mathematics

MDPI

Article

Application of Decimated Mathematical Equations and Polynomial Root-Finding Method in Protection of Text Messages

Borislav Stoyanov *, Gyurhan Nedzhibov, Dimitar Dobrev and Tsvetelina Ivanova

Department of Computer Informatics, Faculty of Mathematics and Informatics, Konstantin Preslavsky University of Shumen, 9700 Shumen, Bulgaria; g.nedzhibov@shu.bg (G.N.); d.d.dobrev@shu.bg (D.D.); ts.r.ivanova@shu.bg (T.I.)

* Correspondence: borislav.stoyanov@shu.bg

Abstract: Cryptography is the process of transforming data so that only the recipient of the message can read it. It uses an algorithm and a key to convert an input into an encrypted output. In this study, we introduce a novel method for protecting readable messages through the utilization of a polynomial root-finding technique in conjunction with the decimated output of Zaslavsky equations. The innovative approach we have developed is a block encryption algorithm that offers both protection for sensitive information as well as adaptability to various block sizes, including dynamically changing block sizes. Precise security analysis is provided for the proposed algorithm using key space analysis and strong statistical tests. The presented results show that the novel block encryption protection provides a high level of security.

Keywords: text encryption; polynomial root-finding methods; mathematical algebraic equations

MSC: 65H04; 68P25

Citation: Stoyanov, B.; Nedzhibov, G.; Dobrev, D.; Ivanova, T. Application of Decimated Mathematical Equations and Polynomial Root-Finding Method in Protection of Text Messages. *Mathematics* **2023**, *11*, 4982. https://doi.org/10.3390/math11244982

Academic Editor: Raúl M. Falcón

Received: 15 November 2023
Revised: 14 December 2023
Accepted: 15 December 2023
Published: 17 December 2023

1. Introduction

The modern communication revolution necessitates faster and more secure methods of ensuring plain message security. It is crucial to have better options in order to be more secure in today's world. Cryptography is the practice of transforming data into an unrecognizable form to protect them from being illegally obtained, altered, modified, or accessed while being transmitted over a network. In recent years, many encryption techniques have been introduced to securely store and transfer data in text format. Message encryption is a deterministic scheme operating on fixed-length encryption groups of bits. These include extensively used classical encryption schemes (RSA, Blowfish, AES, etc.) and novel encryption schemes (based on dynamical systems [1–3] and quantum walks [4–6]).

Due to the increasing resistance to statistical attacks, the use of strong mathematical tools in encryption schemes is becoming more and more popular. A symmetric text encryption technique based on reversible Karhunen–Loeve expansion is proposed in [7]. The proposed algorithm is sufficiently secure based on the security measures. In Ref. [8], a data encryption scheme based on deep learning is presented. It performs well in secrecy capacity. In [9], a data encryption technique based on a numerical method and rotation–translation formula is proposed. Two different ways of constructing nonlinear functions are presented. Text encryption based on logistic, pinchers, and sine-circle equations is described in [10]. Pseudorandom bit extraction from Lorenz and Rössler attractors is presented in [11]. Different numerical methods for solving nonlinear dynamic systems are available. In Ref. [12], a variant of the RSA scheme is used to secure simple input messages. Core-adaptive Fourier decomposition for high encryption performance and security is described in [13].

Decimation is the process of reducing the random or pseudorandom output stream with a filtering function. Irregularly based encryption schemes can be found in [14,15].

1.1. Motivation and Related Works

Our study was inspired by the rising demand for more complex and reliable encryption as a result of the advancement in computer and network technology. The main motivation of this paper is to design and propose an encryption method suitable for readable massages protection. In [9], we introduced an enhanced encryption algorithm. Our research focuses on advancing encryption techniques, emphasizing both efficiency and security in data protection. We developed an algorithm using numerical methods and the rotation–translation equation to create a symmetric-key-based encryption–decryption system. Our goal was to improve the existing encryption methods by incorporating a faster convergent iterative approach, ensuring secure convergence of the numerical scheme, and enhancing security through the use of a rotation–translation formula.

As far as the authors are aware, the studies that examine encryption algorithms with iterative techniques include [9,11,16–19]. Publications [9,16–18] discuss encryption techniques that use numerical methods to solve nonlinear algebraic equations. Meanwhile, in [11], an encryption technique is considered where numerical methods are used to solve differential equations. In [19], the use of the Zaslavsky function to construct a hash function is demonstrated. For some more recent results based on the chaotic-map Zaslavsky equation, we recommend [20–22] and the citations therein.

The present work is a kind of continuation and expansion of the ideas and approach from publication [9]. Our goal is to present a new block encryption technique based on iterative methods for simultaneous approximation of polynomial zeros in conjunction with the decimated output of Zaslavsky equations.

1.2. Contributions

This research paper introduces a novel approach to text encryption, leveraging the fusion of decimated mathematical equations and the polynomial root-finding method. The proposed algorithm offers a robust and secure method for achieving block encryption, enhancing the confidentiality and integrity of textual information.

The main contributions of our research can be summarized as follows:

- We present a decimation-based algorithm for pseudorandom byte output using two Zaslavsky equations, which possesses respectable statistical features.
- We propose secure text encryption based on a mix of the polynomial root-finding method and the decimated output of the novel pseudorandom generator.
- We examine the proposed encryption, and the data show that it has good key space and excellent security properties that can withstand most common theoretical and statistical vulnerabilities.

1.3. Article Structure

The rest of the paper is organized as follows. In Section 2, we propose basics on polynomials, factoring polynomials, and root-finding methods. In Section 3, we propose a novel decimation-based pseudorandom byte generation and text encryption technique. In Section 4, some security cryptanalysis is provided. Finally, Section 5 concludes the paper.

The MATLAB source code and output results are available from [23].

