy to launch counter-attacks, exploit vulnerabilities, and disrupt military coordination. To this end, encryption emerges as a vital line of defense.

Encryption can be divided into two main categories, namely symmetric and asymmetric. In symmetric encryption methods, the process to encrypt and decrypt information is done using the same key [9]. The main advantage of symmetric encryption method lies in its efficiency in faster processing compared to asymmetric encryption [10]. Unlike conventional computers, IoT devices used on the battlefield have limited processing power, memory, and energy resources [11]. Therefore, traditional cryptographic techniques that are usually designed for conventional computers may be too complex and consume a lot of resources on these devices [12]. This is where the importance of lightweight cryptography, which can strike a balance between security and efficiency, ensures that important information circulating on IoT networks remains secure without compromising device performance.

II. THEORY

2.1 Lightweight Stream Cipher

Lightweight encryption is a type of encryption designed primarily for devices with low computing power, limited battery life, small size, small memory, and limited power supply [13][14]. Therefore, traditional cryptographic methods may not work well for smart devices that have limited resources [15]. In the context of securing IoT and IoBT, the use of lightweight stream cipher algorithms becomes crucial as these devices often have significant resource limitations.

Stream ciphers are cryptographic algorithms designed to encrypt and decrypt data in a continuous, stream-like manner, as opposed to block ciphers that process data in fixed-sized blocks [16]. Stream ciphers are well suited for applications where data is sent or received continuously, such as in real-time communication systems and resource-constrained devices [17]. It generates a keystream, a sequence of pseudorandom bits or completely random bits, which is then combined with the plaintext to produce the ciphertext.

2.2 Chacha20

ChaCha20 is a widely recognized and trusted stream cipher and symmetric key encryption algorithm. Developed by Daniel J. Bernstein [18], it offers a high level of security and performance, making it a popular choice in various cryptographic applications. ChaCha20 operates by generating a pseudorandom keystream based on a key, a nonce, and a counter. This keystream is then combined with the plaintext using a simple XOR operation to produce the ciphertext. In addition, ChaCha20 is designed to be highly efficient, making it suitable for use in resource-constrained environments such as IoT devices and mobile platforms. Its efficiency, combined with strong security guarantees, has led to its adoption in various protocols and systems, including secure messaging applications, VPNs, and TLS 1.3, the latest version of the Transport Layer Security protocol.

2.3 Poly1305

Poly1305 is a cryptographic message authentication code (MAC) algorithm designed to provide data integrity and authenticity in secure communication protocols [19]. Poly1305 operates by taking a secret key and a message as input and generating a fixed size authentication tag, which is then attached to the message. This tag acts as a cryptographic fingerprint of the message and key, ensuring that the message is not tampered with during transmission. When ChaCha20 and Poly1305 are combined to create an Authenticated Encryption with Associated Data (AEAD) scheme that provides a level of confidentiality and integrity of encrypted data, the resulting construction is known as ChaCha20-Poly1305 [20].

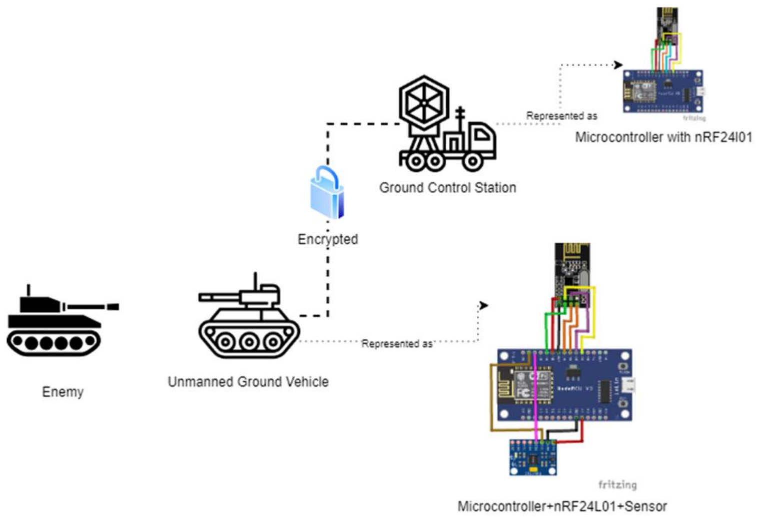
III. Research Methodology

In this research, the data to be collected and analyzed comes from the sensor integrated with the microcontroller used. The following are materials in the form of data that will be taken from the MPU6050 Accelerometer sensor:

a. Acceleration Data: Measures acceleration on three axes (x, y, and z). This data will provide information about the motion and orientation of the device on which the sensor is placed.

b. Gyroscope Data: Measures the angular velocity on three axes. This information is useful to know the change in orientation or rotation of the device.

