Published Online First: March, 2024
https://doi.org/10.21123/bsj.2024.9052
P-ISSN: 2078-8665 - E-ISSN: 2411-7986

Baghdad Science Journal

Parallel lightweight Block Cipher algorithm for Multicore CPUs

Hawraa J. Hamiza* *, *Ahmed Fanfakh

Computer Science Department, College of Science for Women, University of Babylon, Babil, Iraq.
*Corresponding Author.

Received 10/05/2023, Revised 28/08/2023, Accepted 30/08/2023, Published Online First 20/03/2024

Abstract

Data protection has become one of the top issues despite major advancements in communications and technology. For web-based technology to send data quickly and safely, the data must be encrypted. Encryption is the process of turning plain text into ciphred text, which bad people can't read or change. Both the cryptanalysis and decryption procedures required a large amount of time in order to maintain the requisite level of security. However, a number of researchers developed the cryptography approach in parallel in order to reduce the amount of time needed for the encryption and decryption procedures to be finished. The investigation of the issue has produced a number of viable solutions. Researchers were able to attain improved performance levels on the encryption technique by using parallelism to increase the throughput and boost the efficiency of encryption methods. To achieve high performance, lightweight speck cipher algorithms have been presented and implemented on CPU platforms with various improvements. Thus, in this work, a lightweight cipher scheme is proposed which only employs one round of block cipher technique that is applied in parallel over a multicore processor. The proposed message encryption algorithm uses two subblocks of 128 bits of plain message and substitution box and splitmix64 PRNG to encrypt the plain message and obtain two encrypted subblocks, making it a fast technique to encrypt and decrypt blocks of messages. In comparison to the existing method. According to the performance findings, it is able to reach a high data throughput in comparison to some lightweight methods that already exist, with a throughput that is higher than 25 Gigabits per second on an Intel Core i7 central processing unit. The proposed encryption method outperforms the parallel speck method by an average of 14.10 times faster when executed over a multicore CPU. The average speedup compared to the sequential version of the proposed algorithm and its parallel implementation is 4.70. Also, the proposed encryption method offers a substantial amount of randomness and passes PractRand's statistical tests. Thus, the suggested method is a strong contender for high-security implementation on multicore processors.

Keywords: Cryptography, Lightweight, Multicore CPU, One round cipher, Parallel Computing.

Introduction

With the development of new security attacks that take advantage of the attackers' increased processing capacity, data security is experiencing growing difficulties. Attacks on data security might be active or passive. While active attacks may substantially jeopardize data availability, authenticity, and integrity, passive attacks have the potential to seriously threaten data secrecy. While passive attackers just intercept the conveyed data, active attackers may add, delete, or change the substance of the data. Although passive attacks are more difficult to identify, they should be taken into consideration to protect data confidentiality [1]. Cryptography may help you protect your data from hackers more efficiently.

Cryptography is the practice and study of techniques for secure communication in the presence of third parties, known as adversaries. It involves transforming information (referred to as plaintext) into an unreadable form (cipher text) to prevent unauthorized access and then transforming the cipher text back into the original form (plaintext) for authorized access [2]. Cryptography is focused on the safety and confidentiality of data. It consists of a group of algorithms that are aimed at protecting information and data [3].

Cryptology, whose primary goal is to safeguard data from malicious users, can be divided into two main branches: asymmetric cryptography, in which communicating parties do not need to share a common secret beforehand (such as RSA). Asymmetric cryptography, commonly referred to as "public-key cryptography," is a technique for encrypting and decrypting data using two separate keys: a public key that is available to everyone and a private key that is typically kept private and is known only to the owner, while symmetric cryptography, where a secret must be shared beforehand (such as AES), has significantly better performance and smaller implementations [4,5].

Symmetric cryptography, also known as shared-key cryptography, is a method of encryption and decryption where the same secret key is used to both encrypt and decrypt a message. It may be divided into block ciphers and stream ciphers, respectively. Whereas block ciphers need many keys, most recently 64 bits, and only encode as a single entity, stream ciphers encrypt one bit of text at a time [6].

Currently, there are more than 4.95 billion internet users, and the development of information security has profoundly changed our way of life due to the accessibility and availability of information. The rise of internet banking and electronic commercial exchanges has made it possible for operations like banking and trading to be done predominantly online, making it essential to protect all data and resources from numerous security risks. These security functions, which can all be secured using widely accepted cryptographic algorithms, include source authentication, data integrity, and information confidentiality[7]. Moreover, Lightweight ciphers are relevant in wireless communication, as evidence such as the works in [8-10].

Using the principles of parallel computing, a large message may be broken down into more manageable chunks that can then be distributed among several processors. Several researchers implemented the cryptography method in parallel. The research that has been done on the problem has uncovered several potential answers. Researchers used parallelism to improve the throughput of their algorithms, which allowed them to achieve higher performance levels on the encryption algorithm. On the other side, researchers use data-level parallelism to speed up their encryption methods. To meet this need, many parallel platforms were used. As a result, this work makes the following contribution:

1: One round encryption algorithm designed to explore the features of multicore CPUs is proposed.

2: The proposed parallel lightweight encryption algorithm improves the overall performance of the proposed method.

3: The proposed parallel encryption algorithm is compared to the parallel speck lightweight technique.

The remainder of the paper is structured as follows: The lightweight weight-speck algorithm is described in Section 3. Section 4 then goes through the background information required for CPU multicore and compares the proposed encryption technique to similar lightweight ciphers. Then, in Section 5, a thorough description of the suggested one-round block cipher method for CPU multicore implementations is given. In Section 6, the suggested block cipher scheme's resilience is discussed. In order to confirm the effectiveness of the suggested solutions in terms of throughput, speed-up, and execution time, many performance tests were carried out and are discussed in Section 7. In Section 8, a conclusion and future work are offered.

Related works

In this section, the provide several relevant studies that deal with a few encryption techniques used on various parallel systems in an effort to enhance throughput. A cryptographic system's throughput is the number of bits or bytes it can process and encrypt in a certain amount of time, usually measured in seconds or milliseconds. It's essential to keep this in mind while developing cryptographic systems, especially for uses involving the safe processing and transmission of huge volumes of data.

The authors in [11] proposed Speck-R a lightweight symmetric key cryptographic scheme designed specifically for IoT applications. It is designed to provide confidentiality and integrity for data transmitted over wireless networks, while minimizing the computational overhead required for encryption and decryption. One of the most popular lightweight cryptography methods is speck. It used 64-bit blocks, CTR mode, and the 96-bit Speck version. The study's primary success is the decrease in the number of Speck rounds from 26 to 7, while still retaining a good degree of security. By reducing the number of repeats, execution times will be decreased.

Researchers in [12] have concentrated on hardware and lightweight cryptography. Other authors have implemented AES, SIMON, SPECK, PRESENT, LED, and TWINE, which are six block ciphers that can be implemented both in software and hardware employing the proprietary configurable microcontroller. Other authors have also focused on lightweight cryptography. Because they are both implemented on comparable Xilinx Kintex-7 FPGAs, it is possible to make direct comparisons between these architectures in terms of frequency, position, throughput-to-area (TP/A), voltage, and expense.

