UNIVERSITE DE NICE-SOPHIA ANTIPOLIS

ECOLE DOCTORALE STIC
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H E S E

pour l'obtention du grade de

Docteur en Sciences

de l'Université de Nice-Sophia Antipolis

Mention : Automatique, Traitement du Signal et des Images

présentée et soutenue par

Olga TORNEA

CONTRIBUTIONS TO DNA CRYPTOGRAPHY
APPLICATIONS TO TEXT AND IMAGE SECURE TRANSMISSION

Thèse dirigée par Monica BORDA et Marc ANTONINI

soutenue le 13 Novembre 2013

Jury :

Mircea Giurgiu	Université Technique de Cluj-Napoca	Président
William Puech	Université Montpellier 2, LIRMM	Rapporteur
Pedro Gomez-Vilda	Université Polytechnique de Madrid	Rapporteur
Monica Borda	Université Technique de Cluj-Napoca	Directeur de thèse
Marc Antonini	Université de Nice-Sophia Antipolis	Directeur de thèse
Lionel Fillatre	Université de Nice-Sophia Antipolis	Examinateur
María Rodellar Biarge	Université Polytechnique de Madrid	Examinateur
Romulus Terebes	Université Technique de Cluj-Napoca	Examinateur



















Joint PhD Thesis

Contributions to DNA Cryptography
Applications to Text and Image Secure Transmission

at

Technical University of Cluj - Napoca

&

University of Nice Sophia Antipolis

Eng. Olga Tornea

PhD Advisors:

Prof. Dr. Eng. Monica Borda

Prof. Dr. Eng. Marc Antonini

2013

Acknowledgements

I would like to express my gratitude to my professor and PhD advisor, Monica Elena Borda, for her constant support and guidance during my thesis work. I am grateful for her time, ideas and enthusiasm that helped me a lot since I started my research activity and contributed to the achievements of the thesis.

I would like to thank my professor and PhD advisor, Marc Antonini, for the valuable guidance, advice and support. Discussions we had brought clarity on certain aspects of the thesis and also gave new directions in its development.

I gratefully acknowledge the funding received for my PhD from the project "Improvement of the doctoral studies quality in engineering science for development of the knowledge based society-QDOC" contract no. POSDRU/107/1.5/S/78534, project co-funded by the European Social Fund through the Sectorial Operational Program Human Resources 2007-2013. I am also grateful to the funding received through the I3S Laboratory which was helpful to complete a part of the thesis work.

Thanks to the Data Processing and Security Research Centre for a warm, welcoming and professional environment. Special thanks to Romulus Terebes, Emil Raul Malutan, and Bogdan Belean for their friendly and professional advices.

I would like to express my appreciation to the I3S Laboratory for providing me with a good environment. I am also grateful for the support I received from the scientific collaboration with the team MediaCoding. Special thanks to Jean-Luc Peyrot for his friendly support.

I am grateful to my family and all my friends for their encouragement and support in my work.

Abstract

DNA cryptography is a new and promising field in information security. It combines classical solutions in cryptography with the strength of the genetic material. By introducing DNA into the common symmetric key cryptography, it is possible to benefit from the advantages of the classical cryptosystems and solve some of its limitations. There are different ways how DNA can be used to secure information content. It is about using the biological medium of DNA for storing and hiding data. Secret information can be placed in microscopic size of DNA and hidden among a great amount of other DNA structures. Biomolecular computation is possible with specially designed DNA structures. Valuable parts of this type of computation are self-assembling property of DNA molecules and parallel computations. DNA is considered self-assembling because there are complementary bases in its structure that bind to each other through a standalone hybridization process. Parallel computations occur when hybridization happens at the same time between many different DNA structures. Random numbers can be generated from DNA sequences which can be found in genetic databases in digital form. Genetic databases represent a feasible solution to the One-Time-Pad (OTP) symmetric key generation and transmission problem. The one-time use is ensured due to the great variety of the publicly available, very long (thousands of bases) sequences. Transmission of a very long key is not required because each sequence has a unique identification number in the database and this number, or a combination of this numbers can be sent instead.

Compression along with information security have always been topics of interest because, as technology advances, the amount of data that is desired to be transmitted, stored, or used in real time applications is becoming greater. Some of the encryption schemes can increase the size of the data, or bring unwanted additional computations. These drawbacks can be solved by several techniques to combine compression with encryption in one process or by performing a selective encryption of the data.

Résumé

La cryptographie ADN est un domaine nouveau et prometteur pour la sécurité de l'information. C'est une combinaison des solutions classiques de cryptographie avec les avantages du matériel génétique. En effet, il est possible de bénéficier des avantages des systèmes cryptographiques classiques et de les rendre plus efficaces sur certaines méthodes grâce à l'utilisation de l'ADN. Il y a différentes façons d'utiliser l'ADN pour sécuriser le contenu de l'information. Cette thèse propose deux solutions différentes pour utiliser l'ADN dans la cryptographie : sous sa forme biologique ou alors sous forme numérique.

