Journal of Mathematical Sciences and Modelling, 6 (3) (2023) 120-132
Research Article

Journal of Mathematical Sciences and Modelling

Journal Homepage: www.dergipark.gov.tr/jmsm
ISSN 2636-8692
DOI: http://dx.doi.org/10.33187/jmsm.1241420

DNA Secret Writing With Laplace Transform of Mittag-Leffler Function

Mehmet Çağrı Yılmazer[1], Emrah Yılmaz[2*], Tuba Gülşen[3] and Mikail Et[4]

[1, 2*, 3, 4]*Department of Mathematics, Faculty of Science, Firat University, Elazığ, Türkiye*
[*]*Corresponding author*

Article Info

Keywords: *Cryptology, Data encryption, DNA, Mittag-Leffler function, Laplace transform, Statistical tests*
2010 AMS: *33E12, 44A10, 62P99, 68P25, 92D20, 94A60*
Received: *24 January 2023*
Accepted: *25 August 2023*
Available online: *10 September 2023*

Abstract

In this study, we present a new cryptosystem named Deoxyribose Nucleic Acid (DNA) secret writing with the Laplace transform of the Mittag-Leffler function. The method is proper for encrypting large files. In this technique, we consider the original message as binary sequence. These binary streams corresponding to the plain text is transformed to DNA bases by utilizing DNA encoding, then the DNA codes are transformed to positive integers. We apply the Laplace transform to these numbers which are coefficients of the expansion of the Mittag-Leffler function. To provide multi-stage protection, the outcome coefficients are transformed to binary sequences and other level of encryption with cumulative XOR is applied and equivalent MSBs obtained at every iteration are utilized for building cipher text. Decryption is implemented in the opposite way. We employ monobit test, correlation analysis for measuring the reliability of encryption and Python programming language to obtain secret message, the plain text and computations of statistical tests.

1. Introduction

Cryptography is simply the science of utilizing mathematics for encryption and decryption of information. Its essential aim is to provide two groups, to communicate over a channel that is insecure in the manner that an opponent cannot comprehend the message that is for the intended recipient. Information security is becoming increasingly important in the utilization of electronic telecommunication in financial activities. Cryptography is intended to enable safety services and it has become an important technique in many areas for information protection. Encryption is the procedure of converting an original message called plain text to obscuring form named cipher text. We commonly use it for confidentiality and generally for secret communication [1–3].

A cipher is defined as a method to apply encryption and decryption. As mentioned above, the actual message is regarded as plain text and the enciphered form as cipher text. The encrypted message comprises the entire information of the original message but is not in form decipherable by a person or computer unless an appropriate tool to decrypt it. It must look like nonsense to those not aimed to understand it. We usually parameterize ciphers by using a bit of subsidiary information, named as a key. The encrypting process is diversified based on the key, which modifies the elaborated operation of the method. Unless the key is proper, decryption is not possible.

Several methods for cryptography are presented in [4–11]. The mathematical method utilizing matrices for it are presented in Dhanorkar and Hiwarekar [12], Overbey and others [13], Saeednia [14]. A message is encrypted by means of series expansion of $f(t)$ and its Laplace transform [1, 15, 16]. In [17], Hiwarekar uses the Laplace transform of hyperbolic cosine functions for encryption and decryption. Here, we propose Mittag-Leffler function and its variations for encryption and decryption by Laplace transform in DNA secret codes. First, let's talk about the importance of DNA chains in encryption.

It was stated by Watson and Crick that DNA chains are important in the encryption of information [18]. Although DNA encryption is a new and useful, it is not as useful as the conventional method. We can combine it with current cryptographic systems to enable improved

***Email addresses and ORCID numbers*: m.cagri.yilmazer@gmail.com, 0000-0001-9784-838X (M. Ç. Yılmaze), emrah231983@gmail.com, 0000-0002-7822-9193 (E. Yılmaz), tubagulsen87@hotmail.com, 0000-0002-2288-8050 (T. Gülşen), mikailet68@gmail.com, 0000-0001-8292-7819 (M. Et)**
Cite as "M. Ç. Yılmazer, E. Yılmaz, T. Gülşen, M. Et, *DNA secret writing with Laplace transform of Mittag-Leffler function*, J. Math. Sci. Model., 6(3) (2023), 120-132."

security [19–21]. Adleman's research [22] in DNA computing and Viviana Risca's project on DNA steganography gave rise to new fields of DNA cryptography and stenography [23]. In [24], Sukalyan and Moumita Som presented a new technique for DNA encoding by using Laplace transform. For a better understanding of the subject, let's explain the structure of DNA.

DNA is a long polymer consisting of numerous nucleotides in the shape of a double helix storing genetic information. Each spiral chain composed of sugar phosphate as spine and bases are joined to complemental chain by hydrogen bridges between dual bases Adenine(A), thymine(T), guanine(G) and cytosine(C). Adenine and thymine are joined by two hydrogen bridges while guanine and cytosine are joined by three. It is shown that DNA cryptography is very effective in its initial phase. Nowadays, various DNA computing algorithms can be solutions for cryptanalysis and stenography problems. The idea of DNA computing compounded with areas of cryptography and steganography becomes a new technique for nonbreakable algorithms [25, 26].

There exists two complemental strands in the framework of DNA. Each base in DNA has a sugar constituent connected to a phosphate group at one place, and to a nitrogen comprising nucleotide bounded at another place. The strands in DNA possess the phosphate of one base connected to sugar of the next base to form a chain of consecutive sugars and phosphates with dangling nitrogenous bases, as seen in Figure 1.1.

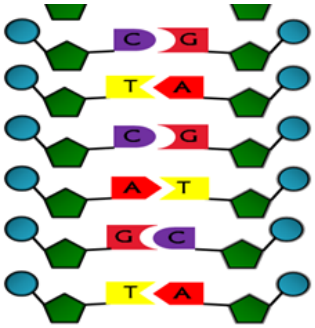

Figure 1.1: Fundamental DNA framework.

DNA includes two such strands, intertwined with each other to constitute a duplex helix with the nucleoids on the inside. Each A on one strand constitutes weak bonds with a T on the other chain, and each C on a chain weakly to a G on the opposite strand. Thus, the two chains are complemental and the sequence in one can be understood from other's sequence. The fundamental DNA framework is illustrated in Figure 1.2.

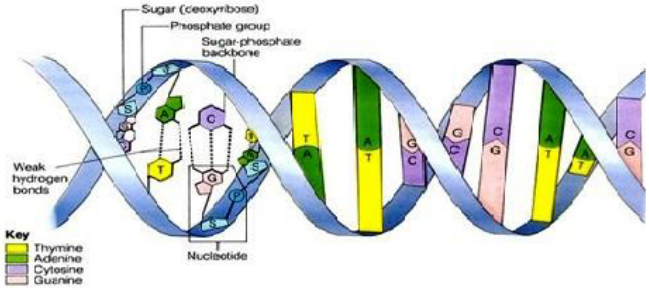

Figure 1.2: Composite of Nucleotide bases in chains.