The authors in [13] presented AES (Advanced Encryption Standard), a symmetric key encryption algorithm that is widely used to secure data in wireless sensor networks. The S-box in AES is a non-linear substitution table used in the encryption process. The S-box splitting technique involves dividing the S-box into multiple sub-tables and using these sub-tables in different rounds of the encryption process. They have compared their proposed enhanced AES (EAES)

algorithm to other encryption techniques used in sensor networks, such as RC5, Blowfish, and Skipjack. When dealing with various base process lengths, key lengths, and rounding, the EAES algorithm performs better. The suggested enhanced EAES algorithm improves the throughput and longevity of the WSN as a consequence.

Researchers in [14] presented FPGAs as a type of reconfigurable digital circuit that can be used to implement various encryption algorithms, including the Advanced Encryption Standard (AES). Pipelined AES encryption systems are designed to process multiple blocks of data in parallel, while parallel AES encryption systems are designed to process different parts of a single block of data in parallel.

Another study, demonstrated in [15], suggested conducting the performance evaluation of the blowfish technique in an alternate setting. The MPI industry standard was used in the development of the technique, and the investigations were performed on the IMAN1 computer. The results of the experiments show that the blowfish system's run time decreases and its performance increases in direct proportion to the number of processors in use. When using 32 processors, it achieves the best performance for a plaintext length of 160 MB. The greatest results can be achieved with 2, 4, or 8 CPUs, and the simultaneous effectiveness can reach 99%, 98%, or 66%, depending on the number of processors used (16, 32, 64, or 128).

Researcher [16] has suggested a compact stream cipher technique built on a dynamic key strategy that combines two distinct pseudo-random number generators (PRNGs). Comparing the method to existing encryption standards like AES, it offers a high degree of safety with less delay and the required support. The suggested encryption is fairly strong, with capacities of more than 115 gigabytes on a Titan X GPU and significantly more than 372 gigabytes on a Tesla V100 GPU. The big crushing of TestU01, frequency, and key responsiveness make it a strong stream cipher alternative due to its high amount of unpredictability.

The authors in [17] presented a proposed encryption called "ORSCA" that only required one round and made use of the dynamic resource approach. The recommended cryptosystem was designed with the characteristics of the GPU in mind. For large-scale applications, this work featured a key stream with just one iteration. It can process more data than previous methods, according to productivity findings, with a capability of around 5 terabits per second on a Tesla A100 GPU. The provided cryptography outperforms the most powerful GPU implementations of AES, Simons, and Speck, making it more appropriate for use in practical applications.

The authors in [18, 19] introduced a one-round encryption technique for the authentication of

Published Online First: March, 2024
https://doi.org/10.21123/bsj.2024.9052
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

messages that may be carried out in parallel across a multicore processor and GPUs.

Overall, while the mentioned studies make significant contributions to the field of encryption in parallel systems, additional research and analysis are needed to address the highlighted research gaps and provide a more comprehensive understanding of the performance, security, and practical applicability of these encryption techniques. However, this work advances the field of data encryption by presenting a strong contender for high-security one-round encryption implemented on multicore processors, offering improved throughput, efficiency, and statistical robustness compared to existing methods.

Table 1ْ Comparison between parallel encryption techniques

Ref	Algorithm	Key Size	Block Size	#Rounds	parallel architecture
11	**Speck-R**	96 bits	64 bits	26	-
11,12	**Speck**	128 bits	128 bits	32	-
12	**Simon**	128 bits	128 bits	64	-
13	**EAES**	128,192, 256 bits	128 bits	10,12, 14	Multi core processors
14	**AES**	128,192, 256 bits	128 bits	10,12, 14	Multi processors platform, Pipeline
15	**Blowfish**	32-448 bits	64 bits	16,18	Multi core processors
16	**ESSENCE**	512 bits	128 bits	One	GPU
17	**ORSCA**	512 bits	256 bits	One	GPU
18	**MEAA**	512 bits	256 bits	One	GPU
19	**MAA**	512 bits	64 bits	One	Multi core CPU
Proposed	**One round**	512 bits	128 bits	One	Multi core CPU

Lightweight Cryptography Algorithms:

Lightweight cryptography is a type of symmetric encryption with minimal computational complexity and/or low memory requirements. It is now going through a process of compiling international standards and recommendations with the aim of increasing the application of cryptography on restricted devices. The use of lightweight cryptography should be simple on a range of hardware and software systems. Such general-purpose, lightweight designs are uncommon, and creating cryptography of this kind is quite difficult. The US National Security Agency (NSA) has unveiled the general-purpose lightweight block cipher family Speck, each of which offers great performance in both hardware and software. Additionally, lightweight cryptography offers the right level of protection. Trade-offs in terms of security are not always made by lightweight encryption. New, simple cryptographic primitives are presented.

Speck Cryptography:

Speck is a lightweight block cipher family that was publicly revealed in June 2013 by the NSA (National Security Agency). Speck has undergone performance improvements for software applications. The add-rotate-xor (ARX) cipher is what the Speck cipher uses, as shown in Fig. 1. Speck is a family of block ciphers with different key sizes and block sizes. The most common versions of Speck are Speck32/64, Speck48/72, and Speck64/128. The numbers in the names refer to the block size and key size of the cipher. The Speck cipher is represented as Speck2n/wn, where it has a 2n-bit block and a wn-bit key. The cipher's round functions involve combining various n-bit word operations [19]:

● Bitwise XOR

● Modular Addition

● Left and right circular shift

Published Online First: March, 2024
https://doi.org/10.21123/bsj.2024.9052
P-ISSN: 2078-8665 - E-ISSN: 2411-7986

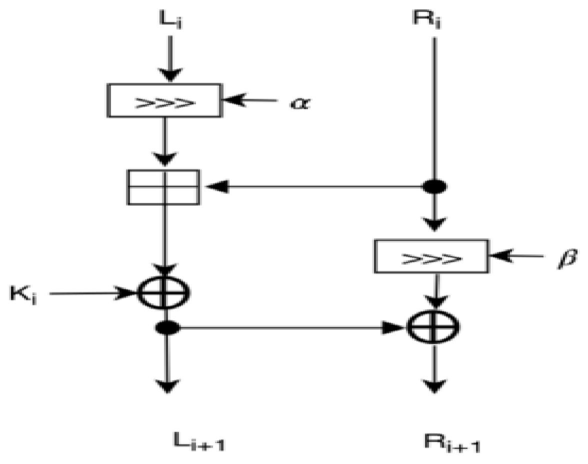


Figure 1. Speck round function.

Background: message encryption and parallel computing

Encryption is a part of our everyday lives, although it is mostly invisible. It is used to secure network connections, make e-commerce and e-banking feasible, prevent eavesdropping on our conversations through mobile phone calls and the Internet, and generally conceal information from prying eyes. New encryption techniques have been developed throughout civilization as previous ones have been cracked. On the other hand, parallel computing describes the use of numerous processors or computers to carry out a single calculation or activity. This may speed up the calculation and considerably decrease the amount of time needed to finish it. The suggested message encryption technique can be executed more quickly than previous lightweight ciphers because of the utilization of parallel computing[20, 21]. A multi-core processor is an integrated circuit having two or more multifunctional processing cores linked to it in order to boost performance while also reducing power consumption, as seen in Fig. 2.

