Application of Catalan Numbers and the Lattice Path Combinatorial Problem in Cryptography

Muzafer Saračević[1], Saša Adamović[2], Enver Biševac[1]

[1] Department of Computer Sciences, University of Novi Pazar, Dimitrija Tucovića bb, 36300, Novi Pazar, Serbia, (muzafers, e.bisevac)@uninp.edu.rs

[2] Faculty of Informatics and Computing, Singidunum University in Belgrade, Danijelova 32, 11000 Belgrade, Serbia, sadamovic@singidunum.ac.rs

Abstract: This paper analyzes the properties of Catalan numbers and their relation to the Lattice Path combinatorial problem in cryptography. Specifically, analyzes the application of the appropriate combinatorial problem based on Catalan-key in encryption and decryption of files and plaintext. Accordingly, we use Catalan numbers for generating keys and within the experimental part we have applied the NIST (National Institute of Standards and Technology) statistical battery of tests for assessing the quality of generated keys was applied. A total of 12 quality assurance tests for Catalan-key were applied. A Java application is presented which allows the encryption and decryption of plaintext based on the generated Catalan-key and combinatorial problem of movement in integer network or Lattice Path. Experimental study yields the comparison of results in text encryption speed for combinatorial encryption methods (such as: Ballot Problem, Stack permutations and Balanced Parentheses) in comparison with Lattice Path method (in Java programming language).

Keywords: cryptography; Catalan numbers; Lattice path; combinatorial problems; encryption

1 Introduction

The subject of this research paper refers to the testing of Catalan numbers and the possibility of their use in cryptography. In addition, this research paper also illustrates the application of Lattice Path combinatorial problems which are based on the properties of Catalan numbers, used for encrypting and decrypting files and plaintext. The idea emerged, based on our previous research in the field of combinatorial problems, number theory and Catalan numbers. The pseudo-random numbers are of paramount importance in these procedures, particularly in key generation [1].

Number theory is most important in the process of key generation and also in the design of the cryptographic algorithm and in the cryptographic analysis [2, 3].

By using Catalan-key in encryption and decryption of files and plaintext, the basic idea is realized: generation of a long and unpredictable binary sequence of symbols from an alphabet on the basis of a short secret key, selected in a random manner. In this case, we will present the Lattice Path combinatorial encryption methods. In this way, a effective system performance for encryption is achieved [4].

2 Related Work

Applied number theory has numerous applications in cryptography, especially in the field of the integer sequences. Previous cryptographic algorithms were designed by using the integer sequences of the Fibonacci sequence and Lucas numbers.

Monograph [5] lists the concrete applications of these numbers with the possible solutions in terms of representation of Catalan numbers. This monograph contains a set of tasks that describe over 60 different interpretations of Catalan numbers. Some can be enumerated: the triangulation of polygons, paired brackets problem, a binary tree, steak permutations, Ballot problem, Lattice Path or the problem of motion through an integer grid, etc. In [6], the enumeration and generation of generalized Dyck words (path) based on Catalan numbers is discussed.

Paper [7] proposes cryptographic algorithms based on integer sequences of Catalan numbers as new methods of encryption. In the proposed encryption method, by using Catalan numbers, a large random number "*n*" is set as a secret key for encryption text in one file. The binary notation of Catalan number C_n, corresponding to the agreed upon secret key. In the mentioned paper, the proposed encryption methods use a logical XOR operation (exclusive-OR) on bits of *ASCII* binary code of messages.

More precisely, in the mentioned paper, the binary records of Catalan numbers and the messages, as well as, the XOR operations between them, simulate a One-Time Password (OTP) code. From the point of cryptanalysis, the proposed algorithm uses Catalan numbers, where it seems that they are resistant to the most known sorts of attacks. Identical characters in plain text are coded with different crypto characters. Thereby, "brute force attack" and the complete key search are difficult to perform. The time for encryption or decryption is independent of the characters in the data block.

Paper [8] presents an advanced technique for encryption based on values that satisfy properties of Catalan numbers. Key generation is based on a series of

Catalan numbers. The primary key value is fixed and defined by the user. In addition, each subsequent key value is double the previous one. The objective of the algorithm is to make cryptanalysis more difficult and to strengthen the algorithm. This paper emphasizes another important application of Catalan numbers, that is, the application in cryptosystem design, with techniques of recursive key generation.

Catalan numbers have the property of reclusiveness and their generation can be efficiently implemented with dynamic programming. Paper [9] gives a proposal of a *Vigenere* cipher modification, with the help of Catalan numbers and double transposition. In this paper, this method is based on Catalan numbers and it is presented as a mathematical method, which is used to create the initial cryptanalysis key with an emphasis on stronger key properties (balance and unpredictability).

From the point of view of cryptanalysis, it is emphasized that such a key is more difficult to detect, due to its specific properties and sequences that correspond to Catalan numbers.

3 The Basic Properties of Catalan Numbers

Catalan numbers (C_n) represent a sequence of numbers which are primarily used in computational geometry and in solving many combinatorial problems. Catalan numbers, $n > 0$, present a series of natural numbers, which appear as a solution to a large number of known combinatorial problems [5] (the number of possible paths in a discrete grid of n x n, problem of balanced parenthesis, stack permutations, binary trees, triangulation of polygons, etc.).

Catalan numbers are defined as [5]:

$$C_n = \frac{(2n)!}{(n+1)!n!} \tag{1}$$

Now we are going to analyze the values which are generated in the C_n set. For the purposes of Catalan numbers validity verification, we will use the binary notation. The basic feature that must be fulfilled is *bit property balance*, in the binary form for a certain number from the C_n set (we will referee to this property as *bit-balance* property).

For example, for the basis $n=29$ we have the space of keys $C_{29}=1,002,242,216,651,368$, i.e. the values that satisfy the property of Catalan number. By increasing the n basis, the key space is also drastically increasing. In order to provide a stronger, i.e. a more resistant mechanism of cryptanalysis encryption, it is necessary to choose keys whose value base is mainly greater than $n=30$.