2. Polynomials, Factoring Polynomials, and Root-Finding Methods

This section provides several fundamental facts about polynomials, which will be used in the following sections. Let $n \in \mathbf{Z} \cup \{0\}$ be a nonnegative integer, and let $a_0, a_1, \ldots, a_n \in \mathbf{C}$ be complex numbers. Then, we call the expression

$$p(x) = a_n x^n + \ldots + a_1 x + a_0 \tag{1}$$

as a polynomial in the variable x with coefficients $a_0, a_1, \ldots, a_n$. The highest power, n, where $a_n \neq 0$, occurring is called the degree of the polynomial. The highest degree term is called the leading term and its coefficient a_n is called the leading coefficient. Moreover, if

the leading coefficient is $a_n = 1$, we call $p(x)$ a *monic polynomial*. Finally, by a *root* (or *zero*) of a polynomial $p(x)$, we mean a complex number α such that $p(\alpha) = 0$.

2.1. Fundamental Theorem of Algebra

The Fundamental Theorem of Algebra has several equivalent formulations and many different proofs. One of the first proofs was provided by the famous mathematician Carl Gauss in his 1799 doctoral dissertation.

Theorem 1 (*Fundamental Theorem of Algebra*)**.** *Given any positive integer $n = 1$ and any choice of complex numbers $a_0, a_1, ..., a_n$, such that $a_n \neq 0$, the polynomial equation*

$$a_n x^n + \ldots + a_1 x + a_0 = 0 \tag{2}$$

has at least one solution $x \in C$.

There are several equivalent formulations of Theorem 1. The theorem is also stated as follows.

Theorem 2. *Let $p(x)$ be a polynomial with complex coefficients of degree n. Then, $p(x)$ has n roots.*

Theorem 3 (*Complex Factorization Theorem*)**.** *Suppose p is a polynomial function with complex number coefficients. If the degree of p is $n \geq 1$, then p has exactly n complex roots, counting their multiplicity. If $\alpha_1, \alpha_2, \ldots, \alpha_k$ are the distinct roots of p, with multiplicities $m_1, m_2, \ldots, m_k$, respectively, then*

$$p(x) = a(x - \alpha_1)^{m_1}(x - \alpha_2)^{m_2} \ldots (x - \alpha_k)^{m_k}, \tag{3}$$

where a is the leading coefficient of $p(x)$.

It follows from Theorem 3 that a polynomial is completely defined by its zeros, their multiplicities, and its leading coefficient.

2.2. Polynomial Root-Finding

Generally, polynomials' zeros cannot be computed exactly nor expressed in closed form. Note that there exist closed formulas for the roots of polynomials, only for polynomials up to degree four. This fact was proved by Abel in 1824, and independently by Galois. As a result, most computational methods for solving the root-finding problem are iterative. Essentially, iterative methods work as follows: they start with an initial approximation x_0 and then construct a sequence of iterations x_k using an iteration formula in the hope that this sequence converges to a root of $p(x)$. There are two main types of iteration methods for polynomial root-finding: those that approximate one root at a time and those that approximate all roots at once. A classic representative of methods for simultaneous approximation of all zeros of $p(x)$ is the famous Weierstrass method, which is defined by the sequence

$$\mathbf{x}_{k+1} = \mathbf{x}_k - W(\mathbf{x}_k) \quad \text{for} \quad k = 0, 1, 2, \ldots, \tag{4}$$

where $W : D \subset \mathbf{C}^n \to \mathbf{C}^n$ is a vector-valued function with components

$$W_i(\mathbf{x}) = \frac{p(x_i)}{\prod_{j \neq i}^{n} (x_i - x_j)}, \quad \text{for } i = 1, \ldots, n, \tag{5}$$

and D is the set of all vectors with distinct components in $\mathbf{C}^n$. Weierstrass sequence (4) is well-defined under appropriate initial conditions and converges to the polynomial p's zero vector. Provided that all p zeros are simple, the iteration has quadratic convergence.

Matrix methods can also be used to approximate polynomial roots. Over the past few decades, they have gained popularity for simultaneously finding polynomial roots. With this approach, the zeros of the input polynomial p are approximated as the eigenvalues

of the companion matrix (or generalized companion matrix). It is known that a matrix $A \in \mathbf{C}^{n\times n}$ is a companion matrix associated with the polynomial $p(x)$ if

$$\det(A - \mathbf{x}I) = p(x) \tag{6}$$

for $\mathbf{x} \in \mathbf{C}^n$. In mathematical terms, p denotes the characteristic polynomial of A. Thus, eigenvalues of A coincide with the zeros $\alpha_1, \ldots, \alpha_n$ of p . Therefore, the polynomial root-finding problem can be reduced to the eigenvalue problem: find a scalar $\lambda \in \mathbf{C}$ and a nonzero vector $\mathbf{x} \in \mathbf{C}^n$ such that

$$A\mathbf{x} = \lambda\mathbf{x}. \tag{7}$$

The most common example of companion matrices is the well-known Frobenius companion matrix of order n

$$A = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ -a_0 & -a_1 & -a_2 & \ldots & -a_{n-1} \end{pmatrix} \tag{8}$$

associated with the polynomial p.

This implies the use of different techniques to find matrix eigenvalues. The eigenvalues of the companion matrix may be found by using standard versions of the balancing QR algorithm. These versions may be found in LAPACK or its precursor EISPACK. This is how the MATLAB roots function performs its computation.

2.3. MATLAB Functions for Root-Finding

MATLAB's built-in functions poly, polyval, and roots can be used in the context of zero-finding of a polynomial. Function poly converts roots to a polynomial equation. The following syntax poly(v), when v is a vector, provides a vector whose elements are the coefficients of the polynomial whose roots are the elements of v.

Function roots finds the polynomial roots. The following syntax roots(c) computes the roots of the polynomial whose coefficients are elements of the vector c.

Function polyval is for polynomial evaluation. The following syntax polyval(p, x) evaluates the polynomial p at each point in x. The argument p is a vector of length $n + 1$ whose elements are the coefficients (in descending powers) of an n-th degree polynomial.