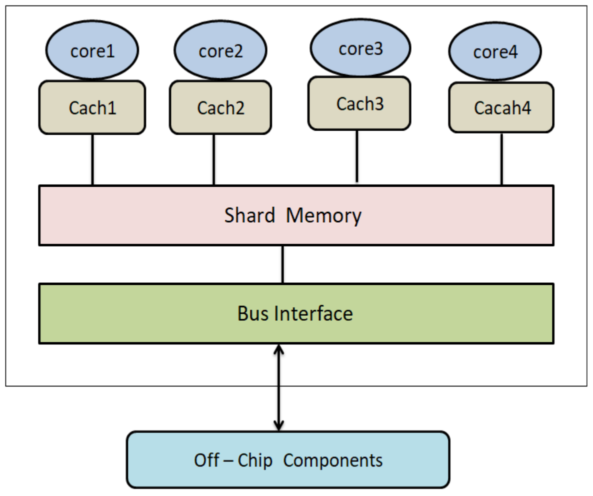


Figure 2. Multi-core architecture

Table 2. Notations table.

Symbol	Definition
K	Shared secret key
IV	Initial vector that changes per input message
CTR	Counter mode
DK	Key dynamically modified for each input message
K1 and K2	Substitution sub-keys
S1 and S2	Tables of substitutions generated with the use of kS1 and kS2, respectively.
KSeed	The seed sub-key, and it is used in the process of producing N seeds.
N	Size of initial seeds
KSA	RC4's Key Establishment Algorithm.
M	Original message
C	Message that is encrypted
BLi	ith original plaintext block
CBLi	ith encrypted plaintext block

Proposed One Round Cryptography Algorithm

This section presents the proposed one-round cipher algorithm, designed to outperform the multi-round Speck cipher. The algorithm has been implemented in parallel to enhance encryption performance. To assess its effectiveness, the proposed cipher underwent randomness tests, including the PractRand test, and its performance was compared to well-known lightweight encryption algorithms such as Speck. The proposed system consists of three main steps: mixing, PNRG (Pseudo-Random Number Generator), and substitution. Each step has been meticulously designed to ensure both the security and efficiency of the cipher. Fig. 3. illustrates the general encryption process, highlighting the various steps involved. Additionally, Table 2 provides a list of notations used throughout this paper. Fig. 3 showcases the scheme of the proposed one-round block cipher.

Algorithm 1 is designed to perform one-round encryption on an input message divided into two 64-bit blocks, represented by the variable 'at'. The input undergoes bitwise XOR operations with the key, block counter CTR, and initial vector using the PRNG xorshift64. The resulting value is stored in another variable. Temporary variables R1 and R2 are then calculated by performing XOR operations

Published Online First: March, 2024
https://doi.org/10.21123/bsj.2024.9052
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

between the initial variables (i) and the dynamic key, followed by applying the splitmix64 function to R1. R2 undergoes byte substitution using Sbox1, which replaces each byte with a new value based on a lookup table. Subsequently, XOR operations are performed between R1 and R2, and the result is stored in R1. Finally, the encrypted values are stored in the output array 'out', and the index variable 'k' is updated.

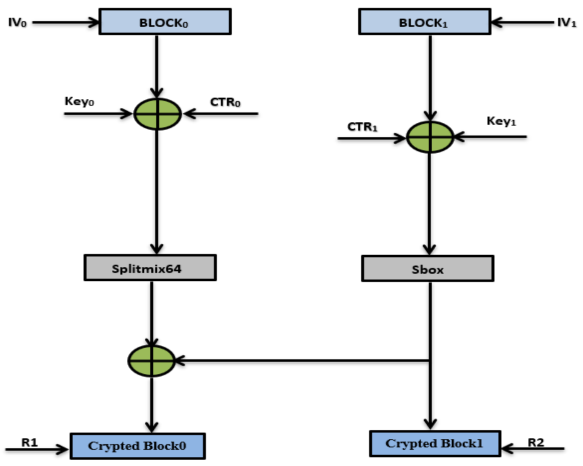


Figure 3. Scheme of the proposed one-round block cipher.

```
Algorithm 1:  Proposed Encryption algorithm
Input:
  in: plain block array of size 64-bit
  Sbox1: arrays of size 256 bytes representing substitution boxes
  DK: array of size 512 bytes
  v: array of initial vector of size 64-bit
  start: Starting index for the encryption loop
  end: Ending index for the encryption loop
Output:
   out: encrypted  block array  of size 64-bit
Start
  1:  Set k to 0
  2:  For i = start to end (increment by 2):
  3:    R1 = ((j) □ (((ulong*) DK) [i&63])) □ v[j]
   4:   R2 = ((j+1) □ (((ulong*) DK) [i&63])) □ v[j+1]
   5:   R1 =  Splitmix64 (R1)
   6:   R2  =  Substitution (R2, Sbox)
     7:  R1  =   R1 ^ R2
     8:  out [k] =  R2 ^ plain [j]
     9:  out [k+1] = R1 ^ plain [j+1]
   10:  Increasing k by 2
   11:  End For
 12: End
```

Dynamic key generation method:

The suggested solution is founded on the dynamic key-dependent methodology, in which a dynamic key DK is utilized in order to generate a collection of dynamic cryptographic primitives. (Substitution tables in addition to a set of N seeds, where each seed can be a word of 32 or 64 bits.) This dynamic key Dk is acquired by performing an XOR operation between an initial vector and CTR (counter mode), which should be refreshed and distinctive for each communication. This process is carried out in the manner depicted in Fig. 4, and it is outlined in the equation as follows:

$$DK = \; (IV \; □ \; CTR) \qquad 1$$

This dynamic key DK is broken up into three sub-keys, with each of the first two sub-keys (K1 and K2) having a length of 128 bits, while the third sub-key (Kseed) having a length of 256 bits [16].

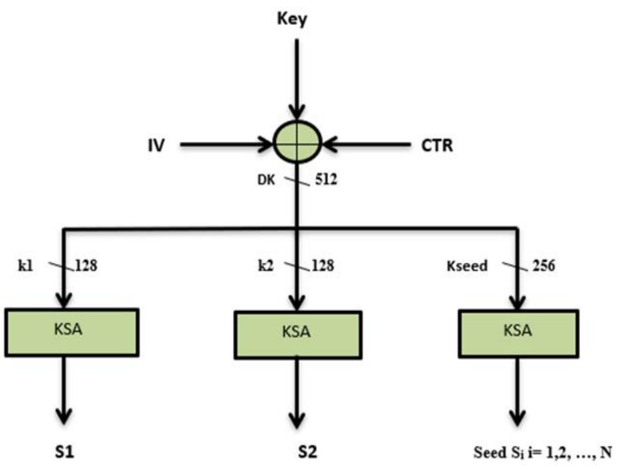


Figure 4. Displays the proposed dynamic key generation and construction cryptographic primitives.

The following provides an explanation of these sub-keys for your reference [17] :

1. K1 is the first substitution sub-key, and it symbolizes the first 128 least significant bits of DK. This is the case because K1 is the first subkey. This sub-key is utilized in the creation of the very first substitution table, which is denoted by S1. In this stage, you are free to employ any technique you choose to generate dynamic-key dependent substitution tables. For instance, the Key Setup Algorithm (KSA) of RC4 was implemented in [16]. Therefore, dynamic substitution tables could be created. At this point, we also make use of the KSA algorithm that is used by RC4.

2. K2 is the second substitution sub-key, and it symbolizes the second of the 128 least important

Published Online First: March, 2024
https://doi.org/10.21123/bsj.2024.9052
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

bits of DK. K2 comes after K1. Using the same process that is used with K1, which is the KSA for RC4, this sub-key is used to generate the second substitution table, which is referred to as S2.

