Multimed Tools Appl (2017) 76:22703–22724
DOI 10.1007/s11042-016-4333-y

A resource-efficient encryption algorithm for multimedia big data

Shadi Aljawarneh[1] · **Muneer Bani Yassein**[1] · **We'am Adel Talafha**[1]

Received: 19 July 2016 / Revised: 24 November 2016 / Accepted: 29 December 2016 /
Published online: 23 January 2017

Abstract Nowadays, multimedia is considered to be the biggest big data as it dominates the traffic in the Internet and mobile phones. Currently symmetric encryption algorithms are used in IoT but when considering multimedia big data in IoT, symmetric encryption algorithms incur more computational cost. In this paper, we have designed and developed a resource-efficient encryption system for encrypting multimedia big data in IoT. The proposed system takes the advantages of the Feistel Encryption Scheme, an Advanced Encryption Standard (AES), and genetic algorithms. To satisfy high throughput, the GPU has also been used in the proposed system. This system is evaluated on real IoT medical multimedia data to benchmark the encryption algorithms such as MARS, RC6, 3-DES, DES, and Blowfish in terms of computational running time and throughput for both encryption and decryption processes as well as the avalanche effect. The results show that the proposed system has the lowest running time and highest throughput for both encryption and decryption processes and highest avalanche effect with compared to the existing encryption algorithms. To satisfy the security objective, the developed algorithm has better Avalanche Effect with compared to any of the other existing algorithms and hence can be incorporated in the process of encryption/decryption of any plain multimedia big data. Also, it has shown that the classical and modern ciphers have very less Avalanche Effect and hence cannot be used for encryption of confidential multimedia messages or

✉ Shadi Aljawarneh
saaljawarneh@just.edu.jo

Muneer Bani Yassein
masadeh@just.edu.jo

We'am Adel Talafha
Watalafha13@cit.just.edu.jo

[1] Faculty of Computer and Information Technology, Jordan University of Science and Technology, Irbid, Jordan

confidential big data. The developed encryption algorithm has higher Avalanche Effect and for instance, AES in the proposed system has an Avalanche Effect of %52.50. Therefore, such system is able to secure the multimedia big data against real-time attacks.

Keywords Internet of things · Multimedia big data · Encryption · AES · GPU

1 Introduction

Currently, the emergence of a new dimension of communication between people and objects, and between objects with each other is called “Internet of Things” (IoT). It is widely spread in several areas, such as industries, academia, wireless networking and communications [20]. The term “Internet of Things” (IoT) can be defined as a collection of objects that have unique addresses which communicate with each other through a network [3, 16].

This technology is composed of two parts, “Internet” and “Things”. The phrase “Things” can be anything in the real world, whether alive or not alive [8, 9, 12, 17]. With the immediate access to information, which is related to the real world and things, there has been a marked improvement in services, productivity, and performance [4]. Consequently, numerous applications have been developed depend on the IoT techniques, such as smarter transportation systems, smarter enterprises, smart grid, factories and multimedia sensing [3]. Experts estimated that by 2020, IoT technology will be encompassed to 50 billion objects [6]. With IoT growth and widespread rate, current processing capacity presents a new obstacle for IoT development as they cannot cope with the massive amount of data generated. So, a big data technology is needed to improve IoT development [5].

Generally, big data can be defined as “the Information asset characterized by such a high volume, velocity and variety to require specific technology and analytical methods for its transformation into value” [5]. However, the big data produced by IoT has several features comparable to general big data due to the various kinds of data that have been collected, such as heterogeneity, un-structure, and high redundancy [5]. As part of big data, multimedia are increasingly becoming the “Biggest big data” as it currently generates 60% of Internet traffic and 70% of mobile phone traffic [23]. One of the most important issues that faces multimedia big data is a security.

Many security systems have been developed to keep the communication channels safe in IoT. One category of these systems employs public-key exchange, such as AES, 3-DES, DES, and Blowfish. Public key exchange is based on a key that can be created using time and mathematical processes. After that, the data can be encrypted and sent using Internet insecure channels. The strength of any symmetric key encryption system depends on the size of the key used [21, 24, 27]. Symmetric key encryption algorithms are suitable for IoT environment; since they need low computational cost and consumes less energy. But, with the huge data introduced by multimedia big data, symmetric encryption algorithms suffer from performance degradation due to the need of frequent key update mechanism.

There is a little research and technical reports in the academia and industry in the security of multimedia big data [1, 2, 22]. In addition, there is no an real-time

encryption system to deal with the multimedia big data. Therefore, we have developed a resource-efficient encryption system that is suitable for multimedia big data. The proposed system attempts to offer a multi-level security model where it consists of a Feistel Encryption System (FES) which is used to manipulate the blocks through the shift and rotate operations [1, 2, 22], AES symmetric key algorithm, and Genetic Algorithms (GA). Note that test evaluation has been conducted on real IoT data that is the medical multimedia big data to the hospital of JUST university, Jordan.

In brief, the encryption phase in such system reads the input files in the form of multi-size blocks. Each block is divided into plaintext and key. The key is ciphered using Feistel Encryption System. Then, the plaintext is encrypted using the ciphered key by AES to generate the cipher text. Finally, the ciphered text and the ciphered key are integrated using genetic algorithm. On the other hand, the decryption phase is the reverse of the encryption process. Using this proposed system, there is no need for key distribution and update mechanism since the key is generated from the data itself. The data to be encrypted using AES is only half of the original one (since the other half is encrypted using FES as a key).

The proposed system is tested on high-performance machine equipped with GPU to handle different real tests to benchmark the symmetric encryption algorithms such as MARS, RC6, 3-DES, DES, and Blowfish in terms of encryption and decryption time, throughput and Avalanche Effect. A GPU executes most of the computations, computer graphics, and image processing extremely efficient compared to a CPU. Furthermore, GPU provides many parallel processing units which are ideal for throughput computing.

The rest of the paper is organized as follows: Section 2 discusses the related work. In Section 3, we have offered the proposed efficient-resource encryption system and its architecture framework. The results and discussions have been showed in Section 4. Finally, Section 5 draws the conclusions and further work.

2 Related works

Many approaches and systems have been proposed to deal with the limitations and constraints of encrypting transmitted data in the IoT. Furthermore, there is a vital need to study further the security of multimedia big data. In this section, we have shown a number of current security protocols and approaches that used to secure the multimedia big data.

Y. Zhao [30] proposed an Intelligence Service Security Application protocol (ISSAP) in order to use a custom data packet encapsulation mechanism and a cross-platform communication combined with a USPIOT platform for encryption and business communication, and signature and authentication algorithms. The author did not provide specific details about the encryption process but, in general, most USPIOT platforms use traditional encryption algorithms which suffers from overhead due to keys creation and management.

Sundaram, B. Vinayaga, et al. [25] introduced a combination of encryption and hash algorithms for IoT. They proposed an encryption algorithm that takes 64 bits' cipher text and 128 bits' master key for 32 operational rounds. The hash algorithm is applied for padding, compression, and truncation. This algorithm cannot deal the huge

data generated by multimedia big data, since it only can encrypt 64 bits per block, and however, it is very slow in terms of performance. The algorithm also suffers from extra overhead represented by key generation and distribution processes.

M. Xin [28] presented a hybrid approach that combines the benefits of both symmetric and asymmetric encryption algorithms to deal with IoT requirements of high security and low computation complexity. The hybrid approach uses the Advance Encryption Standard (AES) and the Elliptic Curve Cryptography (ECC); where AES is considered to be simple, fast, and reliable encryption algorithm for long plaintexts encryption and elliptic curves are used as a digital signature and key management. Although using AES is considered to be fast, key management and digital signature processes are encrypted using elliptic curves which considered to be complex and slow, thus adding extra overhead to the hybrid approach.

