Received: May 24, 2023. Revised: June 16, 2023.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Parallel Message Authentication Algorithm Implemented Over Multicore CPU

Yamamh Alaa Alamari[1] Ahmed Fanfakh[1]* Esraa Hadi[1]

[1]*College of science for women, University of Babylon, Babylon, Iraq*
* Corresponding author's Email: ahmed.fanfakh@uobabylon.edu.iq

Abstract: Currently, there are around 4.95 billion people who use the internet, which creates a large audience with an increasing demand for activities that can be done online. Some examples of these activities include social networking, information sharing, and online shopping. Therefore, there is an urgent need for improved levels of secrecy and privacy. It has recently come to light that one of the most serious challenges to the mainstream adoption of business apps is that of online fraud. As a direct consequence of this, issues pertaining to authentic cation, authorization, and identification have emerged as critical considerations in today's open and accessible society. The act of recognizing an entity, whether it be a human, a computer, or a software program, is referred to as the identification process. Authentication and authorization are complementary processes that are used in security systems to decide which persons are permitted to access information resources over a network. Authentication is performed by the user's device, while authorization is performed by the security system. However, a variety of potential solutions have been suggested, one of which is the use of parallelism in order to boost the efficiency of authentication methods. This work proposes a new message authentication algorithm that is applied in parallel over a multicore processor. The proposed message authentication algorithm uses two PNGRs and two substitution boxes to encrypt and authenticate the plain message. The encryption process has a one-round operation, making it a fast technique to encrypt and decrypt blocks of messages. In comparison to the existing method, the proposed authentication method outperforms the parallel speck-based authentication method by an average of 3.27 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 2.99. The proposed method passes the most difficult randomness test, and the obtained MAC values are tested further to meet other security measurements.

Keywords: Message authentication, Cryptography, Parallel computing, Multicore, One-round encryption.

1. Introduction

Over the past two decades, the proliferation of networks has played a pivotal role in facilitating the explosive growth of computer systems, rendering them indispensable in various domains. The imperative need for internet connectivity has escalated, prompting businesses to increasingly invest in network infrastructures. However, this widespread adoption of computer networks has also led to an upsurge in unauthorized access to computer systems. Consequently, the internet, being a powerful platform that revolutionizes modern-day business, has introduced new security threats that permeate every facet of daily life. This emerging landscape underscores the imperative to tread cautiously and adopt effective measures to mitigate these risks. Notably, the proliferation of internet users has grown exponentially in recent times, further cementing its position as a crucial component of contemporary existence [1]. The chaining mode (CCM) method, also referred to as AES-CCM, employs encryption to achieve confidentiality and authentication. Specifically, the confidentiality aspect is achieved through counter mode operation, while the authentication component utilizes chaining mode operation. The dual functionality of AES-CCM renders it a desirable option as it provides a high level of authentication assurance. Given the ever-increasing demand for online financial and personal data collection on digital platforms, ensuring the security of communication networks has become a

crucial imperative. The escalating popularity of real-time services, particularly in satellite communication, has driven scholars to pursue secure communication solutions with faster transmission rates. This has fueled a surge in the reliance on data security algorithms, particularly for authentication services. The authentication algorithm serves to validate user credentials, with the application domain determining the specific security services to be utilized. In applications that require low-level security, the authentication service is sufficient to offer user identity validation [2]. The Advanced Encryption Standard (AES), a widely adopted symmetric encryption method, can simultaneously provide authentication and confidentiality during message transmission [3]. To achieve authentication, an authentication algorithm is employed to generate the message authentication code (MAC) using the plaintext and a secret key. Upon reception of the plaintext, the MAC is used to verify its validity. The received plaintext is considered valid (MACT) if the message authentication code at the receiver (MACR) matches the transmitted MAC [3]. In contrast, to maintain confidentiality, the plaintext is transformed into ciphertext to prevent unauthorized access [3]. Currently, the number of internet users has surpassed 4.95 billion, and the advent of information security has significantly transformed our way of life, driven by the availability and accessibility of information. With activities such as banking and trading primarily conducted online through the development of internet banking and electronic commercial exchanges, safeguarding all forms of data and resources against various security threats has become paramount. These security functions include source authentication, data integrity, and information confidentiality, all of which can be ensured through commonly used cryptographic techniques [1]. On the other hand, researchers accelerate their authentication algorithms by using data-level parallelism. Various parallel platforms were used to fulfil this demand. Accordingly, this work contributes to the following:

1-A new parallel message authentication algorithm designated to explore the features of multicore CPUs is proposed.

2-The proposed message authentication algorithm is based on a new lightweight one-round encryption function that improves the overall performance of the proposed method.

3-The proposed authentication algorithm is compared to well-known lightweight techniques.

2. Related works

Communication security necessitates the use of Secure-hash algorithms (SHAs), with SHA-256 being the most secure and efficient option. Most signature algorithms, including quantum-resistant ones, utilize SHA-256 for this purpose. A recent work presented in [4] proposes a concurrent SHA-256 implementation for hashing hundreds of messages on the Sunway TaihuLight SW26010 many-core processor, one of the world's fastest and most powerful computers. One of the critical components of AES/GCM is the management of the authentication process, which employs Galois HASH (GHASH). In [5], researchers describe a successful parallel-pipelined GHASH hardware architecture that is agnostic to the underlying mainframe. The Karatsuba-Ofman algorithm (KOA) is the name of the algorithm used for multiplying Galois gields (GF). Unlike previous parallel hardware KOA-based systems, the authors of [5] utilized a single reduction array for all parallel multipliers, which resulted in an optimized design in terms of area utilization. In a recent publication [2], a new authentication mechanism called parallel cipher-based message authentication code (PCMAC) was proposed by researchers. The method was designed in a parallel structure, and it is highly efficient for applications that demand high throughput. A comparative analysis was conducted between the suggested technique and MAC-based authentication schemes. The findings of the study demonstrated that the PCMAC method can authenticate data at a speed of 2.99 GB/s. In order to protect the key against pattern-based attacks, the researchers introduced a key adjustment strategy based on the S-Box byte-key transformation. The suggested AMP-MP was implemented on an 8-bit asynchronous 9-core CPU, which was fabricated using a 65nm CMOS technique. The test results revealed that authentication had a throughput of 13.54 GB/s, whereas encryption and authentication together had a throughput of 8.32 GB/s. These values are 17 and 70 times faster, respectively, than the claimed competitor. Furthermore, the secret key was exposed at 5105 traces, which is 17 times more secure than the conventional ASIC AES-CCM implementation. This was based on power dissipation and EM SCA on the suggested AMP-MP [3]. In [6], researchers present a low-latency architecture for AES in the context of Cipher-based message authentication code (CMAC). The architecture employs the block RAM (BRAM) resources available in current FPGAs and utilizes intense parallel processing per cycle. The performance evaluation of the proposed architecture reports a