3. KSeed serves as a representation for the first 256 most significant bits (MSB) of DK. This sub-key is used as a hidden seed in conjunction with any PRNG in order to generate a key stream with a length of N words, where each word's length can range between 32 and 64 bits. Because of this, KSeed is where one can acquire N seeds. Each process will choose one of these produced samples, which will be generated in a manner that is dynamically simulated to be random.

Splitmix64 PRNG:

SplitMix64 is a fast and highly robust 64-bit random number generator. It is designed to generate random numbers with high quality, uniform distribution and low correlation, even in the presence of multiple concurrent streams. Splitmix64 PRNG employs logical (xor and rotation) and

arithmetic (addition and multiplication) operations. The splitmixt64 algorithm's phases are shown in Algorithm 2. Splitmix64 is not a secure PRNG, but it was chosen since it is quick to develop and efficient (low execution time). The proposed block cipher uses huge key space and the use of dynamic cryptographic primitives provides a high degree of security [16,17].

The Proposed Parallel Encryption Method:

To provide clarity on the operational process of a one-round cipher, Fig. 5 presents a graphical illustration of the parallel implementation of the proposed cipher. The plain messages are divided into groups of blocks, with each block having a size of 64 bits. The block size, size, is computed by dividing the message size by the number of parallel threads. Subsequently, all sub-blocks have the same size, and each one is sent to its respective thread. The encryption algorithm is then applied to each thread, generating a set of encrypted blocks. Moreover, all encrypted blocks are gathered asynchronously.

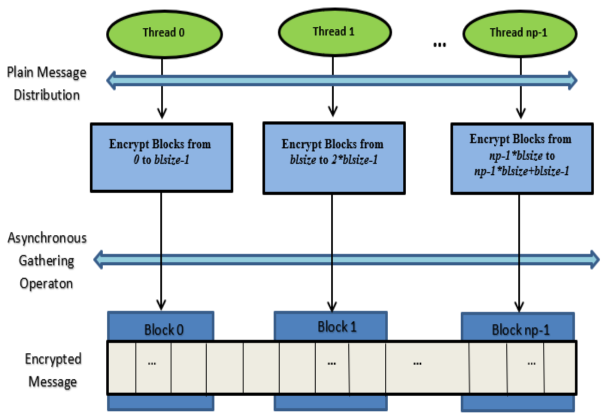

Thread 0
Thread 1
...
Thread np-1
Plain Message Distribution
Encrypt Blocks from 0 to blsize-1
Encrypt Blocks from blsize to 2*blsize-1
Encrypt Blocks from np-1*blsize to np-1*blsize+blsize-1
Asynchronous Gathering Operaton
Block 0
Block 1
Block np-1
Encrypted Message
...

Figure 5. The parallel encryption method.

The parallel proposed Encryption method is a function that takes in several parameters as shown in algorithm 3. In the following the description of the algorithm parameters:

- IV: A pointer to the 64-bit initialization vector.
- Sbox: A pointer to an array of 256 bytes that represents an S-box substitution table.
- ThreadId: the index of the parallel thread dedicated to each block of data.
- Blocksize is the size of block data that each parallel thread computes.
- plain is an array of 64-bit elements representing the plain message.
- encrypt: A subarray of a 64-bit output buffer to store the encrypted cipher text.
- crypted_Block0, crypted_Block1: are the final output for the encrypted blocks per iteration.

Algorithm 3: Parallel One-Round Cipher Encryption

```
Inputs: initial vector (V), Sbox, ThreadId, Blocksize, DK, plain, cipher
output: crypted_Block0, crypted_Block1
1:  Start = ThreadId*blocksize;
2:  End = start+(blocksize-1)
3:  For j = start to end (increment by 2):
4:        Compute R1 = ((j) □ (((ulong*) DK) [i&63])) □ v[j]
5:        Compute R2 = ((j+1) □ (((ulong*) DK) [i&63])) □ v[j+1]
6:            R1 ⟵ Splitmix64 (R1)
7:            R2 ⟵ Substitution (R2, Sbox)
8:            R1 ⟵ R1 ^ R2
9:            cipher [k] = R2 ^ plain [j]
10:           cipher [k+1] = R1 ^ plain [j+1]
11:           Increasing k by 2
12: End For
13: Call MPI_Igather (cipher, bl_size, MPI_INT64_T, allcipher, blocksize,
        MPI_INT64_T,0, MPI_COMM_WORLD)
```

Published Online First: March, 2024
https://doi.org/10.21123/bsj.2024.9052
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

Presented here is algorithm 3, which is a block cipher encryption algorithm that uses a parallel one-round encryption technique. An starting vector (V), a Sbox substitution table, a ThreadId, a Blocksize, a DK (derived key), plaintext (plain), and an encryption function are all inputs that the algorithm receives. The method then generates two encrypted blocks (crypted_Block0 and crypted_Block1) as output. The method performs its operations on each block of the plain message in parallel. It does this by dividing the plain message into sub-blocks of size 'blocksize' and assigning each sub-block to a distinct thread that is identified by 'ThreadId'. Each thread on the assigned sub-block performs parallel processing.

Before processing each sub-block, the method first computes R1 and R2 based on the starting vector V, the derived key DK, and the sub-block index j. This procedure is repeated until all of the sub-blocks have been processed. In the subsequent step, the operations Splitmix64 and Substitution are applied to R1 and R2, respectively. After the R1 and R2 values have been generated, they are joined via the employment of an XOR operation. The resultant values are then utilized to encrypt two plain values (j and j+1) through the utilization of the encryption function that has been given. The 'encrypt' array is being used to hold the values of the cipher message that was generated. Repeating this operation for each and every plaintext sub-block that is allocated to the current thread is the next step.

The method utilizes MPI_Igather to collect the resultant cipher message values from all threads and stores them in the 'allcipher' array when it has finished processing all of the sub-blocks that have been allocated to the current thread before moving on to the next thread. The cipher message values that have been obtained are of the type MPI_INT64_T and have a size of 'blocksize'. These values come from all of the threads that are used in the buffer, which is referred to as allcipher. The use of asynchronous gathering helps to cut down on the amount of time spent communicating. For the most part, the technique that has been shown is a parallel block cipher encryption algorithm. This means that it acts on plain messages in parallel, which makes it appropriate for use in systems that require the encryption of huge amounts of data in an efficient and scalable manner.

Results and Discussion

Security Analysis

A suggested encryption scheme is tested for safety and security using well-known attacks such as statistical, linear, nonlinear, or brute-force assaults [16,17]. Extensive tests are carried out in this part to demonstrate the resilience of the suggested cipher system. While the suggested encryption system may be used for any data type, only the results for multimedia content are presented.

Statistical analysis tests:

A cipher must possess two crucial criteria, namely randomness and uniformity, in order to be regarded as safe against statistical assaults [16]. In order to properly perform text cryptanalysis, methods including probability density function (PDF) analysis, entropy analysis, and correlation between the original and encrypted texts should be utilized. Further to doing PractRand tests. These criteria are detailed in the subsections that follow in order to verify the cryptographic security of newly developed pseudo-random bit generators.