These complemental chains possess codons as basic components. Codons are essentially triads of nucleotide bases. Table 1 below demonstrates codons building DNA sequences in two complemental chains. As can be observed DNA nucleotide bases are included by codons and are complemented each other. We can utilize these codons for encryption and decryption of the information.

2. Preliminaries

In this section, we provide definitions and some features of the Mittag-Leffler function, Laplace transform, and relations between them which are important in applying our method to cryptography.

Definition 2.1. *Let f be a function defined on the set of positive real numbers. Then, $L\{f(t)\}$ or $F(s)$ which is the Laplace transform of $f(t)$ is defined by*

$$L\{f(t)\} = F(s) = \int_0^\infty e^{-st} f(t)\, dt. \tag{2.1}$$

The domain of $F(s)$ is set of values of s which the above improper integral converges. Furthermore, we can write the formula (2.1) as $f(t) = L^{-1}\{F(s)\}$. In this situation, $f(t)$ is defined as inverse Laplace transform of $F(s)$. We call the symbol L which transforms $f(t)$ into $F(s)$ as the Laplace transform operator. On the other hand, we can define the symbol L^{-1} which transforms $F(s)$ to $f(t)$ as the inverse Laplace operator [27].

The Laplace transform is named in memory of the French mathematician, physicist, and astronomer *Pierre-Simon Marquis de Laplace* (1749-1827), who utilized the transform in his inquiries of probability theory.

Assume that the functions f and g have Laplace transforms for $s > c_1$ and $s > c_2$, respectively. If c represents the maximum of the two numbers c_1 and c_2, then for $s > c$ and constant α and β, following expression holds:

$$\int_0^\infty e^{-st}[\alpha f(t) + \beta g(t)]\,dt = \alpha \int_0^\infty e^{-st} f(t)\,dt + \beta \int_0^\infty e^{-st} g(t)\,dt.$$

That is to say L is a linear transform

$$L\{\alpha f(t) + \beta g(t)\} = \alpha L\{f(t)\} + \beta L\{g(t)\}.$$

Moreover, using the features of the definite integral, the Laplace transform of any finite functions of t is the sum of Laplace transforms of single functions. Likewise, the same is held for inverse Laplace transforms [27].

We give some fundamental Laplace transforms as follows [27]

$$\begin{aligned} L\{t^n\} &= \frac{n!}{s^{n+1}}, \\ L^{-1}\{\frac{1}{s^{n+1}}\} &= \frac{t^n}{n!}. \end{aligned}$$

Before touching on the Laplace transform of the Mittag-Leffler function and its variations, we want to mention about Mittag-Leffler function, its properties, and its history.

The exponential function e^z has a very significant role in the theory of integer-order differential equations. One-parameter generalization of it, the function that is now represented by the following definition [28, 29].

Definition 2.2. *One-parametric Mittag-Leffler function is defined by*

$$E_\alpha(z) = \sum_{j=0}^{\infty} \frac{z^j}{\Gamma(\alpha j + 1)} \ (\alpha \in \mathbb{C}),$$

where Γ *is the gamma function. It was presented by Mittag-Leffler [30–32] and investigated also by Wiman [33, 34].*

The two-parameter form of the Mittag-Leffler function, which plays a critical role in the fractional calculus was presented by Agarwal [29,35]. By using Laplace transform method, Humbert and Agarwal get several relations for this function [29, 36]. It could have been named the Agarwal function. However, Humbert and Agarwal greatheartedly left the same notation as for the one-parameter Mittag-Leffler function, and this is why we call the two-parameter function as the Mittag-Leffler function [28, 29, 37].

Definition 2.3. *By using the series expansion, a two-parameter form of the Mittag-Leffler function can be defined as follows [28, 29]*

$$E_{\alpha,\beta}(z) = \sum_{j=0}^{\infty} \frac{z^j}{\Gamma(\alpha j + \beta)}, \ (\alpha, \beta > 0). \tag{2.2}$$

From the definition, we get

$$E_{1,m}(z) = \frac{1}{z^{m-1}} \left\{ e^z - \sum_{j=0}^{m-2} \frac{z^j}{j!} \right\}.$$

Some of the special conditions for Mittag-Leffler function are the hyperbolic sine and cosine:

$$\begin{aligned} E_{2,1}\left(z^2\right) &= \sum_{j=0}^{\infty} \frac{z^{2j}}{\Gamma(2j+1)} = \sum_{j=0}^{\infty} \frac{z^{2j}}{(2j)!} = \cosh z, \\ E_{2,2}\left(z^2\right) &= \sum_{j=0}^{\infty} \frac{z^{2j}}{\Gamma(2j+2)} = \frac{1}{z} \sum_{j=0}^{\infty} \frac{z^{2j+1}}{(2j+1)!} = \frac{\sinh z}{z}. \end{aligned}$$

For $\beta = 1$, we get one-parameter Mittag-Leffler function as

$$E_{\alpha,1}(z) = \sum_{j=0}^{\infty} \frac{z^j}{\Gamma(\alpha j + 1)} = E_\alpha(z).$$

Using term-by-term derivation, we can build from (2.2) the series expansion of derivatives of the two-parameter Mittag-Leffler function [38]

$$\frac{d^k}{dz^k} E_{\alpha,\beta}(z) = \sum_{j=k}^{\infty} \frac{(j)_k}{\Gamma(\alpha j + \beta)} z^{j-k}, \ k \in \mathbb{N},$$

with $(x)_k$ representing the factorial below

$$(x)_k = x(x-1)\ldots(x-k+1).$$

By modifying the index of the summation, we can derive the following expression of the kth order derivative of two-parameter Mittag-Leffler function by

$$\frac{d^k}{dz^k}E_{\alpha,\beta}(z) = \sum_{j=0}^{\infty}\frac{(j+k)_k}{\Gamma(\alpha j+\alpha k+\beta)}z^j.$$

We give the following Laplace transform of $t^{\alpha k+\beta-1}E^{(k)}_{\alpha,\beta}(\pm at^\alpha)$, which has significant role in our study [29],

$$\begin{aligned} L\{t^{\alpha k+\beta-1}E^{(k)}_{\alpha,\beta}(\pm at^\alpha)\} &= \int_0^\infty e^{-st}t^{\alpha k+\beta-1}E^{(k)}_{\alpha,\beta}(\pm at^\alpha)\,dt \\ &= \frac{k!s^{\alpha-\beta}}{(s^\alpha \mp a)^{k+1}},\ \left(Re(s) > |a|^{\frac{1}{a}}\right). \end{aligned}$$