K. N. Prasetyo, et al. [19] suggested the use of symmetric encryption algorithm to be adopted in IoT. Therefore, they proposed an implementation of the Blowfish encryption algorithm on FPGA resource and program it using the VHDL language. The proposed implementation was evaluated by measuring performance metrics such as security, encryption time, avalanche effect, and throughput and found to provide a good performance. The evaluation of this implementation was only conducted on text inputs and did not consider multimedia input in the evaluation.

X. Yao, et al. [29] provided a scheme that defines lightweight no-pairing attribute-based encryption policy that is managed by attribute authority. The scheme is based on a combination of elliptic curve cryptography (ECC) and key policy (KP-ABE). This is a suitable for IoT since it is a lightweight scheme, but it suffers from some limitations like poor flexibility in revoking attribute, poor scalability, and poor generality. Another lightweight encryption algorithm was implemented by J. Y. Lee, et al. [15]. Such encryption algorithm based on XOR manipulation was implemented on hardware to enhance a typical RFID system that is used in IoT. The XOR manipulation is used for anti-counterfeiting and privacy protection.

Another novel approach for IoT was offered by P. N. Mahalle, et al. [18] namedthe Identity Authentication and Capability-based Access Control (IACAC). It uses public key approach that is consistent with the limitation of IoT. The approach also introduces a timestamp technique for authenticating the messages sent between the devices in order to prohibit man-in-the-middle attacks and serve as MAC. Furthermore, the model uses an RC5 algorithm to encrypt data between communicating devices. The proposed model aims to gain over the mutual identity establishment in IoT by filling the existing gaps in both authentication and access control capabilities with the integrated protocol. When considering multimedia big data, the approach suffers from additional overhead due to key generation, update, and distribution.

Finally, A. Kumar, et al. [14] introduced a signal security system that depend on Genetic Algorithms (GA) with Non-Linear Forward-Feedback-Shift-Register (NLFFSR) for encrypting the signals and transforming them into an unknown format (disorder data). The main aspect of the system is the feasible integration with multimedia transmission application and high throughput rate. Using GA in cryptography, the system becomes more powerful in encrypted multimedia data specially the randomness properties of NLFFSR. Subsequently, the best technique is used to facility transferring the data with highly safe and reliable in the domain of multimedia communication. However, their results were very sensitive to parameter fluctuations.

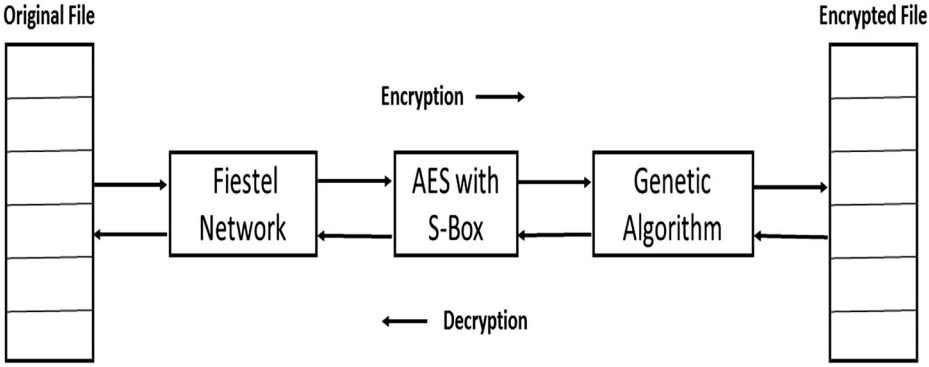

Fig. 1 System framework

3 Proposed encryption/decryption system

In this section, we introduce our resource-efficient encryption system. The proposed system consists of three main components: Feistel network, AES with S-box, and Genetic algorithm. Figure 1 illustrates the proposed system framework.

The input file is divided into a number of equal size blocks, where each block is split to plaintext and key parts. The key part is encrypted/decrypted using Feistel network. The Feistel network component divides the key to small blocks and performs a series of shift and rotation operations on them. The Feistel network produce a cipher key that will be used in AES component.

AES with S-box accepts the plaintext and the ciphered key as input. AES uses the substitution-permutation network and performs 10 rounds of encryption to generate the ciphered text. One of the main challenges in AES is key generation and management, the proposed system alleviates this challenge since each block is encrypted using a unique key generated from the data itself.

The ciphered text and ciphered key are integrated in the genetic algorithm component. The genetic algorithm uses two levels of integrations; crossover and mutation. In crossover, portion of the cipher text and ciphered key are swapped, where in mutation a randomly chosen bit is flipped in both ciphered text and ciphered key. This integration is repeated number of times for the same block. The result of this integration along with the crossover and mutation points are written to the encryption file as a ciphered block. Note that the addition of the crossover and mutation points increase the system security against brute force attacks.

3.1 Functionality overview of the proposed system

The data required for the system is read serially from an input files in form of blocks with equal sizes (each block size is 256 bits). The input files may include image, audio, video or text.

As a start, each block is split into 2 uniform parts; a 128 plain text stream of bits called Left (L) and another 128 key stream of bits called Right (R). As shown in Fig. 2, the encryption steps are as follows:

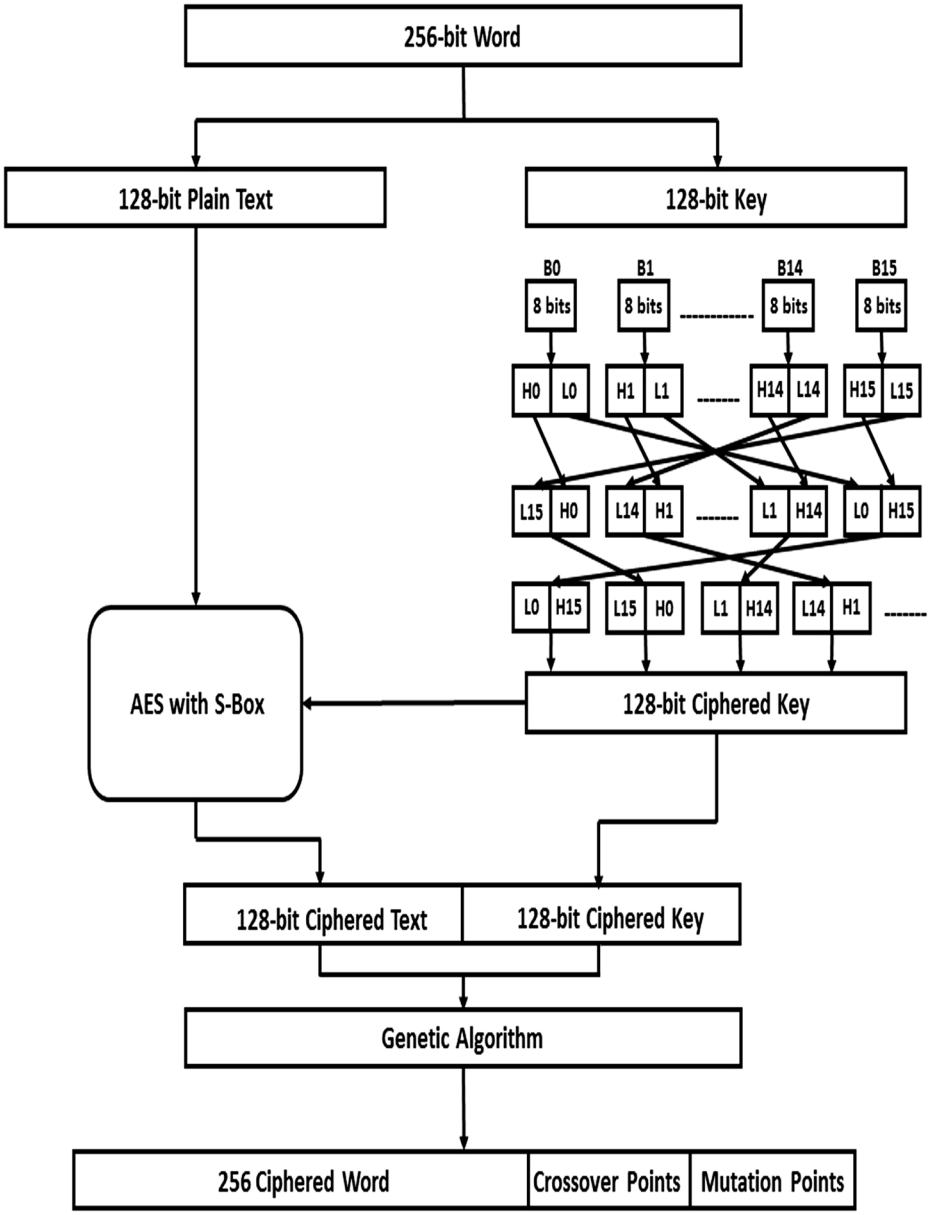

Fig. 2 256-bits encryption process

1. Next, the key (R) is divided into 16 bytes that numbered from B0 to B15. Each byte is split into two nibbles which are symbolizing to lower and upper halves of the byte. Then using symmetric technique, bytes' nibbles are shifted.
2. The key (R) part is ciphered using a Feistel algorithm that depends on three operations; logical right shift to the upper side of a byte, logical left shift to the lower part of another byte and logical OR for both lower and upper sides of each byte.