Histogram analysis

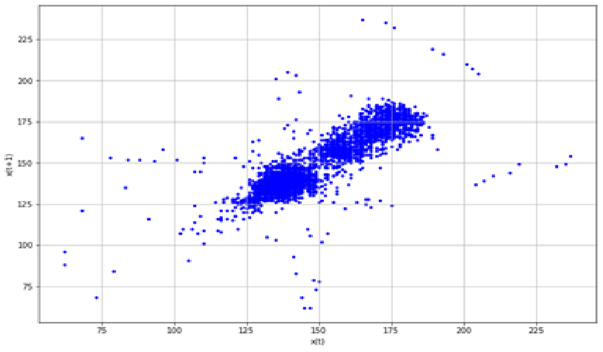
This encryption satisfies the uniformity condition only if the encrypted image has a histogram with a uniform distribution. This indicates that the frequency with which each symbol appears is proportional to the total number of symbols in the message. In other words, must be close to message-size / number of symbols. Figs. 6-c and 6-d is a histogram comparing the original plain-images of size 512 x 512 with their cipher-image counterparts. The histogram of the encrypted images is demonstrated to be quite near to a uniform distribution (about 1024).

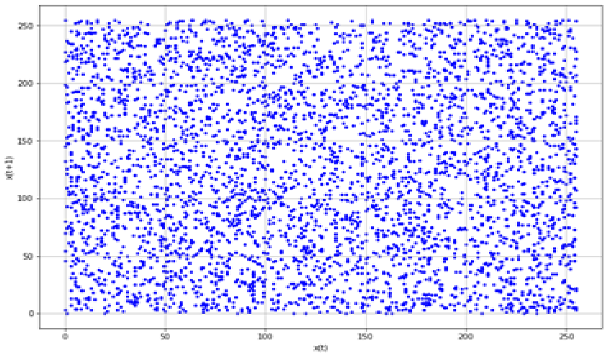
Uniformity analysis:

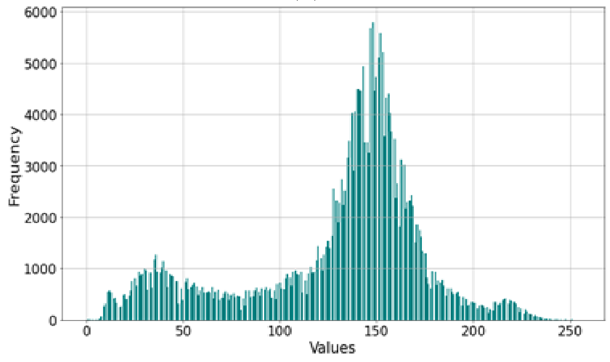
The probability density function (PDF) of the encrypted text is often examined when statistical analysis tests are run on it. The likelihood of an individual value appearing in the cipher text is described in the PDF. One common PDF analysis test is to calculate the frequency distribution of each symbol in the cipher text. This can be done by counting the number of times each letter appears in the text and dividing by the total number of letters in

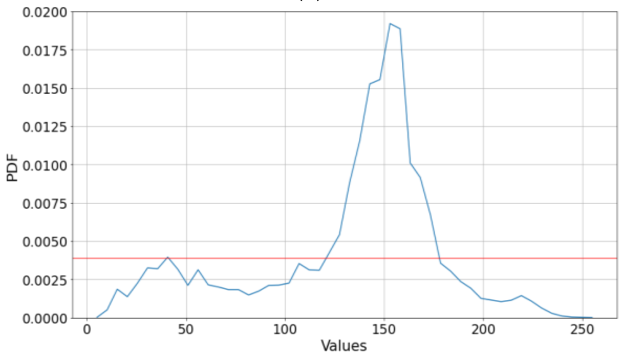
Published Online First: March, 2024
https://doi.org/10.21123/bsj.2024.9052
P-ISSN: 2078-8665 - E-ISSN: 2411-7986

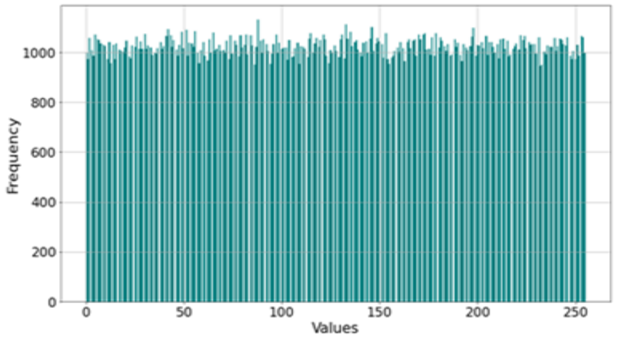
the text. Figs. 6-e and 6-f displays the original PDF and the accompanying encrypted messages. With a value of around 0.039 (1/256 = 3.9 × 103) for all cipher text symbols, the PDFs of the encrypted messages can be thought of as being close to the uniform distribution.


(a)


(b)


(c)


(e)


(d)

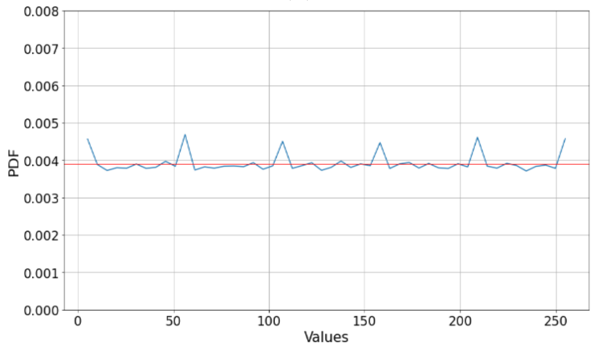

(f)

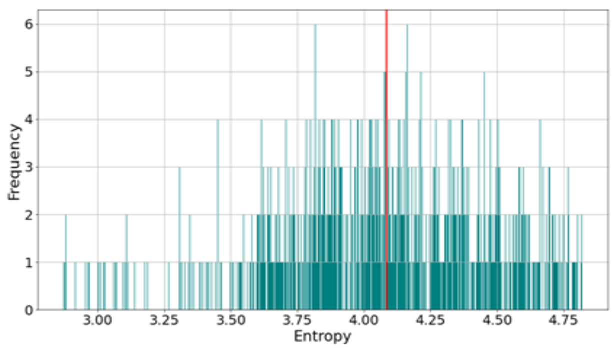
Figure 6. (a) and (b) show the recurrence of the original and cipher message, respectively. (c) and (d) are the histograms of the original and encrypted messages. The PDFs of the original and cipher message are presented in (e) and (f), respectively.

Entropy Analysis:

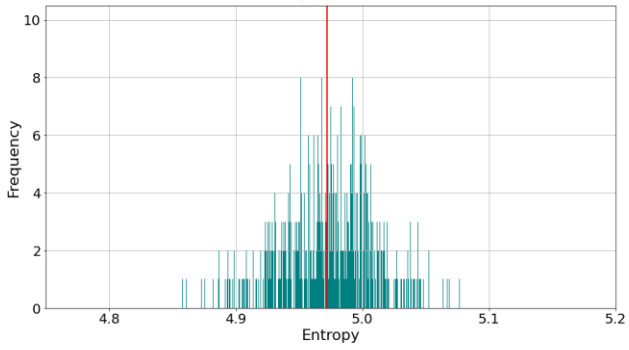
Entropy analysis is a popular statistical analysis test used to assess the randomness or unpredictability of cipher text. Entropy analysis's fundamental goal is to gauge the degree of uncertainty or information present in a transmission. The message's informational entropy M is a metric that defines the degree of a random variable's uncertainty as follows:

$$H(m) = -\Sigma\, p(M_i) \times \log_2 (1/p(M_i)) \qquad 2$$

Where H is the entropy, $p(M_i)$ is the probability of the *ith* symbol occurring in the message, and log2 is the logarithm base 2. The binary secret data should have an entropy value of 1 or close to it for optimum encryption. Fig. 7b shows the distribution of the entropy values obtained from testing encrypted text collected at the byte level. Almost exactly at the theoretical maximum ($\log_2$ (Tb/8) = 5 for *Tb* = 256), the entropy values have a normal distribution with a mean of 4.97 and a standard deviation of 0.0362. The uniformity of the distribution of the entropy values in Fig. 7b indicates that the encrypted message follows the same pattern. As a result, the suggested encryption system is sufficiently safe from any entropy attack.