3. Main Results

For encryption of the message, P, the plain text, is transformed into 8-bit binary codes of ASCII values corresponding to the original text, P_{bin}. Then, we transform P_{bin} into DNA codes, P_{dna} by applying the following assignments

Binary stream	00	01	10	11
DNA Nucleotide Base	A	C	G	T

Table 3.1: DNA Nucleotide bases and their corresponding 2 bit binary stream

where A, T, G, C are DNA nucleotide pairs. Afterwards, P_{dna} is transformed into sequence of integers, P_{int} by utilizing the following table,

A	T	G	C
10	20	30	40

Table 3.2: DNA nucleotide bases and their corresponding decimal values.

we can correspond base pairs to numeric values. Subsequently, every integer in P_{int} can be utilized as the coefficients of the Laplace transform of the function $f(t) = Gt^{\alpha k+\beta-1}E^{(k)}_{\alpha,\beta}(\pm at^\alpha)$, in other words

$$\begin{aligned} L\{Gt^{\alpha k+\beta-1}E^{(k)}_{\alpha,\beta}(\pm at^\alpha)\} &= L\left\{\sum_{j=0}^{\infty}\frac{G_j(j+k)!}{j!}\frac{(\pm a)^j t^{\alpha j+\alpha k+\beta-1}}{\Gamma(\alpha j+\alpha k+\beta)}\right\} \\ &= \sum_{j=0}^{\infty}\frac{G_j(j+k)!(\pm a)^j}{j!}\frac{1}{s^{\alpha j+\alpha k+\beta}}, \end{aligned} \tag{3.1}$$

where $G_j \geq 0$, $\forall j \geq 8$. Thereafter, the coefficients of (3.1) namely, $C_j = \frac{G_j(j+k)!(\pm a)^j}{j!}$ is converted into P_{Lap} by using $C_j \mod 128$. Therefore, we can assign each integer of P_{Lap} to its corresponding 7 binary equivalents and these values are transformed into P_{XOR} by applying cumulative XOR protocol in Figure 3.1. Finally, we can correspond ASCII equivalent of the values in P_{XOR} to letters of encrypted text, E.

For decryption of the secret message, the encrypted text E is transformed into its corresponding ASCII values that is 7 binary equivalent, E_{ASCII}. Afterwards, cumulative XOR protocol is applied and outcome binary sequences are transform back into their corresponding decimal form DE_j, and by using $C_j = 128\cdot key_j + DE_j$, we obtain the coefficients of the Laplace transform again. If inverse Laplace transform is applied to the following

$$G\frac{k!s^{\alpha-\beta}}{(s^\alpha \mp a)^{k+1}} = \sum_{j=0}^{\infty}\frac{C_j}{s^{\alpha j+\alpha k+\beta}},\ C_j = \frac{G_j(j+k)!(\pm a)^j}{j!},$$

we get G_j which is sequence of integers corresponding to DNA nucleotide bases. Subsequently, G_j is transformed back into DNA bases by utilizing Table 3.2 which generate P_{dna}. Then, we can convert the DNA nucleotide bases back to their binary streams i.e. P_{bin} by utilizing Table 3.1. Lastly, the binary sequences in P_{bin} are assigned to their equivalent ASCII values and therefore generating the decrypted text or the plain text, P.

Theorem 3.1. *The plain text string in respect of $G_j, j = 0,1,2,...$ by using the Laplace transform of $Gt^{\alpha k+\beta-1}E^{(k)}_{\alpha,\beta}(\pm at^{\alpha})$ can be transformed to P_{Lap} that is consists of DE_j where*

$$DE_j = C_j - 128 \cdot key_j, \text{ for } j = 0,1,2,\cdots$$

and

$$C_j = \frac{G_j\,(j+k)!\,(\pm a)^j}{j!}, \text{ for } j = 0,1,2,\cdots$$

with private key

$$key_j = \frac{C_j - DE_j}{128}, \text{ for } j = 0,1,2,\cdots.$$

Theorem 3.2. *The encrypted text string in respect of $G_j, j = 0,1,2,...$, with private key $key_j, j = 0,1,2,...$ by using the inverse Laplace transform of*

$$G\frac{k!s^{\alpha-\beta}}{(s^{\alpha} \mp a)^{k+1}} = \sum_{j=0}^{\infty} \frac{G_j\,(j+k)!\,(\pm a)^j}{j!}\frac{1}{s^{\alpha j+\alpha k+\beta}}$$

can be transformed to P_{int} that is consists of G_j, where

$$G_j = \frac{DE_j + 128 \cdot key_j}{\frac{(j+k)!}{j!}(\pm a)^j}, \text{ for } j = 0,1,2,\cdots$$

and

$$C_j = 128 \cdot key_j + DE_j, \text{ for } j = 0,1,2,\cdots.$$

Theorem 3.3. *Encryption of the plain text and decryption of the secret message in terms of (3.1) is independent of the choice of α and β in Mittag-Leffler function.*

Proof. Let us consider the function as follow

$$\begin{aligned} Gt^{\alpha k+\beta-1}E^{(k)}_{\alpha,\beta}(\pm at^{\alpha}) &= \sum_{j=0}^{\infty} \frac{G_j\,(j+k)!}{j!}\frac{(\pm a)^j t^{\alpha j+\alpha k+\beta-1}}{\Gamma(\alpha j+\alpha k+\beta)} \\ &= \frac{G_0\,(0+k)!}{0!}\frac{(\pm a)^0 t^{\alpha\cdot 0+\alpha k+\beta-1}}{\Gamma(\alpha\cdot 0+\alpha k+\beta)} \\ &+ \frac{G_1\,(1+k)!}{1!}\frac{(\pm a)^1 t^{\alpha\cdot 1+\alpha k+\beta-1}}{\Gamma(\alpha\cdot 1+\alpha k+\beta)} \\ &+ \frac{G_2\,(2+k)!}{2!}\frac{(\pm a)^2 t^{\alpha\cdot 2+\alpha k+\beta-1}}{\Gamma(\alpha\cdot 2+\alpha k+\beta)} \\ &+ \cdots + \frac{G_n\,(n+k)!}{n!}\frac{(\pm a)^n t^{\alpha n+\alpha k+\beta-1}}{\Gamma(\alpha n+\alpha k+\beta)} + \cdots. \end{aligned} \tag{3.2}$$