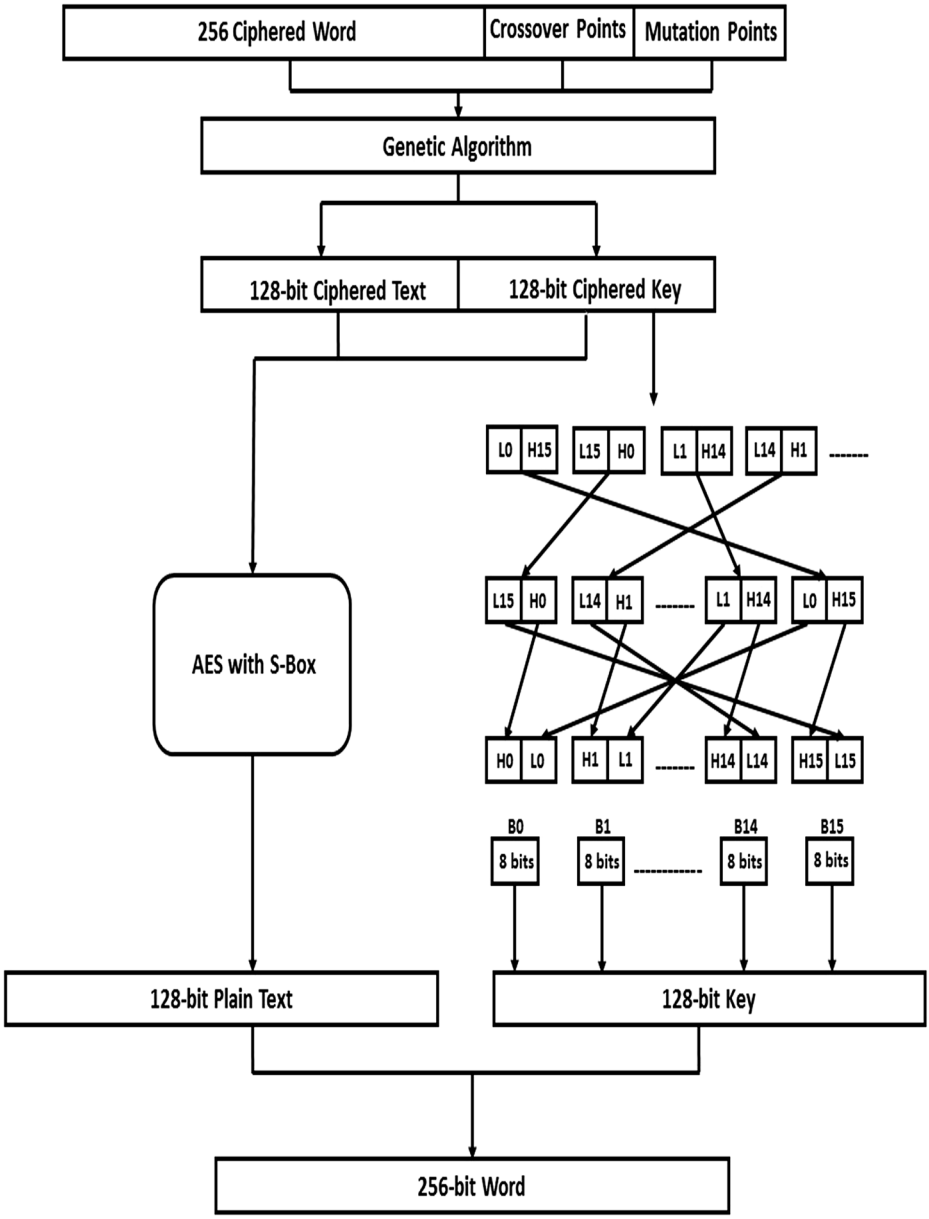

Fig. 3 256-bits decryption process

3. The upper nibble of each byte is rotated to the right and the lower nibble of each byte is shifted according to the proposed algorithm and illustrated in Fig. 1. As a result, the arrangement of all nibbles is adjusted as the lower portion becomes comprise of the upper halves and vice versa; the upper portion became of comprise the lower halves.
4. Rotating, adjusting and swapping are main operations in Feistel algorithm that are used to change the nibbles order completely, providing a 128 bits encrypted key. This encrypted key will be used to encrypt plain text part (L).

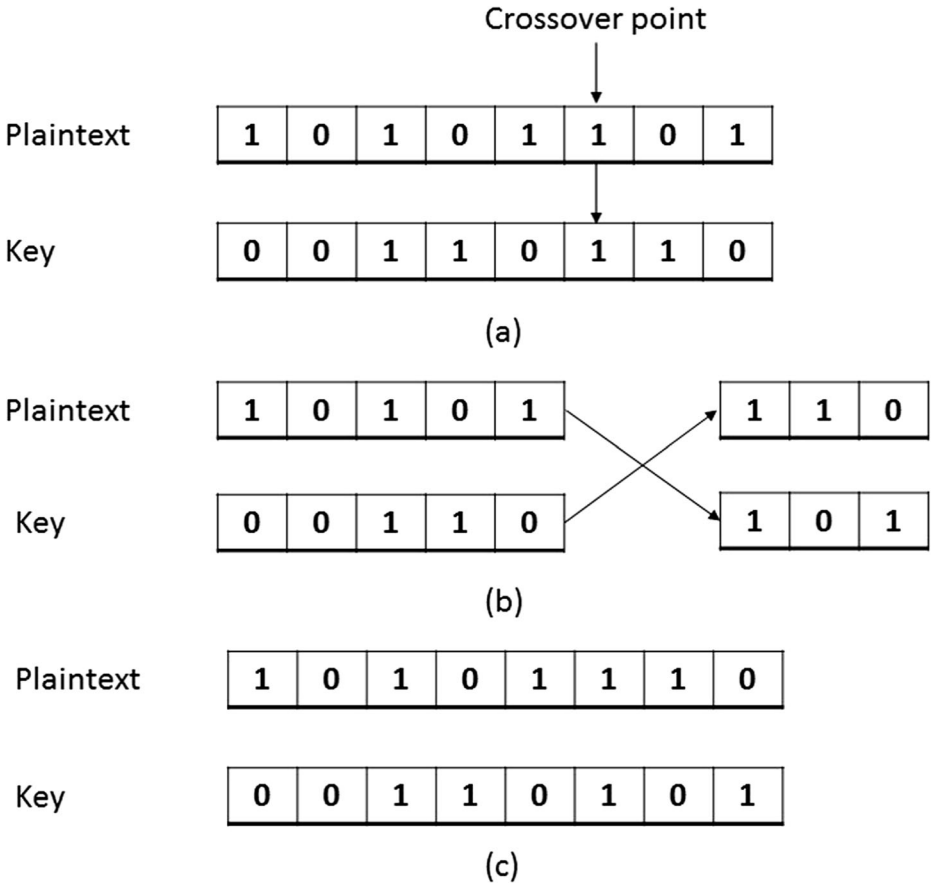

Fig. 4 Single point crossover

5. The plain text part (L) is ciphered using S-Box technique that is mainly based on the implementation of Advanced Encryption Standard (AES) encryption algorithm. AES is used to encrypt plain text part (L) depending on the ciphered key (128 bits) that is already generated from the Feistel algorithm. The key size is very important in AES, it is responsible for determining the number of iteration for conversion rounds, which is essential to convert the plaintext into ciphered text, such as: 10 cycles of repetition for 128-bit key and 12 cycles of repetition for 192-bit keys.
6. The AES algorithm consists of three main rounds. Each round includes many processing steps and each of them comprise of four similar but different stages, while taking into account one stage that depends on the encryption key itself. we can summarize AES as follow:

    - **Initial Round**: this round uses bitwise (XOR) technique that combined the block of the round key with each byte of the state, this is called Add Round Key.
    - **Rounds**: this round encompass on four main processes that can be described as follows:

- **Sub Bytes process**: process of replacing each byte that is located in the state with another byte by take into account the lookup table. Also is called a non-linear substitution process.
- **Shift Rows process:** process of shifting each byte of the state to the left cyclically.
- **Mix Columns process**: This process is implemented on each column, each of them is multiplied with a constant polynomial.
- **Add Round Key process:** In this process the Rijndael's key schedule is responsible for deriving the sub-key from the main key for each round. After that, the bitwise XOR merges the corresponds byte in each state and sub key; due to insert the sub-key.

• **Final Round**: This process contains the same previous processes (Sub Byte, Shift Rows and Add Round Key) without Mix Columns process.

After AES algorithm finish the job completely, the plaintext is encrypted perfectly.

7. The ciphered text and the cipher key are integrated via a Genetic Algorithm (GA). Genetic Algorithm is a method that is trying to solve any optimization problem successfully by modeling it based on the natural selection process. Generally, GA consists of initialization, selection, mutation, crossover, and termination steps. Those steps are repeated for a number of generations.

   Both of the ciphered key (128 bits) generating from step 5 and the ciphered text (128 bits) results from step 7 are integrated using mutation and crossover steps of GA and repeated for 10 generations.
8. Finally, a message that includes 256 bits ciphered word, crossover points and mutation points is sent to the receiver. Algorithm 1 shows a pseudocode for the encryption algorithm.

Figure 3 illustrates the decryption process. Generally, The decryption process is a reversible way of the encryption algorithm. The decryption steps is as follows:

1. When the message arrived at the receiver side, the ciphered word (256 bits), crossover point and mutation point are extracted. GA accepts them as an input; where it divides the ciphered word into two blocks L/R sides that each of them have the same size (128 bits). Mutation and crossover operations will be applied for both blocks in reverse order. These operations will be repeated based on the number of generations.
2. The R block (ciphered key) is entered into Feistel algorithm and decrypted using the proposed key encryption so as to retrieve the original encryption key, this operation is called Key Recovery.
3. The L block (ciphered plaintext) along with the R block (ciphered key) are entered into AES algorithm to be decrypted. Finally, the decrypted plaintext from AES and the recovered key from Feistel algorithm are added up and sent to an output file.
4. It should be noted that the proposed system is carried out on a graphical processing unit (GPU) rather than the CPU, in order to optimize performance through parallel processing. A GPU executes most of the computations, computer graphics, and image processing extremely efficient compared to a CPU. Furthermore, GPU uses high-speed parallel processing compared to the CPU's low-speed sequential processing. So, to encrypt and decrypt multimedia big data in a fast and efficient way, we used of GPU rather than CPU.

The pseudocode of the decryption algorithm is shown in algorithm 2.

Input: 256-bit word

Output: 256-bit ciphered word + Crossover points + Mutation Points

Divide 256-bit word to two 128 bit blocks (Plain text +

Key)

// Generating cipher key

CipherKey ⟵ Feistel Code(Key)

// Plain text encryption

CipherText ⟵ Encrypt AES with S-Box (Plaintext, CipherKey)// Produce generations

For i⟵ 1 to NumOfGenerations

CrossoverPoints (i) ⟵ Randomly choose a crossover point

CrossOver(CipherText, Cipherkey, CrossoverPoints (i))

MutationPoints (i) ⟵ Randomly choose a mutation point

Mutation(CipherText, Cipherkey, MutationPoints (i))

End For

Combine(CipherText, Cipherkey, CrossoverPoints, MutationPoints)

End

Input: 256-bit ciphered word + Crossover Points + Mutation Points

Output: 256-bit word

Divide the encrypted word to two 128 bit blocks, CrossoverPoints, MutationPoints

// Reverse generation production

For i ← NumOfGenerations to 1 step -1

Mutation (CipherText, Cipher key,

MutationPoints (i))

Crossover (CipherText, Cipher key,

CrossoverPoints (i))

End For

// Plain text AES decryption

PlainText ←

DecryptAESwithSBox (CipheredText, CipherKey)

// Regenerating the key

Key ← FiestleCode(CipherKey)

Combine (Text, key)

End

3.2 Genetic algorithm

Genetic Algorithm plays essential role in the proposed system. For each block, it integrates and mixes the cipher plain text and cipher key together based on crossover and mutation operator actions, which will be repeated for a predefined number of iterations (number of generations).

Crossover is an operation in which two blocks are mixed and match their desirable qualities in a random way. Many crossover techniques were proposed in literature; our proposed system uses single-point crossover technique which is illustrated in Fig. 4. The crossover operation starts by randomly choosing the crossover point (Fig. 4a). Then, the portion of the blocks from the crossover point to the least significant bit in the two blocks are exchanged (Fig. 4b, c).

The mutation operation in general is used to maintain the genetic diversity from one generation to another. Similar to crossover, different mutation types were proposed, in our proposed system we use the bit string mutation. Figure 5 illustrates the bit string mutation operation, it starts by randomly choose a mutation point for both cipher plaintext and cipher key (Fig. 5a). Then, the target bit is flipped in both blocks (Fig. 5b). Note that the mutation probability of a bit is 1/length (block), where length (block) is the length of the cipher text or cipher key.

The number of generations represents the number of times mutation and crossover operations are repeated. When this number is high, the system will be more secure since the

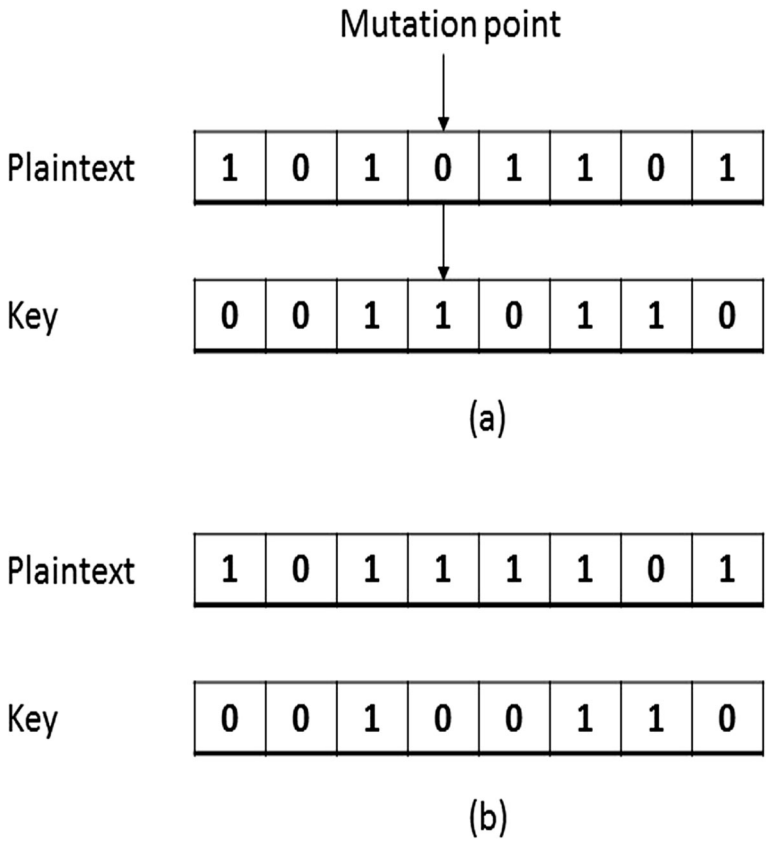

Fig. 5 Bit string mutation

integration is repeated many times but the running time will suffer. On the other hand, when the number is low, the system will be less secure.