Catalan number property [5]: A number can be labeled as a Catalan number when its binary form consists of numbers equal to "1" and "0" and starting with "1". If a binary notation of a Catalan number is connected with another mode of writing, most often with the mode of balanced parentheses, then "1" represents an open parenthesis and "0" represents a closed parenthesis, and it can be said that each opened parenthesis closes, or every bit 1 has its pair and that is the bit 0.

This property is known as a Dyck word. The Dyck words interpretation of Catalan numbers, so that C_n is the number of ways to correctly match *n* pairs of parentheses. C_n is the number of monotonic lattice paths along the edges of a grid (Lattice) with $n \times n$ square cells, which do not pass above the diagonal.

A monotonic path is one which starts in the lower left corner, finishes in the upper right corner, and consists entirely of edges pointing rightwards or upwards. Counting such paths is equivalent to counting Dyck words or number of *valid Dyck path*. Coordinate X stands for "move right" and Y stands for "move up".

A Dyck path of semilength *n* is a lattice path from (0,0) to (2n,0) consisting of *n* up steps of the form (1,1) and *n* down steps of the form (−1,1) which never goes below the x-axis y=0. Every Dyck word *w* of length ≥ 2 can be written in a unique way in the form $w = Xw_1Yw_2$ with Dyck words w_1 and w_2 [5].

Also, in [6] the definition is given for Dyck path of semilength *n*, which can be seen as a Dyck word, this is a word in {0,1} such that any prefix contains at least as many 1's as it contains 0's. Seeing a 1 as an opening parentheses and a 0 as a closing bracket, Dyck words can be seen as *well-formed parentheses systems*.

Dyck words definition [6]: Let B = {0, 1} be a binary alphabet and $X_1X_2 \dots X_n \in B^n$. Let h: B → {−1, 1} be a valuation function with

$$h(0)=1,\ h(1)=-1 \text{ and } h\left(X_1X_2...X_n\right) \ = \sum_{i=1}^{n} h(x_i)$$

A word $X_1X_2 \dots X_n \in B2^n$ is called a Dyck word if it satisfy conditions:

- $h(X_1X_2 \dots X_i) \geq 0$, for $1 \leq i \leq 2n - 1$
- $h(X_1X_2 \dots X_{2n}) = 0$, where *n* is the semi-length of the word

In papers [10, 11, 12, 13] we performed generation testing of all the numbers for a given basis *n* which fulfill the above mentioned Catalan number properties. Based on the given analyzes, we can perform basic characteristics of keys generation based on Catalan number properties:

1. The condition of balance property must be fulfilled
2. It can serve as pseudo-random number generator (PRNG)
3. It can be used for realization of sequential algorithms
4. Based on the key belonging to the *n* basis, or 2*n*-length key, appropriate *n*-permutations can be made

4 Application of Catalan Numbers and the Lattice Path Combinatorial Problem in File Encryption

Catalan numbers have found widespread use in solving many combinatorial problems. In [5, 6], concrete applications of these numbers are given, with possible solutions, when it comes to representation over certain combinatorial problems. The number of combinations and the manner of Catalan number generation represent a solution for certain combinatorial problems.

The binary notation of a Catalan number can be graphically represented in the integer network (another name, discrete grid or *Lattice Path*) which consists of a number of points in the Cartesian coordinate system. The problem is related to the number of calculations of the paths for movement through the integer network. The number of possible valid paths in the network is directly determined by the calculating formula for the C_n set of Catalan numbers. The pathways consist of *2n* steps with the initial point *(0,0)* and the end point *(n, n).* If we want to present the binary record of Catalan number in the form of movement through the integer network, then bit 1 represents movement to the right and bit 0 represents movement to the up.

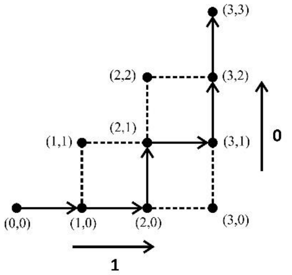


Figure 1

Lattice path based on the Catalan-key K3 = 110100

As shown in the figure, each path in the integer network can be encoded with specific order of vector movement to the right (1,0) and vector movement to the up (0,1). The selection of position movements (label 1, of the total number of *2n* movements) uniquely determines the path in the integer network, because the remaining positions represent the movement to the right (label 0). Thus, if we apply a valid Catalan number, the direction through integer network will do exactly *2n* movements, starting from the center point (0,0), and finishing it at the end point *(n,n)*.

We will check whether the movement in the integer network actually corresponds to Catalan number property. Based on the general case, a restriction has been introduced on the network of the size n × n and it thus, determines how many shortest paths exist in the integer network.

The path never crosses its diagonal. The main requirement is that each subsequent step must be closer to the target point. The number of possible paths in the network of dimension *n* is determined by Catalan number for the basis *n*, and the binary values in the C_n set are determining different combinations of the paths in the network. The presented movement procedure through the integer network based on the *binary* key record can serve as an idea for the system formation of plaintext encryption.

The encryption process based on discrete lattice can be applied to text messages (*String variant*), but also it can be applied to binary messages (*ASCII Text to Binary*). In addition, due to a better understanding, we will show the process of text encryption where the values are taken as a string record (the transposition cipher is obtained), and in the experimental part of this paper, we will present an example of taking an open text in a binary form (the substitution cipher is obtained). Based on the position of bits 0 and 1 in the binary key record, the elements in the text can have two states:

1. *Free element* – a character from the message which is not encrypted, or more precisely, which is not transferred in the cipher text. The free element is conditioned by the appearance of bit 1 (in the key), and it awaits its pair, bit 0.
2. *Engaged element* – a character from the message that is encrypted and transferred in the cipher text. This is an element that is conditioned by the appearance of bit 0 (in the key). In this way, the element is "closed" (transferred in the cipher text because bit 0 has appeared and it closes the corresponding bit 1).