By taking the Laplace transform of (3.2) and using $\Gamma(k+1) = k!$, we have

$$\begin{aligned} G\frac{k!s^{\alpha-\beta}}{(s^{\alpha} \mp a)^{k+1}} &= \frac{G_0\,(0+k)!}{0!}\frac{(\pm a)^0}{(\alpha\cdot 0+\alpha k+\beta-1)!}\frac{(\alpha\cdot 0+\alpha k+\beta-1)!}{s^{\alpha\cdot 0+\alpha k+\beta}} \\ &+ \frac{G_1\,(1+k)!}{1!}\frac{(\pm a)^1}{(\alpha\cdot 1+\alpha k+\beta-1)!}\frac{(\alpha\cdot 1+\alpha k+\beta-1)!}{s^{\alpha\cdot 1+\alpha k+\beta}} \\ &+ \frac{G_2\,(2+k)!}{2!}\frac{(\pm a)^2}{(\alpha\cdot 2+\alpha k+\beta-1)!}\frac{(\alpha\cdot 2+\alpha k+\beta-1)!}{s^{\alpha\cdot 2+\alpha k+\beta}} \\ &+ \cdots + \frac{G_n\,(n+k)!}{n!}\frac{(\pm a)^n}{(\alpha\cdot n+\alpha k+\beta-1)!}\frac{(\alpha\cdot n+\alpha k+\beta-1)!}{s^{\alpha\cdot n+\alpha k+\beta}} + \cdots \\ &= \frac{G_0\,(0+k)!\,(\pm a)^0}{0!}\frac{1}{s^{\alpha\cdot 0+\alpha k+\beta}} + \frac{G_1\,(1+k)!\,(\pm a)^1}{1!}\frac{1}{s^{\alpha\cdot 1+\alpha k+\beta}} \\ &+ \frac{G_2\,(2+k)!\,(\pm a)^2}{2!}\frac{1}{s^{\alpha\cdot 2+\alpha k+\beta}} + \cdots + \frac{G_n\,(n+k)!\,(\pm a)^n}{n!}\frac{1}{s^{\alpha\cdot n+\alpha k+\beta}} + \cdots. \end{aligned}$$

Therefore, the sequence corresponding to numerical values of the encrypted text is obtained as a

$$C_j = \frac{G_j\,(j+k)!\,(\pm a)^j}{j!}, \text{ for } j = 0,1,2,\cdots.$$

This proves the theorem. □

In the following, we provide an example for encryption of the plain text and decryption of the secret message in respect of DNA secret writing with the Laplace transform of the Mittag-Leffler function.

3.1. Encryption

We consider the two-letter word "Go" for demonstrating the algorithm. Here, 71 (01000111) and 111 (01101111) are ASCII values of "G" and "o", respectively. Equivalent binary stream characterization of the word is illustrated as follows.

0	1	0	0	0	1	1	1	0	1	1	0	1	1	1	1

Table 3.3: Binary characterization of ASCII coded "Go"

We code the binary bit utilizing DNA coding exemplified in Table 3.3. Therefore, DNA encoded text transform into CACTCGTT and we apply integer encoding to the script by utilizing Table 3.4. Hence, the equivalent integer encoded text is obtained as follow.

20	10	20	40	20	30	40	40

Table 3.4: Integer encoded script

We implement the Laplace transform taking the above integer codes as following. Let's consider the following expression

$$\begin{aligned} t^{\alpha k+\beta-1}E^{(k)}_{\alpha,\beta}\left(\pm at^{\alpha}\right) &= \sum_{j=0}^{\infty}\frac{(j+k)!}{j!}\frac{(\pm a)^j t^{\alpha j+\alpha k+\beta-1}}{\Gamma(\alpha j+\alpha k+\beta)} \\ &= \frac{(0+k)!}{0!}\frac{(\pm a)^0 t^{\alpha\cdot 0+\alpha k+\beta-1}}{\Gamma(\alpha\cdot 0+\alpha k+\beta)} \\ &+ \frac{(1+k)!}{1!}\frac{(\pm a)^1 t^{\alpha\cdot 1+\alpha k+\beta-1}}{\Gamma(\alpha\cdot 1+\alpha k+\beta)} \\ &+ \frac{(2+k)!}{2!}\frac{(\pm a)^2 t^{\alpha\cdot 2+\alpha k+\beta-1}}{\Gamma(\alpha\cdot 2+\alpha k+\beta)} \\ &+ \cdots+\frac{(n+k)!}{n!}\frac{(\pm a)^n t^{\alpha n+\alpha k+\beta-1}}{\Gamma(\alpha n+\alpha k+\beta)}+\cdots, \end{aligned} \tag{3.3}$$

where $\alpha,\beta>0$ and $a\in\mathbb{R}^+$.

Assume that

$$\begin{aligned} &G_0=20, \quad G_1=10, \quad G_2=20, \\ &G_3=40, \quad G_4=20, \quad G_5=30, \\ &G_6=40, \quad G_7=40, \quad G_n=0, \quad \text{for } n\geq 8. \end{aligned}$$

By choosing $\alpha=2$, $\beta=1$, $k=0$ and $a=2$ in (3.3), we can suppose that,

$$\begin{aligned} f(t)=GE_{2,1}\left(2t^2\right) &= \sum_{k=0}^{\infty}\frac{G_k\cdot\left(2t^2\right)^k}{\Gamma(2k+1)}=\sum_{k=0}^{\infty}\frac{G_k\cdot 2^k t^{2k}}{(2k)!} \\ &= \frac{G_0\cdot 2^0 t^0}{0!}+\frac{G_1\cdot 2^1 t^2}{2!}+\frac{G_2\cdot 2^2 t^4}{4!} \\ &+ \frac{G_3\cdot 2^3 t^3}{3!}+\frac{G_4\cdot 2^4 t^4}{4!}+\frac{G_5\cdot 2^5 t^5}{10!} \\ &+ \frac{G_6\cdot 2^6 t^6}{12!}+\frac{G_7\cdot 2^7 t^7}{14!}+\frac{G_8\cdot 2^8 t^8}{16!}+\cdots \\ &= 15+17\cdot\frac{2t^2}{2!}+14\cdot\frac{2^2t^4}{4!}+5\cdot\frac{2^3t^6}{6!} \\ &+ 4\cdot\frac{2^4t^8}{8!}+18\cdot\frac{2^5t^{10}}{10!}+18\cdot\frac{2^6t^{12}}{12!} \\ &+ 14\cdot\frac{2^7t^{14}}{14!}+17\cdot\frac{2^8t^{16}}{16!}. \end{aligned}$$

By applying Laplace transform to both sides, we get

$$\begin{aligned} L\{f(t)\} &= L\{GE_{2,1}\left(2t^2\right)\} \\ &= L\left\{20+10\cdot\frac{2t^2}{2!}+20\cdot\frac{2^2t^4}{4!}+\cdots+40\cdot\frac{2^7t^{14}}{14!}\right\} \\ &= \frac{20}{s}+\frac{20}{s^3}+\frac{80}{s^5}+\frac{320}{s^7}+\frac{320}{s^9}+\frac{960}{s^{11}}+\frac{2560}{s^{13}}+\frac{5120}{s^{15}}. \end{aligned}$$

By taking modulo 128 on 20, 20, 80, 320, 320, 960, 2560, 5120, we have sequence which consists of 20, 20, 80, 64, 64, 64, 0, 0.