In this research, two microcontrollers will be used: a Raspberry Pi Pico with an MPU6050 sensor and an nRF24l01 radio module; a Raspberry Pi Pico with an nRF24l01 radio module. The following is a design of the system flow for the encryption system that will be implemented.



The UGV is represented as a microcontroller-based device equipped with a radio module and various types of sensors. The main function of the UGV is to collect data from these sensors, encrypt the resulting data, and then transmit the encrypted data through the available radio module. Before encryption is performed, the time will be recorded, then encryption will be performed, and after encryption and the tag is generated, the time will be recorded again. The initial and final times will then be calculated to obtain the encryption time.

The GCS is also represented by a microcontroller-based device with a radio module. The role of the GCS is to listen to the radio signal sent by the UGV, then the decryption will be performed. Before decryption is performed, the time will be recorded, then decryption will be performed, and after the plaintext is retrieved, the time will be recorded again. The initial and final times will then be calculated to obtain the decryption time.

Then performance calculations will be carried out, namely Avalance Effect, Encryption and Decryption Speed, and Throughput.

a. Avalanche Effect: Measures how sensitive the encrypted output is to changes in the input data. This assesses the algorithm's resistance to cryptanalysis.

b. Encryption/Decryption Speed: Examines how fast the system encrypts and decrypts data of varying sizes. This analysis, involving time measurements at different data sizes, reveals the algorithm's data processing efficiency, crucial for applications requiring fast data handling.

c. Throughput: Analyzes the system's performance in managing data flow by investigating how much data can be transmitted per second across the network depending on the data size.

IV. RESULT

To determine the duration of the encryption process accurately, the first step is to record the time. This is done immediately before the encryption process begins. Recording the time before the encryption process begins ensures accuracy in measuring the duration of the encryption process. Once the initial time recording is done, the encryption process can be started. Once the encryption process is finished, the final time recording is done. By subtracting the final time from the initial time of recording, we can determine the duration required for one encryption cycle.

The table below shows the results of the encryption that has been carried out, along with the original plaintext data and other elements used in the encryption process.

