International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020

Mahaviracharya Encryption Standard (MES)

Nagaraju. Bollepalli, Ramkumar.Penu

Abstract: ***Given the current use of the Internet, The most important thing is to provide security to the user's information. Many encryption algorithms already exist for this purpose. Here we discussed a new process called Mahaviracharya Encryption Standard. MES is a symmetric encryption algorithm. Here, this algorithm is cryptanalyzed, and compared with blowfish algorithm. MES algorithm can be used instead off algorithms like AES, Blowfish etc.***

Keywords: Mahaviracharya Encryption Standard, MES, Rasilabdacheda misravibhaga sutram, Symmetric Encryption Algorithm.

I.INTRODUCTION

Cryptography algorithms are very important to offer confidentiality to the data by encryption. There are two types of algorithms, Symmetric ciphers and Asymmetric ciphers. Mahaviracharya encryption standard is a conventional and a block cipher.

Mahaviracharya was a great mathematician born in Bharat (India). He belongs to the 9th century. He wrote a manuscript on geometry and algebra. The name of the book is *Ganitha Saara Sangraha*. In this manuscript, he talk about a formula .i.e. "*Rasilabdacheda misravibhaga sutram*", to separate the unidentified dividend integer, divisor and quotient as of their collective addition [1].

Rasilabdacheda misravibhaga sutram: "any appropriate freely selected integer deducted from the specified collective addition occurs to be the divisor. On dividing, by this divisor as augmented by one, the remainder, the requisite quotient is reached at. The very equal remainder, as reduced by quotient becomes the requisite dividend integer" [1].

A new cipher was constructed using *Rasilabdacheda misravibhaga sutram* labelled as Mahaviracharya Encryption Algorithm (MEA). In this system, plain text considered as divisor (a), secret key considered as quotient (c) and cipher text considered as combined sum (x). While decrypting the ciphertext x, to find plaintext a i.e. divisor in the formula, we must select a integer (k). But, for various k values, we can find distinct a, b, c integers.

Manuscript received on March 15, 2020.
Revised Manuscript received on March 24, 2020.
Manuscript published on March 30, 2020.
* Correspondence Author
B. Nagaraju, Assistant Professor, Department of CSE, University College for Women, Osmania University (OU), Hyderabad, India.
Dr. P. Ramkumar, Professor Department of CSE, University College of Engineering, Osmania University (OU), Hyderabad, India.

Therefore, to find accurate a at this point, we must select an appropriate k. An equation is given in decryption process, which is given below, to calculate the appropriate integer k [2].

A. Encryption Process

$$b=a * c$$
$$x=a+b+c$$

Ciphertext: x

B. Decryption Process

$k=c(x+1)/(c+)$

$a=x-k$

Plain text: a

This decryption process can be changed. The revised decryption process is:

$$a=(x-c)/(c+1)$$

II.BACKGROUND

We published a paper with title "Mahaviracharya Encryption Algorithm (MEA) with Modified Counter Mode and Comparing with AES Algorithm". This paper was described a technique, how to implement counter mode on Mahaviracharya Encryption Algorithm. Commonly, in counter mode, a random number is allocated to the counter and then added by one for each following block. In MEA, counter mode is not used as it is. In MEA entire plaintext is considered as one block. Here, we are applying counter mode to that one block, thus growing of counter number should not requisite [3].

In Mahaviracharya Encryption Algorithm plaintext (P), secrete key (K), counter number(R) are inconstant size integers and must equal or more than the 128 bits. Ciphertext, which is the output of MEA algorithm, size same as the plaintext size.