We transform each of these integer into their equivalent ASCII values (7 bit binary) and then apply cumulative XOR operation repetitively until one bit is obtained. The technique is explicated taking binary stream corresponding to 20, i.e. 0010100 on Figure 3.1.

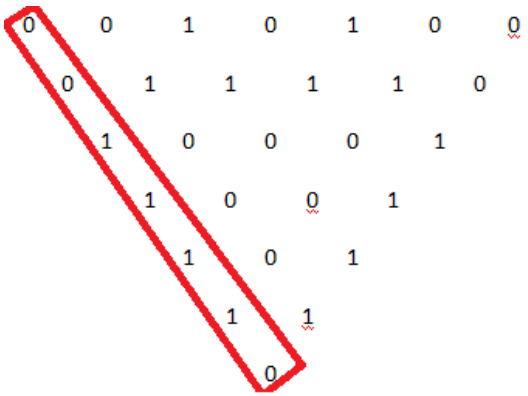

Figure 3.1: Cumulative XOR Method on Binary sequence corresponding to 20 and MSB collection

Hence, the binary version of the encoded cipher character equivalent to 20 is 0011110. By applying similar method to another values, we obtain the following table.

Laplace Coefficients	Binary Equivalents	Cumulative XOR	Cipher Text (ASCII)	Cipher Text (Character)
20	0010100	0011110	30	∧∧
20	0010100	0011110	30	∧∧
80	1010000	1100110	102	*f*
64	0101000	1111111	127	*LeD*
64	0101000	1111111	127	*LeD*
64	0101000	1111111	127	*LeD*
0	0000000	0000000	0	∧@
0	0000000	0000000	0	∧@

Table 3.5: Binary sequences corresponding to Laplace coefficients and their equivalent cipher characters

Therefore, secret message equivalent to ”Go” is ”∧∧∧∧fLeDLeDLeD∧@∧@”.

3.2. Decryption

The encrypted text characters in respect of the ASCII values correspond to 30, 30, 102, 127, 127, 127, 0 and 0. We transform these values into their 7 binary bit streams and apply cumulative XOR operation to them to generate binary sequences corresponding to Laplace coefficients. Therefore, we transform these coefficients back to their decimal values and perform the inverse Laplace transform as follows.

We take into consideration

$$\begin{aligned} G\frac{s}{s^2-2} &= \frac{15}{s}+\frac{34}{s^3}+\frac{56}{s^5}+\frac{40}{s^7}+\frac{64}{s^9}+\frac{576}{s^{11}}+\frac{1152}{s^{13}}+\frac{1792}{s^{15}}+\frac{4352}{s^{17}} \\ &= \sum_{j=0}^{\infty}\frac{q_j}{s^{2j+1}}. \end{aligned} \tag{3.4}$$

By applying inverse Laplace transform to both sides of (3.4), we obtain

$$\begin{aligned} GE_{2,1}\left(2t^2\right) &= 20\cdot 2^0+\frac{10\cdot 2^1\cdot t^2}{2!}+\frac{20\cdot 2^2\cdot t^4}{4!} \\ &+ \frac{40\cdot 2^3\cdot t^6}{6!}+\frac{20\cdot 2^4\cdot t^8}{8!}+\frac{30\cdot 2^5\cdot t^{10}}{10!} \\ &+ \frac{40\cdot 2^6\cdot t^{12}}{12!}+\frac{40\cdot 2^7\cdot t^{14}}{14!}. \end{aligned}$$

Consequently, we get $G_0=20$, $G_1=10$, $G_2=20$, $G_3=40$, $G_4=20$, $G_5=30$, $G_6=40$ and $G_7=40$.

Thus, integer values corresponding to DNA nucleotides are 20, 10, 20, 40, 20, 30, 40 and 40 which are transformed back to their equivalent bases by utilizing Table 3.5.

As a result, the DNA encoded text is coverted to $CACTCGTT$ and utilizing Table 3.1 the binary sequence of original message is transformed to 0100011101101111.

Finally, we decompose the binary sequence into corresponding ASCII equivalent which generates the ASCII values of "G" and "o" are 71 (01000111) and 111 (01101111), respectively. Therefore, the original message "Go" is obtained.

Example 3.1. *Assume the plain text be string 'PROFESSOR'. By applying our results in section 3, we have the following secret messages*

(1) $'\wedge\wedge\wedge J\wedge\wedge"\wedge\wedge\wedge T""'$ *for* $a = 1$, $k = 0$,
(2) $'\wedge\wedge\wedge\wedge <*R4\wedge QLeD'$ *for* $a = 1$, $k = 1$,
(3) $'\wedge\wedge\wedge T4;aF\,];'$ *for* $a = 3$, $k = 0$,
(4) $'Df\,3LeD\,]*LeD\wedge@'$ *for* $a = 3$, $k = 3$,
(5) $'U*U\wedge@LeDLeD\wedge@\wedge@'$ *for* $a = 5$, $k = 5$.