(a)



(b)

Figure 7. Analysis of the entropy of (a) the plain message and (b) the generated cipher message at the sub-matrix level of size 16*16 in comparison to 1,000 random secret keys.

Correlation coefficient (CC):

The correlation coefficient (CC) is a statistical measure that quantifies the strength of the linear relationship between two variables. Its significance demonstrates the differences between them. the plain and encrypted messages. If the correlation coefficient is close to 1, it suggests a strong correlation between the cipher message and the expected plain message frequencies, which in turn suggests that the encryption method may be vulnerable to frequency analysis attacks. If the correlation coefficient is close to 0, it suggests no correlation between the cipher and the expected plain message frequencies, which indicates a stronger encryption method. The following equation is used to compute the correlation coefficient corr(x, y) [17] :

$$\mathbf{Corr(x, y)} = \frac{\mathbf{Cov(x,y)}}{\sqrt{\mathbf{E(x) \times E(y)}}}$$

where:

$$\mathbf{Cov(x, y)} = \frac{1}{N} \times \sum_{k=1}^{n} (\mathbf{xi} - \mathbf{C(x)})\,(\mathbf{yi} - \mathbf{C(y)}) \quad 3$$

$$\mathbf{Cx} = \frac{1}{N} \times \sum_{k=1}^{n} \mathbf{xi}$$

$$\mathbf{Ex} = \frac{1}{N} \times \sum_{k=1}^{n} (\mathbf{xi} - \mathbf{C(x)})^2$$

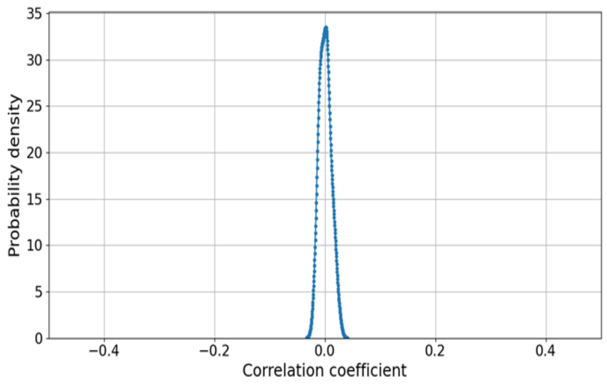


Figure 8. PDF of the correlation coefficients between plain and cipher messages for a group of 1024 bytes.

In Fig. 8, the findings of the correlation test between the original and encrypted messages are shown for one random key per iteration and a total of 1,000 random keys. The findings unmistakably demonstrate that the correlation coefficient is very low, almost very close to zero, which validates the randomness and independence of the created cipher text.

Table 3. The comparison of the security results

Security Measure	**Average results**	
	Proposed cipher	**Speck cipher**
Histogram	748.98	748.92
PDF	0.003942	0.003921
Entropy	4.9712	4.9749
Correlation coefficient	0.000696	0.00496

Table 3 shows the comparison of the security results between the proposed and speck ciphers. It indicates that both ciphers are very close in terms of security level.

Statistical tests with PRACTRAND:

PractRand is a statistical testing suite designed to test the randomness of pseudorandom number generators (PRNGs)[22]. The test suite includes various statistical tests that analyze the distribution, frequency, and correlation of random numbers generated by the PRNG. The purpose of this tool is to identify minor faults or biases in random number generators, which may not be obvious through the use of straightforward statistical tests. As was said earlier,

Published Online First: March, 2024
https://doi.org/10.21123/bsj.2024.9052
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

the proposed block cipher was put through its paces by utilizing 64 seeds and Practrand, and it was able to pass each and every test with flying colors. PractRand will analyze the created sequence and produce a report indicating whether or not the sequence was successful in passing the tests. In order to analyze the cipher message that was generated, the most difficult statistical tests, known as PractRand, were utilized. The results of this test demonstrate that the key stream that was generated satisfies the essential standards for randomization and uniformity.

Performance Analysis

In this part, a comparison is made between the recommended encryption technique and several lightweight ways of encrypting data on CPU-based devices. The comparison is carried out in parallel, utilizing the suggested encryption method. A Linux operating system and the parallel message passing (MPI) platform were utilized in the process of carrying out this investigation. An experiment with an Intel multicore i7-7700HQ processor was used to test how well the recently suggested encryption method works in a parallel computing setting. A comparison of the parallelized lightweight Speck cipher method on two, four, six, and eight threads, as well as on a variety of message sizes—four, eight, sixteen, thirty-two, sixty-four, and one hundred and twenty-six megabytes—was the primary emphasis of the evaluation. This section presents an analysis of the results based on parallel throughput, encryption time, and speedup metrics for both the Speck and proposed ciphers. As a concluding note, Table 4 displays the average of the comparison results for execution time and throughput across all message sizes.

Throughput evaluation:

The throughput metric is the ratio between the message size to the execution time of the encryption/decryption process. The presented results in Fig. 9 show that the proposed encryption technique exhibits significantly higher throughput compared to the modern speck cipher algorithm when executed on a multicore processor. The actual transmission rates achieved are subject to variability based on the quantity of data subjected to encryption and the computational ability of the CPU. The obtained results demonstrate that the suggested algorithm for the one-round cipher offers superior performance in comparison to stream encryption algorithms utilized for a range of message sizes.

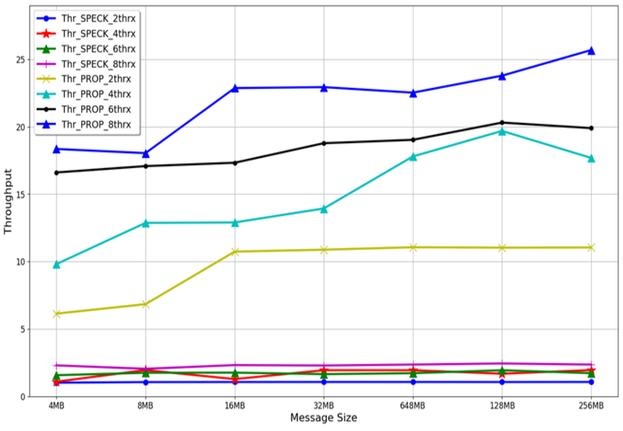


Figure 9. Shows the throughput results of the proposed encryption in comparison to speck ciphers.

Execution time evaluation:

Fig 10 shows the results of the execution time to encrypt or decrypt a message of different sizes executed over different numbers of parallel threads. The chose seven different sizes of data, from 4 megabytes to 256 megabytes. According to the figure, the can see that the encryption time goes down as the number of threads goes up. This can be seen clearly when executing over more parallel threads. However, based on the collected results, the proposed consecutive algorithm for the one-round cipher had the lowest processing time when compared to the Speck algorithm for different message sizes.