The encryption and decryption techniques are:

A. Encryption Method

1. Plaintext: $P=p_1\ p_2\ p_3\ p_4\ \ldots p_m$
   - P has m digits and $P \geq 128$ bits.
2. Counter value: $R=r_1\ r_2\ r_3\ r_4\ \ldots\ldots r_s$
   - R is a SecureRandom[5] integer, has s digits and $R \geq 128$ bits.
3. Secrete key: $K=k_1\ k_2\ k_3\ k_4\ldots k_q$
   - K has q digits and $K \geq 128$ bits.
4. $B=R*K$
5. $X=R+B+K = x_1\ x_2\ x_3\ldots\ldots\ldots x_m\ x_{m+1}\ x_{m+2}\ldots\ldots x_{m+n}$
   - X size $(m+n)$ > Plaintext (P) size.
   - If X size < P size, then X must be extended to x_{m+n}.
6. $X'= x_1 x_2\ x_3\ \ldots\ldots x_m$.
7. Ciphertext: $C=X' \oplus P$

B. Decryption Method

1. Ciphertext: $C=c_1 c_2 c_3 c_4 \ldots c_m$
   - C *has* m digits and $C \geq 128$ bits.
2. Counter number: $R=r_1 r_2 r_3 r_4 \ldots r_s$
   - R is SecureRandom[5] integer, has s digits and $R \geq 128$ bits.
3. Secrete Key: $K=k_1 k_2 k_3 k_4 \ldots k_q$
   - K has q digits and $K \geq 128$ bits.
4. $B=R*K$
5. $X=R+B+K = x_1 x_2 x_3 \ldots\ldots x_m x_{m+1} x_{m+2} \ldots\ldots x_{m+n}$
   - X size $(m+n)$ > Ciphertext(C) size.
   - If X size < C size, then X must be extended to x_{m+n}.
6. $X' = x_1 x_2 x_3 \ldots\ldots x_m$
7. Plaintext: $P=X' \oplus C$

III.CRYPTANALYSIS OF MEA

Generally, the encryption algorithm is being attacked to identifying the secrete key instead of identifying the plaintext of a specific cipher text. Cryptanalysis and brute-force attack are two common procedures to attacking symmetric encryption algorithms.

Cryptanalysis: Cryptanalytic attacks functioning based on the attributes of the encryption procedure and some awareness of the common attributes of the plain text or some model plaintext-ciphertext pairs [4].

Ciphertext-only, Known-plaintext, Chosen-plaintext, Chosen-ciphertext and Chosen-text are the several kinds of cryptanalytic attacks, founded by knowledge known to the cryptanalyst. Table-I briefly explains these cryptanalytic attacks [4].

Table-I: Types of Attacks on Ciphertext

Type of Attack	Known to Cryptanalyst
Ciphertext only	• Encryption algorithm • Ciphertext
Known plaintext	• Encryption algorithm • Ciphertext • One or more plaintext-ciphertext sets generated with the same secret key
Chosen plaintext	• Encryption algorithm • Ciphertext • Plaintext message selected by cryptanalyst, together with its matching ciphertext produced with the same secret key
Chosen ciphertext	• Encryption algorithm • Ciphertext • The intended ciphertext, chosen by the cryptanalyst, along with its associated decrypted plaintext generated by the secret key
Chosen text	• Encryption algorithm • Ciphertext • Plaintext message selected by cryptanalyst, along with its matching ciphertext created using the secret key • The intended ciphertext, chosen by the cryptanalyst, along with its associated decrypted plaintext generated by the secret key

Relatively weak algorithms are only unable to overcome the ciphertext-only attack. Usually, an encryption algorithm is constructed to overcome the known-plaintext attack [4].

A *brute-force attack* is an attack, which is trying each possible key till an understandable conversion of the ciphertext to plaintext is acquired. Generally, fifty percent of the imaginable secrete keys should be experimented to get success. Below table shows how much time is involved for various key spaces [4].

Key Size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 Decryptions/s	Time Required at 10^{13} Decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} years	1.8×10^{56} years
26 characters (permutation)	Monoalphabetic	$2! = 4 \times 10^{26}$	2×10^{26} ns = 6.3×10^{9} years	6.3×10^{6} years

As explained above, an encryption algorithm is designed to overcome, at least, known-plaintext attack. So, ciphertext-only and known-plaintext attacks are analysed for this algorithm along with brute-force attack.

A. Ciphertext-only Attack

In ciphertext-only attack encryption algorithm and ciphertext are known by cryptanalyst or attacker. From this information he wants to get the plaintext or secrete key.