The results in Example 3.1 can be obtained by utilizing the codes below, which we write by the help of Python 3.8.5.

```
import string
import math
import re
string.ascii_lowercase

alph = list(string.ascii_lowercase)

ascii_char_dic = {
        "0":"^@", "1":"^A", "2":"^B", "3":"^C", "4":"^D",
        "5":"^E", "6":"^F", "7":"^G", "8":"^H", "9":"^I",
        "10":"^J", "11":"^K","12":"^L", "13":"^M", "14":"^N",
        "15":"^O", "16":"^P","17":"^Q", "18":"^R", "19":"^S",
        "20":"^T", "21":"^U", "22":"^V", "23":"^W", "24":"^X",
        "25":"^Y", "26":"^Z", "27":"^[", "28":"^'\'", "29":"^]",
        "30":"^^", "31":"^_", "32":" __", "33":"!", "34":" " ",
        "35":"#", "36":"$", "37":"%", "38":"&", "39": "'", "40":"(",
        "41":")", "42":"*", "43":"+", "44":",", "45":"-", "46":".",
        "47":"/", "48":"0", "49":"1", "50":"2", "51":"3", "52":"4",
        "53":"5", "54":"6", "55":"7", "56":"8", "57":"9", "58":":",
        "59":";", "60":"<", "61":"=", "62":">", "63":"?", "64":"@",
        "65":"A", "66":"B", "67":"C", "68":"D", "69":"E", "70":"F",
        "71":"G", "72":"H", "73":"I", "74":"J", "75":"K", "76":"L",
        "77":"M", "78":"N", "79":"O", "80":"P", "81":"Q", "82":"R",
        "83":"S", "84":"T", "85":"U", "86":"V", "87":"W", "88":"X",
        "89":"Y", "90":"Z", "91":"[", "92":"'\'", "93":"]", "94":"^",
        "95":"_", "96":'`', "97":"a", "98":"b", "99":"c", "100":"d",
        "101":"e", "102":"f", "103":"g", "104":"h", "105":"i", "106":"j",
        "107":"k", "108":"l", "109":"m", "110":"n", "111":"o", "112":"p",
        "113":"q", "114":"r", "115":"s", "116":"t", "117":"u", "118":"v",
        "119":"w", "120":"x", "121":"y", "122":"z", "123":"{", "124":"|",
        "125":"}", "126":"~", "127":"LeD"
}

def binary_operator(i):
    list_3 = []
    while i >= 1:
         list_3.append(i % 2)
         i = i // 2
    return list_3[::-1]

def xor(list_z):
    list_ex = []
    for i in range(len(list_z)):
        if i < len(list_z)-1:
            list_ex.append(list_z[i] ^ list_z[i+1])
    return list_ex

def cumulative_xor(list_z):
    list_xor = []
    while len(list_z) > 0:
         list_xor.append(list_z[0])
         list_z = xor(list_z)
    return list_xor
```

```
def cipher_numbers(list_y):
    list_sum = []
    for i in range(len(list_y)):
        list_sum.append((2**(len(list_y)-1-i))*list_y[i])
    return sum(list_sum)

def dna_privatekey_general(list_manipulated, list_mod):
     list_dna_privatekey = [int((i-j) / 128) for i, j in zip(list_manipulated, list_mod)]
     return list_dna_privatekey

def dna_encryption_general(a, n):
    dic_nucleoid_binary = {"00":"A", "01":"C", "10":"G", "11":"T"}
    dic_nucleoid_dec = {"A":10, "C":20, "G":30, "T":40}
    mes = input("Please enter an message for encryption: ")
    mes2 = list(mes)
    ascii_list = [ord(i) for i in mes2]
    list_binary = []
    for i in ascii_list:
        x = [str(j) for j in [0] + binary_operator(i)]
        y = "".join(x)
        list_binary.append(y)
   list_bin = []
   for i in list_binary:
      for j in i:
          list_bin.append(j)
   list_bin_2 = []
   for i in range(0, len(list_bin), 2):
        j = list_bin[i] + list_bin[i+1]
        list_bin_2.append(j)
   list_nucleoid = []
   for i in list_bin_2:
        list_nucleoid.append(dic_nucleoid_binary[i])
   list_coff = []
for i in list_nucleoid:
    list_coff.append(dic_nucleoid_dec[i])
list_manipulated =  [int(list_coff[i]*(math.factorial(i+n)/(math.factorial(i))*(a**i)))
                     for i in range(len(list_coff))]
list_mod = [i % 128 for i in list_manipulated]
 list_bin_3 = []
for i in list_mod:
    list_bin_3.append(binary_operator(i))
list_bin_4 = []
for j in list_bin_3:
   if len(j) < 7:
      if j != []:
           list_bin_4.append((7-len(j))*[0] + j)
       else:
           list_bin_4.append(7*[0] + j)
   else:
       list_bin_4.append(j[0:7])

list_cum_xor = []

for i in list_bin_4:
    list_cum_xor.append(cumulative_xor(i))

list_cipher_numbers = []
for i in list_cum_xor:
    list_cipher_numbers.append(cipher_numbers(i))
list_encrypted = []
for i in list_cipher_numbers:
    list_encrypted.append(ascii_char_dic[str(i)])
list_binary_full = [i for i in list_cum_xor]
list_binary_full_2 = [j for i in list_cum_xor for j in i]
list_binary_plain = [j for i in list_bin_4 for j in i]
```

```
print("*"*114)
print("Your encrypted message is: ", "".join(list_encrypted))
print("*"*114)
print("Your encrypted list is:", list_encrypted)
print("*"*114)
print("Your private key is:", dna_privatekey_general(list_manipulated, list_mod))
print("*"*114)
#print(list_binary)
#print(list_bin)
#print(list_cum_xor)
#print(list_coff)
print("DNA bases:",list_nucleoid)
print("*"*114)
#print(list_binary_full)
#print(list_binary_plain)
print("Results of monobit test:")
print(monobit_test(list_binary_full))
print("*"*114)
print("phi test:", phi_test(list_binary_plain, list_binary_full_2))



def dna_decryption_general(a, n):
    dic_nucleoid_binary = {"00":"A", "01":"C", "10":"G", "11":"T"}
    dic_nucleoid_dec = {"A":10, "C":20, "G":30, "T":40}
    dmes = input("Please enter an secret message for decryption: ")
    dna_privatekey = input("Please enter private key: ").split(", ")
    dna_list_privatekey = [int(i) for i in dna_privatekey]
    #print("dna_list_privatekey",dna_list_privatekey)
    #dmes_list = dmes.split(", ")
    string_list = list(re.findall(r"'(.*?)'", ''.join(dmes)))
    #print(string_list)
    #print("string_list", string_list)
    list_keys = []
    for i in string_list:
        for k, l in ascii_char_dic.items():
            if i == l:
                list_keys.append(k)
    #print("list_keys",list_keys)
    list_keys_num = [int(i) for i in list_keys]
    #print("list_keys_num",list_keys_num)
    list_binary_x = [binary_operator(i) for i in list_keys_num]
    #print("list_binary_x",list_binary_x)
    list_binary_xor = []
    for j in list_binary_x:
       if len(j) < 7:
          if j != []:
              list_binary_xor.append((7-len(j))*[0] + j)
          else:
              list_binary_xor.append(7*[0] + j)
      else:
          list_binary_xor.append(j[0:7])
    #print("list_binary_xor",list_binary_xor)
    list_binary_y = [cumulative_xor(i) for i in list_binary_xor]
    #print("list_binary_y",list_binary_y)
    list_ciphers = [cipher_numbers(i) for i in list_binary_y]
    #print("list_ciphers",list_ciphers)
    list_dec = [i + 128*j for i, j in zip(list_ciphers,  dna_list_privatekey)]
    #print("list_dec",list_dec)
    list_manipulated = [int(list_dec[i] / (math.factorial(i+n)/(math.factorial(i))*(a**i)))
                        for i in range(len(list_dec))]
    #print("list_manipulated",list_manipulated)
    list_dna_nucleoid = []
    for i in list_manipulated:
        for k, l in dic_nucleoid_dec.items():
           if i == l:
```

```
            list_dna_nucleoid.append(k)
#print("list_dna_nucleoid",list_dna_nucleoid)
list_dna_binary = []
for i in list_dna_nucleoid:
    for k, l in dic_nucleoid_binary.items():
        if i == l:
            list_dna_binary.append(k)
#print("list_dna_binary",list_dna_binary)
list_final_bef = []
for i in range(0, len(list_dna_binary), 4):
     list_final_bef.append(list_dna_binary[i:i+4])
#print("list_final_bef",list_final_bef)
list_final = []
for j in list_final_bef:
    list_final.append("".join(j))
#print("list_final",list_final)
list_sum = []
for j in list_final:
    list_num = [int(i) for i in j]
    list_sum.append(cipher_numbers(list_num))
#print("list_sum",list_sum)
list_final_words = []
for i in list_sum:
     for k, l in ascii_char_dic.items():
         if str(i) == k:
            list_final_words.append(l)
print("DNA bases:", list_dna_nucleoid)
print("The original message: " + "".join(list_final_words))
```