MATLAB's built-in functions poly, polyval, and roots are employed for the purpose of locating the roots of a polynomial. The poly function transforms roots into a polynomial equation. When given a vector v, the syntax poly(v) generates a vector containing coefficients corresponding to the polynomial with roots specified by the elements of v. The roots function determines the roots of a polynomial. Using the syntax roots(c), where c is a vector, computes the roots of the polynomial defined by the coefficients in vector c. The polyval function is utilized for evaluating polynomials. Using the syntax polyval(p, x), it calculates the value of the polynomial p at each specified point in x. Here, the argument p is a vector of length $n + 1$, containing coefficients (in descending powers) of an n-th degree polynomial.

3. Secure Text Encryption Using Polynomial Root-Finding Method and Decimated Output of Zaslavsky Equations

3.1. Proposed Decimation-Based Pseudorandom Byte Generation

In this section, we will describe the Zaslavsky-equation-based [24] pseudorandom byte output generation.

The following function provides a definition of the Zaslavsky formula:

$$y_{n+1} = [y_n + \nu(1 + \mu z_n) + \epsilon\nu\mu\cos(2\pi y_n)](\mathrm{mod}\,1), \tag{9}$$

$$z_{n+1} = e^{-r}(z_n + \epsilon\cos(2\pi y_n)), \tag{10}$$

where

$$\mu = \frac{1 - e^{-r}}{r}. \tag{11}$$

It depends on three constants ν, ϵ, and r, where $r = 3.0$, $\nu = 400/3$, and $\epsilon = 0.3$ as in [24].

George M. Zaslavsky introduced the novel strange attractor, a discrete-time nonlinear dynamical system, in 1978. The Zaslavsky equation demonstrates deterministic dynamic behavior, a crucial component of modern security algorithms [25–30]. The diagram of the Zaslavsky equation has been provided in Figure 1. This diagram shows the random behavior of the Zaslavsky map. One can choose starting values in the interval $[-0.001, 0.001]$.

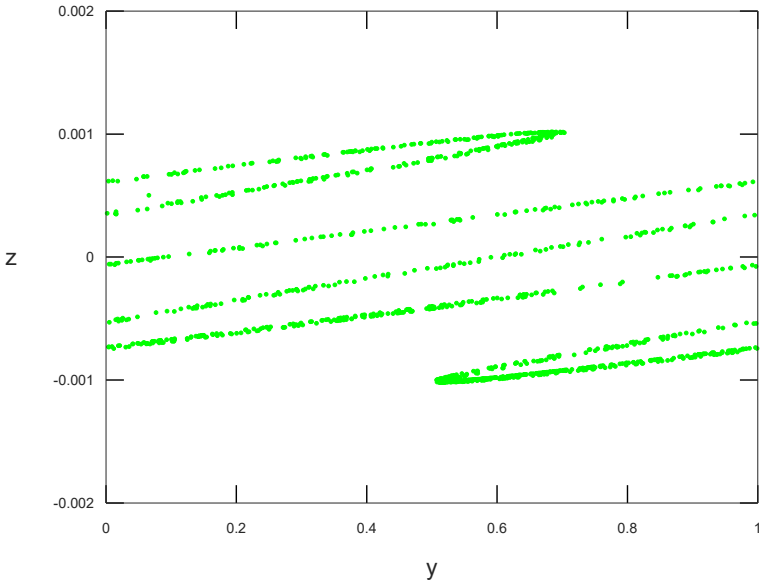

Figure 1. Plot of Zaslavsky equation.

The proposed scheme is based on the following two Zaslavsky equations:

$$\begin{aligned} y_{1,n+1} &= \bmod(y_{1,n} + \nu(1 + \mu z_{1,n}) + \epsilon\nu\mu\cos(2\pi y_{1,n}), 1) \\ z_{1,n+1} &= e^{-r}(z_{1,n} + \epsilon\cos(2\pi y_{1,n})), \\ y_{2,n+1} &= \bmod(y_{2,n} + \nu(1 + \mu z_{2,n}) + \epsilon\nu\mu\cos(2\pi y_{2,n}), 1) \\ z_{2,n+1} &= e^{-r}(z_{2,n} + \epsilon\cos(2\pi y_{2,n})), \end{aligned} \tag{12}$$

where μ is from Equation (11) and $y_{1,0}$, $z_{1,0}$, $y_{2,0}$, *and* $z_{2,0}$, are floating point numbers.

The complete decimation process consists of the following steps:

1. The starting values $y_{1,0}$, $z_{1,0}$, $y_{2,0}$, and $z_{2,0}$ of Equation (12) and an output stream length Z are determined.
2. The Zaslavsky equations are iterated initially for M times.
3. The iteration of the Equation (12) continues, and, as a result, four decimal fractions $y_{1,m}$, $z_{1,m}$, $y_{2,m}$, and $z_{2,m}$, are generated and two of them post-processed as follows:

$$\begin{aligned} s_i &= mod(abs(int(y_{1,m} \times 10^{14})), 2) \\ t_j &= mod(abs(int(z_{2,m} \times 10^{14})), 2), \end{aligned} \tag{13}$$

   where $int(x)$ returns the integer part of x, truncating the value at the decimal point, $abs(x)$ returns the absolute value of x, and $mod(x, y)$ returns the reminder after division. Two bits s_i and t_j, are obtained.
4. If the bit $(s_i \oplus t_j)! = 1$, return to Step 3.
5. Post-process the other two values as

$$\begin{aligned} s_j &= mod(abs(int(z_{1,m} \times 10^{14})), 256) \\ t_i &= mod(abs(int(y_{2,m} \times 10^{14})), 256), \end{aligned} \tag{14}$$

and the byte $(s_j \oplus t_i)$ produces part of the output sequence.