throughput of 3.8 Gbps, a latency of 10 clock cycles for common IoT applications, and resource utilization of 1355 slices and 2 BRAMs. In [7], a hardware implementation of the CMAC algorithm using an FPGA for satellite communication is proposed. The proposed implementation shows an improvement in the effective utilization of time and FPGA area when compared to earlier implementations. In [8], a parallel chaos-based non-chaining MAC algorithm is presented. The structure of proposed MAC algorithm ensures that the MAC value is always sensitive to the message in the same way. The keystream is tightly coupled with the algorithm key, content, and order of each message block through the use of the changeable-parameter mechanism and self-synchronization mechanism. The proposed algorithm modulates the entire message into a chaotic iteration orbit, and the coarse-graining trajectory is extracted to serve as the MAC value. Theoretical analysis and computer simulation results demonstrate that the suggested algorithm meets the performance requirements of a MAC and can be a promising option in a parallel computing environment. In [9], a novel mode of operation named ZMAC is proposed, enabling the generation of a stateless and deterministic message authentication code (MAC) from a tweakable block cipher (TBC). The suggested architecture provides security beyond the birthday limit with respect to the block-length n, and allows for the processing of n bits plus t bits of inputs for each TBC call when a TBC is utilized with n-bit blocks and t-bit tweaks. To accomplish this, a TBC is employed as a variable-input-length pseudo-random function (PRF). In [10], the concise communication proposes the implementation of Hash-based message authentication code (HMAC) and advanced encryption standard-Galois message authentication code (AES-GMAC) algorithms for GOOSE message integrity. Timing experiments are performed in the laboratory to evaluate their performance and feasibility for GOOSE. GPU computing had used to implement message authentication and encryption of the messages to get superior performance as in [11]. In [12], It has been suggested that the new message authentication algorithm (MAA), which will be known as 'DKEMA', would better accommodate the capabilities and notions of the GPU. The dynamic key-dependent scheme with one round of substitution and diffusion operations forms the basis for this system. The results of the experiments indicate that the suggested approach is quite successful on a Titan V100 GPU; the throughput is more than 400 GB/s. Based on existing literature, a majority of cryptographic techniques employ multiple rounds to ensure the attainment of the desired level of security. Unfortunately, this approach tends to adversely impact the efficiency of both encryption and decryption algorithms. Nonetheless, the utilization of parallelism has emerged as a viable strategy to enhance the performance of cryptography programs. However, certain techniques do not fully harness the inherent capabilities of parallel computing and fail to explore its true potential. In light of these observations, the present study introduces a novel message authentication method specifically designed for parallel multicore CPUs. This method relies on a one-round encryption process, thereby circumventing the performance degradation associated with multi-round techniques. By leveraging the computational and communicative advantages offered by parallelism, the proposed approach significantly enhances the overall performance of the application.

It is worth noting that this advancement contributes to the field of cryptography by addressing the computational challenges posed by traditional methods. The parallelization of the authentication process not only optimizes computations but also improves communication efficiency. Consequently, this research serves as a valuable contribution to the realm of cryptography, offering an effective solution for achieving better performance in cryptographic applications compared to other methods.

3. Background: message authentication and parallel computing

Message authentication is a crucial aspect of data security and is used to verify the integrity and authenticity of a message. It involves the use of a message authentication code (MAC) that is computed using a key and the message [2]. The MAC is sent along with the message, and the receiver can use the same key and the received message to re-compute the MAC. If the re-computed MAC matches the received MAC, the receiver can be confident that the message has not been tampered with or altered in any way as in Fig. 1. Parallel computing, on the other hand, refers to the use of multiple processors or computers to perform a single computation or task. This can significantly improve the performance of the computation and reduce the time required to complete it. The relationship between message authentications and parallel computing lies in the fact that message authentication algorithms can be computationally intensive and can take a significant amount of time to compute, particularly for large messages or in high-throughput scenarios. Parallel computing can be used to speed up the computation

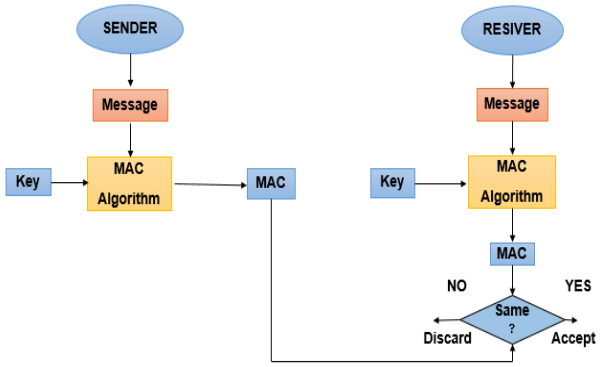


Figure. 1 Message authentication schema

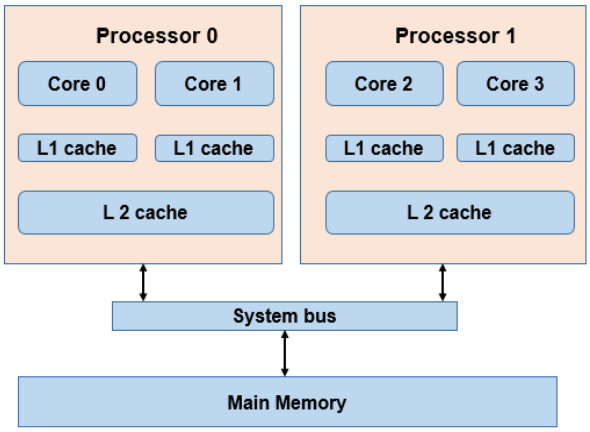


Figure. 2 Architecture of multi-core processor

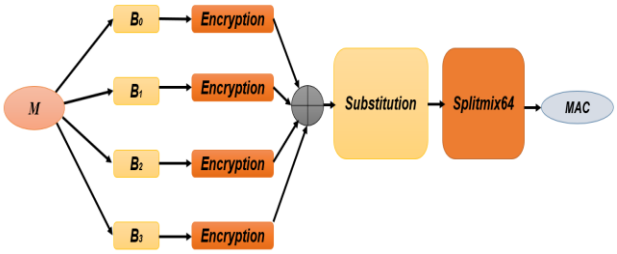


Figure. 3 The Authentication function

of the MAC by dividing the computation across multiple processors or computers. Furthermore, parallelism can be implemented to data level in the authentication algorithm. Using parallel computing can improve the throughput of the authentication algorithm and meet the requirements for MAC selection. This is particularly relevant in applications that require high-speed data transfer and real-time processing, such as satellite communication or financial transactions. In conclusion, the relationship between message authentication and parallel computing is that parallel computing can be used to improve the performance and throughput of the message authentication algorithm, making it more efficient and effective in high-throughput scenarios [1]. In this work, parallel multicore architecture is used to speed up the execution of the proposed message authentication algorithm. A multi-core processor is an integrated circuit that has two or more multipurpose processing cores connected to it with the purpose of increasing performance while simultaneously lowering power consumption, see Fig. 2.

