ISSN: 0067-2904

A Strong AES Algorithm Using Hash Functions for Key Generation

Mays M. Hoobi, Inas Ali Abdulmunem*
Computer Science Department, College of Science, University of Baghdad, Baghdad, Iraq

Received: 15/3/2024 Accepted: 14/8/2024 Published: 30/8/2025

Abstract

The present world depends on communications. The increasing dependence on communications means that information security is more important than ever. Every user wants to keep the data secure by applying a strong data encryption concept. Strong data encryption can be achieved by utilizing a robust encryption algorithm and strong keys. In this research, a method is proposed—the Key Generation Method (KGM)—to generate strong, unique keys for the cryptographic algorithm. This method involves using 22 different types of hash functions to generate 22 strong unique keys with different lengths (128-bit and 256-bit), and then applying these keys with the Advanced Encryption Standard (AES) algorithm. Analysis tools (entropy and autocorrelation) in addition to 4-NIST tests (frequency test, run test, long run test, and serial test) are used in addition to the poker test to evaluate the strength of AES ciphering. The results confirm that the proposed method for key generation can effectively satisfy the criteria of strong encryption against attackers.

Keywords: Encryption, Key Generation Method, Hash function, AES.

خوارزمية *AES* القوية باستعمال دوال الـ *Hash* لتوليد المفاتيح

ميس محمد هوبي, ايناس علي عبدالمنعم*
قسم علوم الحاسوب، كلية العلوم، جامعة بغداد، بغداد، العراق

الخلاصة

العالم الحالي يعتمد على الاتصالات. إن زيادة الاعتماد على الاتصالات يعني أن أهمية أمن المعلومات تتزايد أكثر من أي وقت مضى. يريد كل مستخدم الحفاظ على أمان البيانات من خلال تطبيق مفهوم قوي لتشفير البيانات. يمكن تحقيق التشفير القوي للبيانات باستعمال خوارزمية تشفير قوية ومفتاح قوي. في هذا البحث تم اقتراح طريقة وهي طريقة توليد المفاتيح (KGM) لتوليد المفاتيح الفريدة القوية لخوارزمية التشفير. تتضمن هذه الطريقة استعمال 22 نوعًا مختلفًا من وظائف التجزئة لإنشاء 22 مفتاحًا فريدًا قويًا بطولين مختلفين (128 بت و256 بت)، ثم تطبيق هذه المفاتيح باستعمال خوارزمية معيار التشفير المتقدم (AES). يتم استعمال أدوات التحليل (الانتروبيا والارتباط الذاتي) بالإضافة إلى اختبارات NIST-5 (اختبار التردد، واختبار البوكر، واختبار التشغيل، واختبار المدى الطويل، والاختبار التسلسلي) لتقييم قوة تشفير AES. تؤكد النتائج أن الطريقة المقترحة لتوليد المفاتيح يمكن أن تلبي بشكل فعال معايير التشفير القوي ضد المهاجمين.

*Email: inas.ali@uobaghdad.edu.iq

1. Introduction

Sending encrypted messages is essential for ensuring secure message interactions. The encryption algorithm's strength and the secret key's confidentiality are the two fundamental factors that determine a block cipher's security. The security of an encryption algorithm is obtained through both the linear and nonlinear functions/component substitution box(s) or by using multiplication functions to add nonlinearity to the algorithm. Permutations and circular shifts are examples of linear functions. These functions enable the encryption algorithm to conceal the relationship between plain text and cipher text [1] and [2].

For lightweight cryptography, symmetric and asymmetric algorithms are divided into two categories. Asymmetric key encryption (public-key encryption) encrypts data using a public key and decrypts it using a private key, whereas symmetric encryption utilizes the same secret key for both operations. Two further methods of symmetric key encryption are the stream cipher and block cipher [3] and [4].

Confusion and diffusion are two fundamental principles of cryptography that must be achieved in the ciphertext in order to create a strong cipher and prevent attackers from attacking. Shannon defines these principles as follows: Confusion is a complex and involved relationship between the ciphertext and the symmetric key, whereas diffusion is the dispersion of the plaintext's statistical structure over the bulk of the ciphertext. This complexity is typically implemented using a repeated series of substitutions and permutations, which involve manipulating the bit order in accordance with a permutation algorithm and substituting specific bits with other bits in accordance with "substitution" rules. The overall cipher's resistance against linear and differential attacks is increased with a strong key. Modern encryption algorithms rely heavily on key scheduling algorithms, whose security is just as vital as the encryption algorithms themselves. Lots of research has been conducted on the cryptographic strength assessment of encryption algorithms; however, the strength assessment of key scheduling algorithms often receives less attention, which may indicate a potential vulnerability in the encryption process as a whole [5] [6] [7].

The main problem of this research, as one of the basics upon which the strength of encryption depends for all encryption algorithms in general and the AES algorithm in particular, is the strength of the key and ensuring that its value is not a repeated or unique key. This aim is satisfied by the proposed system of this research when generating a single, strong key through the use of many types of hash functions, which represent a one-way encryption method, and the attacker cannot return the hash function value to the original content. This implies a high level of complexity, making it difficult for attackers to crack the cipher text.

The remaining sections of this research are: Section 2: The related work will be reviewed. Section 3 briefly describes the AES algorithm. In Section 4, different types of hash functions will be covered, while in Section 5, analysis tools and NIST methods will be explained briefly. Section 6 will consist of the proposed system structure and details. Consequently, the last two sections, 7 and 8, discuss the experimental results and conclusions.

2. Related Works

There are plenty of studies that have occurred in the field of AES encryption; each suggests improving the AES algorithm to enhance randomness. The most useful studies are presented in the following:

The researcher in [8] proposed a model that combines a strong memory-hard hashing function (SCrypt) with a memory-accelerated AES algorithm. The model's structure is mainly partitioned into three phases, which are key derivation using the Scrypt algorithm, AES-256 encryption in Electronic Code Book (ECB) mode, and AES-265 decryption in ECB mode.

The results have shown that, as compared to others, this hybrid model is highly efficient and has high performance, and thus it successfully increases the password security of an online user from brute force attack.

In order to improve durability against biclique and brute force attacks, the researchers from [9] added a dynamicity character to the AES S-Box (substitution box). Additionally, they used MD4, SHA3, or SHA5 hashing techniques in conjunction with the AES algorithm to achieve a variance in security. A novel key dispersion strategy is presented in order to boost the AES algorithm's security avalanche impact with the advancement of technology. The experiment's results indicated that the suggested algorithm is more complex and slower than the original AES, but these differences can be disregarded for two reasons: First, given the prevalence of computers in modern society, the algorithm's increased time and complexity can be overlooked. The second reason is that the algorithm's benefits make new attacks on symmetric key ciphers extremely difficult.

The researchers in [10] modified the AES algorithm to improve the original AES's security. The researchers made two changes to optimize the process. The first generates keys for the AES method using the DES key, substitution layer, and one-way function (MD5). The second modification generates variable values for the key used to transfer the state matrix rows instead of the static keys (AES). The modified AES output, which is used to encrypt all kinds of data, is evaluated using five metrics: five basic statistical tests, encryption runtime, brute force attacks, and analytical attacks. The results showed that the sub-keys passed the test and achieved a full spread of bits.

The researchers in [11] proposed a safe method that supports the deduplication feature and makes use of conventional secure encryption. Users who own the same file can receive the same encryption key over the Internet without a third party's assistance. These keys are generated by taking a key from the file's hash and adding more random salt to make the key more difficult to crack and resistant to attacks. Moreover, because the key creation process happens within the variable environment, the server cannot access these keys. In order to minimize network bandwidth usage and efficiently de-duplicate data on the cloud server, the data mark value of every file is frequently compared. To protect the system from attacks, encryption is done using the AES-CBC technique, which is lightweight in comparison to other security algorithms. The results proved that the suggested technique is effective at getting rid of redundant data while maintaining security against attacks.
Some of the major drawbacks with the existing AES core system—such as low data rate, attack susceptibility, reliability, and discriminative key management—are examined and resolved in [12] with the use of the proper hierarchical transformation techniques. In order to decrease the path delay, inner-stage pipelining is used over composite field-based S-box transformation measures. Furthermore, the research presented a bit-level masking technique for AES, which covers operations like modulo and inverter that are part of simpler arithmetic. This transformation technique of bit masking mitigates key management-related concerns in AES with enhanced diffusion and confusion measures. According to a thorough analysis of data rate performance that showed the proposed AES model offered improved system performance when compared to the conventional AES core, the robustness of the improved FF masking model and related security improvements in the AES system are also demonstrated with suitable test input stimulus models.

The goal of [13] is to identify secure methods for preserving security, privacy, and integrity in healthcare management systems. Using the Keyed-Hash Message Authentication

Code (HMAC) and the Secure Hash method 256 (SHA-256), the AES algorithm is used to encrypt data transferred while offering a way to verify the integrity of the data and increase its resistance to attacker approaches. While a firewall system is ineffective against such attacks on its own, the danger of exposure to attackers can be prevented by combining honeypot systems with intrusion detection systems (IDSs). The encryption algorithm's performance is compared for varying plain-text lengths, encryption duration, memory and CPU consumption, and entropy in order to assess the efficacy of the proposed security health information management system. Furthermore, it is evident that the longer the AES key size is employed, the more memory and time are needed. Therefore, if hard real-time operation of the system is required, then the 128-bit AES key is recommended.