4 Results and analysis

This section presents the evaluation of our proposed encryption system. The evaluation was conducted to show that the proposed system provides a balance between security and performance, which make it suitable for multimedia big data environment.

4.1 System implementation and environment

The proposed system was implemented in JAVA language and evaluated against benchmark symmetric encryption algorithms, such as Data Encryption Standard (DES), 3-DES with 168 bits' key size, MARS, Rivest Cipher (RC6) and Blowfish with 64 bits' key size in terms of running time, throughput for encryption and decryption processes, and the avalanche effect. The results were measured by a timing process within the program itself. The nature of the experiment was to replace the AES algorithm with each of the previously mentioned algorithms and compute the running time for each one of them (This is called Benchmarking). Each test was conducted for several times and the results were averaged. In general, benchmarking is a process used to measure the performance of an algorithm using specific indicator then compared the results with others [13].

All the experiments were conducted on an identical platform; HPC resources of Jordan's National Supercomputing (IMAN1) which includes 5 IBM cell processor machines. Each node is equipped with 32 GB of memory, Two Intel Xeon Processor E5-2609 v2 4C 2.5 GHz 10 MB Cache 1333 MHz 80 W CPU, and NVidia tesla k20 GPU which includes 2496 processor cores and 706 MHz core clock. Figure 6 shows the TESLA k20 block diagram [11].

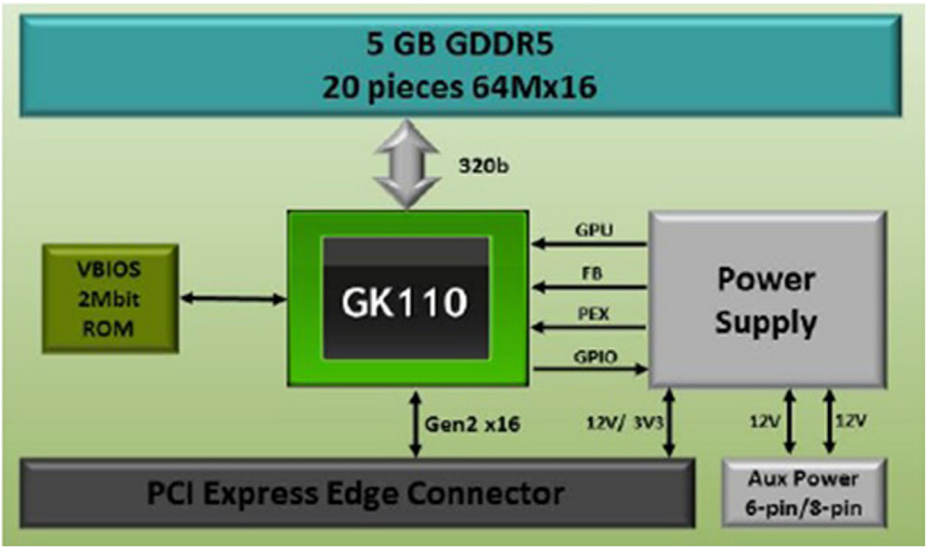

Fig. 6 TESLA K20 block diagram [10]

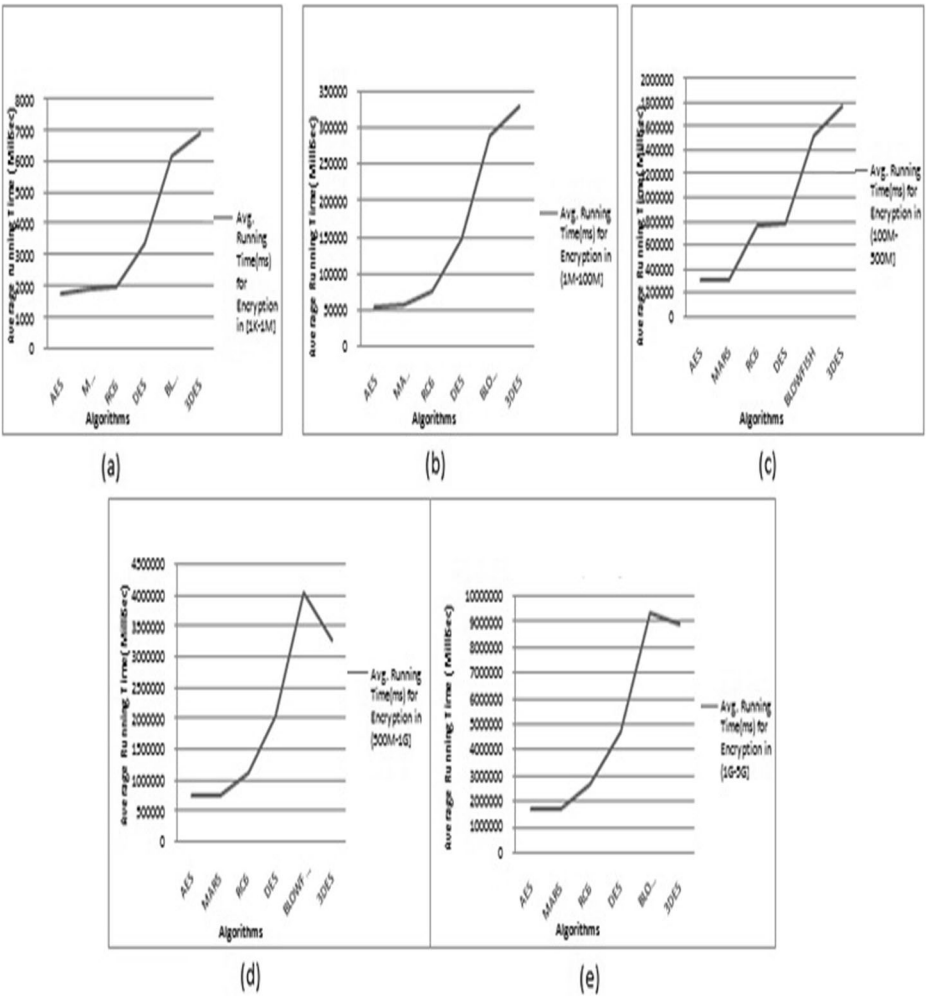

Fig. 7 Average Encryption Time for Intervals: **a** [1 KB – 1 MB], **b** (1 MB – 100 MB], **c** (100 MB – 500 MB], **d** (500 MB – 1 GB], **e** (1 GB – 5 GB]

The data used in the experiments are collection of real IoT data that is the medical multimedia big data that are taken from the hospital of JUST university, Jordan. We have tested the proposed algorithm with different input files sizes, those files were selected randomly to fall into one of the following intervals: [1 K–1 M], (1 M–100 M], (100 M–500 M], (500 M–1 G], and (1 G–5 G].

4.2 Performance analysis

In this section, we present and discuss the results of our proposed system in terms of performance metrics such as Running Time and Throughput for a number of symmetric key algorithms like MARS, RC6, DES, 3- DES, Blowfish.