The Code for encryption process based on movement through *LatticePath:*

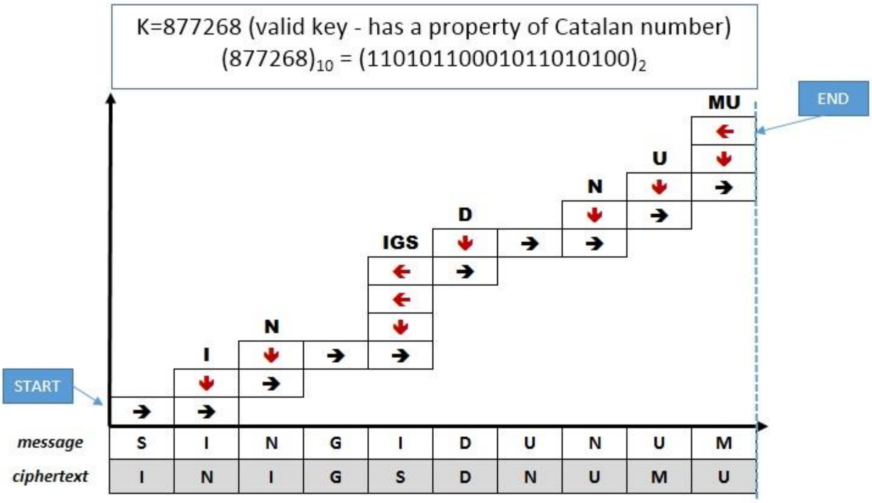
```
n=0, A[n]=1; segment = 60;
1. count = file.length() / segment;
2. for (j = 0; j < count; j++) {
3. in.read(text, 0, segment);
4. EndPoint = segment - 1;
5. for (i=n; i>=0; i--) {
6. if (A[i] == 1) {
   path[Free] = text[EndPoint];
   Free ++;
   EndPoint --;
   }
7. else {
   ciphertext[Engaged] = path[Free-1];
   Engaged ++; Free --;
   path[Free] = 0;  }
   }
```

```
    8. out.write(ciphertext);
       }
```

Explanation of source code:

(1) Splits the message (a file) on n-bit segments (fits to the basis of the n key); (2) The cycle for inclusion of the file segments starting from 0th up to the last segment; (3) In each segment, the reading of the elements is performed, from the first up to the last element in the segment; (4) Minimizing the position of the bits in the segment; (5) The cycle for reading the bits in the key; (6) if it is bit 1 then movement to the right follows (increasing the number of the free characters, and reducing the number of steps to the end point); (7) if it is bit 0 then movement to the up follows (takes character, which means that it increases the number of engaged characters and reduces the number of free ones); (8) at the end, when all the segments of the message are completed, the complete ciphertext is printed.

Example 1: The given key **K=877268**, which possesses the Catalan number property based on the binary record $(877268)_{10}$ = $(11010110001011010100)_2$ for the given plaintext **P=SINGIDUNUM**. Moving through the integer network based on the binary representation of the key, starting from the source to the end point, we get the ciphertext *C=INIGSDNUMU.*



K=877268 (valid key - has a property of Catalan number)
$(877268)_{10}$ = $(11010110001011010100)_2$

message	S	I	N	G	I	D	U	N	U	M
ciphertext	I	N	I	G	S	D	N	U	M	U

Figure 2

Encryption based on the principle of movement through a discrete grid

From the aforementioned process, we can identify two basic rules of movement: (1) Movement through the integer network never crosses its diagonal; (2) In every next step, the movement must be further from the START and closer to the END point.

Table 1

State of the characters in P = "SINGIDUNUM" based on Lattice Path

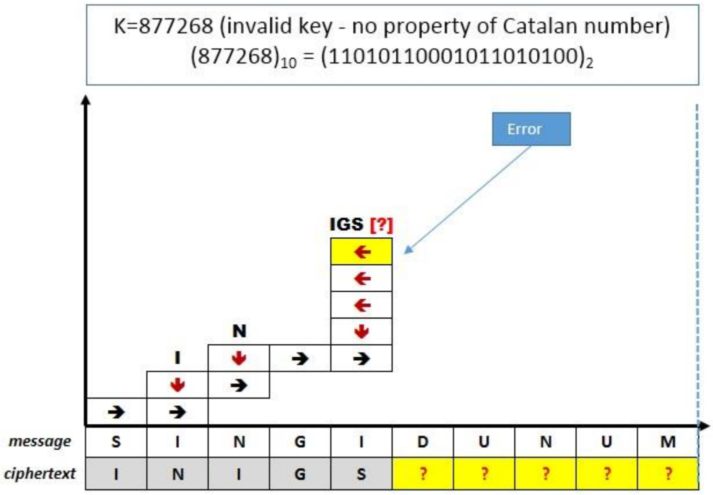
Bit in key	**Free elements - occurrence of bit 1**	**Engaged element - occurrence of bit 0**	**Parameters of the counter**
1	S		EndPoint=19, Free=1, Engaged=0
1	S, I		EndPoint=18, Free=2, Engaged=0
0	S,	I	EndPoint=17, Free=1, Engaged=1
1	S,N,	I	EndPoint=16, Free=2, Engaged=1
0	S,	I, N	EndPoint=15, Free=1, Engaged=2
1	S, G	I, N	EndPoint=14, Free=2, Engaged=2
1	S, G, I	I, N	EndPoint=13, Free=3, Engaged=2
0	S, G	I, N, I	EndPoint=12, Free=2, Engaged=3
0	S	I, N, I, G	EndPoint=11, Free=1, Engaged=4
0		I, N, I, G, S	EndPoint=10, Free=0, Engaged=5
1	D	I, N, I, G, S	EndPoint=9, Free=1, Engaged=5
0		I, N, I, G, S, D	EndPoint=8, Free=0, Engaged=6
1	U	I, N, I, G, S, D	EndPoint=7, Free=1, Engaged=6
1	U, N	I, N, I, G, S, D	EndPoint=6, Free=2, Engaged=6
0	U	I, N, I, G, S, D, N	EndPoint=5, Free=1, Engaged=7
1	U,U	I, N, I, G, S, D, N	EndPoint=4, Free=2, Engaged=7
0	U	I, N, I, G, S, D, N, U	EndPoint=3, Free=1, Engaged=8
1	U, M	I, N, I, G, S, D, N, U	EndPoint=2, Free=2, Engaged=8
0	U	I, N, I, G, S, D, N, U, M	EndPoint=1, Free=1, Engaged=9
0		I, N, I, G, S, D, N, U, M, U	EndPoint=0, Free=0, Engaged=10