From MEA algorithm

$$C=X' \oplus P \qquad (1)$$

$$c_1 c_2 \ldots\ldots c_m = x_1 x_2 x_3 \ldots x_m \oplus p_1 p_2 p_3 p_4 \ldots p_m$$

$$X=R+B+K$$

$$X = R + R*K + K$$

$$x_1 x_2 x_3 \ldots\ldots x_m x_{m+1} x_{m+2} \ldots\ldots x_{m+n} = R + R*K + K$$

$$X = X' x_{m+1} x_{m+2} \ldots\ldots x_{m+n} = R + R*K + K \qquad (2)$$

In (1) ciphertext C and in (2) random number R only known by cryptanalyst. From (1), to get plaintext P cryptanalyst required X'. To get the X', from (2), secrete key K is required, but it is not known by cryptanalyst. So, mathematically identifying plaintext P, secrete key K is not possible in ciphertext-only attack.

B. Known-plaintext attack

In known-plaintext attack encryption algorithm, ciphertext and plaintext are known by cryptanalyst. From this information he wants to get the secrete key.

Case 1:

if cryptanalyst known only one pair of ciphertext (C) and plaintext (P)

From MEA algorithm

$$C = X' \oplus P$$

$$X' = C \oplus P \qquad (3)$$

$$x_1 x_2 x_3 \ldots x_m = c_1 c_2 \ldots c_m \oplus p_1 p_2 p_3 p_4 \ldots p_m$$

$$X=R+B+K$$

$$X = R + R*K + K$$

$$x_1 x_2 x_3 \ldots\ldots x_m x_{m+1} x_{m+2} \ldots\ldots x_{m+n} = R + R*K + K$$

$$X' x_{m+1} x_{m+2} \ldots\ldots x_{m+n} = R + R*K + K$$

$$X' x_{m+1} x_{m+2} \ldots\ldots x_{m+n} - R = R*K + K$$

$$X' x_{m+1} x_{m+2} \ldots\ldots x_{m+n} - R = K(R+1)$$
$$K = (X' x_{m+1} x_{m+2} \ldots\ldots x_{m+n} - R) / (R+1) \quad (4)$$

Initially, in (3) all values ciphertext *C*, plaintext *P* and *X'* are known by cryptanalyst or attacker. In (4) random number *R* known by cryptanalyst or attacker. To get secrete key *K*, from (4), *X'* and *R* can be applied in (4).

Without knowing the $x_{m+1} x_{m+2} \ldots\ldots x_{m+n}$, mathematically it is not possible to identifying secrete key *K* in Known-plaintext attack.

Example:

R= 912489429610476151398672553890893914984 6
P= 578935737319434261391331626600117527861 0798
K=379411324809664174035248110784028779350563 106848
X=R+B+K
=346208823363325561214055648887275864075099 830574881305011939756607401298888205308300 2102
X'=34620882336332556121405564888727586407509 98
C= X' ⊕ P
= 867521690102181608602719951010072859302136

In this attack, cryptanalyst knows *R*, *P*, *C*. From this information he can get *X'* value by using $X' = C \oplus P$. Then, he can substitute these values in (4)

$K = (X' x_{m+1} x_{m+2} \ldots\ldots x_{m+n} - R) / (R+1)$
$K = (462088233633255612140556488872758640750998 x_{m+1} x_{m+2} \ldots\ldots x_{m+n} - 912489429610476151398672553890893914984 6) / (912489429610476151398672553890893914984 6 + 1)$ (5)

In (5), without knowing the $x_{m+1} x_{m+2} \ldots\ldots x_{m+n}$ finding the *K* value is mathematically not possible.

Case 2: if cryptanalyst known more than one pair of ciphertext (*C*) and plaintext (*P*) which are using same secret key (*K*).