Figure 7a–e shows the average running time of the encryption process for input files in the intervals ([1 KB – 1 MB], (1 MB – 100 MB], (100 MB – 500 MB], (500 MB – 1 GB], (1 GB –

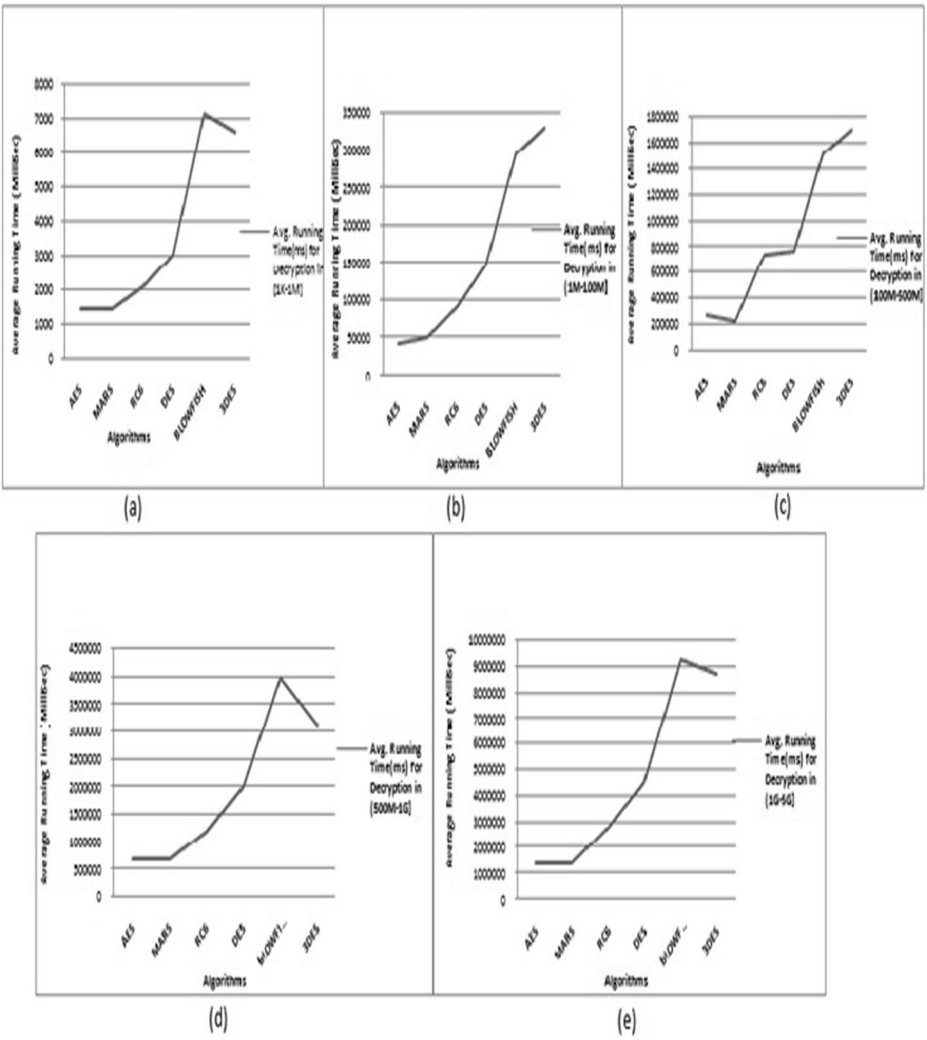

Fig. 8 Average decryption time for intervals: **a** [1 KB – 1 MB], **b** (1 MB – 100 MB], **c** (100 MB – 500 MB], **d** (500 MB – 1 GB], **e** (1 GB – 5 GB]

5 GB]). In general, running time is the time that an encryption/decryption algorithm takes to produce a cipher text/plaintext from a plaintext/cipher text.

All evaluated algorithms have a main advantage which is the ability to eliminate weak and semi-weak keys as the cipher and its inverse use different components. However, all of them are different from each other's in terms of number of encryption/decryption rounds, key size, and structure. Note that DES and 3-DES have the same number of rounds, but 3-DES runs DES three times, for the same plaintext, with three different keys.

The results demonstrate that AES has the lowest running time in comparison with all other algorithms which does not exceed 1700 s for the [1 GB – 5 GB] interval. AES is considered to be very fast in practice; it works more efficiently in hardware and software, more strong and resistant against all known attacks. MARS achieved better security than RC6 because, in

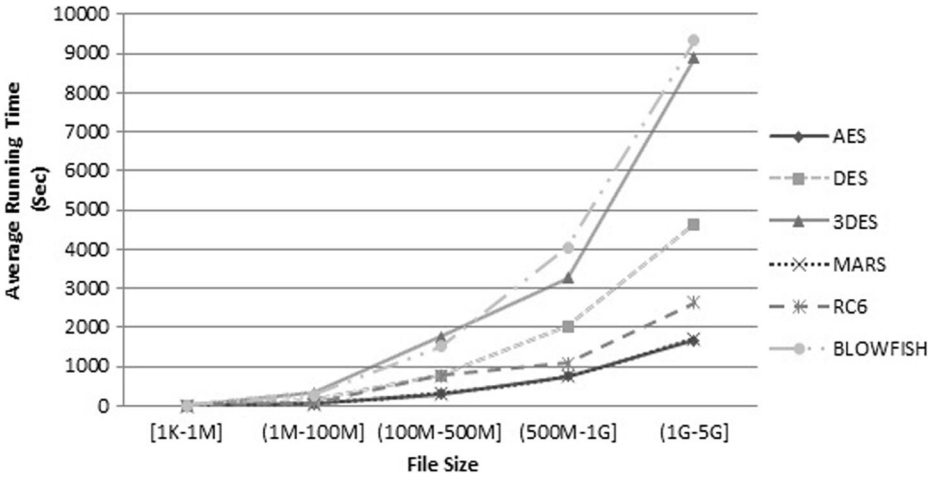

Fig. 9 Average running time of encryption process for all intervals

general, it has key size between 128 and 448 bits whereas RC6 support key sizes of 128, 192, and 256 bits. RC6 has many rounds and does use an extra multiplication operation to make the rotation dependent on every bit in the word, not only the least significant bits, which increase the encryption time. From the other side, RC6 is more flexible than MARS because it has 20 rounds than MARS, which is 32 rounds, and it uses less space in memory because it does not have a table lookup. MARS uses a table lookup, similar to DES S-box, which is a single table of 512 32-bit word. This leads to a slow software implementation since at least 3 instructions are needed per lookup. MARS needs less encryption time than RC6, DES, 3-DES, and Blowfish algorithms because it has heterogeneous structure (most of the rounds are jacketed by unkeyed mixing rounds) and full optimized implementation.

Nevertheless, MARS has lower performance than AES because it has a table lookup (at least 3 instructions per look-up) similar to S-Box in DES which is lead to slow software