4. Proposed parallel message authentication method

The proposed method which authenticates the message divides it into a number of blocks, the size of $\frac{N}{\mathrm{NP}}$, where N is the size of the message and NP is the number of parallel processes. The message's blocks are evenly distributed among parallel processes. Subsequently, encryption and authentication procedures are performed on each block to derive the local message authentication code (MAC). The PRNG of Splitmix64 and diffusion functions via substitution are employed to enhance the randomness and non-linearity and confusion to the MAC value, see Eq. (1). Following these operations, a global MAC is generated by employing the parallel asynchronous gathering operation on all MAC values. The global MAC is obtained through the XOR operation applied to the aggregated MAC values obtained from all processes, as depicted in Fig. 4. Furthermore, both substitution and splitmix64 techniques are applied again to the final MAC value to augment its level of randomness and nonlinearity.

$$MAC\ (b1,b2,b3,b4) = splitmix64\ (\ substitution\ (c1 \oplus c2 \oplus c3 \oplus c4)\ ) \quad (1)$$

Where b1, b2, b3, b4 are the blocks of a plain message and c1, c2, c3, c4 are the encrypted blocks using the proposed one-round function. For more illustration, Fig. 3 presents the authentication function that corresponds to Eq. (1).

4.1 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

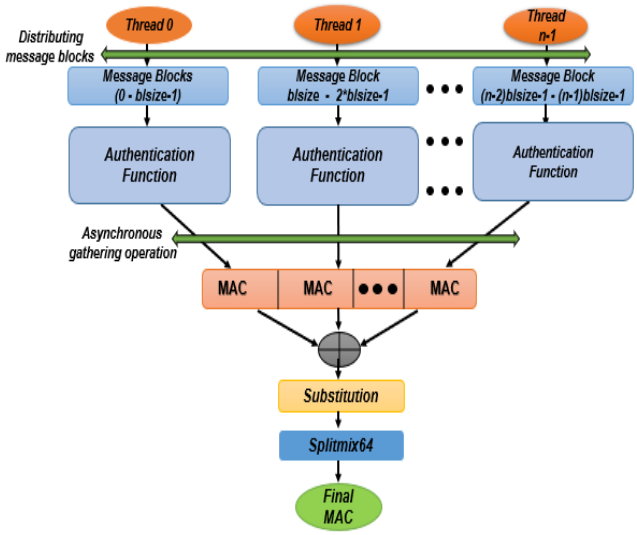


Figure. 4 The structure of the proposed parallel authentication method

Algorithm 1: The proposed one-round cipher

Input:
in: plain block data of size 64-bit
ctr: block counter
Sbox1, **Sbox2**: arrays of size 256 bytes representing substitution boxes
key: array of size 512 bytes
v: array of initial vector of size 1024

Output:
R1: encrypted block of size 64 bit

Start

```
R1 = ((ctr) ^ (((ulong*) key) [ctr & 63])) ^ (xorshift64(v[key[ctr & 511] & 1023]))
uchar*vv =(uchar*)&R1;
vv[0]= Sbox1[vv[0]] + Sbox2[vv[7]]
vv[1]= Sbox1[vv[1]] + Sbox2[vv[6]]
vv[2]= Sbox1[vv[2]] + Sbox2[vv[5]]
vv[3]= Sbox1[vv[3]] + Sbox2[vv[4]]
vv[4]= Sbox1[vv[4]] + Sbox2[vv[3]]
vv[5]= Sbox1[vv[5]] + Sbox2[vv[2]]
vv[6]= Sbox1[vv[6]] + Sbox2[vv[1]]
vv[7]= Sbox1[vv[7]] + Sbox2[vv[0]]
R1= splitmix64(R1)
R1= in ^ R1
Return R1
```

End

manner depicted in Fig. 4, and it is outlined in the Eq. (2) as follows:

$$DK = (IV \oplus CTR \oplus key) \quad (2)$$

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 [12].

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

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 as in [13]. 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 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 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,
4. 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.

4.2 Encryption method

The Algorithm 1 is designed to perform one-round encryption on a given input message, represented by the variable *in*, using a set of operations. The input is processed by applying bitwise XOR operations with key, the block counter CTR and random values generated by a pseudorandom number generator (PRNG) using xorshift64 that apply to initial vector v and the key, see Algorithm 2. The resulting value is stored in the variable *R1*. Next, the algorithm applies a substitution-permutation network (SPN) to R1 using two substitution boxes, represented by the arrays *Sbox*1 and *Sbox2*. The SPN consists of eight steps of substitution, where each step operates on a byte of *R1*. The S-boxes map each byte of *R1* to a new value based on a lookup table.

Algorithm 2: xorshift64
Input: state (pointer to a 64-bit unsigned integer) **Output**: x (64-bit unsigned integer) **Start** *1. Set x as the value pointed to by state.* *2. Perform the XOR operation between x and the left shift of x by 13 bits.* *3. Perform the XOR operation between the result of step 2 and the right shift of x by 7 bits.* *4. Perform the XOR operation between the result of step 3 and the left shift of x by 17 bits.* *5. Update the value pointed to by state with the value of x.* *6. Return the value of x.* **End**

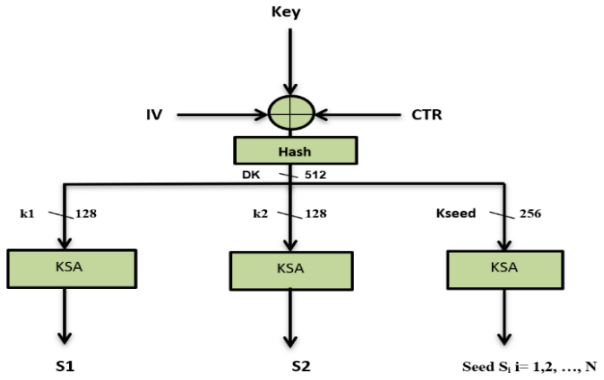


Figure. 5 Displays the proposed dynamic key generation and construction cryptographic primitives