6. Return to Step 3 until the byte stream length Z is reached.

The novel decimation-based generator is implemented in MATLAB (R2023b, MathWorks, Natick, MA, USA).

3.2. Proposed Text Encryption Technique

In this section, we describe the encryption and decryption processes of our approach. The complete encryption process consists of the following steps (Algorithm 1):

Algorithm 1: Encryption

Input: Original text $inpT$, Block length Len and K PR bytes.
Output: Encrypted text $encT$
1: **Procedure** ENCR($inpT$, Len, K)
2: n = length($inpT$); *(Extract the string length)*
3: $numBlocks$ = ceil(n/Len); *(Compute the number of blocks)*
4: **For** $k = 1 : numBlocks$ *(Count the blocks)*
5: $currentBlock = inpT(k)$; *(Extract the current block)*
6: K = GenPRG(Len) *(Random generated array)*
7: $ascii\_codes\_xor$ = double($currentBlock$) xor K; *(Vector of ASCII codes after bit-wise XOR)*
8: P = roots($[1 \ \ ascii\_codes\_xor]$); *(Solving polynomial)*
9: $encT(k) = P$; *(Encrypted block)*
10: **endFor**
11: **End Procedure**

Decryption is the inverse process of encryption. With ciphertext, block length, and decryption keys, one can obtain the original text following decryption operations. The algorithmic steps of the decryption process are summarized as follows (Algorithm 2).

Algorithm 2: Decryption

Input: Encrypted text $encT$, Block length Len and K PR bytes.
Output: Decrypted text $ReconT$
1: **Procedure** DECR($encT$, Len, K)
2: n = length($encT$); *(Extract the string length)*
3: $numBlocks$ = ceil(n/Len); *(Compute the number of blocks)*
4: **For** $k = 1 : numBlocks$ *(Count the blocks)*
5: $currentBlock = encT(k)$; *(Extract the current block)*
6: K = GenPRG(Len) *(Random generated array)*
7: p = poly($currentBlock$); *(Construct the monic polynomial)*
8: $ascii\_codes = p(2 : end)xorK$; *(Vector of ASCII codes after reverse XOR)*
9: $ReconT(k)$= native2unicode($ascii\_codes$); *(Decrypted block)*
10: **endFor**
11: **End Procedure**

In both algorithms described above, a syntax of the type $currentBlock = encT(k)$ is used, which means extracting the k-th block of $encT$ with specified block length. At step 6 in both algorithms, an additional function $GenPRG$, not described as pseudocode, is used, which generates an array of random numbers of length Len using the above-described Zaslavsky-based pseudorandom generation scheme. We have used the following Zaslavsky generator parameters: $\epsilon = 0.3, \nu = 0.2$, and $r = 5$.

The encryption and decryption schemes are shown in Figures 2 and 3.

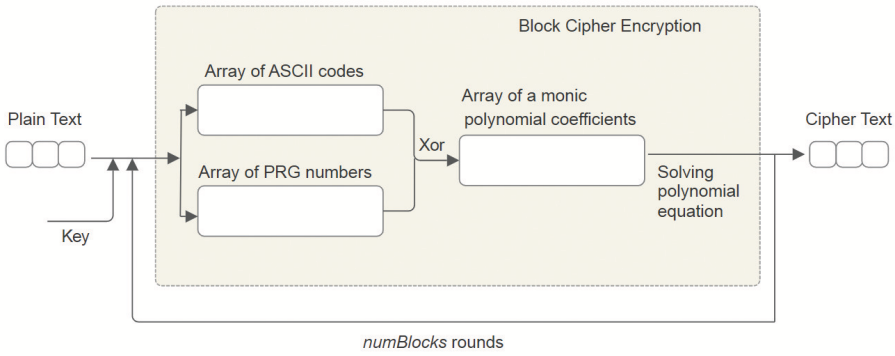

Figure 2. Encryption scheme.

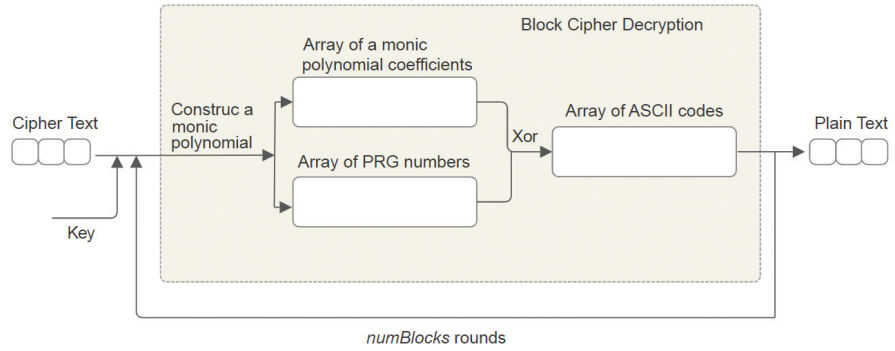

Figure 3. Decryption scheme.

3.3. An Example of Encryption

We will use the example below to illustrate the suggested algorithm. The input text is $inpT$ = "Shumen University" and the encrypted symbols, using two types of block encryption, with block lengths $Len = 4$ and $Len = 5$ are shown in Table 1. Table 1 shows only the result of the encryption process (Procedure $ENCR$). The result of the decryption (Procedure $DECR$) is actually the ASCII codes shown in Column 2 of Table 1.

Table 1. Example of encryption/decryption process.