Before using phi coefficient test and monobit test for the results above, we want to provide some information on these test.

Numerous studies have been done to examine the reliability of a cryptographic algorithm. Inhomogeneity, frequency distribution, and bit rate are usually utilized reliability techniques. The technique to be performed in this article is the monobit test. The monobit test is utilized to determine whether the frequency of 0's and 1's in bit sequences in the encrypted text. Let and represent the number of 0's and 1's in bit sequences respectively. The calculated value obtained with is compared with the critical of value at 1 degree of freedom. If the calculated values of value is smaller than the critical of value, it means that the bit sequences passed the monobit test. The formula of the Monobit test is denoted by

$$\chi^2 = \frac{(n_0 - n_1)^2}{n},$$

where n_0, n_1 and n represent the number of zeroes, ones, and both of them, respectively [39].

Correlation analysis used in statistics investigates whether there is a relationship between two or more variables. The phi coefficient will be utilized to find the correlation value. The phi coefficient is the coefficient of the relationship between two variables with a binary data structure. The goal is to get a completely different or low relationship between plain text and cipher text. If the obtained phi coefficient approaches 1, there is a strong relationship between them. If the phi coefficient approaches 0, there is a very weak relationship between them [40–42].

By using the table as follows,

	$y=1$	$y=0$	$total$
$x=1$	n_{11}	n_{10}	$n_{1\cdot}$
$x=0$	n_{01}	n_{00}	$n_{0\cdot}$
$total$	$n_{\cdot 1}$	$n_{\cdot 0}$	n

Table 3.6: 2×2 table for two random variables x and y

we get the following phi coefficient formula

$$\phi = \frac{n_{11}n_{00} - n_{10}n_{01}}{\sqrt{n_{\cdot 1}n_{\cdot 0}n_{1\cdot}n_{0\cdot}}},$$

where n_{11}, n_{10}, n_{01} and n_{00} denote the observation frequencies.

By utilizing the monobit test and correlation analysis, we have the following results for cipher texts in Example 3.1.

Cipher Text	Calculated value		Calculated value	
	Phi Coefficient	Monobit Test	Phi Coefficient	Monobit Test
∧∧∧J∧∧”∧∧∧T””	0.464150	2.571428	0.5	3.8415
∧∧∧∧<*R4∧QLeD	0.410720	0.285714	0.5	3.8415
∧∧∧T4;aF];	0.292307	0.285714	0.5	3.8415
Df 3LeD]*LeD∧@	0.083333	1.142857	0.5	3.8415
U∗U∧@LeDLeD∧@∧@	0.203653	0.642857	0.5	3.8415

Table 3.7: Results for Monobit Test and Correlation Analysis

The values of Table 3.7 can be attained by utilizing following codes, which we write by means of Python 3.8.5.

```
def phi_test(list_1, list_2):
    list_3 = [i for i in zip(list_1, list_2)]
    list_oo = [i for i in list_3 if i == (1,1)]
    list_oz = [i for i in list_3 if i == (1,0)]
    list_zz = [i for i in list_3 if i == (0,0)]
    list_zo = [i for i in list_3 if i == (0,1)]
    list_po = [i for i in list_3 if i[1] == 1]
    list_pz = [i for i in list_3 if i[1] == 0]
    list_op = [i for i in list_3 if i[0] == 1]
    list_zp = [i for i in list_3 if i[0] == 0]
    a = (len(list_oo)*len(list_zz)) - (len(list_oz)*len(list_zo))
    b = math.sqrt(len(list_op)*len(list_zp)*len(list_po)*len(list_pz))
    phi = a / b
    return phi

def monobit_test(list_ex):
    list_z = []
    list_zeros = []
    list_ones = []
    for i in list_ex:
        for j in i:
            list_z.append(j)
    for k in list_z:
        if k == 0:
            list_zeros.append(k)
    elif k == 1:
        list_ones.append(k)
    chi = (len(list_zeros) - len(list_ones))**2 / (len(list_zeros) + len(list_ones))
    print("Zeros:",len(list_zeros))
    print("Ones:",len(list_ones))
    print(chi)
```

The computed values of χ^2 and ϕ are less in relation to critical values of them. This implies that these binary streams corresponding to cipher texts pass the Monobit test and correlation analysis in respect of phi coefficient. According to the correlation analysis, it is seen that there is a weak relationship between the first text and the last text. In addition, the values obtained in the Monobit test show the power of the function and transformation used.

4. Conclusions

In this study, a text that was handled by using DNA codes effectively was encrypted using the Mittag-Leffler function and Laplace transform, and a new text with different characters was obtained. Here, the strength of the password created by applying the XOR operation in the method has been increased. The difference of this study is the simultaneous use of the Mittag-Leffler function and the Laplace transform. At the end of the study, the reliability of the encryption technique was checked with the Monobit test. The results obtained are quite good. In addition, with the correlation test, it was examined whether there was a relationship between the first and the last text. As a result of this examination, it was seen that there was an acceptable, weak relationship..

Article Information

Acknowledgements: The authors would like to thank the referees for their contribution to the development of the article.

Author's contributions: The authors have no conflicts of interest to declare. All co-authors have seen and agree with the contents of the manuscript and there is no financial interest to report.

Conflict of interest disclosure: No potential conflict of interest was declared by the author.

Supporting/Supporting organizations: No grants were received from any public, private or non-profit organizations for this research.

Ethical approval and participant consent: It is declared that during the preparation process of this study, scientific and ethical principles were followed and all the studies benefited from are stated in the bibliography.