TABLE I. ENCRYPTION RESULT

Key	Nonce	AAD	Plaintext	Ciphertext	Tag	Encryption Time (ms)
0xc8dac0e4f2d62563360aa88db38b98c4d27d9fb791a3fee0ddc1970e8d24cf5b	b'\x935\\\x90\xb8\xfa\xfdTu\x94X '	b'Encrypt on RPI pico'	b"{'gy': 0, 'gz': 1, 'tem': 28.93, 'gx': -4, 'ax': 0.06, 'ay': -0.0, 'az': 0.86}"	bytearray(b'|A\xf1\t23\xb8\x15\xf2\xad\x1f\x8f\xdb\x13\xb8G-\xb9\xf7\xbaL\xc8\xe6\x97\x19G\xbeF=%m\x84O\xe1Z\xdc\x8cJ2\xeb^\xf2\x04\xae\xf5\xea_Y\xc9\x99Y\xd3\x9c\x0e\xdcP/\xe1\x80))\xbc\xbd\xd3\xde*A\xc7\x99\xe4\xdc\x93h\x8e\xf0\xc6?')	bytearray(b'3\xc6\xc9`\x8d\x9a\xed\xc9\x9e}\xfd\x899\x85\xac\n')	406
0xc8dac0e4f2d62563360aa88db38b98c4d27d9fb791a3fee0ddc1970e8d24cf5b	b'\xde\xdc=\xc1\xe6\xa7\xb5\xe1\xa6\xf0\xec '	b'Encrypt on RPI pico'	b"{'gy': 0, 'gz': 0, 'tem': 28.88, 'gx': -4, 'ax': 0.06, 'ay': -0.0, 'az': 0.86}"	bytearray(b'+\xb30\xca\xd9\x1f\x8f)\x9dKsx\xe1\x12\xb2\xe7\xac\xfd\xbb\xc4\xd5\xdd\xfeG\x18\x9b\xc4\xfa\x1b\xcbB\xf70\x83\xcbY\xcfG\x14\xe9o\x15\xcd\xdf)\xfa\xf6\xb1\xbe\x0e\xbc\xc3#i\xa00\x91\x0e\x04\\\xd9\x90\x13c\x8f\xc6v\xda%\xf5(da+LV0\x91')	bytearray(b'\xdft\x07\\x8d\xd1eC\xcf\xf0\x8e\x8fG\x19\xdeb')	407
0xc8dac0e4f2d62563360aa88db38b98c4d27d9fb791a3fee0ddc1970e8d24cf5b	b'\xc2\xcb[\x98\x8e\x12\x7f\xa5\xf3\xa8q\x8d'	b'Encrypt on RPI pico'	b"{'gy': 0, 'gz': 1, 'tem': 28.84, 'gx': -4, 'ax': 0.06, 'ay': -0.0, 'az': 0.86}"	bytearray(b'\x02r\x04\x87y\x8b\xd5\xdaQ\xf4\xdb\x1a\x0e\x04\x82S\xa3\x98\xe8\xf0\xd6y\xb4%K\xb6\xcf%\xc6\xe4\xf6\xee\x7f\x84\xa2\x10\x16\xfcO\x8df\x8b[\xb7\xb7D\x90y0\x07\xf9\x11\x06\xdef\xba\xc4\x88\xd6J\x80\xc9\xe6\x95\xc1\x80\x07]\xda\xdb9\xdb@\x9c~\xaa1\x14')	bytearray(b'\x8e\xe2\x16-\xa3"\xed\xaa|z?\xefj,Y\xde')	406

0xc8dac0e4f2d62563360aa88db38b98c4d27d9fb791a3fee0ddc1970e8d24cf5b	b'\xde5\xb6@\xb7\x7f\xc2.A\x1c\xefA'	b'Encrypt on RPI pico'	b"{'gy': 0, 'gz': 0, 'tem': 28.88, 'gx': -4, 'ax': 0.07, 'ay': -0.009999999, 'az': 0.87}"	bytearray(b"\x0em\xd4\\P\xa0W\xfd\xddo\x11\xc6aa~\xcdR+n\xe9\x8a\xe1\xb2pg\x00\xe0XFx7?\xe8{PX\xb7\xb8\xb9\xf5\xe5\x9d\xb2r\xf2\xfd\xf8\xf9(\xd0\xa6\x17\x8d\xfaVl\xea^@\xb2\x1f\\a\x1aU\xd5\xb7\x93\x8c\xc5\xf2\x01\xce\xa6\xfb6\tz\xab'\x16[\xc9\xc2\xe6")	bytearray(b'\x9b)\x0b5@wTd\n3&\xa4\xbe=\xac9')	408
0xc8dac0e4f2d62563360aa88db38b98c4d27d9fb791a3fee0ddc1970e8d24cf5b	b'\x0f\xf0\xcfV\xa0/\xd7\x92/\xf7\xc4\xc6'	b'Encrypt on RPI pico'	b"{'gy': 0, 'gz': 1, 'tem': 28.93, 'gx': -4, 'ax': 0.06, 'ay': -0.009999999, 'az': 0.86}"	bytearray(b'c=\x10\x95=\xcd)\xbe\xf5\x18(\x0e\xbe\x07\x95\x87\xd7cZp\xd6"/\xf6\x9ce\xd8/\xaex)\xb0\x83t6\x1e\x18\xdb\xef\xae\xf7\xe5\xa3e\xf6\n9yKY\xad\xe0_p\xd3z\xe3(\xde\x8c\xb0\x03\xa0X\xcc\xb4j\x13f;\x83\x1d\xd2~N\xe6\x0c\xf5\xb1\xf7\xd2\x115\xe7K\xba')	bytearray(b'{\xfc\xc6r\xc7vS\xae7%\x8aoW\xa5\x08\xe3')	408

Each row in the table depicts one instance of the encryption process, showing variations in encryption time and helping in further analysis of the consistency and reliability of the encryption algorithm under different conditions.