3. Advanced Encryption Standard Algorithm (AES)

The National Institute of Standards and Technology validated the symmetric encryption method AES in 2001. AES is quick in both hardware and software because it depends on a substitution-permutation network, which combines the substitution and permutation processes.

The block size specified for the AES algorithm is 128 bits/16 bytes, while key sizes of 128 bits/16 bytes, 192 bits/24 bytes, and 256 bits/32 bytes are supported. Table 1 illustrates the variable rounds' number (rounds' number based on key length and block size) and how the block sizes might match those of the keys [14] [15].

Table 1: Summary of the AFM information of CdS QDs [14]

	Key Length (Nk)	Block Size (Nb)	Number of rounds(Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Figure 1 shows the outline of the AES framework. The four basic parts are key expansion, initial round key addition, 9, 11, or 13 rounds of encryption, and final round encryption. There would be ten encryption rounds overall using a 128-bit key. AES uses a 4x4 column-major order array with 16 bytes for operation. The transformation rounds' number required to change the input, known as plain text, into the final output, known as cipher text, is determined by the key size employed in an AES cipher.

AES uses four components, in order, for every encryption round: SubBytes, ShiftRows, MixColumns, and AddRoundKeys. For the first nine rounds (with 128 bits of key length), the encryption procedure remains unchanged, while the final encryption round skips the MixColumns step. Using an 8-bit S-Box, each byte in the state array is changed to a SubBytes value in the SubBytes step. The SubBytes step is the only non-linear process within AES encryption. The bytes in each state row are cyclically shifted by a particular offset via the ShiftRows step. Using a reversible linear transformation, the four bytes from each state column are combined in the MixColumns step. Four bytes are input into the MixColumns function, which outputs four bytes, with each input byte impacting all four output bytes. The AddRoundKeys step links the sub-key and the state. Rijndael's key schedule is used to derive a sub-key from the main key for every round. Every sub-key's size matches the state's size. Using bitwise XOR, each byte of the state and the corresponding byte of the sub-key are combined to create the sub-key [14], [16], [15].

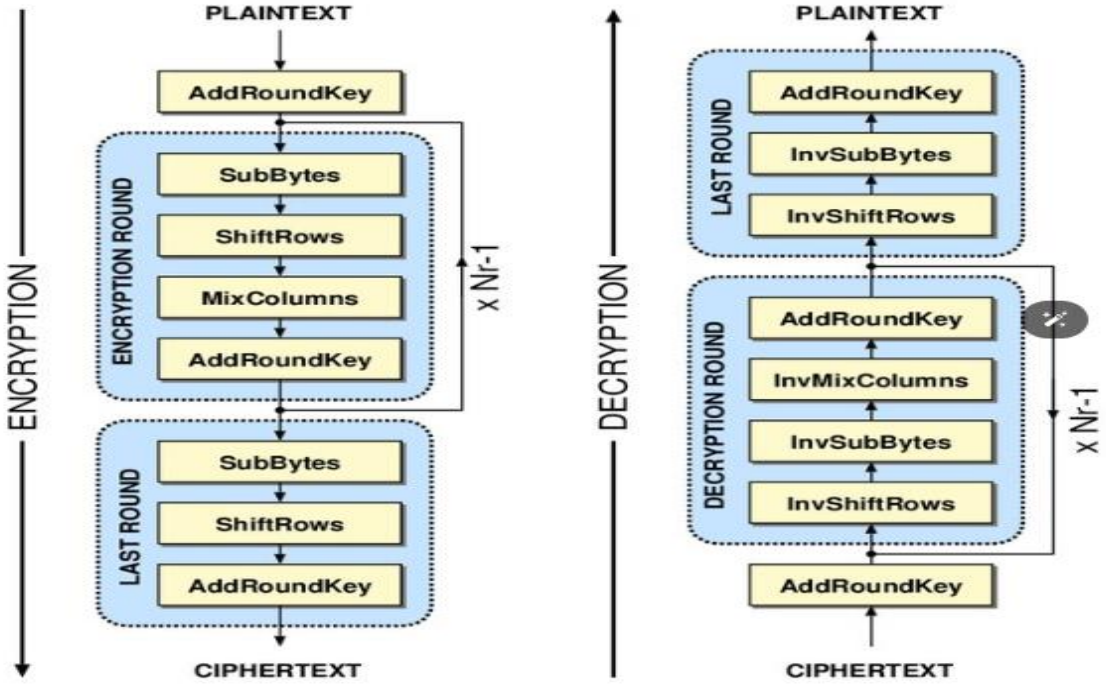


Figure 1: Structure of AES [17]

4. Hash Functions

A hash function is one that generates an output of a defined size from an arbitrary quantity of input. Cryptographic hash functions start with the standard hash function. At the moment, MD5 and SHA variants are the most widely used hash algorithms [18]. Since a message digest cannot be used to retrieve the original information, a hash function is a one-way operation. Since a given plain text would always result in an identical message digest, there is no need for a key. The output and the input are approximately the same size when using symmetric key cryptography. The algorithm always determines the result of a hash function to be a predetermined length (256 bits for SHA2-256, for example). Hash functions, unlike encryption algorithms, find application in a wide range of contexts [19] and [20].

Data integrity plays an important role in a very secure system. Users can create a message digest using the cryptographic hash function method to identify any illegal file changes. This is particularly crucial for critical systems and sensitive databases. Hashing functions can be used in conjunction with other common cryptographic techniques to confirm the origins of data. When hashing algorithms combine with encryption to identify the data's original source, they create unique message digests known as message authentication codes. The standard algorithm, HMAC, is currently in use.

The SHA1 and MD5 algorithms are regarded as secure as there is no known method other than brute force to identify collisions. A brute force attack attempts random inputs and monitors the results until it detects a collision. Although SHA1 is more expensive to use for a message digest than MD5, it is more secure [18].

5. Analysis Tools

Several studies show that there are a number of statistical tests and analysis tools that make up the necessary criteria for evaluating the strength of cryptographic algorithms. The illustration below [21] illustrates some of these crucial concepts:

a. Entropy Test

It is a security factor of knowledge where there is ambiguity or randomness in the ciphertext. In order to accomplish confusion and diffusion features, cipher designers consider Shannon's principle when creating an encryption algorithm [22] [23]. Entropy, which is typically expressed in bit units, quantifies the anticipated value and measures the randomness of the data in a message. A random variable X's Shannon entropy is defined by Eq. (1), and P_i's is defined by equation 2, where x_i represents the ith possible value of X out of n and P_i indicates the possibility that $X = x_i$.

$$H(X) = H(P,\dots,Pn) = \sum_{i=1}^{n} P_i \log_2 P_i \quad (1)$$

Where $P_i = P_r(X = x_i)$ [24] [25].

b. Autocorrelation Test

The purpose of this test is to determine how dependent a sequence (Si) is on its shifted sequence ($Si+d$). If d is a fixed integer such that $1 \leq d \leq (n/2)$, the dependence between (Si) and ($Si+d$) can be examined using the test statistics below:

$$Z(d) = \sum_{i=0}^{n-d-1} S_i \oplus S_{i+d} \quad (2)$$

Where $n - d \geq 10$ and then $Z(d)$ follows an $N(0, 1)$ distribution [26] [27].

c. Frequency Test

This test seeks to ascertain whether the percentage of one to zero in the created stream is equal to that of zero, where the ratio should be almost 50%. It is implied that the generated key does not meet the fundamental condition of randomness if the generated stream is unable to pass the frequency test. Eq. (3) provides an example of the frequency test, which is briefly described here. The statistics utilized are as follows, where a0 and a1 represent the number of 0s and 1s in an n-bit sequence:

$$Z = \frac{(a_0 - a_1)^2}{n} \quad (3)$$

Here, Z approximately follows a $\chi 2$ (chi-square) distribution with 1 degree of freedom if $n \geq 10$ [28] [29].

d. Poker Test

This test determines the P-bit block's frequency of appearance in an n-bit sequence. Split the sequence into B blocks of length P, none of which overlap. Assume that b_i is a P-bit sequence's ith bit. To evaluate the uniformity distribution of P-bit blocks, the following statistics are employed:

$$Z = \frac{2^P}{B} \sum_{i=1}^{2^P} (b)^P - B \quad (4)$$

Where $4 \leq P \leq 8$ and Z approximately follows a $\chi 2$ (chi-square) distribution with $2P - 1$ degree of freedom [26] [27].

e. Run Test

In an n-bit sequence, let O_i represent the number of runs of 0 of length i and l_i the number of runs of 1 of length i. The expected continuous "0" or "1" has a length of $(n-i-3)/2i+2$. With $1 < i \leq L$, let L be the largest of i. The length of runs in a sequence is assessed using the following statistics:

$$Z = \sum_{i=1}^{L} \frac{(I_i - e_i)^2}{e_i} + \sum_{i=1}^{L} \frac{(O_i - e_i)^2}{e_i} \quad (5)$$