For instance, here we are assuming cryptanalyst known two pairs of plaintext and ciphertext, those are (C_1, P_1) and (C_2, P_2)

From (3) we can write

$$X_1' = C_1 \oplus P_1 \quad \text{and} \quad X_2' = C_2 \oplus P_2$$

From (4) we can write

$K = (X_1' x_{m+1} x_{m+2} \ldots\ldots x_{m+n} - R_1) / (R_1 + 1)$ (6)
$K = (X_2' x_{i+1} x_{i+2} \ldots\ldots x_{i+j} - R_2) / (R2 + 1)$ (7)

From (6) and (7)

$(X_1' x_{m+1} x_{m+2} .. x_{m+n} - R_1)/(R_1+1) = (X_2' x_{i+1} x_{i+2} ... x_{i+j} - R_2)/(R_2+1)$ (8)

From (8), without knowing the $x_{m+1} x_{m+2} \ldots\ldots x_{m+n}$ and $x_{i+1} x_{i+2} \ldots\ldots x_{i+j}$ cryptanalyst or attacker can't solve the equation to find the secrete key *K* value.

C. Brute-Force Attack

To get secrete key *K* from (2) cryptanalyst can apply brute-force attack. In brute-force attack cryptanalyst experimenting each probable key till an understandable conversion of the ciphertext to plaintext is gained. But, here one problem, that is, secrete key *K* size. *K* size is not fixed, it is variable length, and it is greater than or equal to 128 bits. Without knowing the length of the secrete key, trying every possible key not possible. If we assume *K* length is 128 bits, though, it is taking 5.3 X 10^{17} years (10^{13} decryption/sec).

To get secrete key *K* from (4) and (6) cryptanalyst should know the value $x_{m+1} x_{m+2} \ldots x_{m+n}$. Here, cryptanalyst can apply brute-force attack to find $x_{m+1} x_{m+2} \ldots x_{m+n}$, which means he can apply some series of digits randomly in the place of $x_{m+1} x_{m+2} \ldots x_{m+n}$. But it is not easy, because without knowing the *n* value, the selection of random number for replacing $x_{m+1} x_{m+2} \ldots x_{m+n}$ is very big problem.

To get secrete key *K* from (8) is more difficult than the previous case, which is explained in above paragraph. Because, in this situation attacker or cryptanalyst has to choose two numbers to replace the $x_{m+1} x_{m+2} \ldots x_{m+n}$ and $x_{i+1} x_{i+2} \ldots x_{i+j}$. Without knowing the *n*, *j* values, which are may or may not equal, it is a very big problem to the cryptanalyst or attacker.

So, from the above analysis, we can say that, Mahaviracharya Encryption Algorithm is secured from brute-force attack.

When checking the above mentioned concept practically, we observed that, if *n*, *j* values are small, then it is easy to identify the secrete key in known-plaintext attack. This is explained below with two examples.

Example 1:

Random number (*R*) =*356*, Secrete key (*K*) = *2741*, Plaintext (*P*) = *3194*.

From (2):

*X= R+R*K+K*
*X= 356+356*2741+2741*
X= 978893
X'= 9788 (from the MEA algorithm, Plaintext *P* length = *X'* length)

We can calculate ciphertext *C* value using (1)

C = X' ⊕ P = 10822

In known-plaintext attack, attacker can knows *P*, *C*, and *R*. From (1), *X'* value will be easily founded as below:

C = X' ⊕ P
X' = C ⊕ P
X' = 10822 ⊕ 3194 = 9788

From (4)

$K = (X' x_{m+1} x_{m+2} \ldots\ldots x_{m+n} - R) / (R+1)$
$K = (9788\ x_{m+1} x_{m+2} \ldots\ldots x_{m+n} - R) / 357$

Case 1:

If we assume, $x_{m+1} x_{m+2} \ldots\ldots x_{m+n}$ is only one digit (*n=1*) then *X* value will be *97880* or *97881* or *97882* or .....*97889*. If we try to find *K* value by applying these *X* values, for example, if we consider 97880 as *X* value *K= (97880-356) / 357 = 97524 / 357 = 273.176471* *K* value should be an integer.

So, above assumption is not correct. Like that, if we consider remaining numbers as *X*, we cannot get a perfect integer value.