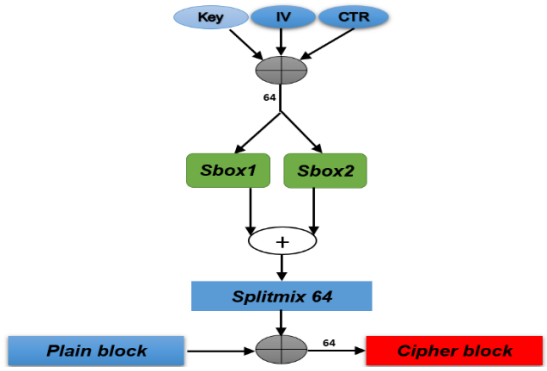


Figure. 6 Proposed lightweight encryption function

The resulting value is sent to splitmix64 PRNG function and the output stored back in R1. Overall, the algorithm combines the PRNG, bitwise XOR operations, S-boxes, and a final XOR operation to perform encryption on the input *in* as illustrated in Fig. 6.

4.2.1. Splitmix64

The splitmix64 PRNG has a number of benefits, the most notable of which are its quick execution time and straightforward implementation. A quick splitable PRNG is provided by the"Splitmix64" algorithm, which may be found in Algorithm 3. A 64-bit state is for both the input and the output of this function. Moreover, it is not recommended for use in cryptography or security applications since the sequences of pseudo-random values that are created may be predicted. This is due to the mixing functions being readily invertible, and the fact that two consecutive outputs are all that is required to reconstruct the system's internal state. In the method that has been given, Splitmix-64 is used with dynamic seeds in addition to a non-linear operation known as confusion. This non-linear operation is the substitution and mixing that is accomplished via the usage of non-invertible binary diffusion. In this instance, Splitmix-64 makes use of a shift/rotate-based linear transformation that has been meticulously built. This assures that there will be a large decrease in latency as well as a high degree of randomization when using the related resources [12].

Algorithm 3 Splitmix64
Input: state (pointer to a 64-bit unsigned integer) **Output**: z (64-bit unsigned integer) **Start** *1. Add UINT64_C(0x9E3779B97F4A7C15) to the value pointed to by state, and assign the result to z.* *2. Perform the XOR operation between z and the right shift of z by 30 bits.* *3. Multiply the result of step 2 by UINT64_C(0xBF58476D1CE4E5B9), and assign the result back to z.* *4. Perform the XOR operation between z and the right shift of z by 27 bits.* *5. Multiply the result of step 4 by UINT64_C(0x94D049BB133111EB), and assign the result back to z.* *6. Perform the XOR operation between z and the right shift of z by 31 bits.* *7. Return the value of z.* **End**

4.2.2 Speck lightweight cryptography

The national security agency (NSA) disclosed Speck, a lightweight block cipher family, to the public in June 2013. The overall software performance of Speck has been enhanced. As can be seen in Fig. 7, the Speck cipher is an implementation of the add-rotate-xor (ARX) encryption. Different

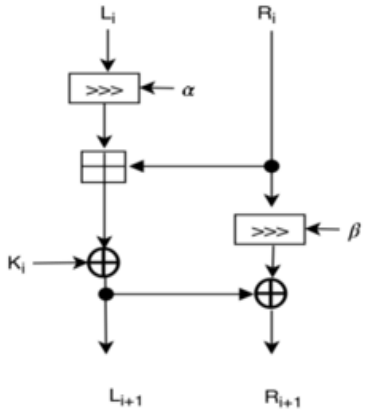


Figure. 7 Lightweight speck cipher

Algorithm: 4 Parallel authentication algorithm (PAA-256)

Input:
data*: array of uint64_t data blocks*
v*: initial vector of type uint64_t*
size*: size of data array*
numtasks*: number of parallel tasks*
id*: Index of the current task*
Sbox1*: array of 256 bytes used for S-box substitution*
Sbox2*: array of 256 bytes used for S-box substitution*
DK*: array of bytes used for encryption key*
Master*: master thread that set to 0*
lmac*: array of four uint64_t values used to store the local MAC of a thread*
all-mac*: array of size 4*numtasks of type uint64_t*
Output:
final_mac: *array of size 256 bits*
Start

```
1.Set lmac[0], lmac[1], lmac[2], and lmac[3] to 0.
  threadbl = ceil (size / numtasks)
  start = id * threadbl
  end = start + (threadbl- 1)

2. for (int i= start; i < end; i+=4)
{
    // Encrypt each data block using the encrypt
      function:
     r1 = encrypt(data[i], i, Sbox1, Sbox2, DK, v)
     r2 = encrypt(data[i+1], i+1, Sbox1, Sbox2,
  DK, v)
     r3 = encrypt(data[i+2], i+2, Sbox1, Sbox2,
  DK, v)
     r4 = encrypt(data[i+3], i+3, Sbox1, Sbox2,
  DK, v)
     t = r1 ^ r2 ^ r3 ^ r4
     uchar *rr=(uchar*)&t;
     for (j = 0; j < 8; j++)
         rr[j]=Sbox1[rr[j]];

     t = splitmix64(t)
     lmac[0] ^= r1 ^ t
     lmac[1] ^= r2 ^ t
     lmac[2] ^= r3 ^ t
     lmac[3] ^= r4 ^ t
}
3.Applying an asynchronous gathering operation
  for
    all mac values from all threads
  MPI_Igather (lmac, 4, MPI_INT64_T, all_mac,
  4,
  MPI_INT64_T, MASTER,
  MPI_COMM_WORLD)
4.if (id==MASTER)
     Apply the authentication steps for the
     all_mac array as in the steps b, c and d
     while storing the results in the final_mac
  array
5.Return final_mac array
End
```

speck block ciphers use different-sized keys and blocks. Speck128/128, Speck128/192, and Speck128/256 are the most often used variants in the CTR mode. The cipher's block size and key size are reflected in the corresponding numbers in the names. The Speck cipher, consisting of a 2n-bit block and *wn*-bit key, is written as Speck2n/wn. Combining distinct procedures on n bits of text constitutes the round functions of the cipher [14]. Thus, in this work speck 128/128 employing CTR mode are used in the comparison of the results.