Here, Z follows $\chi 2$ (chi-square) distribution with $(2L–2)$ degrees of freedom [28] [29].

f. Long Run Test

Finding the longest run of ones within M-bit blocks is the main goal of the test. The purpose of this test is to determine whether the longest run of ones in the sequence under test and the longest run of ones predicted in a random sequence have the same length. It is important to keep in mind that an inconsistency in the longest run of ones' predicted length involves an irregularity in the longest run of zeroes' expected length. As a result, only a test for one is required [30] [31].

g. Serial Test

The focus of this test is the frequency of all possible overlapping *m*-bit patterns across the entire sequence. This test's goal is to find if the number of occurrences of the *2m m*-bit overlapping patterns is roughly equal to what would be expected for a random sequence. Uniformity exists in random sequences, which means that every *m*-bit pattern has the same chance of appearing as every other *m*-bit pattern. It should be noted that the serial test is equivalent to the frequency test when *m = 1*.

$$X^2 = \sum_{i=0}^{m-1} \sum_{j=0}^{1} \frac{(n_{ij} - \mu_{ij})^2}{\mu_{ij}} \quad (7)$$

Where $\mu_{ij} = \frac{n-k+1}{2^{k+1}}$ [32] [31]

6. Proposed System- Key Generation Method (KGM)

In the current era of modern block ciphers, breaking the cipher is relatively harder and requires a lot of resources and time, so attackers are more interested in finding the secret key in other ways, such as brute force attacks and dictionary attacks. The key strength directly affects the security of an encryption algorithm. The key generation method must be robust enough to prevent an attacker from calculating the entire secret key, even if they gain knowledge of some of its bits.

The proposed system is called the Key Generation Method (KGM). This system concerns the aspect of generating a strong unique key that can be used with the AES cryptography algorithm to provide a high security level with a more complex cipher against attacks. KGM takes the plain text, which consists of 600 characters, as illustrated below:

(*The plain-text fed to AES algorithm, then manipulated until it becomes the encrypted-text, is represented as a square of 4 rows and 4 columns. The representation of internal square of AES algorithm is called a state, then different transformations executed during the (encryption - decryption) operations are regrouped in rounds each round including a different round key. The round keys are derived from the first main key. Two invertible transformations used by AES algorithm. In order to substitute a byte, it was interpreted as two hexadecimal digits.)*

In this system, use this plain text as input for 22 different types of hash functions to generate 22 keys, divided into 12 keys generated by using a 128-bit hash function and 10 keys generated by using a 256-bit hash function. The output of all these types of hash functions is considered a unique key. Use the AES algorithm in 22 different scenarios, each involving a different type of hash function. After that, measure and analyze the strength of the cipher text for each case, such as entropy, autocorrelation, poker test, and 4 NIST measurements included (frequency test, run test, long run test, and serial test). Figure 2 provides additional illustrations.

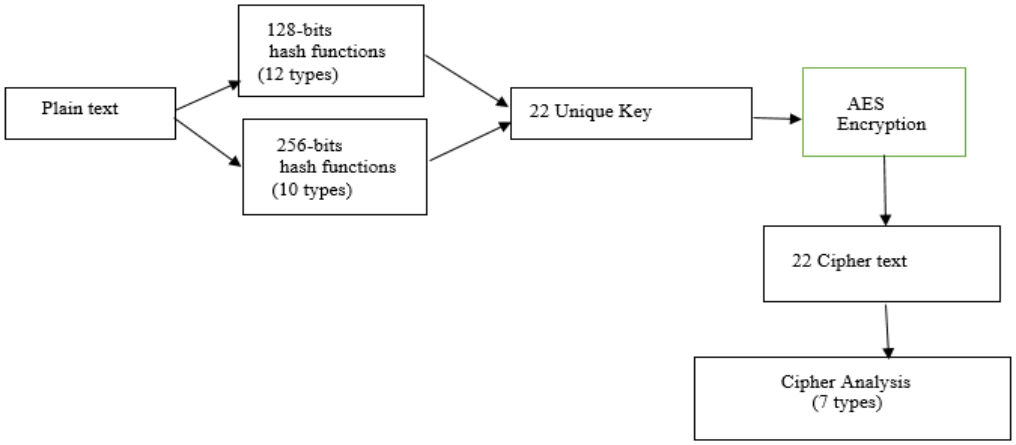


Figure 2: Structure of Proposed System (KGM)

7. Experimental Results and Discussions

The proposed KGM system tests the security strength of AES ciphering using 22 different types of hash functions and 22 generated keys. Several evaluation criteria are applied to the AES algorithm, such as entropy, autocorrelation, poker test, and four NIST measurements (frequency test, run test, long run test, and serial test). Two cases are applied with AES key generation based on key length.

Case 1: key length = 128-bits generated using 12 types of hash functions (MD2, MD4, Murmur3c, Tiger 128,4, Ripemd128, Haval 128,4, Murmur3f, XXh128, Haval 128,5, Tiger 128,3, MD5, and Haval 128,3).

Case 2: key length = 256-bits generated by using 10 types of hash functions (Haval256,3, Gost crpto, Haval 256,5, Ripemd256, SHA 512/256, Snefru 256, Gost, SHA3-256, Haval 256,4, SHA-256).

In this research, high values of entropy were achieved for the AES cipher, as illustrated in Table 2. This table shows the highest entropy value of the AES cipher obtained using Haval 128,3 = 7.66/8 and the lowest entropy value obtained using MD2 = 7.56/8. This is the best case for generating a unique key with 128 bits when using the Haval 128.3 hash function.

Table 2: (Case-1) Entropy Values for Plain text and Different Types of 128-bit Hash Functions

Plain text Entropy	
4.09/4.70	
Hash Function /128 bits	**Entropy**
MD2	7.56/8
MD4	7.58/8
Murmur3c	7.58/8
Tiger 128,4	7.60/8
Ripemd128	7.61/8
Haval 128,4	7.61/8
Murmur3f	7.62/8
XXh128	7.62/8
Haval 128,5	7.63/8
Tiger 128,3	7.63/8
MD5	7.64/8
Haval 128,3	7.66/8

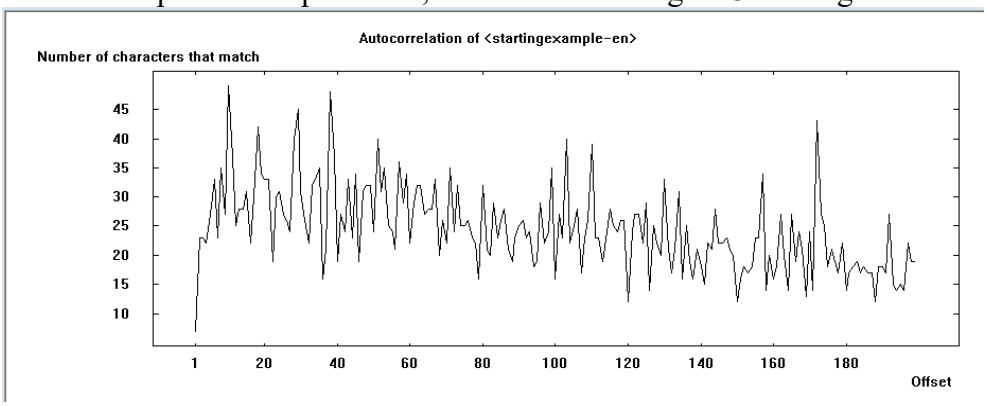
In case 2, the highest value of entropy obtained was 7.66/8 when using SHA-256, while the lowest value was 7.57/8 when using Haval256, as shown in Table 3.

Table 3: (Case-2) Entropy Values for Plain text and Different Types of 256-bit Hash Functions

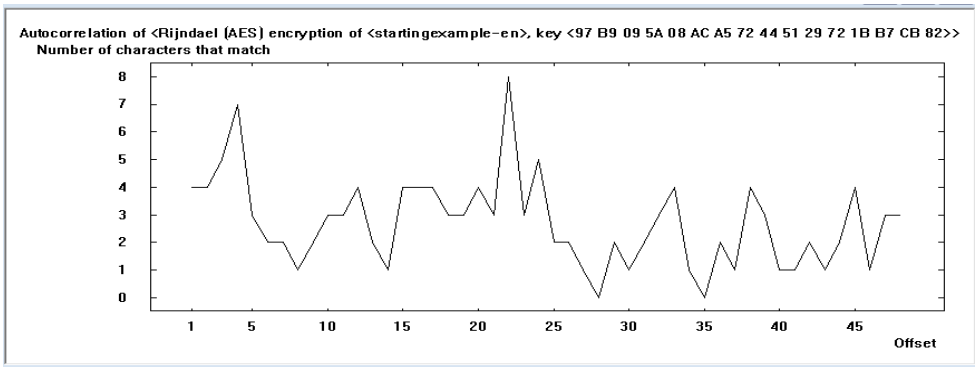
Plain text Entropy	
4.09/4.70	
Hash Function /256 bits	**Entropy**
Haval256,3	7.57/8
Gost crpto	7.60/8
Haval 256,5	7.60/8
Ripemd256	7.61/8
SHA 512/256	7.62/8
Snefru 256	7.62/8
Gost	7.63/8
SHA3-256	7.64/8
Haval 256,4	7.65/8
SHA-256	7.66/8

The above table clearly demonstrates that the AES cipher achieved high entropy values. These results show that the highest entropy value of the AES cipher was obtained by using SHA-256 = 7.66/8 and the lowest entropy value was obtained by using Haval256,3 = 7.57/8. That means the best case for generating a unique key with 256 bits is when using the SHA-256 hash function.