TABLE II. DECRYPTION RESULT

Decrypted Text	**Latency (ms)**	**Decrypted Time (ms)**
b"{'gy': 0, 'gz': 1, 'tem': 28.93, 'gx': -4, 'ax': 0.06, 'ay': -0.0, 'az': 0.86}"	51	406
b"{'gy': 0, 'gz': 0, 'tem': 28.88, 'gx': -4, 'ax': 0.06, 'ay': -0.0, 'az': 0.86}"	50	410
b"{'gy': 0, 'gz': 1, 'tem': 28.84, 'gx': -4, 'ax': 0.06, 'ay': -0.0, 'az': 0.86}"	50	406
b"{'gy': 0, 'gz': 0, 'tem': 28.88, 'gx': -4, 'ax': 0.07, 'ay': -0.009999999, 'az': 0.87}"	48	410
b"{'gy': 0, 'gz': 1, 'tem': 28.93, 'gx': -4, 'ax': 0.06, 'ay': -0.009999999, 'az': 0.86}"	56	406

2.4 Avalanche effect

To assess the avalanche effect, the same plaintext is encrypted twice, with only a single-bit change in the nonce. The resulting ciphertexts are compared, and the number of differing bits quantifies the effect.

TABLE III. AVALANCHE EFFEECT

No	Nonce	Modified Nonce	Plaintext	Ciphertext	Mod Ciphertext	Avalanche (%)
1	b'\xb72L\xc3~g;\xd9\x86\xa7 +'	b'\xb72\xcc\xc3~g;\xd9\x86\xa7 +'	b"{'gy': 0, 'gz': 0, 'tem': 28.32, 'gx': -4, 'ax': 0.06, 'ay': 0.0, 'az': 0.87}"	bytearray(b'\x9a\xb0\x00\x8b\x13\xaa0jL\xd5\x8c\x0b\x1f\xc2\x02\x92\xc7\xcdd\x8b\xca\x93\xd3\xae2\xaa\xd4^\xb8\xe3\xc5j\xb1\xdb\x995\xc4\xb3\x82\x9f*dN\x1f\x02J\x97x\xa7\xa4\x08]\xfd\x11\xff\xe7\xdb\x85pw\xa8\xa9\x8cN%y\xb8\x1d\x99\xa5\xb6\x1e\xbd\xa3\x03\xe6\xe0')	bytearray(b"q\x0b\xa0\x8e\x19\xfc\xb8r\xc3y\xfa\x07\\\xc1\x0c\x19\xd9\xbb\xabl\xc0O\x1d\xa1l\x13}1E\x99'C\xf5,\xd9\xfes\xc8\xcb\x1e\x96\x95\xfb\xcd\x08\x99l\x99K\xf9\xa0N>\xef\xb4\x8e+\xd4\xe5\xe0W7E&^C\xf9J\xfe\x9f!'\xd7\x00\xaf8E")	53.08442
2	b'\xc8\xb0\xba\x06\xc2\xc4\xfc\x0e\xf2\xf5\x10\xec'	b'\xe8\xb0\xba\x06\xc2\xc4\xfc\x0e\xf2\xf5\x10\xec'	b"{'gy': 0, 'gz': 1, 'tem': 28.84, 'gx': -4, 'ax': 0.06, 'ay': -0.0, 'az': 0.86}"	bytearray(b'c\xf2\xa1\x8fG}\xb9\x9a\xb1\x99\xe5\x9c\xb4\xdd\xbf\xe0q\x91J\x19\xc8,\x80K.\x1b\xc9\x98\x1f\xa6eT\xa6m\xca\x02\xe3\x97Dp\x86\x1b\x9a\xb6\xd1m\xaa*X\x05bg8\x81\xe8\xc9k\x0f K\x9b\xea\xe0\xb9\xac\xbb\x8d\xe4\xccD\xb3\xa61\xdb5x\xfd\xe6')	bytearray(b'\x8b\x98Y\x8a\x99=%h\xa8\xf3+\xecfe\xa9\xa59}\xd7:\x10Jj\xa7\xc9\x07\x1f^\xf2)D\x84\'A\xf4\x91\xe0\xf2"r\xcb\xd3o\x9b\xe9\x18\x8c\xaf\xf2X\x83\xf6\xd7\x90\xe7\x9e\x7f\xa17\xbbM\xf8\xd4\xd9\xb3uR\xfd\x07\x10\xba\xbb\x85f\x05\xd0\xa7\r')	48.55769

While the table above showcases only two examples, we conducted 10 experiments. The average avalanche effect of ChaCha20-poly1305 in this experiment is 50,526%, indicating a significant change in ciphertext with small plaintext modifications.