Letter (Char)	ASCII Code	PR Number	Polynomial Coefficient	Encrypted Letter (Block Length = 5)	Encrypted Letter Block Length = 4
S	83	18	65	−62.8745	−0.6287
h	104	239	135	0.3295 + 1.0652i	−0.0200
u	117	45	88	0.3295 − 1.0652i	−0.0006 − 0.0105i
m	109	230	139	−1.3922 + 0.1368i	−0.0006 + 0.0105i
e	101	252	153	−1.3922 − 0.1368i	−1.5241
n	110	52	90	−89.8704	−0.0133
	32	46	14	0.4709 + 1.6081i	0.0037 − 0.0096i
U	85	131	214	0.4709 − 1.6081i	0.0037 + 0.0096i
n	110	143	225	−0.5357 + 0.5174i	−2.2438
i	105	229	140	−0.5357 − 0.5174i	−0.0001 − 0.0105i
v	118	136	254	−253.4214	−0.0001 + 0.0105i
e	101	246	147	0.2639 + 0.9013i	−0.0059
r	114	16	98	0.2639 − 0.9013i	−0.9553
s	115	159	236	−1.0233	−0.0262
i	105	122	19	−0.0830	0.0008 − 0.0076i
t	116	229	145	−143.3466	0.0008 + 0.0076i
y	121	148	237	−1.6533	−2.3700

The last two columns of Table 1 contain the output array of complex numbers, which is the encrypted text the receiver obtains.

4. Security Analysis

The key space is a group of numbers that can be used in the initial of the pseudo-random generation. The proposed technique has five initial values $y_{1,0}$, $z_{1,0}$, $y_{2,0}$, $z_{2,0}$, M, and Block length. As mentioned in the IEEE Standard for Floating-point Arithmetic [31], the computation accuracy of the 64-bit double-precision number is about 10^{14}. Consequently, the key space exceeds 2^{256} bits. This is substantial enough to thwart brute-force attempts [32,33], and it is larger than the key size of the pseudorandom output algorithms proposed in [9,11,19,34–37].

Using NIST [38] (National Institute of Standards and Technology) and Ent [39] applications, we attempted to determine the algorithm's unpredictability.

Array E outputs numbers (e_i) as bytes according to the formula $s_i = mod(abs(int(e_i \times 10^{14})), 256)$. Generating pseudorandom bytes using decimation, 2000 arrays of 125,000 bytes are produced.

The NIST application (version sts-2.1.2) includes 15 tests for randomness. The software calculates how many streams pass the particular test. Based on the confidence interval, an acceptable proportion range can be determined as

$$\hat{p} \pm 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}},$$

where $\hat{p} = 1 - \alpha$ and m is the number of binary streams. There should be at least 1000 sequences of 1,000,000 bits. The value of m in our file is 3000. In this case, the confidence level is

$$0.99 \pm 3\sqrt{\frac{0.99(0.01)}{2000}} = 0.99 \pm 0.0066746.$$

The proportion should lie above 0.9833254 with exception of random excursion and random excursion variant tests. Only when a sequence has more than 500 cycles do these two tests become relevant.

Ten subintervals make up the interval between 0 and 1. Each subinterval's corresponding p-values are counted. Applying a χ^2 test and figuring out a p-value for the goodness-of-fit check on the p-values acquired for any test, p-value of the p-values, can also be used to specify uniformity:

$$\chi^2 = \sum_{i=1}^{10} \frac{(F_i - s/10)^2}{s/10},$$

where s is the sample size and F_i is the number of p-values in the subinterval i. A p-value is calculated as $p\text{-}value_T = IGAMC(9/2, \chi^2/2)$, where $IGAMC$ is the complementary incomplete gamma function. If $p\text{-}value_T \geq 0.0001$, it is subsequently possible to regard the sequences as uniformly distributed.

We will present the particular tests briefly [38]: the ratio of ones to zeros throughout the whole sequence is the *frequency* test's main focus. The goal of this test is to assess whether a sequence has about the same amount of ones and zeros as would be predicted for a really random sequence. The percentage of ones in M-bit blocks is the *block frequency* test's main emphasis. The goal of this test is to evaluate if an M-bit block has a frequency of ones that is around M/2, as would be predicted by the randomness assumption. The maximum excursion (from zero) of the random walk, which is determined by the cumulative total of the adjusted (−1, +1) numbers in the stream, is the main focus of the *cumulative sums*. Finding out if the cumulative sum of the partial streams that occur in the tested stream deviates too much or too little from the cumulative sum's expected behavior for random streams is the aim of the test. The primary issue with the sequence is the overall number of runs, where a run is an unbroken string of identical bits. A run of length k is limited before and after by a bit with the opposite value, and it is made up of precisely k identical bits. The runs test is used to check if there are as many runs of ones and zeros as one would anticipate from a random sequence. The *longest run of ones within M-bit blocks* is the test's main focus. The aim of this test is to see if the longest run of ones in the sequence being evaluated is

consistent with the longest run of ones that would be anticipated in a random sequence. The rank of the entire sequence's disjoint submatrices is the *rank* main concern. This test looks for linear dependency between the original sequence's fixed-length substrings. The discrete Fourier transform of the sequence's peak heights is the main emphasis of *Fourier*. This test's objective is to find periodic features—repetitive patterns that are close to one another in the tested sequence that would contradict the idea of randomness.

The focus of *nonoverlapping templates* is the number of occurrences of prespecified target strings. The purpose of this test is to detect generators that output too many occurrences of a given aperiodic pattern. The focus of the *overlapping template matching* is the number of occurrences of prespecified target streams. *Overlapping template matching* is concerned with the number of instances of prespecified target streams. In the *universal* test, the number of bits between matching patterns is measured (this is related to the length of the compressed sequence). The purpose of the test is to detect without loss of information whether or not the stream can be significantly compressed. We can consider a significantly compressible stream to be nonrandom. By using *approximate entropy*, the frequency of overlapping blocks of two consecutive/adjacent lengths (m and $m + 1$) is compared against the expected result for a random sequence. While using serial, the frequency of all possible overlapping m-bit patterns in the entire stream is determined. The goal is to determine whether the number of occurrences of 2m m-bit overlapping patterns is approximately the same as expected for a random stream. There is uniformity in random sequences, meaning that every m-bit pattern has the same chance of appearing as every other m-bit pattern. *Linear complexity* refers to the length of a linear feedback shift register (LFSR). Using this test, we can determine whether the stream is complicated enough to be considered random. Counting the number of cycles with exactly K visits in a cumulative sum random walk is known as *random excursion*. In a cumulative sum random walk, partial sums are used to transfer a (0, 1) sequence into the correct $(-1, +1)$ stream.