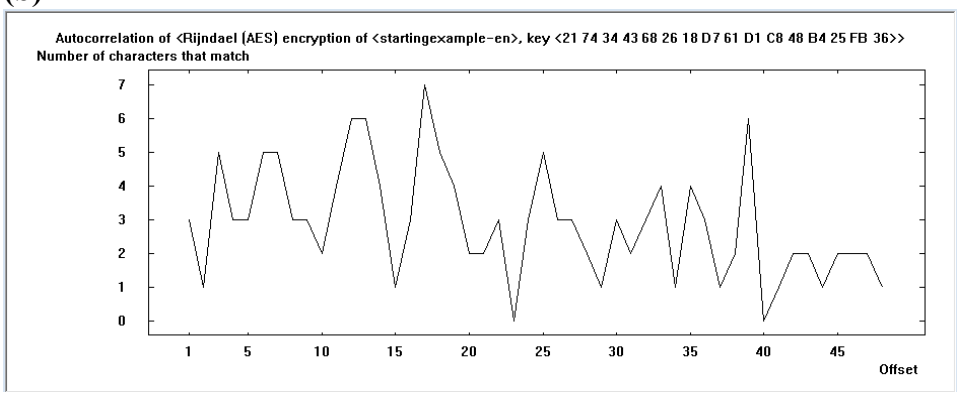
The autocorrelation results indicated that the autocorrelation of cipher text in cases 1 and 2 decreased when compared with plain text, as illustrated in Figure 3 and Figure 4.



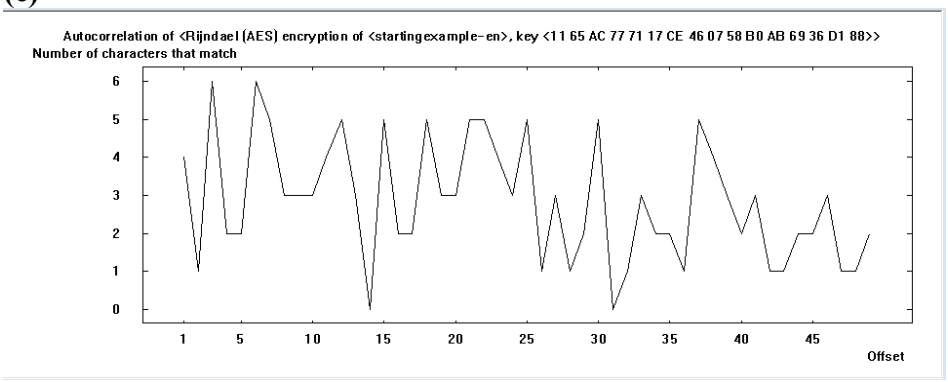
(a)


Autocorrelation of <Rijndael (AES) encryption of <startingexample-en>, key <97 B9 09 5A 08 AC A5 72 44 51 29 72 1B B7 CB 82>>
Number of characters that match
Offset

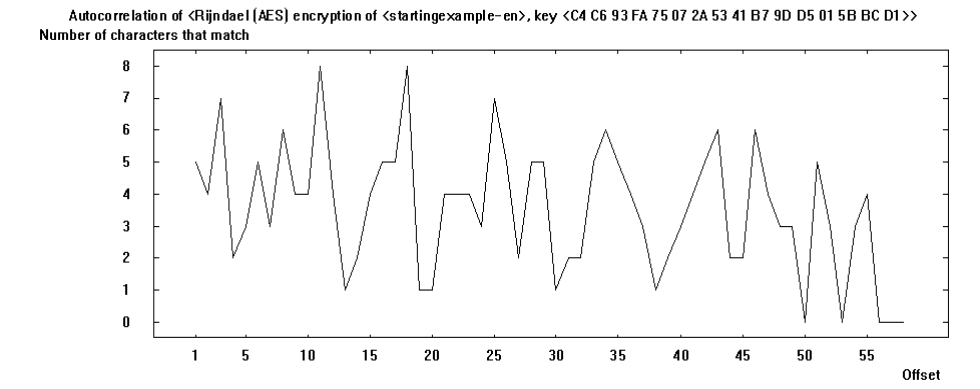
(b)


Autocorrelation of <Rijndael (AES) encryption of <startingexample-en>, key <21 74 34 43 68 26 18 D7 61 D1 C8 48 B4 25 FB 36>>
Number of characters that match
Offset

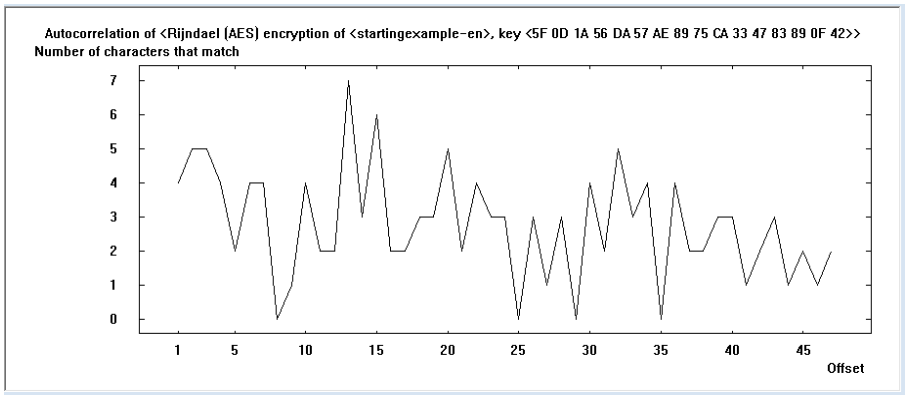
(c)


Autocorrelation of <Rijndael (AES) encryption of <startingexample-en>, key <11 65 AC 77 71 17 CE 46 07 58 B0 AB 69 36 D1 88>>
Number of characters that match
Offset

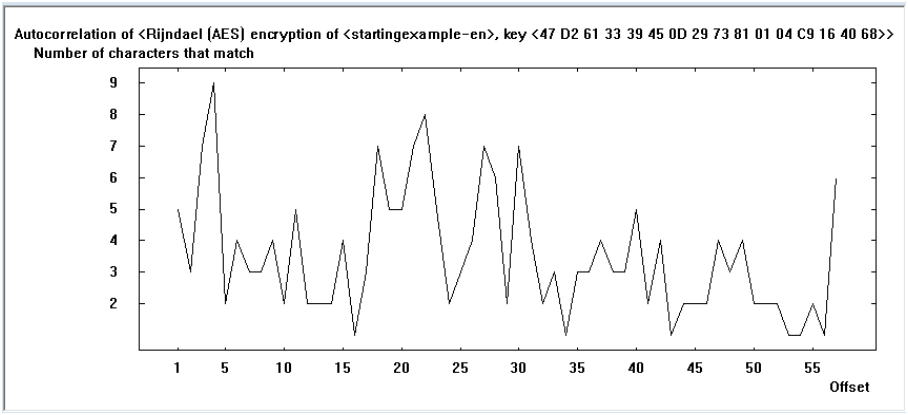
(d)


Autocorrelation of <Rijndael (AES) encryption of <startingexample-en>, key <C4 C6 93 FA 75 07 2A 53 41 B7 9D D5 01 5B BC D1>>
Number of characters that match
Offset

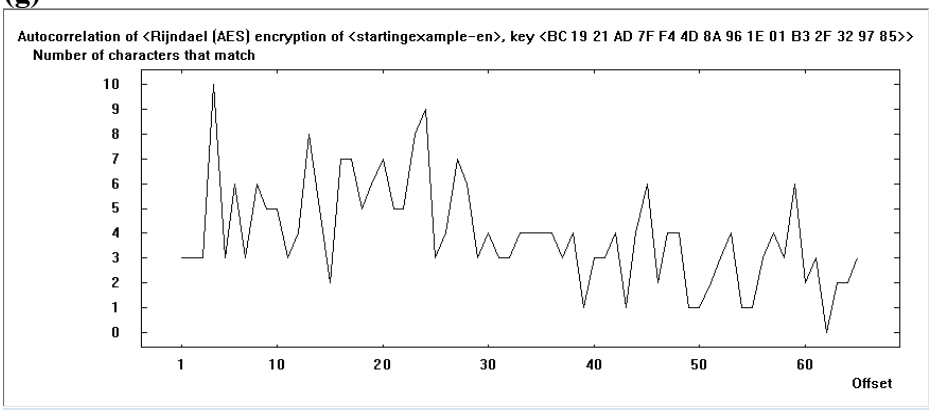
(e)


Autocorrelation of <Rijndael (AES) encryption of <startingexample-en>, key <5F 0D 1A 56 DA 57 AE 89 75 CA 33 47 83 89 0F 42>>
Number of characters that match
Offset