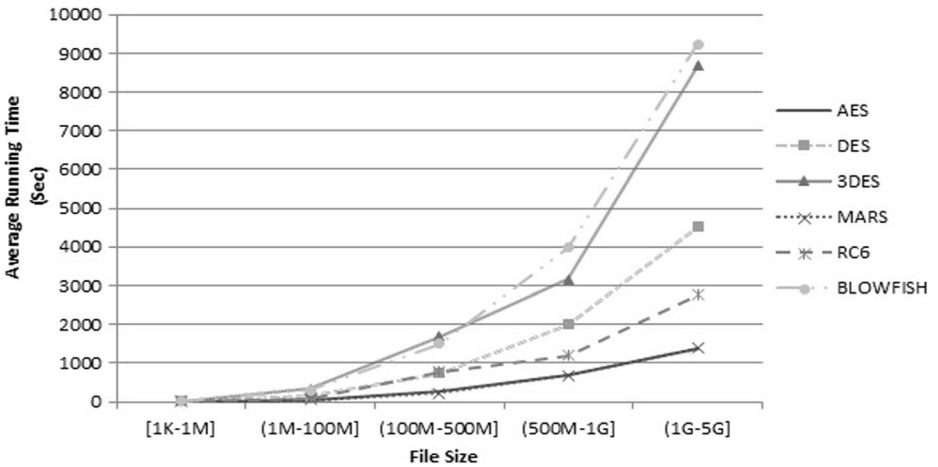

Fig. 10 Average running time of decryption process for all intervals

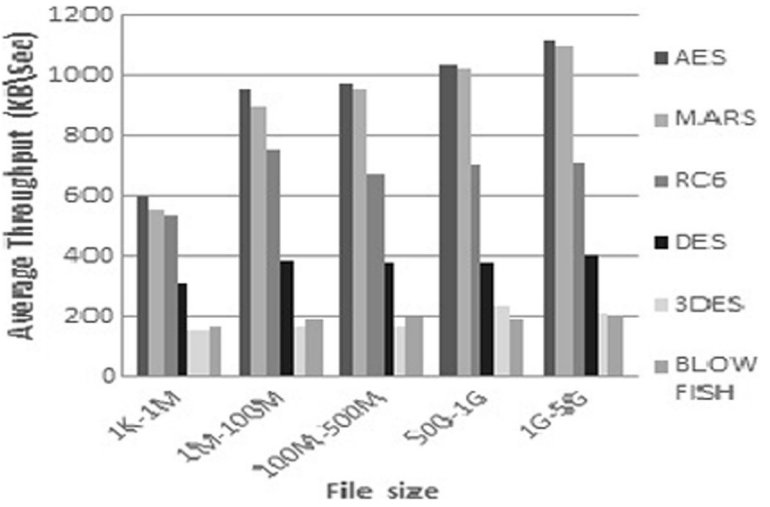

Fig. 11 Average throughput for encryption process for all intervals

implementation. From above Fig. 4a–c, it can be seen that Blowfish algorithm behavior better than 3-DES algorithm, although it uses 128-bit block size compared 168 bit for 3-DES algorithm. For small input sizes, the difference in key length, which reflects the number of blocks to be encrypted, does not have a significant effect on encryption time since the number of blocks to be encrypted are small. On the other hand, results shown in Fig. 4d–e shows that 3-DES is better than Blowfish because it needs more blocks to be encrypted and the number of blocks to be encrypted is huge (input files sizes 500 MB – 5 GB).

Finally, it was found that 3-DESworse than DES since 3-DES implement DES three times in sequence. It consists of three different keys (K1, K2 and K3) and has an effective key length

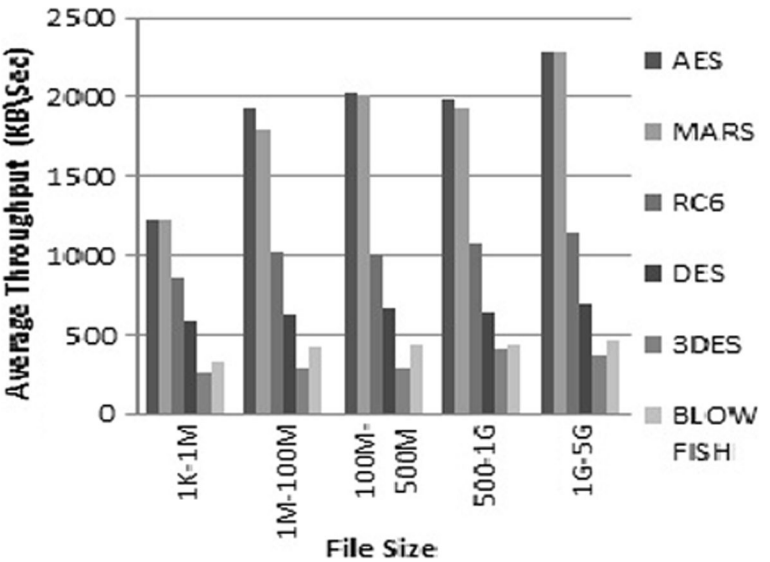

Fig. 12 Average throughput for decryption process for all intervals

of 168 bits. On the other hand, 3-DES more secure than DES because it has 168 bits' key size, whereas 56 bits' is the key size for DES.

Figure 8a–e illustrates the average decryption time for files with input files in the intervals ([1 KB – 1 MB], (1 MB – 100 MB], (100 MB – 500 MB], (500 MB – 1 GB], (1 GB – 5 GB]). The pattern of the results is the same as the encryption process; this is due to the fact that the decryption process in symmetric key algorithms is the reverse of the encryption process.

Figures 9 and 10 show the average running time for each algorithm in milliseconds versus different file sizes (1 KB – 5 GB). The figures show that the overall running time of encryption and decryption processes for AES is lower than the overall average running time for DES, 3-DES, MARS, RC6 and Blowfish. The average overall running time for MARS is very close to AES, but MARS does not satisfy the requirements of IoT.

Figures 11 and 12 present the average throughput for the encryption/decryption processes for all intervals. The throughput of the encryption/ decryption scheme is calculated by dividing the total plaintexts in Kilobytes over the total encryption/decryption time [7]. So, as the throughput of an algorithm increased, the algorithm is considered to have a high speed [7].

$$\text{Throughput} = \text{Plain text (Kilobyte)}/\text{Encryption Time (Second)}.$$

Figure 10 shows the superiority of the AES encryption algorithm over all other algorithms for all intervals, from small multimedia file sizes to big multimedia file sizes, with a more than 1120 KB\S throughput. This indicates that AES has high speed for encrypting multimedia big data per second in comparison with the other algorithms since it uses a substitution-permutation structure with 10 encryption rounds. On the other hand, 3-DES has the lowest with throughput that did not exceed 240 KB/S. 3-DES is considered to be a very slow encryption algorithm since it runs DES three times with three different keys for the same plaintext block. MARS has the second highest throughput since it uses Type-3 Feistel network with 32 encryption rounds, followed by RC6, DES, Blowfish, and finally 3-DES. Another point can be noticed that Blowfish has a very low throughput, this due to the small key size used compared with the other algorithms.

Figure 11 describes the throughput of the decryption process for all intervals. Similar to the encryption process, AES has the highest throughput with more than 2290 KB/S followed by MARS, RC6, DES, Blowfish, and 3-DES with 2280, 1144, 460, and 410 KB/S respectively.

4.3 Security analysis

In this section, we have evaluated our developed algorithm by the security metric which is the Avalanche Effect. Note that the key advantage of Avalanche Effect is measure the strength of the proposed algorithm against cracking and hacking threats and real time attacks such as brute

Table 1 The avalanche effect for different encryption algorithms

Encryption Technique	No. of bits flipped	%
DES	29	24.2
3-DES	37	30.8
CR6	56	46.6
MARS	59	49.2
Blowfish	31	48.4
Proposed Technique	63	52.5

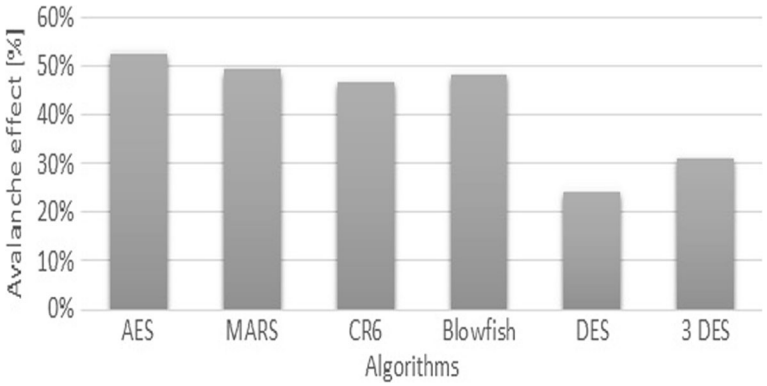

Fig. 13 The avalanche effect for different encryption algorithms

force attacks. In key encryption algorithms, changing a small number of bits in plain text in order to avalanche changes in subsequent rounds resulting in a large number of cipher text bit changes. For an algorithm to satisfy the avalanche criterion, the change of one plaintext bit is expected to result in one-half of the cipher text bits changing [26]. The avalanche effect can be calculated as:

$$\text{Avalanche effect} = \text{No. of flipping bits in the cipher text}/\text{No. of bits in the cipher text} \times 100\%$$

For example, considering a two-byte plaintext (0100100001101011), the most significant bit is flipped (1100100001101011). After encryption, the original ciphered text is (1111010101001l0) and the modified ciphered text is (1011000100000100). The Avalanche effect can be calculated by counting the number of flipped bits in the cipher text, which is 8, divided by the number of bits in the cipher text. So, the avalanche effect is 50%. The results in Table 1 are obtained after calculating the respective Avalanche Effects.