Random excursion variant determines how many times a particular state is visited in a cumulative sum random walk. We use this test to determine how the random walk deviates from the expected number of visits to various states.

The output results for the first 13 tests can be found in Table 2. With the exception of the random excursion variant test, the minimum pass rate for each statistical test is approximately 1966 for a sample size of 2000 byte arrays. For a sample size of 1237 bytes of strings, the random excursion variant test has a minimum pass rate of approximately 1214. Table 3 tabulates eight p-values calculated by the random excursion test.

Table 2. NIST test suite results.

NIST Test	p-Value	Success Rate
Frequency	0.028244	1979/2000
Block frequency	0.941144	1982/2000
Cumulative sums forward	0.653773	1979/2000
Cumulative sums reverse	0.335324	1978/2000
Runs	0.288249	1978/2000
Longest run of ones	0.092877	1981/2000
Rank	0.692112	1978/2000
Fourier	0.133788	1975/2000
Nonoverlapping templates	0,477237	1980/2000
Overlapping templates	0.144915	1981/2000
Universal	0.304126	1982/2000
Approximate entropy	0.666245	1979/2000
Serial one	0.138464	1985/2000
Serial two	0.698285	1979/2000
Linear complexity	0.510153	1976/2000

Table 3. NIST random excursion test results.

State	*p*-Value	Success Rate
−4	0.667562	1228/1237
−3	0.510572	1228/1237
−2	0.998566	1222/1237
−1	0.638812	1224/1237
+1	0.803584	1225/1237
+2	0.758478	1217/1237
+3	0.852332	1226/1237
+4	0.559731	1232/1237

Table 4 shows that the random excursion variant test outputs 18 randomness probability numbers: *p*-values.

Table 4. NIST random excursion variant test results.

State	*p*-Value	Success Rate
−9	0.894055	1225/1237
−8	0.343294	1226/1237
−7	0.626963	1226/1237
−6	0.019209	1223/1237
−5	0.460086	1228/1237
−4	0.750448	1226/1237
−3	0.594875	1228/1237
−2	0.331738	1226/1237
−1	0.772760	1226/1237
+1	0.040947	1229/1237
+2	0.593192	1231/1237
+3	0.460086	1227/1237
+4	0.406370	1232/1237
+5	0.368547	1230/1237
+6	0.403493	1233/1237
+7	0.013779	1234/1237
+8	0.122036	1232/1237
+9	0.248871	1225/1237

From the results in Tables 2–4, we see that all the *p*-values are uniformly distributed over the $(0, 1)$ interval. For all the performed tests, the total numbers of acceptable arrays are within the expected confidence levels. Based on this, the novel pseudorandom byte generator passed without error in NIST suite.

The Ent application includes six tests on bit or byte sequences. We tested a stream of 250,000,000 bytes (2,000,000,000 bits) of the decimation-based pseudorandom scheme and tabulated the output data in Table 5. In addition, there are 999,998,680 zero occurrences with fraction 0.499999, and 1,000,001,320 ones with fraction 0.500001. The novel algorithm passed all the Ent tests successfully.

Table 5. Ent test results.

Ent Test	Input of Bits	Input of Bytes
Entropy	1.000000	7.999999
Optimum compression	Reduce size by 0%	Reduce size by 0%
χ^2 square	0.00, exceed 95.29 %	241.59, exceed 71.73%
Arithmetic mean value	0.5000	127.5015
Monte Carlo for π	3.14140229 (error 0.01%)	3.14140229 (error 0.01%)
Serial correlation	−0.000039	−0.000004

The proposed algorithm has been shown to be of high quality through various statistical tests. To the best of our knowledge, papers using iterative methods and chaotic functions for data encryption are [9,11,16–18]. Table 6 provides a summary of the computed values of our proposed scheme in comparison to other algorithms. In [16,18], there is a lack of sufficient statistical data to enrich the comparative analysis.

Table 6. Comparison of our proposed symmetric encryption with other schemes.

Encryption Schemes	Key Size	Correlation	Entropy	Arithmetic Mean
Proposed	2^{256}	−0.000004	7.999999	127.5015
Stoyanov 2020 [9]	2^{248}	−0.000002	7.999999	127.5055
Ali-Pacha 2022 [11]	-	-	7.981570	-
Murillo-Escobar [34]	2^{128}	−0.002100	7.994500	-
Hana 2020 [19]	2^{128}	-	7.983400	-
AES-128 [35,36]	2^{128}	−0.002100	7.954880	127.5281
Stoyanov 2015 [37]	2^{100}	0.000001	7.999998	127.4982
Othman 2015 [17]	-	-	7.21	-

Based on the positive test outputs and theoretical investigations, we can infer that the proposed readable text encryption based on a numerical method and rotation–translation formula demonstrates satisfactory statistical characteristics and offers a reasonable level of security. From the obtained results, it is confirmed that the new algorithm is not inferior in terms of parameters to the similar techniques indicated in the above tables. The disadvantages of the described algorithm are that in some cases the encrypted text is slightly larger than the input text due to the fact that the output symbols are in complex numbers.

The suggested encryption is dependent on the chaotic system's initial key parameters and system parameters, which were constructed using the two Zaslavsky equations. If any initial key values change, the encrypted output will be changed significantly; see Column 5 and Column 6 from Table 1. Consequently, the suggested encryption can survive the chosen plain text attack [40].