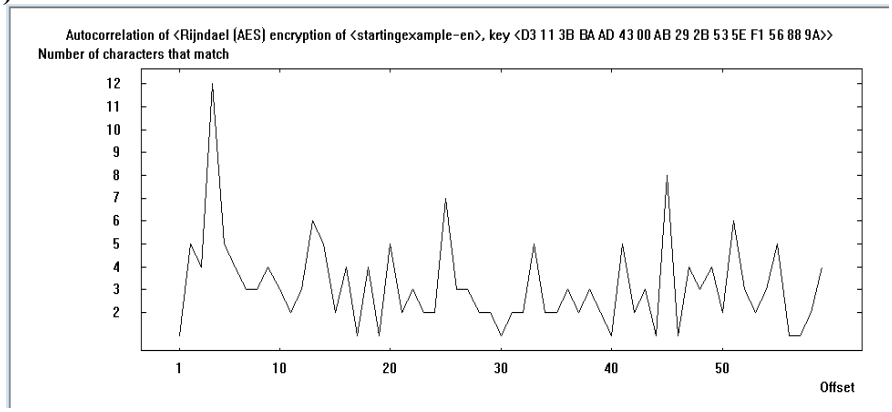
(f)


Autocorrelation of <Rijndael (AES) encryption of <startingexample-en>, key <47 D2 61 33 39 45 0D 29 73 81 01 04 C9 16 40 68>>
Number of characters that match
Offset

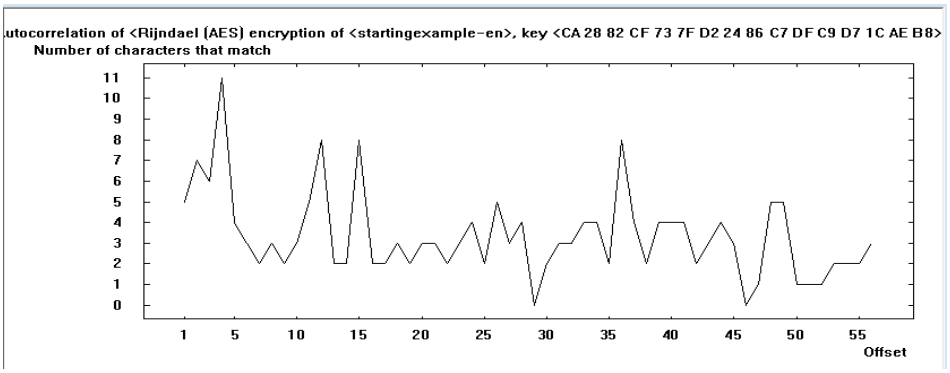
(g)


Autocorrelation of <Rijndael (AES) encryption of <startingexample-en>, key <BC 19 21 AD 7F F4 4D 8A 96 1E 01 B3 2F 32 97 85>>
Number of characters that match
Offset

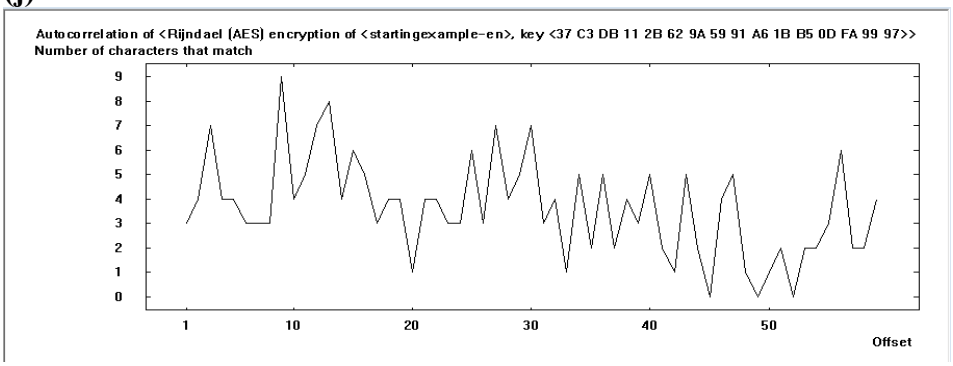
(h)


Autocorrelation of <Rijndael (AES) encryption of <startingexample-en>, key <D3 11 3B BA AD 43 00 AB 29 2B 53 5E F1 56 88 9A>>
Number of characters that match
Offset

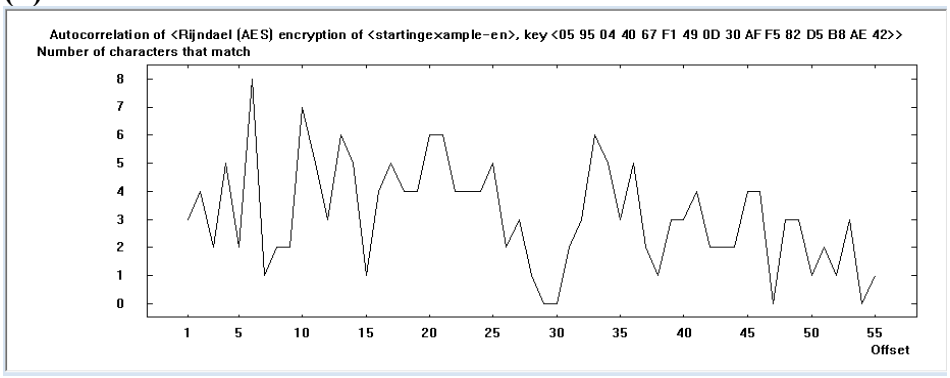
(i)



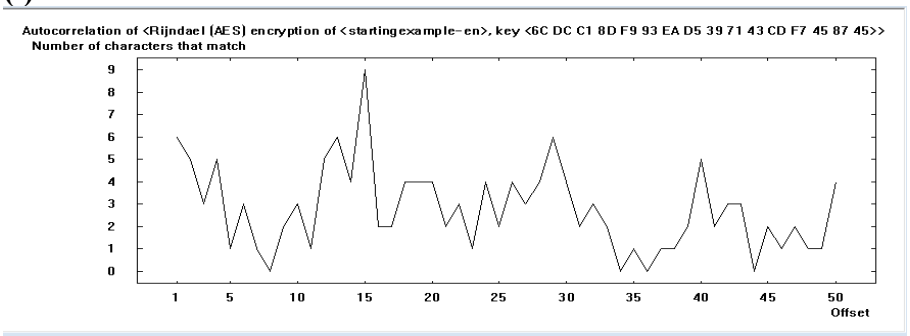
(j)



(k)

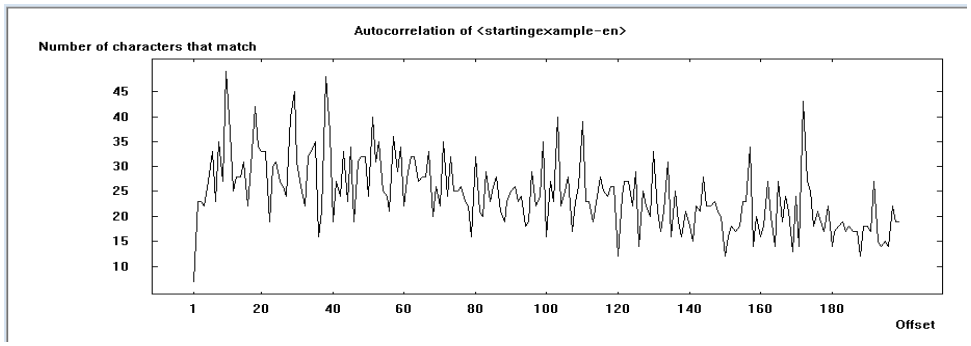


(l)

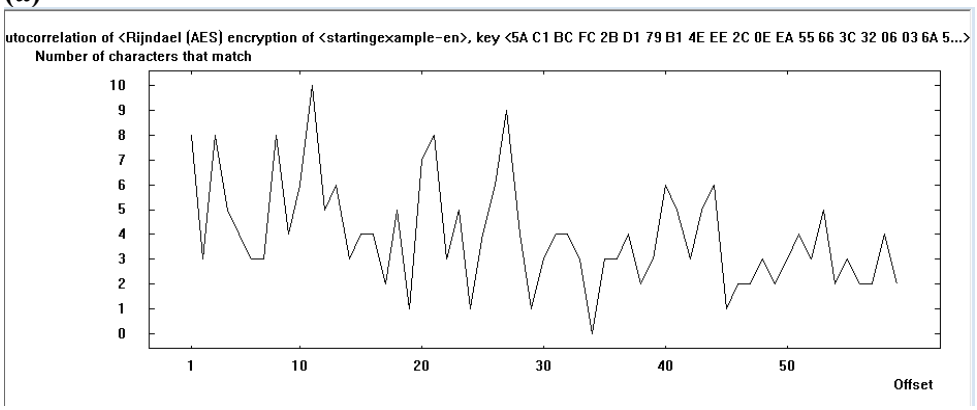


(m)