Plagiarism statement: This article was scanned by the plagiarism program. No plagiarism detected.

Availability of data and materials: Not applicable.

References

[1] G. N. Lakshmi, B.R. Kumar, A.C. Sekhar, *A cryptographic scheme of Laplace transforms*, Int. J. Math. Arch., **2** (12) (2011), 2515-2519.
[2] A. P. Stakhov, *The golden section in the measurement theory*, Comput. Math. Appl., **17** (4-6) (1989), 613–638.
[3] A. P. Stakhov, *The "golden" matrices and a new kind of cryptography*, Chaos, Solit. Fractals, **32** (3) (2007), 1138–1146.
[4] T. H. Barr, *Invitation to Cryptology*, Pearson, Prentice Hall, 2002.
[5] J. A. Buchmann, *Introduction to Cryptography*, New York, Springer, 2009.
[6] E. Cole, R. Krutz, J.W. Conley, *Network Security Bible*, Indianapolis, Wiley Publishing, 2009.
[7] W. Stallings, *Network Security Essentials: Applications and Standards*, Boston, Prentice Hall, 2001.
[8] W. Stallings, *Cryptography and Network Security*, London, Pearson Education Ltd, 2005.
[9] A. Stanoyevitch, *Introduction to Cryptography with Mathematical Foundations and Computer Implementations*, Boca Raton, CRC Press, 2010.
[10] Z. M. Z Muhammad, F. Özkaynak, *Security problems of chaotic image encryption algorithms based on cryptanalysis driven design technique*, IEEE Access, **7** (2019), 99945-99953.
[11] F. Özkaynak, A. B. Özer, *Cryptanalysis of a new image encryption algorithm based on chaos*, Optik, **127** (13) (2016), 5190-5192.
[12] G. A. Dhanorkar, A. P. Hiwarekar, *A generalized Hill cipher using matrix transformation*, Int. J. Math. Sci. Eng. Appl., **5** (4) (2011), 19-23.
[13] J. Overbey, W. Traves, J. Wojdylo, *On the keyspace of the Hill cipher*, Cryptologia, **29** (1) (2005), 59-72.
[14] S. Saeednia, *How to make the Hill cipher secure*, Cryptologia, **24** (4) (2000), 353-360.
[15] A. P. Hiwarekar, *A new method of cryptography using Laplace transform*, Int. J. Math. Arch., **3** (3) (2012), 1193-1197.
[16] A. P. Hiwarekar, *A new method of cryptography using Laplace transform of hyperbolic functions*, Int. J. Math. Arch., **4** (2) (2013), 208-213.
[17] A. P. Hiwarekar, *Application of Laplace transform for cryptographic scheme*, Proc. World Congr. Eng., **1** (2013), 1-6.
[18] J. D. Watson, F.H.C. Crick, *Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid*, Am. J. Psychiatry, **160** (4) (2003), 623-624.
[19] G. Cui, L. Qin, Y. Wang, X. Zhang, *Information security technology based on DNA computing*, ASID, (2007) 288-291.
[20] G. Z. Cui, Y. Liu, X. Zhang, *New direction of data storage: DNA molecular storage technology*, Comput. Eng. Appl., **42** (26) (2006), 29-32.
[21] X. Wang, Q. Zhang, *DNA computing-based cryptography*, Fourth Int. Conf. Bio-Inspired Comput., (2009), 1-3.
[22] L. M. Adleman, *Molecular computation of solutions to combinatorial problems*, Science, **266** (5187) (1994) 1021-1024.
[23] C. T. Clelland, V. Risca, C. Bancroft, *Hiding messages in DNA microdots*, Nature, **399** (6736) (1999), 533-534.
[24] S. Som, M. Som, *DNA secret writing with Laplace transform*, Int. J. Comput. Appl., **975** (8887) (2012), 43-50.
[25] R. J. Lipton, *Using DNA to solve NP-complete problems*. Science, **268** (4) (1995), 542-545.
[26] S. T. Amin, M. Saeb M, S. El-Gindi, *A DNA-based implementation of YAEA encryption algorithm*, Comput. intel., (2006), 120-125.
[27] D. G. Zill, *Advanced Engineering Mathematics, Burlington*, Jones & Bartlett Learning, 2020.
[28] R. P. Boas Jr, *Higher transcendental functions*, Science, **122** (3163) (1955), 290-290.
[29] I. Podlubny, *Fractional Differential Equations*, 198, San Diego, California, USA, Academic Press, 1999.
[30] G. M. Mittag-Leffler, *Sur la nouvelle fonction* $E_\alpha(x)$, C. R. Acad. Sci. Paris, **137** (2) (1903), 554-558.
[31] G.M. Mittag-Leffler, *Sopra la funzione* $E_\alpha(x)$, Rend. Acad. Dei Lincei, **13** (5) (1904), 3-5.
[32] G. M. Mittag-Leffler, *Sur la represention analytique d'une branche uniforme d'une fonction monog´ene*, Acta Math., **29** (1) (1905), 101-181.
[33] A. Wiman, *Über den fundamentalsatz in der teorie der funktionen* $E_a(x)$, Acta Math., **29** (1) (1905), 191-201.
[34] A. Wiman, *Über die nulstellen der funktionen* $E_a(x)$, Acta Math., **29** (1) (1905), 217-234.
[35] R. P. Agarwal, *A propos d'une note de M Pierre Humbert*, C. R. Acad. Sci., **236** (21) (1953), 2031-2032.
[36] P. Humbert, R.P. Agarwal, *Sur la fonction de Mittag-Leffler et quelques-unes de ses g´en´eralisitions*, Bull. des Sci. Math., **77** (2) (1953), 180-185.
[37] M. M. Dzhrbashyan, *Integral Transforms and Representations of Functions in the Complex Domain*, Moscow, Nauka (in Russian), 1966.
[38] R. Garrappa, M. Popolizio, *Computing the matrix Mittag-Leffler function with applications to fractional calculus*, J. Sci. Comput., **77** (1) (2018), 129-153.
[39] A. Ruk, *A statistical test suite for the validation of random number generators and pseudo-random number generators for cryptographic applications*, NIST, 2001.
[40] H. Cramer, *Mathematical Methods of Statistics*, Princeton, Princeton Univ Press, 1946.
[41] G. Nagalakshmi, A.C. Sekhar, N.R. Sankar, K. Venkateswarlu, *Enhancing the data security by using RSA algorithm with application of Laplace transform cryptosystem*, Int. J. Recent Technol. Eng., **8** (2) (2019), 6142-6147.
[42] B. W. Matthews, *Comparison of the predicted and observed secondary structure of T4 phage lysozyme*, Biochim. Biophys. Acta (BBA)-Protein Struct., **405** (2) (1975), 442-451.