We can conclude that the characters are taken from the plaintext at the moment when we get an ordered pair of 1 and 0. As long as the corresponding bit 1 does not get its pair of bit 0, the character has the status of "free", specifically it is not transferred in the ciphertext. The moment it gets its pair, the character will receive the status "engaged" and it is transferred to the ciphertext. Decryption is performed in reverse order of reading the binary key record, starting from the last bit and ending at the first bit in the key. In this case, (1,0)→(0,1) applies, and the occurrence of bit 0 indicates an open pair and 1 closed pair.

Example 2: Now let us take a value (for the key) that does not have Catalan number property, specifically for that value the rule of bit balance does not apply. We will try to use that key in the encryption of the text based on the movement through the integer network.

Case 1: The given key is *K=877011*, for which we can determine that it does not possess Catalan number property based on the binary record *(877011)*$_{10}$ = *(11010110000111010011)*$_{2.}$ And in the given plaintext *P=SINGIDUNUM.* Moving through the integer network, starting from the source point we cannot get to the end point, or more precisely, we cannot successfully complete the process.

From this case, we can see that the problem originated in the 11th step, more accurately, the problem is in the 11th bit in the key (bit 0), because it does not have its pair (877011)$_{10}$ = (1101011000$\boxed{\mathbf{0}}$111010011)$_2$

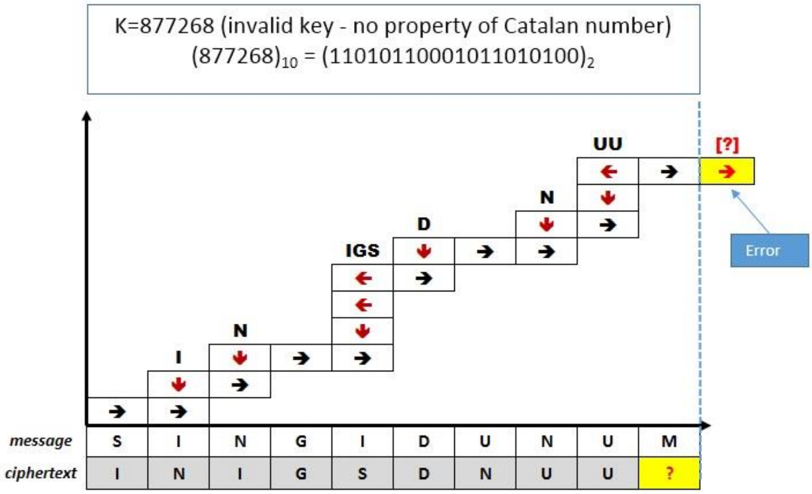


message	S	I	N	G	I	D	U	N	U	M
ciphertext	I	N	I	G	S	?	?	?	?	?

Figure 3

Unsuccessful encryption based on the key that is not Catalan (case 1)

Case 2: The given key is *K=877267*, for which we determined that it does not possess the bit balance property based on its binary record *$(877267)_{10}$* = *$(11010110001011010011)_2$.* And in the given plaintext *P=SINGIDUNUM*. Moving through the integer network, starting from the source point we cannot get to the end point. In this case the movement exceeds the space of the network.



message	S	I	N	G	I	D	U	N	U	M
ciphertext	I	N	I	G	S	D	N	U	U	?

Figure 4

Unsuccessful encryption based on the key that is not Catalan (case 2)

From this case, we can see that the problem originated in the 20th step, more accurately, the problem is in the 20th bit in the key (bit 1), because it does not have its pair $(877267)_{10} = (1101011000101101001\boxed{\mathbf{1}})_{2}$.

In both cases we can conclude that the balance conditions are not fulfilled, and they are the basis for determining Catalan number properties.

This basic rule has caused the two basic conditions of movement through the integer network not to be fulfilled:

1. In the first case, the condition which requires that with every next step it has to be closer to the end point and that it must start and finish within the points that determine the diagonal is not fulfilled.
2. On the other hand, in the second case the introduced restriction on the network size $n \times n$ is not complied, so moving outside is impossible because it is exactly determined how many shortest paths there are in the integer network that never exceed its diagonal, and that is the C_n number.

The conclusion is that the key that does not possess Catalan number properties is not functional in the movement of the integer network. It is noteworthy that this process of encryption can be applied to the textual format of the message (string transposition) and reading the message in the binary form can be performed, hence the application of the described process in the binary message record is conducted.

Example 3: Given is the plaintext *P=“SINGIDUNUM“*. If we apply convertor *ASCII Text to Binary* on *P*, then we get set of bits: *01010011 01001001 01001110 01000111 01001001 01000100 01010101 01001110 01010101 01001101*.

If we apply the key $K = (930516)_{10} = (11100011001011010100)_{2}$ based on which we will perform a permutation of bits from the message, then we will get the following binary record of the ciphertext: *C=01001001011010000 10001100110010101100011010000100101001001000 11001101101010001011*.