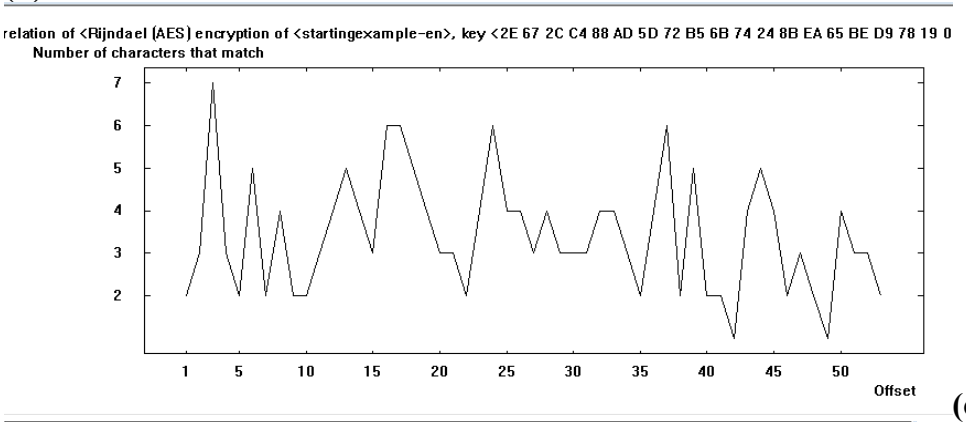
Figure 3: (Case-1) Autocorrelation for a)Plain text, b)MD2_128-bit Hash Function c)MD4_128-bit Hash Function, d)Murmur3c _128-bit Hash Function, e)Tiger 128,4_128-bit Hash Function, f)Ripemd128_128-bit Hash Function, g)Haval 128,4_128-bit Hash Function, h)Murmur3f _128-bit Hash Function, i)XXh128_128-bit Hash Function, j)Haval 128,5_128-bit Hash Function, k)Tiger 128,3_128-bit Hash Function, l)MD5_128-bit Hash Function, m)Haval 128,3_128-bit Hash Function


Autocorrelation of <startingexample-en>
Number of characters that match
45
40
35
30
25
20
15
10
1
20
40
60
80
100
120
140
160
180
Offset

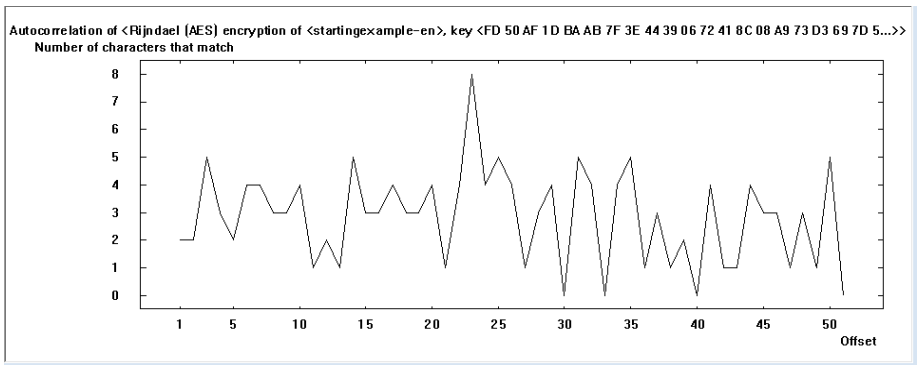
(a)


utocorrelation of <Rijndael (AES) encryption of <startingexample-en>, key <5A C1 BC FC 2B D1 79 B1 4E EE 2C 0E EA 55 66 3C 32 06 03 6A 5...>
Number of characters that match
10
9
8
7
6
5
4
3
2
1
0
1
10
20
30
40
50
Offset

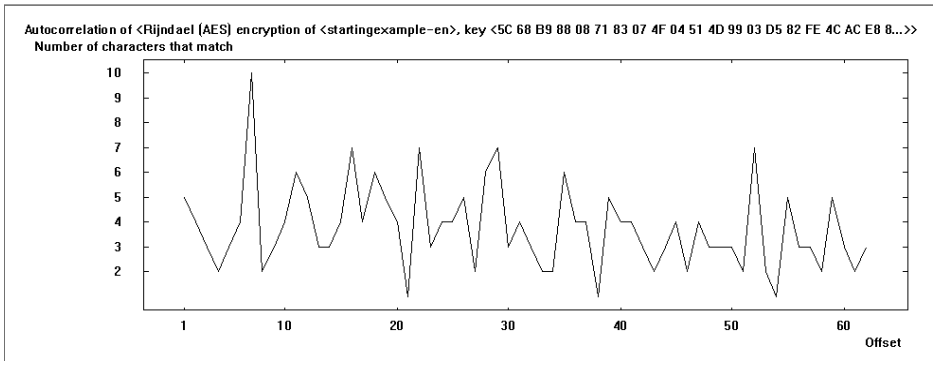
(b)


relation of <Rijndael (AES) encryption of <startingexample-en>, key <2E 67 2C C4 88 AD 5D 72 B5 6B 74 24 8B EA 65 BE D9 78 19 0
Number of characters that match
7
6
5
4
3
2
1
5
10
15
20
25
30
35
40
45
50
Offset

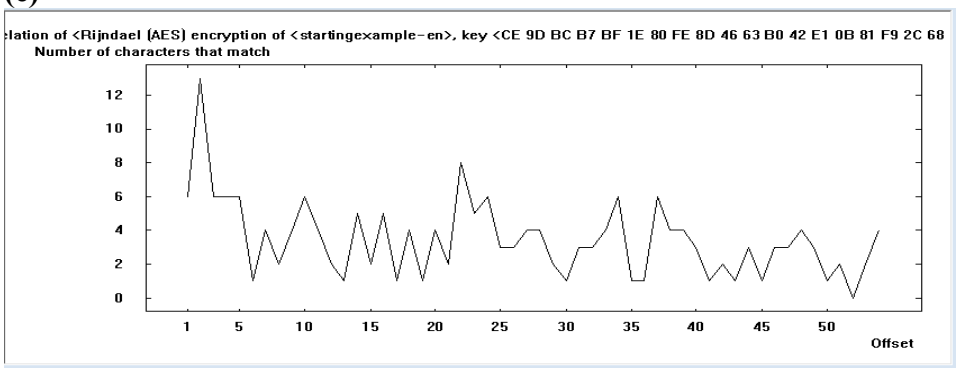
(c)


Autocorrelation of <Rijndael (AES) encryption of <startingexample-en>, key <FD 50 AF 1D BA AB 7F 3E 44 39 06 72 41 8C 08 A9 73 D3 69 7D 5...>>
Number of characters that match
8
7
6
5
4
3
2
1
0
1
5
10
15
20
25
30
35
40
45
50
Offset

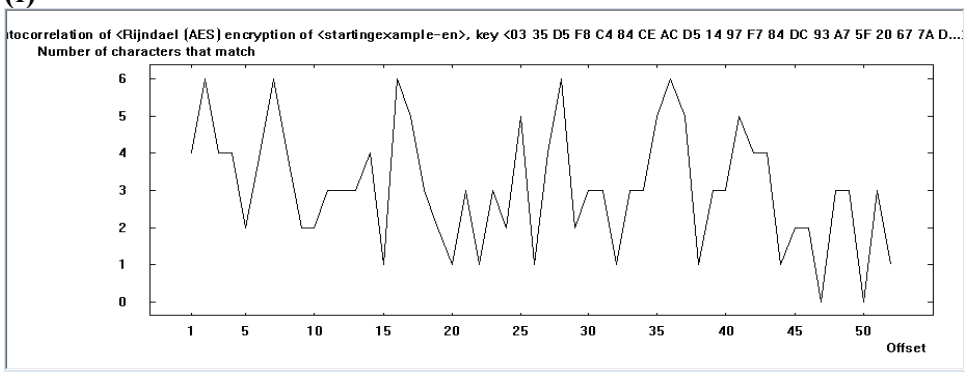
(d)


Autocorrelation of <Rijndael (AES) encryption of <startingexample-en>, key <5C 68 B9 88 08 71 83 07 4F 04 51 4D 99 03 D5 82 FE 4C AC E8 8...>>
Number of characters that match
10
9
8
7
6
5
4
3
2
1
10
20
30
40
50
60
Offset

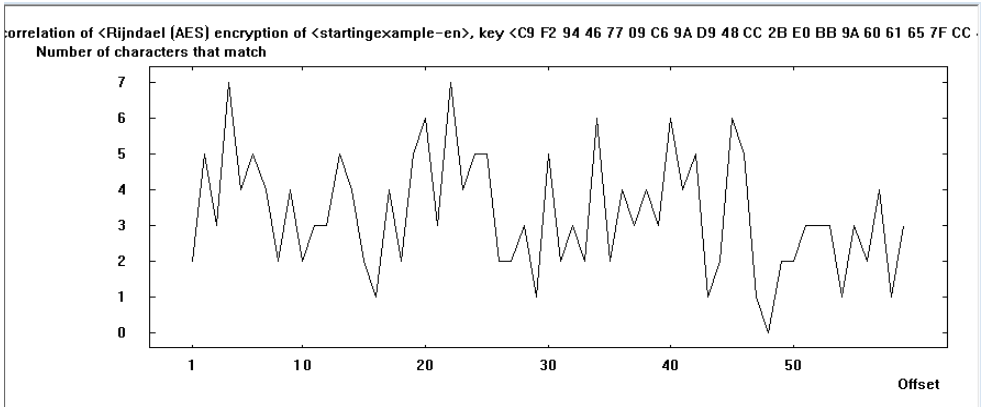
(e)


lation of <Rijndael (AES) encryption of <startingexample-en>, key <CE 9D BC B7 BF 1E 80 FE 8D 46 63 B0 42 E1 0B 81 F9 2C 68
Number of characters that match
12
10
8
6
4
2
0
1
5
10
15
20
25
30
35
40
45
50
Offset