Case 2:

If we assume, $x_{m+1} x_{m+2} \ldots\ldots x_{m+n}$ is two digits (*n=2*), then X value will be *978800* or *978801* or *978802* or .....*978899*. If we try to find *K* value by applying these *X* values, for example, if we consider *97880* as *X* value *K= (978800-356) / 357 = 978444 / 357 = 2740.7395* *K* value should be an integer.

So, above assumption is not correct. Like that, if we assuming remaining numbers as *X*, we can get a perfect integer value for *978893* only.

$K= (978893-356) / 357 = 978537 / 357 = 2741$ (9)

so, we can think that, secrete key *K* may be 2741.

Case 3:

If we assume, $x_{m+1}\, x_{m+2}\ldots\ldots x_{m+n}$ is three digits ($n=3$), then *X* value will be *9788000* or *9788001* or *9788002* or .....*9788999*. If we try to find *K* value by applying these *X* values, we can get a perfect integer value *for 9788225*, *9788582*, and *9788939* only and *K* values are *27417*, *27418*, and *27419* respectively.

Like this, we can get so many K values as per our assumption for $x_{m+1}\, x_{m+2}\ldots\ldots x_{m+n}$.

Example 2:

Random number (*R*) =*491*, Secrete key (*K*) = *2741*, Plaintext (*P*) = *2587*.

From (2):

$X= R+R*K+K$

$X= 491+491*2741+2741$

$X= 1349063$

$X'= 1349$ (from the MEA algorithm, Plaintext *P* length = *X'* length)

We can calculate ciphertext *C* value using (1)

$C = X' \oplus P = 3934$

In known-plaintext attack, attacker can knows *P*, *C*, and *R*. From (1), *X'* value will be easily founded as below:

$C = X' \oplus P$

$X' = C \oplus P$

$X' = 3934 \oplus 2587 = 1349$

From (4)

$K= (X'\, x_{m+1}\, x_{m+2}\ldots\ldots x_{m+n} - R\ ) / (R+1)$

$K= (1349\, x_{m+1}\, x_{m+2}\ldots\ldots x_{m+n} -491) / 492$ *Case 1:*

If we assume, $x_{m+1}\, x_{m+2}\ldots\ldots x_{m+n}$ is only one digit ($n=1$) then *X* value will be *13490* or *13491* or *13492* or ..... *13499*. If we try to find *K* value by applying these *X* values using above equation, we cannot get a perfect integer value.

Case 2:

If we assume, , $x_{m+1}\, x_{m+2}\ldots\ldots x_{m+n}$ is two digits ($n=2$), then *X* value will be *134900* or *134901* or *134902* or ..... *134999*. If we try to find *K* value by applying these *X* values using above equation, we cannot get a perfect integer value.

Case 3:

If we assume, $x_{m+1}\, x_{m+2}\ldots\ldots x_{m+n}$ is three digits ($n=3$), then *X* value will be *1349000* or *1349001* or *1349002* or ..... *1349999*. We can get a perfect integer value for *1349063*

$K=(1349063 – 491) / 492 = 1348572 / 492 = 2741$ (10)

We can find another number, that is, *1349555*, and *K* value for this number is *2742*.

Case 4:

If we assume, $x_{m+1}\, x_{m+2}\ldots\ldots x_{m+n}$ is three digits ($n=4$), we can find so many K values.

If we consider example *1* or example *2* individually, that means if attacker applies known-plain text attack using single pair (*C*, *P*), it is difficult to guess the *K* value, because attacker can get so many number of *K* values depends upon the *n* value as we saw in case *2*, case *3* in example *1* and case *3*, case *4* in example *2*.

If we consider example *1* and example *2* together, that means if attacker knows two are more pairs of (*C*,*P*) which are using same secrete key , it is easy to guess the *K* value by identifying the common *K* value in different pairs of (*C*, *P*). In our situation, *K* value in (9) and *K* value in (10) are equal. So using (8) attacker can easily guess the *K* value, that is *2741*.

The above mentioned contexts will possible when *n*, *j* values are small enough; otherwise there is no problem by using this algorithm.