If we apply the *Binary to ASCII Text* convertor to the ciphertext C, we will get the textual record of the ciphertext *C=”IhFecBRFmK“*.

In this way, we get the stronger ciphertext, that is, one character is replaced by some completely different symbol depending on the obtained permutation of the bits. In this case, we do not have the classical transposition code as in the previous examples; instead we have the code of substitution or replacement.

It is important to mention that here a conventional substitution is not realized, because one character from the message is not always replaced with the same character in the ciphertext. The mode of replacement depends on the key itself and its length, as well as from the disposition of the bits in a key. Also, it depends on the length of the message and the size of the message segments that are taken in the encryption process.

If from the previous example we compare the plaintext P="*SINGIDUNUM*", and the obtained ciphertext C="*IhFecBRFmK*", we can see that the first character "*I*" is replaced with "h", and the second character "*I*" with "*c*". It is the same case with the character "*U*", where the first has been replaced with "*R*" and the second one with "m". In this manner we provide a stronger encryption mechanism.

Paper [14] is related to investigating properties of Catalan numbers and their possible application in the procedure of data hiding in a text, more specifically in the area of steganography. The objective of this paper is to explain and investigate the existing knowledge on the application of Catalan numbers, with an emphasis on dynamic Catalan-key generation and their application in data hiding.

Based on many other research papers, we can observe that *Lucas–Catalan–Fibonacci* numbers are used, which are far superior compared to *Fibonacci* technique for data hiding. In the papers [15,16], new techniques for data hiding are suggested that use combinations of *Catalan–Fibonnaci* and *Catalan–Lucas* numbers sequences, which represents an improvement compared to the existing techniques for data hiding.

5 Experimental Work with Case Studies

Our Java software solution for this case study consists of three segments [4,11]. The first phase involves finding Catalan numbers (keys) based on the given *n* basis. This phase involves the next steps: (1) On the input *n* is assigned, (2) On the basis of *n*, the set C_n (space of keys) is calculated, (3) Selecting one Catalan number (key) from the C_n set, (4) The selected key is converted from the decimal to the binary record.

For implementing the Java software solution for *LatticePath* encryption method, it is important to note that we did not use a ready-made Java classes from the two standard APIs (JCA, JCE).

For details regarding the used Java classes, GUI and application functionality for *LatticePath* encryption method see our paper [11].

The form of the *Java GUI application* has the following options:

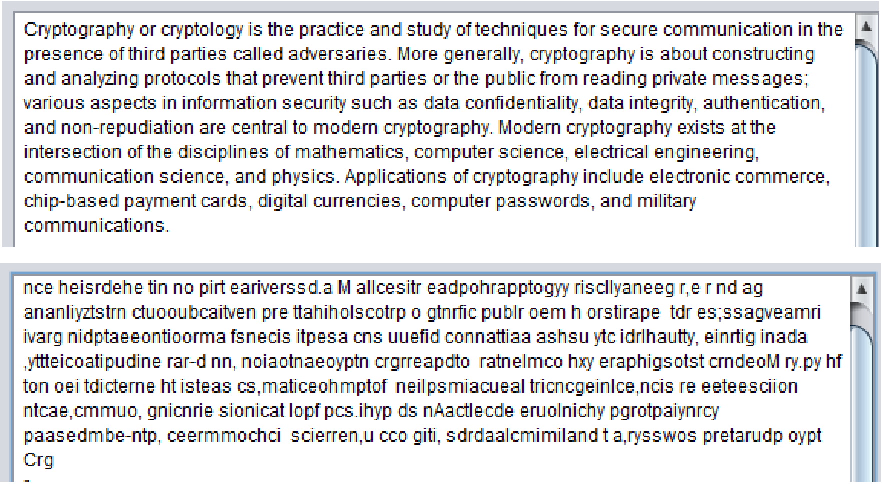
- Loading a Catalan-key
- Loading a file with text
- *LatticePath* encryption
- *LatticePath* decryption
- Checking the number of valid Catalan-keys for given *n*
- Generating the Catalan-keys that have Catalan number properties

The condition to start generating the entire space of Catalan-keys for a particular *n* basis is to determine the file in which the entire space of Catalan-keys will be recorded. After that, generation and recording of keys starts. This process may take time, depending on the input of *n*.

When starting a Java application, the first step is loading the Catalan-key from an external file. There is an algorithm for generating a complete space of Catalan-keys for a specific *n* basis. We use the method of manual taking of one of those values and storing them in the file *Cat-Key.TXT*.

After that, we include the active Catalan-key. We can create multiple Catalan-keys, but in one process we have to determine which key is active for the encryption or decryption process.

After loading the Catalan-key, the next step is loading the plaintext. After successful loading of the key and the message, the "*LatticePath encryption*" button becomes enabled. By clicking the button "*LatticePath decryption*", we can decrypt ciphertext and compare it to the message.



Cryptography or cryptology is the practice and study of techniques for secure communication in the
presence of third parties called adversaries. More generally, cryptography is about constructing
and analyzing protocols that prevent third parties or the public from reading private messages;
various aspects in information security such as data confidentiality, data integrity, authentication,
and non-repudiation are central to modern cryptography. Modern cryptography exists at the
intersection of the disciplines of mathematics, computer science, electrical engineering,
communication science, and physics. Applications of cryptography include electronic commerce,
chip-based payment cards, digital currencies, computer passwords, and military
communications.

```
nce heisrdehe tin no pirt eariverssd.a M allcesitr eadpohrapptogyy riscllyaneeg r,e r nd ag
ananliyztstrn ctuooubcaitven pre ttahiholscotrp o gtnrfic publr oem h orstirape  tdr es;ssagveamri
ivarg nidptaeeontioorma fsnecis itpesa cns uuefid connattiaa ashsu ytc idrlhautty, einrtig inada
,yttteicoatipudine rar-d nn, noiaotnaeoyptn crgrreapdto  ratnelmco hxy eraphigsotst crndeoM ry.py hf
ton oei tdicterne ht isteas cs,maticeohmptof  neilpsmiacueal tricncgeinlce,ncis re eeteesciion
ntcae,cmmuo, gnicnrie sionicat lopf pcs.ihyp ds nAactlecde eruolnichy pgrotpaiynrcy
paasedmbe-ntp, ceermmochci  scierren,u cco giti, sdrdaalcmimiland t a,rysswos pretarudp oypt
Crg
```

Figure 5

Example of *LatticePath* encryption of plaintext and displaying the ciphertext

Paper [11] examines the possibilities of applying of appropriate combinatorial problems in encryption and decryption of files and plaintext. A comparison is given for combinatorial encryption methods based on Catalan-key, such as: *Ballot Problem, Stack permutations* and *Balanced Parentheses*.