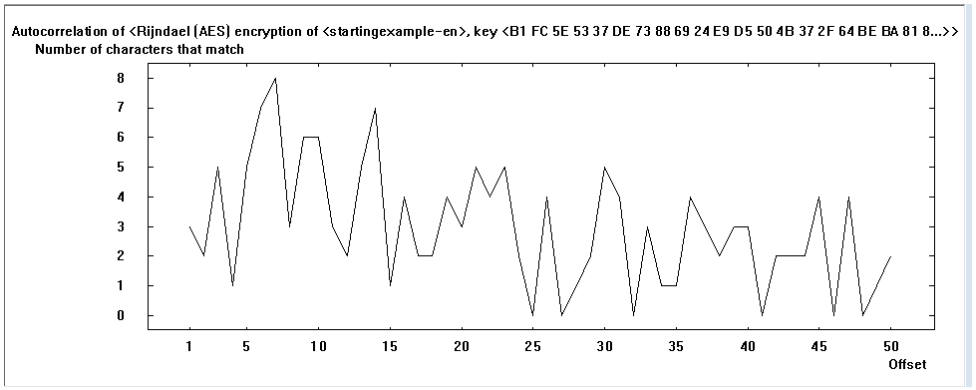
(f)


tocorrelation of <Rijndael (AES) encryption of <startingexample-en>, key <03 35 D5 F8 C4 84 CE AC D5 14 97 F7 84 DC 93 A7 5F 20 67 7A D...
Number of characters that match
6
5
4
3
2
1
0
1
5
10
15
20
25
30
35
40
45
50
Offset

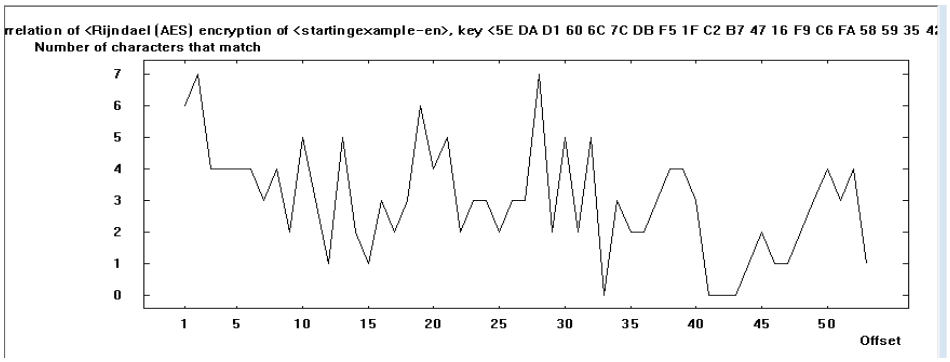
(g)


correlation of <Rijndael (AES) encryption of <startingexample-en>, key <C9 F2 94 46 77 09 C6 9A D9 48 CC 2B E0 BB 9A 60 61 65 7F CC
Number of characters that match
7
6
5
4
3
2
1
0
1
10
20
30
40
50
Offset

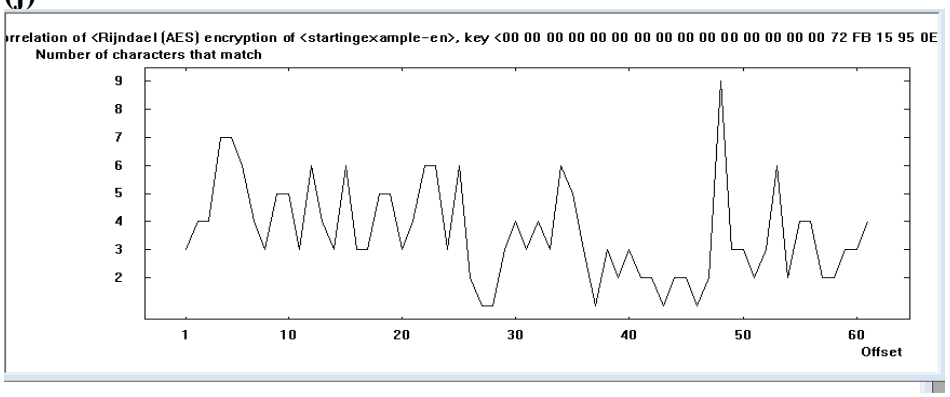
(h)



(i)



(j)



(k)

Figure 4: (Case-2) Autocorrelation for a)Plain text, b)Haval256,3_256-bit Hash Function c)Gost crpto _256-bit Hash Function, d)Haval 256,5_256-bit Hash Function, e)Ripemd256_256-bit Hash Function, f)SHA 512/256_256-bit Hash Function, g)Snefru 256_256-bit Hash Function, h)Gost_256-bit Hash Function, i)SHA3-256_256-bit Hash Function, j)Haval 256,4_256-bit Hash Function, k)SHA-256_256-bit Hash Function

Also, several statistical tests are used to assess the randomness of AES ciphers. Statistical analysis is assessed using 4 metrics from NIST in addition to the poker test. In case 1 of this research, all results showed that the AES cipher achieved a high level of randomness using the poker test and four NIST metrics (frequency test, run test, long run test, and serial test). However, in case-2, only two metrics failed with the Gost crpto hash function, and one metric failed with the SHA-256 hash function. See Table 4 and Table 5.

Table 4: (Case-1) Poker test and 4-NIST Metrics for AES Cipher

Hash Function /128	*Frequency Test Max=3.8410*	*Poker Test Max=14.07*	*Run Test Max=9.488*	*Long Run Test Max=34*	*Serial Test Max=5.99*	*Result Pass or Fail*
MD2	0.5142	3.2679	3.6626	10	0.6223	**Pass 5/5**
MD4	0.2285	11.3482	8.9831	11	1.5448	**Pass 5/5**
Murmur3c	6.0571	13.8452	0.7733	13	1.0009	**Pass 5/5**
Tiger 128,4	0.1080	13.4594	6.9669	11	2.9790	**Pass 5/5**
Ripemd128	1.5750	3.8466	6.2760	11	2.1872	**Pass 5/5**
Haval 128,4	0.0223	7.6188	1.9122	14	0.1471	**Pass 5/5**
Murmur3f	1.1571	2.4534	5.6838	14	3.5267	**Pass 5/5**
XXh128	1.2223	8.2190	8.2258	13	2.3487	**Pass 5/5**
Haval 128,5	0.0437	1.7354	4.6518	14	1.1196	**Pass 5/5**
Tiger 128,3	0.5142	8.3369	2.8385	16	0.7286	**Pass 5/5**
MD5	0.2892	3.8680	3.1218	12	1.7234	**Pass 5/5**
Haval 128,3	0.0321	8.4655	1.6047	15	0.1033	**Pass 5/5**

Table 5: (Case-2) Poker test and 4-NIST Metrics for AES Cipher

Hash type	*Frequency Test Max=3.8410*	*Poker Test Max=14.0700*	*Run Test Max=9.4880*	*Long Run Test Max=34*	*Serial Test Max=5.9910*	*Result Pass or Fail*
Haval256, 3	0.2580	4.1252	3.8424	12	1.3849	**Pass 5/5**
Gost crpto	0.7000	**17.8211 Fail**	**13.1086 fail**	12	5.9673	**Pass 3/5**
Haval 256,5	0.1080	9.9765	3.8056	11	3.4862	**Pass 5/5**
Ripemd256	0.0223	7.8439	8.8126	11	0.2962	**Pass 5/5**
SHA 512/256	1.0937	8.0904	5.7889	9	1.3369	**Pass 5/5**
Snefru 256	0.3937	10.6838	8.2762	10	1.8967	**Pass 5/5**
Gost	0.2287	8.6262	4.4130	9	0.3758	**Pass 5/5**
SHA3-256	1.2223	7.8546	9.0356	17	1.4762	**Pass 5/5**
Haval 256,4	0.3133	10.7916	2.1817	13	0.7126	**Pass 5/5**
SHA-256	2.5080	17.3924	**9.8384 fail**	13	2.9224	**Pass 4/5**

One of the most important advantages of the proposed system is that it generates encryption keys using 22 different hash function methods, ensuring that a strong, unique (non-repeatable) key for strong encryption with the highest complexity is generated.

8. Conclusions

A good key schedule is one that exhibits good diffusion and confusion. This research calculated the randomness of the key generated by the proposed system, KGM. The NIST test determines whether the ratio of randomness and the NIST statistical tests are used to evaluate the strength of the proposed KGM's generated key. These tests include the poker test and 4-NIST tests, such as the frequency test, run test, long run test, and serial test.