5. Conclusions

In conclusion, this research has presented an advancement in the field of text encryption, offering a promising avenue for enhancing data security in applications ranging from secure communication to data storage. By amalgamating decimated mathematical equations and polynomial root-finding methods, this innovative approach reveals new horizons for the development of encryption algorithms, with practical and theoretical implications. Our proposed encryption technique is based on a polynomial root-finding method and the output of the Zaslavsky equations. The proposed scheme uses monic polynomials and adds additional randomness by using the decimated Zaslavsky equations. The encryption approach proposed in this paper can be implemented using another chaotic function instead of the Zaslavsky one, such as Lorenz or Rössler equations.

Through extensive experimentation and analysis, we have demonstrated the efficacy and resilience of our method against a spectrum of cryptographic attacks. Our security analysis shows that the novel encryption technique can be successfully used in communication security. As the digital environment continues to evolve and threats to data security persist, approaches based on hybrid techniques open new avenues for encryption techniques with both theoretical reliability and practical utility. The potential implications of this research are profound, promising a more secure and adaptive future for information protection. This study represents a step towards enhancing the confidentiality and integrity of digital information, paving the way for further exploration and refinement in the field of cryptographic methods.

We believe that the areas in which our proposed encryption technique may have application include secure messaging and communication platforms, email encryption, data storage, and IoT (Internet of Things) security. It could also be applied in some key moments in blockchain technologies, such as key management. Blockchain systems rely on cryptographic keys for various purposes, including wallet addresses and transaction signing. Our encryption approach should seamlessly integrate with key management systems. On the other hand, blockchain networks often require fast transaction processing and validation. Encryption methods that are computationally intensive may not be ideal for use within a blockchain network. Efficiency and speed are crucial factors to consider.

In future steps, we want to experiment with encryption-then-compression of readable messages in order to decrease output file size. We also intend to use turbo codes with secret keys to provide high error correction levels. One of our objectives is to implement the proposed algorithm on a reprogrammable integrated circuit.

Author Contributions: B.S., G.N., D.D. and T.I. wrote and edited the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chen, C.; Lin, C. Text encryption using ECG signals with chaotic Logistic map. In Proceedings of the 2010 5th IEEE Conference on Industrial Electronics and Applications, Taichung, Taiwan, 15–17 June 2010; pp. 1741–1746. [CrossRef]
2. Chen, G.; Mao, Y.; Chui, C.K. A symmetric image encryption scheme based on 3D chaotic cat maps. *Chaos Solitons Fractals* **2004**, *21*, 749–761. [CrossRef]
3. Lian, S.; Sun, J.; Wang, J.; Wang, Z. A chaotic stream cipher and the usage in video protection. *Chaos Solitons Fractals* **2007**, *34*, 851–859. [CrossRef]
4. El-Latif, A.A.A.; Abd-El-Atty, B.; Mazurczyk, W.; Fung, C.; Venegas-Andraca, S.E. Secure Data Encryption Based on Quantum Walks for 5G Internet of Things Scenario. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 118–131. [CrossRef]
5. Rohde, P.P.; Fitzsimons, J.F.; Gilchrist, A. Quantum Walks with Encrypted Data. *Phys. Rev. Lett.* **2012**, *109*, 150501. [CrossRef] [PubMed]
6. El-Latif, A.A.A.; Abd-El-Atty, B.; Venegas-Andraca, S.E.; Elwahsh, H.; Piran, M.J.; Bashir, A.K.; Song, O.Y.; Mazurczyk, W. Providing End-to-End Security Using Quantum Walks in IoT Networks. *IEEE Access* **2020**, *8*, 92687–92696. [CrossRef]
7. Al-Saffar, N.F.H.; Al-Saiq, I.R. Symmetric text encryption scheme based Karhunen Loeve transform. *J. Discret. Math. Sci. Cryptogr.* **2022**, *25*, 2773–2781. [CrossRef]
8. Wang, P.; Li, X. TEDL: A Text Encryption Method Based on Deep Learning. *Appl. Sci.* **2021**, *11*, 1781. [CrossRef]
9. Stoyanov, B.; Nedzhibov, G. Symmetric key encryption based on rotation-translation equation. *Symmetry* **2020**, *12*, 73. [CrossRef]
10. Akgül, A.; Kaçar, S.; Arıcıoğlu, B.; Pehlivan, I. Text encryption by using one-dimensional chaos generators and nonlinear equations. In Proceedings of the 2013 8th International Conference on Electrical and Electronics Engineering (ELECO), Bursa, Turkey, 28–30 November 2013; pp. 320–323. [CrossRef]
11. Ali-Pacha, H.; Hadj-Said, N.; Ali-Pacha, A.; Özer, Ö. Numerical methods for differential equations as encryption key. *J. Interdiscip. Math.* **2022**, *25*, 2209–2237. [CrossRef]
12. Rachmawati, D.; Budiman, M.A. On Using The First Variant of Dependent RSA Encryption Scheme to Secure Text: A Tutorial. *J. Phys. Conf. Ser.* **2020**, *1542*, 012024. [CrossRef]
13. Dai, L.; Ye, Z.; Zhang, L.; Qian, T. A Novel Text Encryption Algorithm Based on Core Adaptive Fourier Decomposition. In *ACAI '19: Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*; Association for Computing Machinery: New York, NY, USA, 2020; pp. 491–497. [CrossRef]
14. Meier, W.; Staffelbach, O. The self-shrinking generator. In *Advances in Cryptology—EUROCRYPT'94*; De Santis, A., Ed.; Springer: Berlin/Heidelberg, Germany, 1995; pp. 205–214. [CrossRef]
15. Kanso, A.; Smaoui, N. Irregularly Decimated Chaotic Map(s) for Binary Digits Generations. *Int. J. Bifurc. Chaos* **2009**, *19*, 1169–1183. [CrossRef]
16. Ghosh, A.; Saha, A. A Numerical Method Based Encryption Algorithm with Steganography. *Comput. Sci. Inf. Technol.* **2013**, *3*, 149–157. [CrossRef]
17. Othman, H.; Hassoun, Y.; Owayjan, M. Entropy model for symmetric key cryptography algorithms based on numerical methods. In Proceedings of the 2015 International Conference on Applied Research in Computer Science and Engineering (ICAR), Beiriut, Lebanon, 8–9 October 2015; pp. 1–2. [CrossRef]
18. Hassoun, Y.; Othman, H. Symmetric Key Cryptography Algorithms Based on Numerical Methods. In Proceedings of the NumAn 2014 Conference, Crete, Greece, 22–27 June 2014; pp. 151–155.
19. Hana, A.P.; Naima, H.S.; Adda, A.P. Image encryption by using a specific adaptation of Lehmer's algorithm. *J. Discret. Math. Sci. Cryptogr.* **2020**, *23*, 949–971. [CrossRef]
20. Abduljabbar, Z.A.; Abduljaleel, I.Q.; Ma, J.; Sibahee, M.A.A.; Nyangaresi, V.O.; Honi, D.G.; Abdulsada, A.I.; Jiao, X. Provably Secure and Fast Color Image Encryption Algorithm Based on S-Boxes and Hyperchaotic Map. *IEEE Access* **2022**, *10*, 26257–26270. [CrossRef]
21. S, A.; Krishnan, M. Enhanced Audio Encryption using 2-D Zaslavsky Chaotic Map. In Proceedings of the 2022 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 25–27 January 2022; pp. 1–4. [CrossRef]