Table 2 shows the comparison of results in text encryption speed for these combinatorial encryption methods and Lattice Path method (in *Java* programming language).

Table 2

Comparison of different combinatorial encryption methods based on Catalan-key

Length of text (in characters, with spaces)	Text encryption methods (time in seconds)			
	Lattice Path combinatorics	*Stack permutations*	*Ballot Problem*	*Balanced Parentheses*
1000	0.001	0.001	0.001	0.001
5000	0.003	0.004	0.004	0.004
30000	0.024	0.026	0.025	0.025
50000	0.040	0.041	0.040	0.041
100000	0.085	0.087	0.085	0.087
500000	0.407	0.417	0.411	0.419
1000000	0.989	0.995	0.991	0.996

6 NIST Statistical Test Battery for the Catalan-Key

Considering that we use a Catalan number for key generation, we will apply the NIST (*National Institute of Standards and Technology*) statistical battery of tests for assessing the quality. The NIST's package of tests consist of multiple statistical tests that have been developed for testing the randomness of binary sequences that produce software or hardware on the basis of random or pseudo-random numbers. These statistical tests focus on the different types of inconsistencies that might exist in the sequence. Some tests are divided into subtests [17]. A total of 12 quality assurance tests for Catalan-key were applied.

For an input parameter we took the key that satisfies the property of Catalan number ε = *1110110101100101001010011111101010010010010001110111010100010100 10 0110111100101010100001011010110000*.

1) The objective of the test for examining the frequency in the series, based on an analysis of the relationship of 1s and 0s in a series of bits, more precisely, is to observe the equality of 1 of 0 occurrences. An approximate number of ones and zeroes is necessary in the sequence, which means that Catalan number property or the bit balance property is always a good result for this test. All subtests derived from this test are directly dependent on its successfulness. The invoke of this test is done through the method ***Frequency(n)*** where *n* is the bit length of the sample. The method uses the parameter *ε* (a series of bits in the key).

```
FREQUENCY TEST - COMPUTATIONAL INFORMATION:

  (a) The nth partial sum = 6
  (b) S_n/n             = 0.006000

SUCCESS       p_value = 0.849515
```

Since $P \geq 0.01$, it is considered that the sequence is random.

2) The test for examining the frequency in the block shows the ratio of 1s and 0s in *n*-bit blocks. The purpose of this test is to detect equality number of 1s and 0s in each *n*-bit block. This test is invoked through the method ***BlockFrequency(M,n)*** where *M* is the length of every block, and *n* is the length of the bit sample. The same as with the test for testing frequency in the series, this test uses the parameter *ε*.

```
BLOCK FREQUENCY TEST - COMPUTATIONAL INFORMATION:

(a) Chi^2 = 1.718750
(b) # of substrings = 7
(c) block length = 128
(d) Note: 104 bits were discarded

SUCCESS              p_value = 0.973758
```

The test displays that the sequence is random, since $P \geq 0.01$.

3) The test for examining the successive repetition of the same bits in the series refers to the total number of successive repetitions of a number in the series. The purpose of the test is to determine whether the number of consecutive repetitions of 0s and 1s matches the expected random sequence. This test is invoked through the method ***Runs(n)*** where *n* is the length of the bit sample. As an additional parameter, input ε is used.

```
RUNS TEST - COMPUTATIONAL INFORMATION:

 (a) Pi                        = 0.503000
 (b) V_n_obs (Total # of runs) = 465
 (c) V_n_obs - 2n pi (1-pi)
      ----------------------- = 1.564499
      2 sqrt(2n) pi (1-pi)

SUCCESS              p_value = 0.026930
```

It is considered that the sequence is random, because the result is $P \geq 0.01$.

4) The test for examining the longest consecutive repetition of units in n-bit blocks determines whether the length of the longest continuous repetition matches the length which is expected in the series of random numbers. This test is invoked through the method ***LongestRunOfOnes(n)*** where *n* is the length of the bit sample. The test is set in the way that for M (the length of each block) three values are used: M = 8 (where *n* is minimum 128), M = 128 (where *n* is minimum 6272), and M = 104 (where *n* is minimum 750,000). In this case, the test is M = 8. As an additional parameter, the input ε is used.

```
LONGEST RUNS OF ONES TEST - COMPUTATIONAL INFORMATION:

 (a) N (# of substrings)  = 125
 (b) M (Substring Length) = 8
 (c) Chi^2                = 3.500324
                       F R E Q U E N C Y
<=1     2     3    >=4   P-value Assignment
  26   47    22    30    SUCCESS p_value = 0.320720
```

The test result shows that $P \geq 0.01$, so the sequence is considered random.

5) The test for examining the state of the binary matrix refers to the verification of the linear dependence between the subseries of fixed length from the original series. The test is invoked through the method ***Rank(n)*** where *n* is the length of the bit sample. The method uses the parameter *ε* (series of bits in the key record).

```
RANK TEST - COMPUTATIONAL INFORMATION:
 (a) Probability P_32 = 0.288788
 (b)             P_31 = 0.577576
 (c)             P_30 = 0.133636
 (d) Frequency   F_32 = 0
 (e)             F_31 = 1
 (f)             F_30 = 0
 (g) # of matrices    = 1
 (h) Chi^2            = 0.731373
 (i) NOTE: 0 BITS WERE DISCARDED.
SUCCESS      p_value = 0.693720
```

It is considered that the sequence is random, because the result is $P \geq 0.01$.