2.5 Encryption and decryption speed

This analysis explores the crucial relationship between data size and encryption/decryption speed. Understanding how data length impacts algorithm performance is vital for selecting the optimal solution for real-world applications, where data volumes and processing demands can vary greatly.

TABLE IV. ENCRYPTION AND DECRYPTION TIME RESULT

Plaintext Length (bytes)	Encryption Time (ms)	Decryption Time (ms)
16	261	266
32	262	266
64	265	279
128	402	409
256	666	672
512	1208	1218
1028	2278	2419
2048	4424	4468
4096	8762	8802
8192	17472	17598

The table shows how long it takes to encrypt and decrypt data when the data size changes. As the data gets bigger (from 16 bytes to 8192 bytes), it takes longer to encrypt and decrypt it. This makes sense, because the encryption algorithm has to work harder with more data.

2.6 Throughput

This section analyzes the throughput of the tested encryption system. Throughput quantifies the system's data processing speed and is calculated by dividing the plaintext length by the encryption/decryption time. These experiments employed a range of plaintext lengths, from 16 bytes to 8192 bytes, to provide a comprehensive understanding of the system's performance across different data sizes. Encryption and decryption times were measured in milliseconds for each experiment.

TABLE V. THROUGHPUT RESULT

Plaintext Length (bytes)	Encryption Time (ms)	Decryption Time (ms)	Throughput Encryption (Bps)	Throughput Decryption (Bps)
16	261	266	61.30268199	60.15037594
32	262	266	122.1374046	120.3007519
64	265	279	241.509434	229.390681
128	402	409	318.4079602	312.9584352
256	666	672	384.3843844	380.952381
512	1208	1218	423.8410596	420.3612479

1028	2278	2419	451.2730465	424.9689955
2048	4424	4468	462.9294756	458.3706356
4096	8762	8802	467.4731796	465.3487844
8192	17472	17598	468.8644689	465.507444

From the data in the table, it can be seen that as the length of the plaintext increases, the time taken to perform the encryption and decryption processes also increases. However, the throughput does not decrease in proportion to the increase in plaintext size; this indicates that the encryption and decryption efficiency remains relatively stable despite the increase in workload.

V. Conclusion

This research investigated the use of a ChaCha20-Poly1305 to secure data transmission from Unmanned Ground Vehicles (UGVs) using a Raspberry Pi Pico microcontroller. A high avalanche effect exceeding 50,526% indicates significant ciphertext changes even with minor plaintext alterations, enhancing security against pattern-based deciphering attempts. While encryption and decryption times increased with larger data sizes, throughput remained stable, suggesting good efficiency for handling varying data loads. A trade-off between data size and throughput might be necessary depending on the application's specific needs, with smaller data sizes preferred for real-time processing or low latency situations. Overall, the chosen encryption system offers a blend of strong security, scalable performance, and efficient processing, making it a viable solution for securing UGV data transmissions on the Raspberry Pi Pico platform. However, further optimization or adjustments might be necessary depending on the specific requirements and constraints of the intended application.

References

[1] H. Ismiyanto, Kemalsyah, and Bastari, "INTELLIGENCE INTERCONNECT COMMUNICATION SYSTEM (IICS) PADA NETWORK CENTRIC WARFARE OPERASI UDARA," *J. Strateg. Pertahanan Udar.*, vol. 9, no. 1, 2023, doi: https://doi.org/10.33172/jspu.v9i1.8213.

[2] D. A. Sungheetha and D. R. Sharma R, "Real Time Monitoring and Fire Detection using Internet of Things and Cloud based Drones," *J. Soft Comput. Paradig.*, vol. 2, no. 3, pp. 168–174, 2020, doi: 10.36548/jscp.2020.3.004.

[3] A. Chandy, "a Review on Iot Based Medical Imaging Technology for Healthcare Applications," *J. Innov. Image Process.*, vol. 1, no. 01, pp. 51–60, 2019, doi: 10.36548/jiip.2019.1.006.