When using all generated keys with AES encryption, the 22-generated keys for both cases 1 and 2 produced good results for entropy values, autocorrelation, poker tests, and 4-NIST tests. All the results obtained from the proposed system indicate that the AES cipher is satisfied with very good randomization. KGM used hash functions to generate 22 strong keys that improve the AES cipher's security level.

The increasing randomness of cryptographic algorithms indicates greater security. A low randomness value indicates a weak security in the algorithm, potentially enabling a cryptanalyst to partially or completely break it.

In this research, high values of entropy were obtained for AES cipher, as shown in previous tables, with the highest value of entropy of AES cipher obtained by using Haval 128,3 = 7.66/8 and the lowest entropy value obtained by using MD2 = 7.56/8. That means the best case for generating a unique key with 128 bits when using Haval 128, a 3 hash function.

In addition, it is also clear from the results of this research that high values of entropy were achieved for the AES cipher. Also, the results showed that the highest entropy value of the AES cipher obtained by using SHA-256= 7.66/8, and the lowest entropy value obtained by using Haval256, 3 = 7.57/8. That means it is the best case for generating a unique key with 256 bits when using the SHA-256 hash function. One of the most important advantages of the proposed system is that it generates encryption keys using 22 different hash function methods, ensuring that a strong, unique (non-repeatable) key for strong encryption with the highest complexity is generated.

References

[1] C. Li, F. Zhao, C. Liu, L. Lei, and J. Zhang, "A hyperchaotic color image encryption algorithm and security analysis," *Security and Communication Networks,* p. 8132547, 2019.

[2] Meryam Saad Fadhil, Alaa Kadhim Farhan and Mohammad Natiq Fadhil, "A lightweight AES Algorithm Implementation for Secure IoT Environment," *Iraqi Journal of Science,* vol. 62, no. 8, pp. 2759-2770, 2021.

[3] H. Wu and H. Wu, "Research on Computer Network Information Security Problems and Prevention Based on Wireless Sensor Network," *2021 IEEE Asia- Pacific Conference on Image Processing, Electronics and Computers (IPEC),* pp. 1015-1018, 2021.

[4] K. Gupta, D. Gupta, S. K. Prasad, and P. Johri, "A Review on Cryptography based Data Security Techniques for the Cloud Computing," *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE),* pp. 1039-1044, 2021.

[5] Mohanad, M. A. and Abdul Wahab, H. B., "Proposed New Blockchain Consensus Algorithm," *International Journal of Interactive Mobile Technologies (iJIM),* vol. 16, no. 20, pp. 162-176, 2022.

[6] Y. Harmouch and R. El Kouch, "The benefit of using chaos in key schedule algorithm," *Journal of Information Security and Applications,* vol. 45, pp. 143-155, 2019.

[7] R. E. J. Paje, A. M. Sison, and R. P. Medina, "Multidimensional key RC6 algorithm," *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy—ICCSP'19,* pp. 33-38, January 2019.

[8] V. Bidhuri, "Enhancing Password Security Using a Hybrid Approach of SCrypt Hashing and AES Encryption," MSc Thesis, School of Computing, National College of Ireland, Dublin, 2019.

[9] Gousia Nissar, Dinesh Kumar Garg, and Burhan Ul Islam Khan, "Implementation of Security Enhancement in AES by Inducting Dynamicity i n AES S-Box," *International Journal of Innovative Technology and Exploring Engineering (IJITEE),* vol. 8, no. 11, pp. 2364-2373, 2019.

[10] Hassan Rahmah Zagi, Abeer Tariq Maolood, "A New Key Generation to Greate Enhanced Security Version of AES Encryption Method," *Journal of College of Education,* vol. 2, pp. 1-16, 2021.

[11] Nourah Almrezeq, Mamoona Humayun, A. A. Abd El-Aziz and NZ Jhanjhi, "An Enhanced Approach to Improve the Security and Performance for Deduplication," *Turkish Journal of Computer and Mathematics Education,* vol. 12, no. 6, pp. 2866-2882, 2021.

[12] Vaibhavi Haveri, Vilaskumar Patil, "Hierarchical Dynamic Key Generation and Selective Transformation Model for Optimized AES Block Cipher Core," *International Journal of Scientific Research in Science and Technology,* vol. 9, no. 4, pp. 318-325, 2022.

[13] Alaa B. Baban and Safa A.Hameed, "Securing a Web-Based Hospital Management System Using a Combination of AES and HMAC," *Iraqi Journal for Electrical And Electronic Engineering,* vol. 19, no. 1, pp. 93-99, 2023.

[14] Thanikodi Manoj Kumar, Kasarla Satish Reddy, Stefano Rinaldi, Bidare Divakarachari Parameshachari and Kavitha Arunachalam, "A Low Area High Speed FPGA Implementation of AES Architecture for Cryptography Application," *MDPI, Electronics,* vol. 10, no. 16, pp. 1-22, 2021.

[15] Daemen, J., and Rijmen, V., "The Advanced Encryption Standard Process," in *Information Security and Cryptography*, Springer, Berlin, Heidelberg,, 2002.

[16] E. S. I. Harba, "Advanced Password Authentication Protection by Hybrid Cryptography & Audio Steganography," *Iraqi Journal of Science,* vol. 59, no. 1C, pp. 600-606, 2018.

[17] Aditya Pradeep, Vishal Mohanty, Adarsh Muthuveeru Subramaniam and Chester Rebeiro, "Revisiting AES SBox Composite Field Implementations for FPGAs," *IEEE Embedded Systems Letters,* vol. 11, no. 3, pp. 85-88, 2019.

[18] P. P. Pittalia, "A Comparative Study of Hash Algorithms in Cryptography," *International Journal of Computer Science and Mobile Computing,* vol. 8, no. 6, pp. 147-152, 2019.

[19] L. Hughes, "Basic Cryptography: Hash Function.," in *Pro Active Directory Certificate Services*, Apress, Berkeley, CA, 2022, pp. 19-22.

[20] Hala Abdulsalam and Assmaa A. Fahad, "Evaluation of Two Thresholds Two Divisor Chunking Algorithm Using Rabin Finger print, Adler, and SHA1 Hashing Algorithms," *Iraqi Journal of Science,* vol. 58, no. 4C, pp. 2438-2446, 2018.

[21] Mariam Duraid Abdul-Jabbar and Yousra abdul alsahib S.aldeen, "A Key Based Hybrid Approach for Privacy and Integrity in Multi-Cloud," *Iraqi Journal of Science,* vol. 64, no. 11, pp. 5952-5963, 2023.

[22] S. Afzal, U. Waqas, A. M. Mubeen, and M. Yousaf, "Statistical analysis of key schedule algorithms of different block ciphers," *Science International,* vol. 27, no. 3, p. 212608024, 2015.

[23] Kawther E. Abdullah and Nada Hussein M. Ali, "A Secure Enhancement for Encoding/ Decoding data using Elliptic Curve Cryptography," *Iraqi Journal of Science,* vol. 59, no. 1A, pp. 189-198, 2018.

[24] Yue Wua, Yicong Zhou, George Saveriades, Sos Agaian, Joseph P. Noonana and Premkumar Natarajan, "Local Shannon entropy measure with statistical tests for image randomness," *Information Sciences, ELSEVIER,,* vol. 222, pp. 323-342, 2018.

[25] Fatin K. Nasser and Suhad F. Behadili, "Breast Cancer Detection using Decision Tree and K-Nearest Neighbour Classifiers," *Iraqi Journal of Science,* vol. 63, no. 12, pp. 4987-5003, 2022.

[26] Barker EB, Roginsky AL and Davis R, "Recommendation for Cryptographic Key Generation," *National Institute of Standards and Technology, Gaithersburg, MD, NIST Special Publication 800-133,* p. Revision 2, 2020.

[27] Barker EB and Roginsky AL, "Transitioning the Use of Cryptographic Algorithms and Key

Lengths," *National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-131A,* p. Revision 2, 2019.

[28] Grassi L, Leander G, Rechberger C, Tezcan C and Wiemer F, "Weak-Key Distinguishers for AES," Cryptology ePrint Archive, 2019. [Online]. Available: https://eprint.iacr.org/2019/852.

[29] Leurent G and Pernot C, "New Representations of the AES Key Schedule," 2020. [Online]. Available: https://eprint.iacr.org/2020/1253.

[30] O. K. Dheyab, "Secure Storage in Cloud Computing Based on Hybrid Cipher Algorithms," *MSc Thesis, University of Baghdad, College of Science, Computer Science Department,* 2021.

[31] Abdullah A. Ghazi and Faez H. Ali, "Robust and Efficient Dynamic Stream Cipher Cryptosystem," *Iraqi Journal of Science,* vol. 59, no. 2C, pp. 1105-1114, 2018.

[32] D. H. Abbas, "Elliptic Curve Cryptosystems for Digital Voice Signals," *MSc Thesis, University of Baghdad, College of Science, Computer Science Department,* 2022.

[33] J. T. Abdulsattar, "Security Risks of the Metaverse World," *International Journal of Interactive Mobile Technologies,* vol. 16, no. 13, pp. 4-14, 2022.