IV.MAHAVIRACHARYA ENCRYPTION STANDARD (MES) – A NEW VERSION TO INCREASE SECURITY

Mahaviracharya Encryption Algorithm (MEA) which is explained in section II is giving sufficient security from attacks like ciphertext-only attack, known-plaintext attack and brute-force attack. There is one drawback in this algorithm explained in previous section.

In known-plaintext attack, attacker can get *X'* value. To get secrete key he/she required *X* value i.e. $X'\ x_{m+1}\, x_{m+2}\ldots x_{m+n}$. In *X* value *X'* is known, so attacker has to find $x_{m+1}\, x_{m+2}\ldots x_{m+n}$. From the algorithm, finding the $x_{m+1}\, x_{m+2}\ldots x_{m+n}$ value mathematically is not possible. We can apply brute-force algorithm to find $x_{m+1}\, x_{m+2}\ldots x_{m+n}$. If $x_{m+1}\, x_{m+2}\ldots x_{m+n}$ value is less than *128* bits, then it will be find in sufficient time.

To overcome this drawback, we added a new condition, $x_{m+1}\, x_{m+2}\ldots x_{m+n}$ should be greater than or equal to 128 bits, to 5th point in encryption and decryption methods of the algorithm. The new encryption and decryption methods are:

A. Encryption Method

1. Plain text: $P=p_1\, p_2\, p_3\, p_4\, \ldots p_m$
   - *P* has *m* digits and $P \geq 128$ bits.
2. Counter number: $R=r_1\, r_2\, r_3\, r_4\, \ldots\ldots r_s$
   - *R* is SecureRandom[5] integer, has *s* digits and $R \geq 128$ bits.
3. Secrete key: $K=k_1\, k_2\, k_3\, k_4\ldots k_q$
   - *K* has *q* digits and $K \geq 128$ bits.
4. $B=R*K$
5. $X=R+B+K = x_1\, x_2\, x_3\ldots x_m\, x_{m+1}\, x_{m+2}\ldots x_{m+n}$
   - *X* size ($m+n$) > Plaintext (P) size.
   - $x_{m+1}\, x_{m+2}\ldots x_{m+n}$ size ≥ 128 bits.
   - If *X* size < *P* size then *X* must be extended to x_{m+n}. To extend *X*, here, BigInteger(*X*.getBytes("usascii")) java class is used repeatedly.
6. $X'= x_1 x_2\, x_3\, \ldots\ldots x_m.$
7. Ciphertext: $C=X' \oplus P$

Retrieval Number: F9485038620/2020©BEIESP
DOI: 10.35940/ijrte.F9485.038620
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

B. Decryption Method

1. Ciphertext: $C=c_1\,c_2\,c_3\,c_4\,\ldots c_m$
   - C has m digits and $C \geq 128$ bits.
2. Counter number: $R=r_1 r_2\, r_3\, r_4\, \ldots\ldots r_s$
   - R must be same (encryption method), shared by sender and receiver.
3. Secrete Key: $K=k_1\, k_2\, k_3\, k_4 \ldots k_q$
4. $B=R*K$
5. $X=R+B+K = x_1\, x_2\, x_3 \ldots\ldots\, x_m\, x_{m+1}\, x_{m+2} \ldots\ldots x_{m+n}$
   - X size $(m+n)$ > ciphertext (C) size.
   - $x_{m+1}\, x_{m+2} \ldots x_{m+n}$ size ≥ 128 bits.
   - If X size < P size then X must be extended to x_{m+n}. To extend X, here, BigInteger(X.getBytes("usascii")) java class is used repeatedly[5].
6. $X' = x_1\, x_2\, x_3\, \ldots\ldots x_m$
7. Plaintext: $P=X' \oplus C$.

V.CRYPTANALYSIS OF MES

Mahaviracharya Encryption Standard (MES) providing more security than previous version which is explained in section II. As explained in section III, Mahaviracharya Encryption Standard (MES) is providing security against ciphertext-only attack and known-plaintext attack.