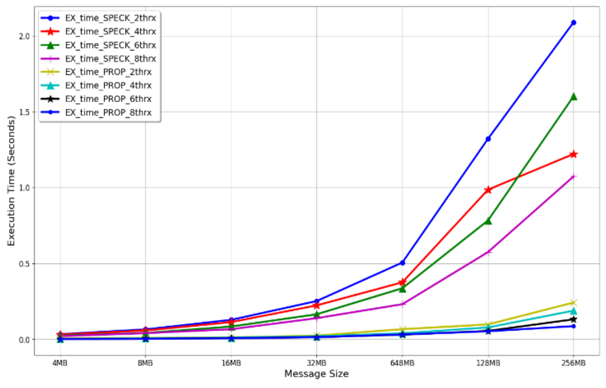
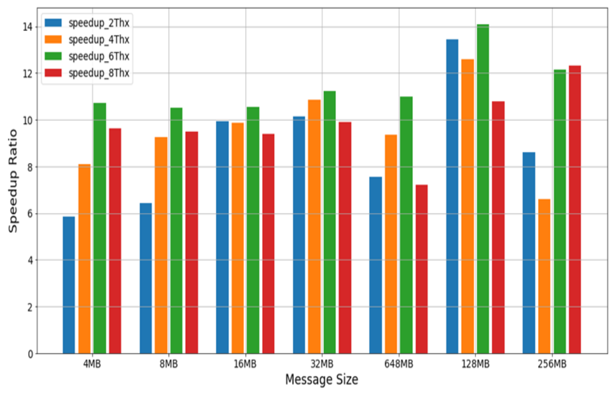


Figure 10. The execution time comparison results

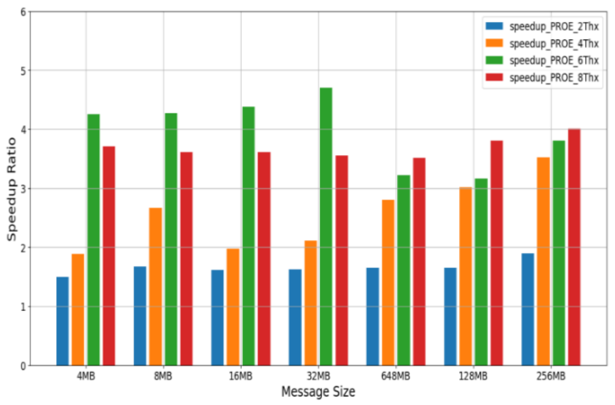
Speedup evaluation

In parallel computing, the speedup factor is the ratio of the execution time of a sequential application divided by the execution time of its parallel implementation. Moreover, it can indicate any acceleration between two different speeds [18]. In Figs. 11 a and b, it is demonstrated that, on average, the proposed

encryption algorithm is significantly faster than the parallel implementations of the Speck cipher. Specifically, in Fig. 11 a, the proposed algorithm is 14.10 times faster than the parallel implementation of Speck. These findings indicate that the proposed algorithm may be a more efficient option for encryption compared to existing parallel implementations of other popular lightweight ciphers. On the other hand, the parallel execution of the proposed one-round cipher is compared to its sequential version in Fig. 11 b. The latter shows that, on average, it is 4.27 times faster than a sequential implementation. Indeed, the speedup ratio is limited to the communication and computation ratios. In other words, the proposed parallel cipher is limited to the number of communications in the parallel system.



(a)



(b)

Figure 11. The speedup ratio of the parallel implementation of the Speck and proposed cipher and, (b) the speedup ratio of the sequential and parallel proposed methods.

Table 4. The comparison results

#threads	**Average overall message sizes**			
	Execution time (s)		Throughput Gbits/s	
	Proposed	Speck	Proposed	Speck
2	0.098	2.088	11.062	1.069
4	0.18	1.22	19.69	1.95
6	0.13	1.60	20. 31	1.94
8	0.087	1.073	25.69	2.44

Conclusion

In conclusion, this work proposes an efficient, optimized one-round cipher scheme. It is designed to be specifically fulfilled on parallel platforms. However, this study provides a ground-breaking method for concurrent message encryption that makes use of a new lightweight cipher. Our methodology entails segmenting the plain message into two sub blocks of 128 bits, each of 64 bits, and subsequently executing encryption operations within a single round. Empirical evaluations show that our suggested method outperforms competing approaches, such as the Speck method, which demands numerous rounds and, in contrast to our method, shows reduced speed and performance. The suggested encryption method, when executed on a multicore CPU, is 14.10 times faster on average than the parallel speck approach, making it more suitable for real-time applications. When implemented in parallel, the suggested approach outperforms its sequential version by a factor of 4.27. Additionally, the recommended encryption method offers a high amount of randomness and has passed the demanding PractRand randomness test. This accomplishment highlights the suggested approach's dependability and robustness. The use of the suggested techniques on GPUs with many cores offers promise as a future research direction for additional performance improvement and reaching faster processing rates.

Acknowledgment

The researchers would like to thank the University of Babylon - College of Science for Women for supporting this work.

Authors' Declaration

- Conflicts of Interest: None.
- We hereby confirm that all the Figures and Tables in the manuscript are ours. Furthermore, any Figures and images, that are not ours, have been included with the necessary permission for re-publication, which is attached to the manuscript.
- Ethical Clearance: The project was approved by the local ethical committee in University of Babylon.

Authors' Contribution Statement

In the research study titled "Parallel lightweight Block Cipher Algorithm for Multicore CPUs," the contributions of the authors are as follows. H.J played a significant role in the conceptualization and methodology of the algorithm. A.F, the corresponding author, contributed to the methodology, software implementation, validation, writing, and project administration. Together, these authors collaborated to develop and implement the Parallel lightweight Block Cipher Algorithm for Multicore CPUs.