4.3 Parallel authentication algorithm

The algorithm 4 takes *data* as an array of plain data of 64-bit integers, a 64-bit integer array *v*, two 8-bit unsigned character arrays *Sbox1* and *Sbox1*, a character array *DK*, four 64-bit integer pointers *lmac*, an integer size, an integer *numtasks*, and an integer *id* representing parallel thread index. It calculates a message authentication code (MAC) for the given data. The algorithm initializes four 64-bit integer pointers *lmac* to 0. It then determines the number of blocks *numtasks* in the data array of dimension *size* and divides it by *numtasks* to determine the block size *threadbl*. The start and end indices are then calculated for each thread. The algorithm then iterates through the blocks of data for each thread, with a step size of 4. For each block, four 64-bit integers *r1*, *r2*, *r3*, and *r4* are generated by calling the encrypt function with the appropriate parameters. These four integers are then XORed together to generate a temporary 64-bit

integer *t*. The eight bytes of *t* are then passed through Sbox1 to obtain eight new bytes, which are stored back into *t*. The splitmix64 function is then called with *t* as the input to generate a new 64-bit integer, which is also XORed with each of the four *r* integers, and the resulting four integers are XORed with the appropriate element of the *lmac* array. The resulting *lmac* array contains the MAC for the given data. Fig. 4 gives an illustration for the proposed parallel authentication function. An asynchronous gathering operation is used to gather all *lmac* arrays from all threads into the *all_mac* array. In the master thread, *thread 0*, authentication steps are applied to compress the *all_mac* array to the *final_mac* array. The implemented authentication steps are the same as those used in thread-level authentication without using encryption.

5. Security analysis of the proposed MAC

For the purpose of ensuring the safety and security of the proposed authentication scheme, well-known attacks such as statistical, linear, differential, or brute-force assaults are put through their paces. Extensive testing is carried out in this section to demonstrate that the proposed authentication scheme is secure and reliable. It is important to note that the suggested system may be used for any form of data; however, in the following, only the results for multimedia contents will be presented.

5.1 Statistical analysis tests

Randomness and uniformity are two criteria necessary authentication and encryption against statistical attacks. The following statistical integrity tests are carried out to evaluate the level of randomness, analysis of the PDF, histogram, entropy, and the correlation of the original and encrypted messages [15].

5.1.1. Histogram analysis

In order test the uniformity of proposed one-round encryption used in the authentication method, the histogram of the encrypted message must have a normal distribution. This indicates the occurrence of each symbol is proportionate to the total number of symbols. Equal to or less than the ratio of message size to the n umber of symbols. A histogram is shown in Fig. 8 (c) and (d), contrasting the 512 x 512 symbol plain-message with their cipher-message equivalents. It shows that the histogram of encrypted messages is quite close to a normal distribution. In our case, the histogram of the encrypted message is close to (512x512)/256, which equals 1024.

5.1.2. Uniform security test

The probability density function, also known as the PDF, of the encrypted message has to be uniform in order for this test to pass, since it is the test that is regarded as having the highest weight. When the ciphertext is generated, each symbol that is included in it has a probability of occurrence that is pretty close to being equal to *1/n*, where *n* is the total number of symbols that are used. In Fig. 8 (a) and (b), both the PDF of original and encrypted message is presented. The probability density functions (PDFs) of the encrypted messages may be thought of as being close to the uniform distribution, with a value that is close to 0.039 ($1/256 = 3.9\times10^{3}$) for all of the ciphertext symbols.

5.1.3. Information entropy analysis

Entropy of the information contained in a particular communication. *h* is a metric that may be defined as the following:

$$h(m) = -\sum_{i=1}^{h2} p(mi)\log_2\frac{1}{p(mi)} \tag{3}$$

It is used to assess the degree of uncertainty associated with a random variable. Bits are the unit of measurement for entropy, $p(m_i)$ is the chance that the symbol m_i will appear, and *NS* is the total number of symbols. Entropy is measured in bits. If the entropy of the ciphertext is the same as or very near to $\log_2(NS)$, then it may be considered as coming from a genuine random source that has a normal distribution. Fig. 9 depicts an examination of the cipher message (encrypted messages') entropy performed at the submatrix level with a dimension of *16 x16* (256 elements) while making use of a random dynamic key. This was done in order to complete the task. In addition, the data that were obtained show that the cipher message that were created have an entropy that is equivalent to the value that was wanted, which was 5. Thus, the encryption scheme that was presented is sufficiently safe against any entropy attack that may be used. The obtained results indicate that the produced cipher message have an entropy of uniform form which is the best. The proposed encryption scheme is sufficiently secure against any given entropy attack.

5.1.4. Randomness test using PractRand

Both the one round cipher and the proposed authentication algorithm were put to the test with 1000 seeds using Practrand [15], and they passed every test with flying colures. This was noted above.