Mahaviracharya Encryption algorithm which is explained in section II has one drawback that is discussed in III.C subsection. Mahaviracharya Encryption Standard (MES) overcome this drawback by adding the condition in 5^{th} point in IV.A, IV.B subsections, that is $x_{m+1}\, x_{m+2} \ldots x_{m+n}$ length should be minimum 128 bits.

Applying the brute-force attack for (4) is very difficult because replace the $x_{m+1}\, x_{m+2} \ldots x_{m+n}$ with any number must be ≥ 128 bits. If we try, it is taking $5.3X10^{17}$ years of time to applying 2^{128} alternatives for $x_{m+1}\, x_{m+2} \ldots x_{m+n}$.

So, Mahaviracharya Encryption Standard (MES) is providing sufficient security against ciphertext-only attack, known-plaintext attack and brute-force attack.

These attacks were discussed in section III and V while random number (R) is known by attacker. If random number (R) shared between sender and receiver in encrypted form, MES is very difficult to break.

VI.RESULTS AND ANALYSIS OF EXPERIMENT

Java code is used to implement the MES. MES outcomes are compared with blowfish algorithm outcomes.

Here, encryption and decryption times are computed for various sizes of text data. Jdk-13.0.1_windows-x64_bin java language, Apache Net Beans IDE 11.2 version, Windows 10 64 bit operating system, Intel(R) core(TM) i5-7500 CPU with 3.40 GHz processor and 4GB RAM desktop is used to do this experiment.

A. Comparing using Encryption Time

We compute the encryption times of various sizes of text messages using MES and blowfish. The details are showed in table-II (KB- kilobytes, ms - milliseconds). From this table, we can easily understand, that is MES is fast than blowfish algorithm.

Table-II: Encryption process

Data	MES	Blowfish
10KB	63 ms	100 ms
20KB	98 ms	162 ms
30KB	205 ms	272 ms
40KB	265 ms	301 ms
50KB	344 ms	398 ms

B. Comparing using Decryption Time

We compute the encryption times of various sizes of text messages using MES and blowfish. The details are showed in table-II (KB- kilobytes, ms - milliseconds). From this table, we can easily understand, that is MES is fast than blowfish algorithm.

Table-III: Decryption Process

Data	MES	Blowfish
10KB	70 ms	105 ms
20KB	100 ms	170 ms
30KB	250 ms	284 ms
40KB	300 ms	312 ms
50KB	402 ms	409 ms

VII.CONCLUSION

Mahaviracharya Encryption Standard is a new algorithm which has the good features like easy to understand, easy to implement. It is giving good security against brute-force attack, ciphertext-only and known-plaintext attacks. It is taking less time for encryption and decryption methods than the blowfish algorithm. With all of this in mind, we can say, Mahaviracharya Encryption Standard is a newfound algorithm.

REFERENCES

1. Rangacarya.M, "Ganitha-Sara- Sangraha of Mahaviracarya", Cosmo Publication New Delhi, India.
2. B.Nagaraju, P.Ramkukar, "A New Method for Symmetric Key Cryptography", International Journal of Computer Applications (0975 – 8887), Volume 142 – No.8, May 2016.
3. Bollepalli.Nagaraju, and Penu.Ramkumar, "Mahaviracharya Encryption Algorithm (MEA) with Modified Counter Mode and Comparing with AES Algorithm" , Proceedings of the International Conference on Emerging Trends in Engineering (ICETE), Vol. 1.,Springer publications-2019.
4. W. Stallings, "Cryptography and network security: principles and practices". Pearson Education India, 20014.
5. https://docs.oracle.com/javase/7/docs/api/java/security/SecureRandom.html

AUTHOR PROFILE

B. Nagaraju is assistant professor(c) in University College for Women, Osmania University (OU), Hyderabad. He is also a research scholar in University College of Engineering (OU). His area of interest is Cryptography and Network Security. He published 2 papers in international journals and presented a research paper in international conference.

Dr. P. Ramkumar is a Retd. Professor in Dept. of CSE, University College of Engineering, OU. He published several papers in international journals. He worked as in charge & member of a team in Software Industry for 8 years. He is supervising 7 Ph.D. students and 1 student is awarded with Ph.D. degree.