6) The test for examining the discrete Fourier transform aims to detect periodic functions. In the tested series, the aim is to indicate deviation from a random assumption (refers to the highest value set of discrete Fourier transform). The intention is to find out whether the number of the highest values exceeding *95%* is significantly different from the remaining *5%*. This test is invoked through the method ***DiscreteFourierTransform(n)*** where *n* is the length of the bit sample.

```
FFT TEST - COMPUTATIONAL INFORMATION:
 (a) Percentile = 95.000000
 (b) N_l        = 475.000000
 (c) N_o        = 475.000000
 (d) d          = 0.000000
SUCCESS             p_value = 1.000000
```

7) The test for examining the non-overlapping samples is based on the analysis of the frequency of occurrence of all possible *n*-bit patterns where there is no overlapping in the entire examined series. The test detects whether the number of non-overlapping patterns is approximately equal to the number expected for the series of random numbers. This test is invoked through the method ***NonOverlappingTemplateMatching (m,n)*** where *m* is the length of the bits and *n* is the length of the bit sample.

```
NONPERIODIC TEMPLATES TEST - COMPUTATIONAL INFORMATION

LAMBDA = 0.228516   M = 125       N = 8  m = 9  n = 1000
F R E Q U E N C Y - Template W_1 W_2 W_3 W_4 W_5 W_6 W_7 W_8
Chi^2   P_value Assignment Index

1.769891 0.987272 SUCCESS   148
128 Templates = SUCCES, 20 Template = FAILURE
```

8) The test for examining the overlapping samples examines the frequency of occurrence of all possible *n*-bit patterns where overlapping occurs in the entire examined sequence. The test should detect whether the number of overlapping patterns is approximately equal to the number expected for the series of random numbers. The test is invoked through the ***OverlappingTemplateMatching (m,n)*** where *m* is the length of the bits and *n* is the length of the bit sample.

```
OVERLAPPING TEMPLATE - COMPUTATIONAL INFORMATION:

 (a) n (sequence length)      = 1000
 (b) m (block length of 1s)   = 9
 (c) M (length of substring)  = 1032
 (d) N (number of substrings) = 0
 (e) lambda [(M-m+1)/2^m]     = 2.000000
 (f) eta                      = 1.000000

F R E Q U E N C Y
0   1   2   3   4 >=5   P-value Assignment
0   0   0   0   0   0   SUCCESS
```

9) The test for examining the linear complexity determines whether the sequence is sufficiently complex to be considered random. This test is invoked through the method ***LinearComplexity (M, n)*** where *M* is the length of every block, and *n* is the length of the bit sample. The method uses an additional parameter *ε*.

```
LINEAR COMPLEXITY - COMPUTATIONAL INFORMATION:

M (substring length)     = 500
N (number of substrings) = 2
F R E Q U E N C Y
```

```
C0   C1   C2   C3   C4   C5   C6    CHI2    P-value
Note: 0 bits were discarded!
0    0    0    2    0    0    0  2.000106 0.919689
```

Since $P \geq 0.01$, it is considered that the sequence is random.

10) The serial test is used to determine the frequency of all possible overlapping of the *n*-bit sequence in the entire sequence. This test is invoked through the method ***Serial (m, n)*** where *m* is the length of every block, and *n* is the length of the bit sample. Also, here an additional parameter *ε* is used.

```
SERIAL TEST - COMPUTATIONAL INFORMATION:
 (a) Block length    (m) = 16
 (b) Sequence length (n) = 1000
 (c) Psi_m               = 146324.928000
 (d) Psi_m-1             = 97500.608000
 (e) Psi_m-2             = 64765.376000
 (f) Del_1               = 48824.320000
 (g) Del_2               = 16089.088000
SUCCESS             p_value1 = 0.011201
SUCCESS             p_value2 = 0.949014
```

Since P_1 and $P_2 \geq 0.01$, it is considered that the sequence is random.

11) The test for examining the approximate entropy examines the frequency of occurrence of all possible overlapping of *n*-bit patterns in the series. The aim is to compare the frequency of overlapping blocks with the expected results. This test is invoked through the method ***ApproximateEntropy (m, n)*** where *m* is the length of every block, and *n* is the length of the bit sample.

```
APPROXIMATE ENTROPY TEST - COMPUTATIONAL INFORMATION:
 (a) m (block length)     = 10
 (b) n (sequence length)  = 1000
 (c) Chi^2                = 973.830581
 (d) Phi(m)               = -5.747581
 (e) Phi(m+1)             = -5.953813
 (f) ApEn                 = 0.206232
 (g) Log(2)               = 0.693147
Note: The blockSize = 10 exceeds recommended value of 4
SUCCESS             p_value = 0.867027
```

12) The test for examining the random summations determines whether the cumulative sum of the partial sequences that occur in the series which are tested is too big or too small in relation to the expected behavior of this cumulative sum of random sequences. The test is invoked through the method ***CumulativeSums (mode, n)*** where there is *mode = 0* or *mode = 1*, and *n* is the length of the bit sample.

```
CUMULATIVE SUMS (FORWARD) TEST - COMPUTATIONAL INFORMATION:
(a) The maximum partial sum = 14
SUCCESS      p_value = 0.997649
CUMULATIVE SUMS (REVERSE) TEST - COMPUTATIONAL INFORMATION:
(a) The maximum partial sum = 20
SUCCESS      p_value = 0.941731
```

We find out that P_1 and $P_2 \geq 0.01$, more precisely, for the *forward* and the *reverse* test, so it is considered that the sequence is random.