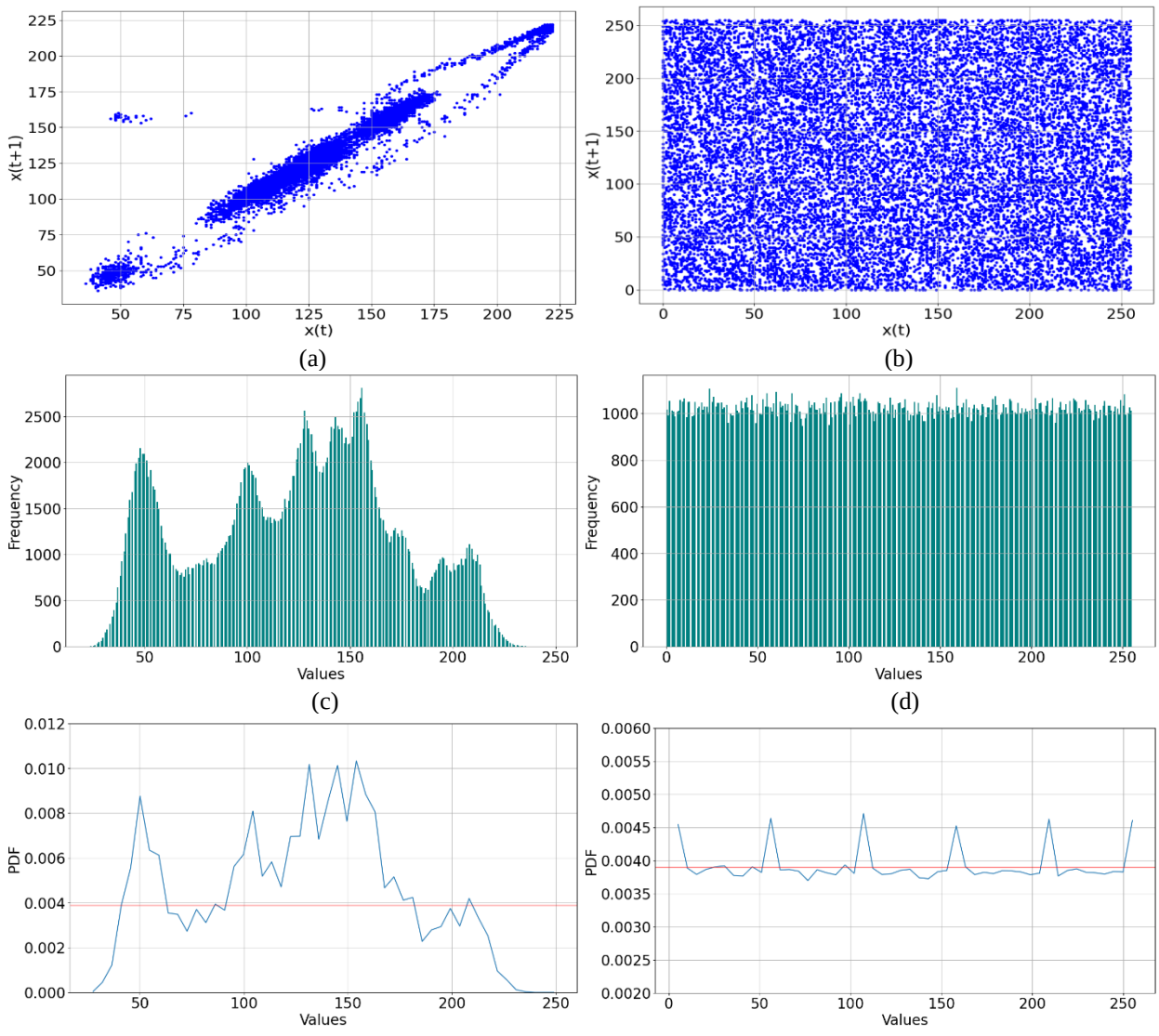


Figure. 8 The recurrence of the original message: (a) Cipher message, (b) Produced by using the proposed stream cipher for a random dynamic key, (c) Histogram of the plaintext, (d) Histogram of the cipher message, (e) The corresponding PDF of the original message, and (f) Pdf of the cipher message

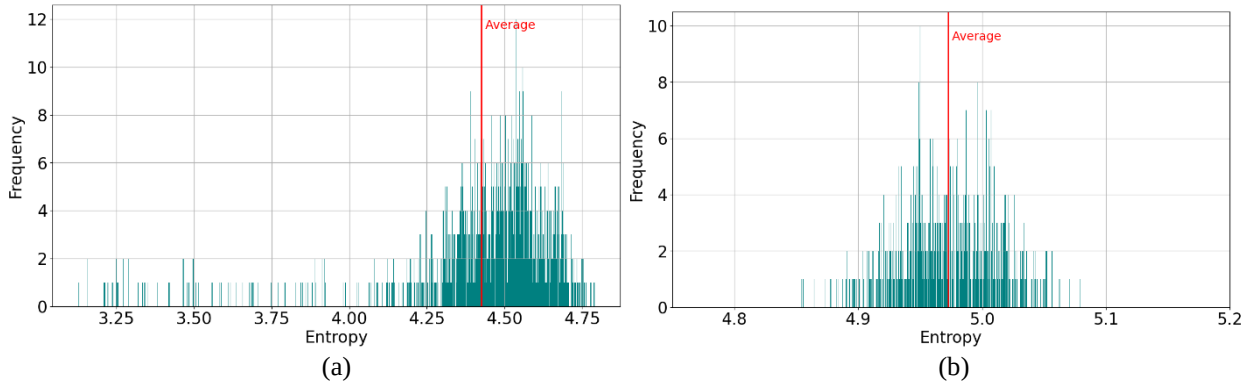


Figure. 9 The entropy of the: (a) plain and (b) cipher message

In point of fact, a message with dimensions of *512 x 512* is the only one that is used. All of the message's components are initialized to zero, and the key is only set up once, at the very beginning. Additionally, each other variable is only initialized a single time. PractRand is one of the most difficult statistical tests, were applied in order to analysis the cipher message

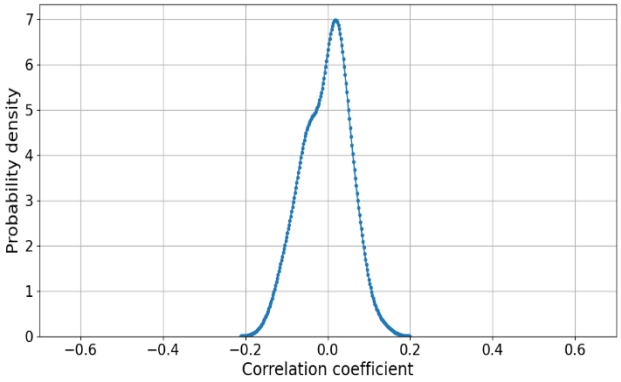


Figure. 10 Distribution of the correlation coefficient between the plain and the encrypted message for 1,000 randomly generated keys

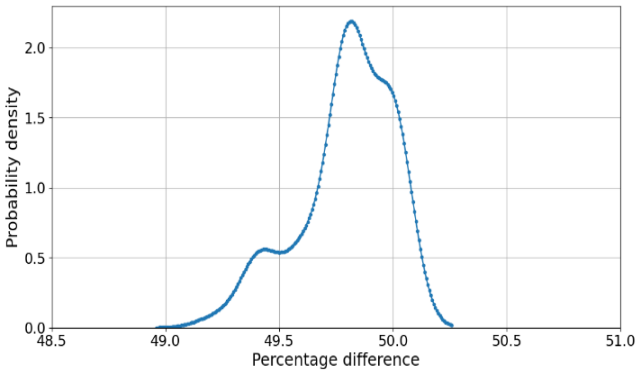


Figure. 11 The percentage difference of the sensitivity test between 1000 MAC values generated by changing randomly one bit in the key for the same message

that was generated as a consequence. Through these examinations, it will be possible to ascertain whether or not the keystream that was generated fulfils the necessary randomization and uniformity standards. Moreover, practrand test suite focuses on detecting specific types of non-randomness that may go undetected by more traditional statistical tests. PractRand employs a variety of statistical tests, including gap tests, birthday spacings tests, and others. It also provides measures of statistical quality, such as the number of "TB" (terabytes) of output it would take to detect a certain level of non-randomness [16].