Figure 13 presents the avalanche effect for AES and other benchmark encryption algorithms (DES, 3-DES, MARS, RC6 and Blowfish). The results show that AES algorithm outperforms all other algorithms with 52.5%, Followed by MARS with 49.2%, Blowfish 48.4%, CR6 46.7%, 3-DES 30.8%, and finally DES 24.2%. It can be seen that AES is the only encryption algorithm to satisfy the avalanche effect criterion; subsequently it needs more time to be cracked. That is why AES is widely distributed and much more dependable than other benchmark algorithms.

From the evaluations, it's concluded that the proposed framework with AES has the lowest running time, highest throughput, and highest avalanche effect. Thus, the proposed framework is fast, efficient, secure, reliable, and scalable.

5 Conclusion

In this article, a novel efficient GPU encryption system has been developed to secure the Multimedia Big Data against real-time attacks such as DoS and tampering attacks. It should be noted that the developed encryption algorithm combines the process of scrambling of bits and substitution boxes resulting in high avalanche effect. The results showed that the developed

system is promising and encouraging in terms of the security and performance objectives. Note that we have run the experiments on IoT medical multimedia big data from JUST University Hospital. As a part of future work, the developed system will be run on other IoT multimedia big data, such as IoT military and education multimedia big data.

References

1. Aljawarneh S (2011) Cloud security engineering: avoiding security threats the right way. Int J Cloud Appl Comput (IJCAC) 1(2):64–70. doi:10.4018/ijcac.2011040105
2. Aljawarneh S, Raja M, Abdelsalam M (2016) Investigations of automatic methods for detecting the polymorphic worms signatures. Futur Gener Comput Syst 60:67–77
3. Atzori L, Iera A, Morabito G (2010) Internet of things: a survey. Comput Netw 54:2787–2805
4. Bandyopadhyay D, Sen J (2011) Internet of things: applications and challenges in technology and standardization. Wirel Pers Commun 58:49–69
5. Chen M, Shiwen M, Yunhao L (2014) Big data: a survey. Mob Netw Appl 19:171–209
6. Dave E (2016) The internet of things: how the next evolution of the internet is changing everything. Cisco
7. Elminaam DS, Abdual-Kader HM, Hadhoud MM (2010) Evaluating the performance of symmetric encryption algorithms. IJ Netw Secur 10:216–222
8. Evans D (2011) The internet of things how the next evolution of the internet is changing everything. Cisco Internet Business Solutions Group (IBSG)
9. Hart J, Martinez K (2015) Toward an environmental internet of things. AGU Earth Space Sci 2:194–200
10. http://www.nvidia.com/content/pdf/kepler/tesla-k20-passive-bd-06455-001-v07.pdf. Accessed 14 July 2016
11. http://www.sesame.org.jo. Accessed 30 Aug 2016
12. Kalyani V, Gaur P, Vats S (2015) IoT: 'machine to machine' application a future vision. J Manag Eng Inf Technol (JMEIT) 2:2394–8124
13. Kuehner H, Hartenstein H (2016) decentralized secure data sharing with attribute-based encryption: a resource consumption analysis. Proceedings of the 4th ACM International Workshop on Security in Cloud Computing
14. Kumar A, Rajpal N, Tayal A (2005) New signal security system for multimedia data transmission using genetic algorithms. NCC05
15. Lee JY, Lin WC, Huang YH (2014) A lightweight authentication protocol for internet of things. Proceedings of the International Symposium on Next Generation Electronics
16. Li S, Da Xu L, Zhao S (2015) The internet of things: a survey. Inf Syst Front 17:243–259
17. Madakam S (2015) Internet of things: smart things. Int J Future Comput Commun 4:250–253
18. Mahalle PN, Anggorojati B, Prasad NR, Prasad R (2013) Identity authentication and capability-based access control (IACAC) for the internet of things. J Cyber Secur Mob 1:309–348
19. Prasetyo KN, Purwanto Y, Darlis D (2014) An Implementation of data encryption for internet of things using blowfish algorithm on FPGA. International Conference on Information and Communication Technology
20. Santucci G (2009) From internet to data to internet of things. Proceedings of the International Conference on Future Trends of the Internet
21. Schneier B (1996) Applied cryptography, 2nd edn. Wiley
22. Schweitzer D, Boleng J (2009) Designing web labs for teaching security concepts. J Comput Sci Coll 25:39–45
23. Smith J (2013) Riding the multimedia big data wave. In: Proceedings of the International ACM SIGIR Conference on Research and development in Information Retrieval (SIGIR'13)
24. Stallings W (1999) Cryptography and network security principle and practice, 2nd edn. Prentice Hall
25. Sundaram BV, Ramnath M, Prasanth M, Sundaram V (2015) Encryption and hash based security in internet of things. Proceedings of International Conference on Signal Processing, Communication and Networking
26. Tavares SE, Heys HM (1995) Avalanche characteristics of substitution-permutation encryption networks. IEEE Trans Comput 44:1131–1139
27. Wang W, Hu Y, Chen L, Huang X, Sunar B (2012) accelerating fully homomorphic encryption using GPU. Proceedings of the IEEE Conference on High-Performance Extreme Computing (HPEC)
28. Xin M (2015) A mixed encryption algorithm used in internet of things security transmission system. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery
29. Yao X, Chen Z, Tian Y (2015) A lightweight attribute-based encryption scheme for the internet of things. Futur Gener Comput Syst 49:104–112
30. Zhao Y (2013) Research on data security technology in internet of thing. Proceedings of the International Conference on Mechatronics and Control Engineering

Shadi Aljawarneh is an associate professor, Software Engineering, at the Jordan University of Science and Technology, Jordan. He holds a BSc degree in Computer Science from Jordan Yarmouk University, a MSc degree in Information Technology from Western Sydney University and a PhD in Software Engineering from Northumbria University-England. He worked as an associate professor in faculty of IT in Isra University, Jordan since 2008. His research is centered in software engineering, web and network security, e-learning, bioinformatics, Cloud Computing and ICT fields. Aljawarneh has presented at and been on the organizing committees for a number of international conferences and is a board member of the International Community for ACM, Jordan ACM Chapter, ACS, and IEEE. A number of his papers have been selected as "Best Papers" in conferences and journals.

Dr. Muneer Masadeh Bani Yassein received his B.Sc. degree in Computing Science and Mathematics from Yarmouk University, Jordan in 1985 and M. Sc. in Computer Science, from Al Al-bayt, University, Jordan in 2001. And PhD degrees in Computer Science from the University of Glasgow, U.K., in 2007, He is currently an associate professor in the Department of Computer science at Jordan University of Science and Technology (JUST), Muneer served as Chairman of the department of Computer science from 2008 to 2010, as Vice Dean of the Faculty of Computer and Information Technology from 2010 to 2012, and from 2013 to 2014. Muneer is currently conducting research in Mobile Ad hoc Networks, Wireless sensors Networks, Cloud Computing, simulation and modelling, development/analysis of the performance probabilistic flooding behaviours in MA-NET, optimizations and the refinement of service discovery and routing algorithms for mobile device communications in heterogeneous network environments. Bani Yasin has published over 90 technical papers in well reputed international journals and conferences. During his career, he has supervised more than 50 graduate and undergraduate students. Dr. Bani Yasin is member of IEEE and he is a member of the technical programs of several journals and conferences.

We'am is a postgraduate student in the CS Department, JUST, Jordan. She has received her BC from Jordan Yarmouk University with excellent rate. We'am has published a number of papers in very good conferences and journals.