Conclusion and Further Work

From the achieved results, we can say that we have given a few new applications for Catalan numbers, primarily as a generator of pseudo-random numbers in combination with several combinatorial problems, with the purpose of text encryption and decryption. We emphasized the application of movement through the integer network methods in text encryption. Within the theoretical part of the research the tested basic Catalan number properties are mentioned, and the focus was on the bits balance property in the binary notation of Catalan number.

We then provided examples and experimental results of text encryption speed for some combinatorial encryption methods (such as: Ballot Problem, Stack permutations and Balanced Parentheses) in comparison with Lattice Path method.

An experimental study is given that includes specific algorithms for LatticePath encryption and decryption, which are implemented in the Java programming language. The implemented GUI application has all the necessary elements for easy and efficient file encryption and decryption, loading of Catalan-keys, displaying the content of the encoding text, key generation, etc. In the experimental section of this paper, we applied the NIST statistical test battery to assess the quality of the keys based on Catalan number properties.

The proposed methods can be further improved and adapted to the modern approaches in cryptography. Some studies are dealing with the application of number theory in the realization of visual cryptography algorithms and in solving the problem of sharing secrets. The visual cryptography is primarily based on cryptographic methods that perform encryption and data hiding within a set of images, and the reconstruction of the protected or encrypted data is done by a direct, visual examination. Additionally, number theory finds increasing application in the realization of basic cryptographic techniques dealing with secure data exchange.

Beside steganography and visual cryptography, some other suggestions for future work in the field of application of Catalan numbers in cryptography can be given. In [18] the authors give the possibility of applying Catalan numbers in quantum cryptography. In many scientific studies, papers and monographs, when discussing the future of cryptography, quantum cryptography is indicated, which emerged as a result of discoveries in the field of quantum computing [19]. It is very important to mention that quantum cryptography and DNA, in the near future, will present the basis for the protection of confidential documents. According to that, a proposal for future work could relate directly to the application of Catalan numbers in quantum cryptography and the improvement of existing algorithms and methods.

With regard to the fact that cryptography is a very dynamic and widespread field, this paper covers only part of the mathematical concepts and provides a contribution for the application of number theory, in the field of cryptography.

References

[1] Horak, P., Semaev, I., Tuza, I. Z. An application of Combinatorics in Cryptography, Electronic Notes in Discrete Mathematics, 2015, Vol. 49, pp. 31-35

[2] Higgins, P. M. Number Story: From Counting to Cryptography, Springer Science & Business Media, Berlin, Germany, 2008

[3] Lachaud, G., Ritzenthaler, C., Tsfasman, M. A. Arithmetic, Geometry, Cryptography, and Coding Theory, American Mathematical Society, United States, 2009

[4] Saračević, M. Application of Catalan numbers and some combinatorial problems in cryptography (Bachelor's thesis), Faculty of Informatics and Computing, Singidunum University in Belgrade, 2017

[5] Koshy, T. Catalan Numbers with Applications, Oxford University Press, New York, 2009

[6] Duchon, P., On the enumeration and generation of generalized Dyck words, Discrete Mathematics, 2000, Vol. 225, No. 3, pp. 121-135

[7] Amounas, F., El-Kinani, E. H., Hajar, M. Novel Encryption Schemes Based on Catalan Numbers, International Journal of Information & Network Security, 2013, Vol. 2, No. 4, pp. 339-347

[8] Srikantaswamy, S. G., Phaneendra, H. D. A Cryptosystem Design with Recursive Key Generation Techniques, Procedia Engineering, 2012, Vol. 30, pp. 170-173

[9] Pratama, G. M., Tamatjita, E. N. Modifikasi algoritma vigenère cipher menggunakan metode Catalan number dan double columnar transposition, Journal Compiler, 2015, Vol 4, No 1, pp. 31-40

[10] Mašović, S., Saračević, M., Stanimirović, P. Alpha-Numeric notation for one Data Structure in Software Engineering, Acta Polytechnica Hungarica: Journal of Applied Sciences, 2014, Vol. 11, No. 1, pp. 193-204

[11] Saračević, M., Koričanin, E., Biševac, E. Encryption based on Ballot, Stack permutations and Balanced Parentheses using Catalan-keys, Journal of Information Technology and Applications, 2017, Vol. 7, No. 2, pp. 69-77

[12] Saračević, M. Methods for solving the polygon triangulation problem and their implementation (PhD thesis), Faculty of Science and Mathematics, University of Niš, 2013

[13] Stanimirović, P., Krtolica, P., Saračević, M., Mašović, S. Decomposition of Catalan numbers and Convex Polygon Triangulations, International Journal of Computer Mathematics, 2014, Vol. 91, No. 6, pp. 1315-1328

[14] Saračević, M., Hadžić, M., Koričanin, E. Generating Catalan-keys based on dynamic programming and their application in steganography, International Journal of Industrial Engineering and Management, 2017, Vol. 8, No. 4, pp. 219-227

[15] Pund-Dange, S., Desai, C.G., Data Hiding Technique using Catalan-Lucas Number Sequence, Indian Journal of Science and Technology, 2017, Vol. 10, No. 4, pp. 12-17

[16] Aroukatos, N., Manes, K., Zimeras, S., Georgiakodis, F. Data hiding techniques in steganography using Fibonacci and Catalan numbers. In Information Technology: New Generations (ITNG), Ninth International Conference, 2012, pp. 392-396

[17] Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E. A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST Special Publication, USA, 2010

[18] Cohen, E., Hansen, T., Itzhaki, N. From entanglement witness to generalized Catalan numbers, Scientific Reports, 2016, Vol.6, Art. No: 30232

[19] Kościelny, C., Kurkowski, M., Srebrny, M. Modern Cryptography Primer: Theoretical Foundations and Practical Applications, Springer Science & Business Media, Berlin, Germany, 2013