5.1.5. Independence

To make sure the suggested encryption system is secure, it is crucial to get rid of any connection between the sequences of the components [15]. When the correlation coefficient is small (very close to zero), it indicates that the cipher scheme is quite random. Selecting pairs of nearby pixels from the original message and their matching encrypted ones is how the correlation test is carried out. Directions in the horizontal, vertical, and diagonal planes are all supported by the correlation. The following equations may be used to get the r_{xy} correlation coefficient:

$$r_{xy} = \frac{\mathrm{cov(x,y)}}{\sqrt{D(x)\times D(Y)}} \quad (4)$$

$$\mathrm{cov(x,y)} = \frac{1}{n}\sum_{i=1}^{n}(x_i - E(x))(y_i - E(y)) \quad (5)$$

$$E_{x=}\frac{1}{n}\times\sum_{i=1}^{n}x_i \quad (6)$$

$$D_{x=}\ \frac{1}{n}\times\ (\sum_{i=1}^{n}(x_{i-}\ \mathrm{E(x)})^2 \quad (7)$$

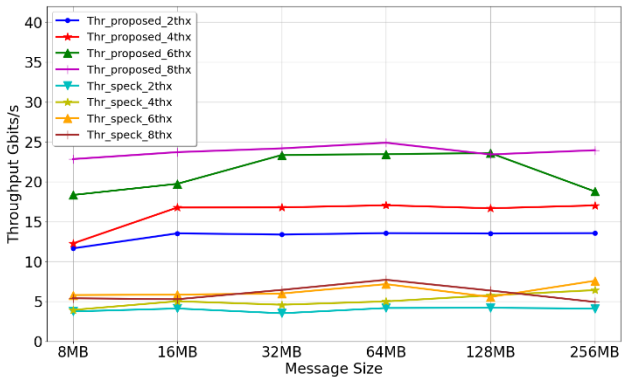
The present study employed a method wherein a thousand random keys were used one at a time to encrypt messages, and the results of a correlation test between the original and encrypted messages are illustrated in Fig. 10. The outcomes suggest that the correlation coefficient between the messages is negligible, indicating that the ciphertext possesses a high degree of randomness and autonomy.

5.2 Sensitivity test

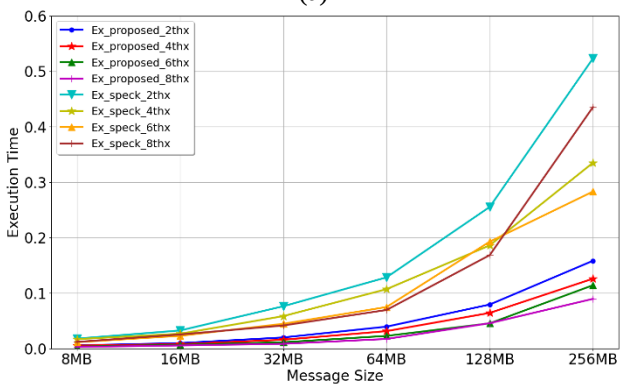
Differential attacks concentrate on examining the link between two encrypted communications that emerges from a little change, such as a one-bit difference between two original messages. The sensitivity tests must demonstrate that even a tiny alteration to the plain message or key can modify the MAC value and result in a different one. The sensitivity of the suggested authentication system improves with increasing differences. Fig. 11 demonstrates that for *Nb =256* and similarly for larger values of *Nb*, the difference between the MAC values is between 1000 MAC values for the same message, where each value is created by randomly modifying a single bit of the key. The obtained results show that at least *Nb/2* bits and thus, 50% of MAC bits are changed. However, the proposed MAC ensured high unpredictability against statistical attacks.

6. Performance evaluation and comparison results

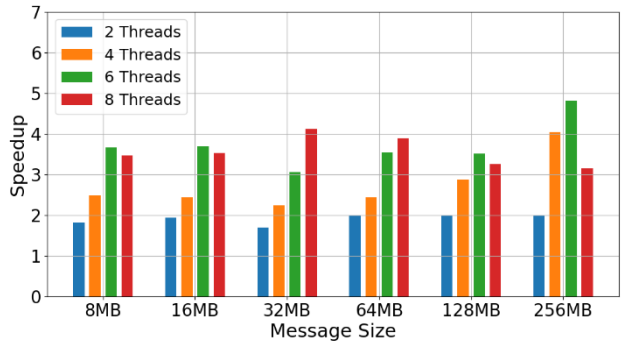
In this section, the performance of the proposed parallel authentication algorithm is presented in terms of throughput, execution time, and speedup. The proposed method has multiple steps to compress the data message to a 256-bit MAC value. It depends on the one-round lightweight encryption method that produced a 64-bit block at one round operation. However, in this section, it is important to evaluate



(a)



(b)



(c)

(d)

Figure. 12 Performance outcomes of the proposed and speck-based authentication algorithms. The results are showcased across four metrics: (a) execution time, (b) throughput, (c) speedup achieved by employing the proposed method in a parallel fashion compared to its sequential counterpart, and (d) speedup observed when comparing the parallel versions of the speck-based and proposed authentication algorithms

the performance of the proposed authentication method when using different lightweight ciphers. The Speck encryption method is used in comparison to the proposed encryption to produce a MAC value. To fairly compare both methods, a lightweight speck of 128 block size and 128 key length is used. The call of speck inside algorithm 4 can be done twice to produce four encrypted blocks of size 64 bits. The proposed parallel authentication and authentication-based Speck algorithms are evaluated on an Intel multicore (R) i7-7700HQ CPU to determine their performance. The frequency of each core is 2.8 GHz. The processor has 4 physical cores and 8 logical ones. The conducted experiments investigate three performance indicators by utilizing various numbers of threads (2, 4, 6, and 8) in conjunction with different message sizes (8, 16, 32, 64, 128, and 256 megabytes). Fig. 12 (a) illustrates the execution time of both algorithms when executed in parallel across multiple threads. It is observed that the proposed algorithm exhibits lower execution times in comparison to the speck-based authentication algorithm. Additionally, Fig. 12 (b) presents the throughput results, demonstrating superior throughput for the proposed parallel authentication algorithm when compared to alternative speck approach. Another crucial factor for evaluating performance in parallel computing is the speedup. Speedup is defined as the ratio of the execution time of a sequential application to the execution time of its parallel counterpart, and it provides insights into the acceleration achieved when comparing different speeds [17]. In Fig. 12 (c), we present the speedup ratio between the sequential and parallel authentication algorithms. Across all thread scenarios, the parallel algorithm consistently outperforms the sequential version by an average speedup factor of 2.99. Additionally, Fig 12 (d) showcases the speedup ratios for the parallel implementations of the proposed authentication algorithm and the speck-based authentication algorithm. The average speedup achieved in this context is 3.27. Consequently, the proposed method exhibits a reduced execution cost compared to the speck-based authentication algorithm. Finally, Table 1 summarizes the average of the comparison results over all message sizes for execution time and throughput.