[4] M. M. Rathore, A. Ahmad, A. Paul, J. Wan, and D. Zhang, "Real-time Medical Emergency Response System: Exploiting IoT and Big Data for Public Health," *J. Med. Syst.*, vol. 40, no. 12, 2016, doi: 10.1007/s10916-016-0647-6.

[5] L. Zhu, S. Majumdar, and C. Ekenna, "An invisible warfare with the internet of battlefield things: A literature review," *Hum. Behav. Emerg. Technol.*, vol. 3, no. 2, pp. 255–260, 2021, doi: 10.1002/hbe2.231.

[6] S. Russell and T. Abdelzaher, "The Internet of Battlefield Things: The Next Generation of Command, Control, Communications and Intelligence (C3I) Decision-Making," *Proc. - IEEE Mil. Commun. Conf. MILCOM*, vol. 2019-Octob, pp. 737–742, 2019, doi: 10.1109/MILCOM.2018.8599853.

[7] A. Petrovski, M. Radovanović, A. Behlic, and S. Ackovska, "Advantages of Implementation of C6Isr in Low Budget Armies," pp. 47–60, 2023, doi: 10.18509/gbp23047p.

[8] J. Bader and A. L. Michala, "Searchable Encryption with Access Control in Industrial Internet of Things (IIoT)," *Wirel. Commun. Mob. Comput.*, vol. 2021, 2021, doi: 10.1155/2021/5555362.

[9] L. E. Hughes, "Basic Cryptography: Symmetric Key Encryption," in *Pro Active Directory Certificate Services: Creating and Managing Digital Certificates for Use in Microsoft Networks*, Berkeley, CA: Apress, 2022, pp. 3–17.

[10] S. F. S. Adnan, M. A. M. Isa, and H. Hashim, "Energy analysis of the AAβ lightweight asymmetric encryption scheme on

an embedded device," *IEACon 2016 - 2016 IEEE Ind. Electron. Appl. Conf.*, pp. 116–122, 2017, doi: 10.1109/IEACON.2016.8067366.

[11] K. Shahzad, T. Zia, and E. U. H. Qazi, "A Review of Functional Encryption in IoT Applications," *Sensors*, vol. 22, no. 19, pp. 1–50, 2022, doi: 10.3390/s22197567.

[12] P. Panahi, C. Bayılmış, U. Çavuşoğlu, and S. Kaçar, "Performance Evaluation of Lightweight Encryption Algorithms for IoT-Based Applications," *Arab. J. Sci. Eng.*, vol. 46, no. 4, pp. 4015–4037, 2021, doi: 10.1007/s13369-021-05358-4.

[13] M. Aikawa, K. Takaragi, S. Furuya, and M. Sasamoto, "A lightweight encryption method suitable for copyright protection," *IEEE Trans. Consum. Electron.*, vol. 44, no. 3, pp. 902–910, 1998.

[14] S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, "Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions," *J. Ambient Intell. Humaniz. Comput.*, vol. 0, no. 0, pp. 1–18, 2017, doi: 10.1007/s12652-017-0494-4.

[15] M. S. Turan *et al.*, "NISTIR 8268 Status: Status Report on the First Round of the NIST Lightweight Cryptography Standardization Process," *Nistir 8309*, pp. 1–27, 2021, doi: https://doi.org/10.6028/NIST.IR.8369.

[16] R. A. Rueppel, "Stream ciphers," *Anal. Des. Stream Ciphers*, pp. 5–16, 1986, doi: 10.1007/978-3-642-82865-2_2.

[17] L. Jiao, Y. Hao, and D. Feng, "Stream cipher designs: a review," *Sci. China Inf. Sci.*, vol. 63, no. 3, pp. 1–25, 2020, doi: 10.1007/s11432-018-9929-x.

[18] D. J. Bernstein, "ChaCha, a variant of Salsa20," *Work. Rec. SASC*, pp. 1–6, 2008, [Online]. Available: http://cr.yp.to/chacha/chacha-20080120.pdf.

[19] D. J. Bernstein, "The poly1305-AES message-authentication code," *Lect. Notes Comput. Sci.*, vol. 3557, pp. 32–49, 2005, doi: 10.1007/11502760_3.

[20] Y. Nir and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols," 2018.