22. Deepa, N.R.; Sivamangai, N.M. A State-of-Art Model of Encrypting Medical Image Using DNA Cryptography and Hybrid Chaos Map—2d Zaslavaski Map: Review. In Proceedings of the 2022 6th International Conference on Devices, Circuits and Systems (ICDCS), Coimbatore, India, 21–22 April 2022; pp. 190–195. [CrossRef]
23. Available online: https://github.com/gyurhan/Encryption-by-Polynomial-Root-Finding-Method-and-Zaslavsky-Equations.git (accessed on 14 December 2023).
24. Zaslavsky, G. The simplest case of a strange attractor. *Phys. Lett. A* **1978**, *69*, 145–147. [CrossRef]
25. Farsana, F.; Devi, V.; Gopakumar, K. An audio encryption scheme based on Fast Walsh Hadamard Transform and mixed chaotic keystreams. *Appl. Comput. Inform.* **2019**, *ahead-of-print*. [CrossRef]
26. Stoyanov, B.; Kordov, K. Novel Zaslavsky Map Based Pseudorandom Bit Generation Scheme. *Appl. Math. Sci.* **2014**, *8*, 8883–8887. [CrossRef]
27. Zellagui, A.; Hadj-Said, N.; Ali-Pacha, A. A new hash function inspired by sponge construction using chaotic maps. *J. Discret. Math. Sci. Cryptogr.* **2023**, *26*, 529–559. [CrossRef]
28. Mohammed, A.H.; Shibeeb, A.K.; Ahmed, M.H. Image Cryptosystem for IoT Devices Using 2-D Zaslavsky Chaotic Map. *Int. J. Intell. Eng. Syst.* **2022**, *15*, 543–553. [CrossRef]
29. Stoyanov, B.; Todorova, M.; Ivanova, T.; Borboryan, G.; Hasanov, A. Two Zaslavsky maps in pseudorandom byte generation. *AIP Conf. Proc.* **2019**, *2164*, 120013. [CrossRef]
30. Farsana, F.; Gopakumar, K. A Novel Approach for Speech Encryption: Zaslavsky Map as Pseudo Random Number Generator. *Procedia Comput. Sci.* **2016**, *93*, 816–823. [CrossRef]
31. *IEEE Std 754-2019 (Revision of IEEE 754-2008)*; IEEE Standard for Floating-Point Arithmetic. IEEE: Piscataway, NJ, USA, 2019; pp. 1–84. [CrossRef]
32. Alvarez, G.; Li, S. Some basic cryptographic requirements for chaos-based cryptosystems. *Int. J. Bifurc. Chaos* **2006**, *16*, 2129–2151. [CrossRef]
33. Alvarez, G.; Amigó, J.M.; Arroyo, D.; Li, S. Lessons Learnt from the Cryptanalysis of Chaos-Based Ciphers. In *Chaos-Based Cryptography: Theory, Algorithms and Applications*; Kocarev, L., Lian, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 257–295. [CrossRef]
34. Murillo-Escobar, M.; Cruz-Hernández, C.; Cardoza-Avendaño, L.; Méndez-Ramírez, R. A novel pseudorandom number generator based on pseudorandomly enhanced logistic map. *Nonlinear Dyn.* **2017**, *87*, 407–425. [CrossRef]
35. Abubaker, S.; Wu, K. DAFA—A Lightweight DES Augmented Finite Automaton Cryptosystem. In *Security and Privacy in Communication Networks*; Keromytis, A.D., Di Pietro, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 1–18. [CrossRef]
36. Mushtaq, M.F.; Jamel, S.; Disina, A.H.; Pindar, Z.A.; Shakir, N.S.A.; Deris, M.M. A Survey on the cryptographic encryption algorithms. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 333–344. [CrossRef]
37. Stoyanov, B.; Kordov, K. Image Encryption Using Chebyshev Map and Rotation Equation. *Entropy* **2015**, *17*, 2117–2139. [CrossRef]
38. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, A.; et al. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Application*; NIST Special Publication 800-22: Revision 1a; Lawrence E. Bassham III: Gaithersburg, MD, USA, 2010.
39. Walker, J. *ENT: A Pseudorandom Number Sequence Test Program*; Fourmilab: Neuchatel, Switzerland, 2008.
40. Barrera, J.F.; Vargas, C.; Tebaldi, M.; Torroba, R. Chosen-plaintext attack on a joint transform correlator encrypting system. *Opt. Commun.* **2010**, *283*, 3917–3921. [CrossRef]