References

1. Logunleko KB, Adeniji OD, Logunleko A. A Comparative Study of Symmetric Cryptography Mechanism on DES , AES and EB64 for Information Security. Int J Sci Res. Comput Sci Eng. 2020; 8(1): 45-51.
2. Hemamalini V, Zayaraz G, Susmitha V, Gayathri M, Dhanam M. A Survey on Elementary , Symmetric and Asymmetric Key Cryptographic Techniques. Int J Adv Comput Sci Appl. 2016; 5(1): 11-26.
3. Asaad R, Abdulrahman S, Hani A. Advanced Encryption Standard Enhancement with Output Feedback Block Mode Operation. Acad J Nawroz Univ. 2017; 6(3): 1-10. https://dx.doi.org/10.25007/ajnu.v6n3a70
4. ABood OG, Guirguis SK. A Survey on Cryptography Algorithms. Int J Sci Res Publ. 2018; 8(7): 7978-23. https://doi.org/10.29322/ijsrp.8.7.2018.p7978
5. Shukur WA, Qurban LK, Aljuboori A. Digital Data Encryption Using a Proposed W-Method Based on AES and DES Algorithms. Baghdad Sci J. 2023; 20(4): 1414–1424. https://dx.doi.org/10.21123/bsj.2023.7315
6. Suhael SM, Ahmed ZA, Hussain AJ. Proposed Hybrid Cryptosystems Based on Modifications of Playfair Cipher and RSA Cryptosystem. Baghdad Sci J. 2023; 20(5): 1-10. https://doi.org/10.21123/bsj.2023.8361
7. Sleem L. Design and implementation of lightweight and secure cryptographic algorithms for embedded devices Lama: HAL Id : tel-03101356. 2021.
8. Al-Shareeda MA, Manickam S. A Systematic Literature Review on Security of Vehicular Ad-Hoc Network (VANET) Based on VEINS Framework. IEEE Access. 2023; 11: 46218–28
9. Al-Mekhlafi ZG, Al-Shareeda MA, Manickam S, Mohammed BA, Alreshidi A, Alazmi M, et al. Efficient Authentication Scheme for 5G-Enabled Vehicular Networks Using Fog Computing. Sensors. 2023; 23(7): 3543 –18.
10. Al-Shareeda MA, Manickam S. COVID-19 Vehicle Based on an Efficient Mutual Authentication Scheme for 5G-Enabled Vehicular Fog Computing. Int J Environ Res Public Health. 2022; 19(23):15618 – 16. https://doi.org/10.3390/ ijerph192315618
11. Sleem L, Couturier R. Speck-R: An ultra light-weight cryptographic scheme for Internet of Things. Multimed Tools Appl. 2021; 80(11): 17067-17102. https://doi.org/10.1007/s11042-020-09625-8
12. Dutta IK, Ghosh B, Bayoumi M. Lightweight cryptography for internet of insecure things: A survey. 2019 IEEE 9th Annu Comput Commun Work Conf CCWC 2019. Published online 2019: 475-481. https://doi.org/10.1109/CCWC.2019.8666557
13. Gupta M, Sinha A. Enhanced-AES encryption mechanism with S-box splitting for wireless sensor networks. Int J Inf Technol. 2021; 13(3): 933-941. https://doi.org/10.1007/s41870-021-00626-w
14. Nabil M, Khalaf AAM, Hassan SM. Design and implementation of pipelined and parallel AES encryption systems using FPGA. Indones J Electr Eng Comput Sci. 2020; 20(1): 287-299. https://doi.org/10.11591/ijeecs.v20.i1.pp287-299
15. Asassfeh MR, Qatawneh M, Al Azzeh FM. Performance evaluation of blowfish algorithm on supercomputer IMAN1. Int J Comput Networks

Commun. 2018; 10(2): 43-53. https://doi.org/10.5121/ijcnc.2018.10205
16. Couturier R, Noura HN, Chehab A. ESSENCE: GPU-based and dynamic key-dependent efficient stream cipher for multimedia contents. Multimed Tools Appl. 2020; 79(19-20): 13559-13579. https://doi.org/10.1007/s11042-020-08613-2
17. Fanfakh A, Noura H, Couturier R. ORSCA-GPU: one round stream cipher algorithm for GPU implementation. J Supercomput. 2022; 78(9): 11744-11767. https://doi.org/10.1007/s11227-022-04335-4
18. Fanfakh A, Noura H, Couturier R. Simultaneous encryption and authentication of messages over GPUs. Multimed Tools Appl . 2023;29:1-22. https://doi.org/10.1007/s11042-023-15451-5
19. Alaa Y, Fanfakh A., Hadi E. Parallel Message Authentication Algorithm Implemented Over Multicore CPU. Int. J Intell Eng Syst. 2023; 16(4):642–54. https://doi.org/10.22266/ijies2023.0831.52
20. Aldahdooh RMN, Mahmoud AY. Parallel Implementation and Analysis of Encryption Algorithms.2018:76. https://www.researchgate.net/publication/324747960 .
21. Fanfakh ABM. Predicting the Performance of MPI Applications over Different Grid Architectures. J Univ Babylon Pure Appl Sci. 2019; 27(1):468–77.
22. Sleem L, Couturier R. TestU01 and Practrand: Tools for a randomness evaluation for famous multimedia ciphers. Multimed Tools Appl. 2020; 79(33–34): 24075–88. https://doi.org/10.1007/s11042-020-09108-w

خوارزمية تشفير كتلة خفيفة الوزن المتوازية لوحدات المعالجة المركزية متعددة النواة

حوراء جابر حمزة، احمد بدري مسلم فنفخ

قسم علوم الحاسوب، كلية العلوم للبنات ، جامعة بابل ، بابل ، العراق.

الخلاصة

أصبحت حماية البيانات واحدة من أهم القضايا على الرغم من التقدم الكبير في الاتصالات والتكنولوجيا. لكي ترسل التكنولوجيا المستندة إلى الويب البيانات بسرعة وأمان ، يجب تشفير البيانات. التشفير هو عملية تحويل النص العادي إلى نص مشفر ، لا يستطيع الأشرار قراءته أو تغييره. تتطلب كل من إجراءات تحليل التشفير وفك التشفير قدرًا كبيرًا من الوقت من أجل الحفاظ على المستوى المطلوب من الأمان. ومع ذلك ، طور عدد من الباحثين نهج التشفير بالتوازي من أجل تقليل مقدار الوقت اللازم لإنهاء إجراءات التشفير وفك التشفير. أنتج التحقيق في هذه القضية عددًا من الحلول القابلة للتطبيق. تمكن الباحثون من تحقيق مستويات أداء محسّنة في تقنية التشفير باستخدام التوازي لزيادة الإنتاجية وتعزيز كفاءة طرق التشفير. لتحقيق أداء عالٍ ، تم تقديم خوارزميات تشفير البقعة خفيفة الوزن وتنفيذها على منصات وحدة المعالجة المركزية مع تحسينات مختلفة. وبالتالي ، في هذا العمل ، تم اقتراح مخطط تشفير خفيف الوزن يستخدم جولة واحدة فقط من تقنية تشفير الكتلة التي يتم تطبيقها بالتوازي على معالج متعدد النواة. تستخدم خوارزمية تشفير الرسائل المقترحة كتلتين فرعيتين من 128 بت من الرسائل العادية ومربع الاستبدال و splitmix64 PRNG لتشفير الرسالة العادية والحصول على كتلتين فرعيتين مشفرتين ، مما يجعلها تقنية سريعة لتشفير وفك تشفير كتل الرسائل. بالمقارنة مع الطريقة الحالية. وفقًا لنتائج الأداء ، فإنه قادر على الوصول إلى إنتاجية عالية للبيانات مقارنة ببعض الأساليب خفيفة الوزن الموجودة بالفعل ، مع معدل نقل أعلى من 25 كيكابت في الثانية على وحدة المعالجة المركزية Intel Core i7. تتفوق طريقة التشفير المقترحة على طريقة البقع المتوازية بمعدل 14.10 مرة أسرع عند تنفيذها على وحدة معالجة مركزية متعددة النواة. متوسط التسريع مقارنة بالنسخة التسلسلية للخوارزمية المقترحة والتنفيذ المتوازي لها هو 4.70. أيضًا ، توفر طريقة التشفير المقترحة قدرًا كبيرًا من العشوائية وتجتاز الاختبارات الإحصائية لـ PractRand. وبالتالي ، فإن الطريقة المقترحة هي منافس قوي للتنفيذ عالي الأمان على المعالجات متعددة النواة.

الكلمات المفتاحية: التشفير, التشفير خفيف الوزن, وحدة معالجة مركزية متعددة النواة, التشفير احادي الجولة, حوسبة المتوازية.