7. Conclusion

In conclusion, this research paper introduces a pioneering framework for parallel message authentication employing a new lightweight cipher.

Our methodology entails segmenting the message into blocks of 64 bits each, subsequently executing encryption and authentication operations within a single round. Empirical assessments reveal that our proposed approach surpasses alternative techniques, including the Speck method, which necessitates multiple rounds and consequently exhibits reduced speed and performance in contrast to our approach. On average, the proposed authentication technique is 3.27 times quicker than the parallel speck-based authentication approach when run on a multicore CPU. The proposed method is improved upon by a factor of 2.99 when compared to its sequential counterpart when implemented in parallel.

Table 1. The comparison results

#threads	Average overall message sizes			
	Execution time (s)		Throughput Gbits/s	
	Proposed	Speck Auth.	Proposed	Speck Auth.
2	0.052	0.172	13.20	3.97
4	0.041	0.121	16.10	5.12
6	0.034	0.104	21.05	6.32
8	0.028	0.125	23.99	5.85

Furthermore, the proposed authentication method, along with its encryption technique, has successfully passed the rigorous randomness test known as PractRand. This achievement underscores the robustness and reliability of the proposed approach. As a potential avenue for future research, the application of the proposed methods on many core GPUs holds promise for further enhancing performance and achieving accelerated processing speeds.

Acknowledgment

The researchers would like to thank the University of Babylon (Iraq) / College of Science for Women for supporting this work.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

In the research study titled "Parallel Message Authentication Algorithm Implemented Over Multicore CPU," the contributions of the authors are as follows. Yamamh Alaa Alamari played a significant role in the conceptualization and methodology of the algorithm. Ahmed Fanfakh, the corresponding author, contributed to the methodology, software implementation, validation, writing, and project administration. Esraa Hadi provided support in the data curation and visualization aspects of the research. Together, these authors collaborated to develop and implement the parallel message authentication algorithm over a multicore CPU**.**

References

[1] A. Alaa, Y. Yamamh, A. Fanfakh, and E. Hadi, "A Survey of Parallel Message Authentication and Hashing Methods", *Journal of University of Babylon for Pure and Applied Sciences*, Vol. 1, No. 1, pp. 100-110, 2023.

[2] S. J. H. Pirzada, A. Murtaza, and J. Liu, "The Parallel CMAC Authentication Algorithm", In: *Proc. of IEEE 11th International Conference on Communication Software and Networks (ICCSN)*, pp. 1-5, 2019.

[3] A. A. Pammu, S. Z. Ahmed, M. S. S. Ali, and K. S. Esam, "A high throughput and secure authentication-encryption AES-CCM algorithm on asynchronous multicore processor", *IEEE Transactions on Information Forensics and Security*, Vol. 14, No. 4, pp. 1023-1036, 2018.

[4] Z. Wang, Y. Li, Z. Cao, and Z. Wu, "Parallel SHA-256 on SW26010 many-core processor for hashing of multiple messages", *The Journal of Supercomputing*, Vol. 79, No. 2, pp. 2332-2355, 2023.

[5] K. M. Abdellatif, R. C. Avot, and H. Mehrez, "Efficient parallel-pipelined GHASH for message authentication", In: *Proc. of International Conference on Reconfigurable Computing and FPGAs*, pp. 1-8, 2012.

[6] I. B. Dhaou, S. Belhassan, and F. Drira, "Low-latency hardware architecture for cipher-based message authentication code", In: *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1-4, 2017.

[7] S. J. H. Pirzada, A. Murtaza, and J. Liu, "Implementation of CMAC Authentication Algorithm on FPGA for Satellite Communication", In *Proc. of IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pp. 1-5, 2019.

[8] D. Xiao, Q. Fu, and T. Xiang, "A Non-Chaining Message Authentication Code Based on Chaos", *IEICE Proceeding Series*, Vol. 47, No. 10, pp. 550-553, 2015.

[9] T. Iwata, K. Nei, and K. Sasaki, "ZMAC: a fast tweakable block cipher mode for highly secure message authentication", In: *Proc. of Advances*

in Cryptology – CRYPTO 2017, pp. 559-588, 2017.

[10] S. M. Hussain, S. M. Farooq, and T. S. Ustun, "Analysis and implementation of message authentication code (MAC) algorithms for GOOSE message security", *IEEE Access*, Vol. 7, pp. 80980-80984, 2019.

[11] A. Fanfakh, H. Noura, and R. Couturier, "Simultaneous encryption and authentication of messages over GPUs", *Multimedia Tools and Applications*, pp. 1-20, 2023.

[12] H. N. Noura, R. Couturier, O. Salman, and K. Mazouzi, "DKEMA: GPU-based and dynamic key-dependent efficient message authentication algorithm", *Journal of Supercomputing*, Vol. 78, No. 12, pp. 14034-14071, 2022.

[13] R. Couturier, H. N. Noura, and A. Chehab, "ESSENCE: GPU-based and dynamic key-dependent efficient stream cipher for multimedia contents", *Multimedia Tools and Applications*, Vol. 79, Nos. 19-20, pp. 13559-13579, 2020.

[14] L. A. Muhalhal and I. S. Alshawi, "A hybrid modified lightweight algorithm for achieving data integrity and confidentiality", *International Journal of Electrical and Computer Engineering (IJECE)*, Vol. 13, No. 1, pp. 833-841, 2023.

[15] A. Fanfakh, H. Noura, and R. Couturier, "ORSCA-GPU: one round stream cipher algorithm for GPU implementation", *The Journal of Supercomputing*, Vol. 78, No. 9, pp. 11744-11767, 2022.

[16] C. D. Humphrey, "PractRand: A Practical Random Number Test", 2007. [Online]. Available: http://pracrand.sourceforge.net/.

[17] A. B. M. Fanfakh, "Predicting the Performance of MPI Applications over Different Grid Architectures", *Journal of University of Babylon for Pure and Applied Sciences*, Vol. 27, No. 1, pp. 468-477, 2019.