D 'une part, l'ADN biologique peut être utilisé pour le stockage et pour cacher des données à l'intérieur de celui-ci. L'information secrète est placée dans une molécule de l'ADN et caché parmi d'autres molécules d'ADN. Le calcul biomoléculaire est possible grâce à des structures synthétiques d'ADN. Les éléments importants pour ce type de calcul sont la propriété d'auto-assemblage des molécules d'ADN et le parallélisme du processus grâce à l'hybridation.

D'autre part, les nombres aléatoires peuvent être générés à partir de séquences numériques d'ADN. En effet, les bases informatiques de données génétiques contiennent des séquences d'ADN sous forme numérique. Ils représentent une solution pour la génération et la transmission des clés OTP (One-Time-Pad) symétriques. La transmission d'une très longue clé de cryptage n'est pas nécessaire, car chaque séquence possède un numéro d'identification unique dans la base de données. Ce numéro, ou une combinaison de ces numéros, peut alors être transmis.

Enfin, la sécurité et la compression sont très importantes lors de la transmission et du stockage des données informatiques. Cependant, la plupart des systèmes de cryptage peuvent augmenter la taille des données, ou encore augmenter la complexité calcul. Ces inconvénients peuvent être résolus en combinant la compression de données avec le cryptage dans un seul processus ou en effectuant le cryptage sélectif des données.

Table of Contents

List of Figures ..... **x**

List of Tables ..... **xiii**

List of Abbreviations ..... **xiv**

Chapter 1 Introduction **1**

1.1 Motivation and Objectives of the Thesis 1

1.2 Thesis Structure 3

Chapter 2 State of the Art **5**

2.1 Elements of Cryptography 5

2.1.1Conventional Cryptography (Symmetric) 6

2.1.1.1 Stream and Block Ciphers 8

2.1.2 Public Key Cryptography (PCK) 9

2.1.3 One-Time-Pad 10

2.1.4 Cryptanalysis 12

2.1.5 Key Management 13

2.1.6 Cryptosystems for Duplex Transmission and Storage 13

2.2 Data Compression 15

2.2.1 Data Compression Chain 16

2.2.2 Rate Distortion Theory 19

2.2.3 Types of Data Compression 20

2.2.4 Classification of the Multimedia Compression – Encryption methods 21

2.3 DNA Cryptography 24

2.3.1 Molecular Biology Principles 25

2.3.1.1 DNA Structure 26

2.3.2 Cryptography at the Molecular Level 27

2.3.2.1 Techniques for DNA Analysis 27

2.3.2.2 Molecular Storage and Computations 29
2.3.3 Digital DNA in Cryptography 30
2.4 Performance Evaluation Aspects 31
2.4.1 Computational Complexity 31
2.4.2 Security Level 32
2.4.3 Compression Ratio 35
Chapter 3 DNA Indexing Cipher 37
3.1 Principle of the Algorithm 38
3.2 Design and Implementation Details 40
3.3 Performance Evaluation 44
3.3.1 Computational Complexity of the Algorithm 44
3.3.2 Security Level of the Algorithm 48
3.3.3 Comparison with another cipher of similar principle 51
3.4 Modifications of DNA Indexing Cipher for Better Compression Ratio and Security 52
3.4.1 Improvements of Direct Encryption 52
3.4.1.1 Change of Indexes in Key Dictionary 54
3.4.1.2 Homophonic Substitution 55
3.4.1.3 Encryption and Compression through Longest Common Subsequences in DNA Sequence 56
3.4.2 Joint Compression – Encryption 58
3.4.2.1 Analysis Block - Bit Allocation 60
3.4.2.2 Entropy Coding of the Key Dictionary 603
3.4.2.3 Experimental Results of R(D) Functions 65
3.4.2.4 Comparison of Compression Efficiency between different Versions of the Algorithm 68
3.5 Contributions 70

Chapter 4 Generation of Random Sequences using Genetic Databases and proposal of OTP Cryptosystems 73
4.1 DNA Based Random Sequence Generation Method 74
4.1.1 Distribution of the DNA Sequence 74
4.1.2 Transformation from DNA to Binary 75
4.1.3 Access to the DNA Sequence 75
4.1.4 Length of the DNA Sequence 78
4.1.5 Format of the Secret Key 81
4.1.6 Overview of the Method and its Advantages 82
4.2 OTP Encryption Systems for Transmission and Storage 83
4.2.1 OTP Cryptosystem for Duplex Transmission Based on DNA-RS 83
4.2.2 OTP Cryptosystem for Storage, based on DNA-RS key 86
4.3 Performance Evaluation 87
4.3.1 Computational Complexity of the Algorithm 87
4.3.1.1 Time Evaluation Functions 87
4.3.1.2 Experimental Measurements of the Computational Time 89
4.3.2 Security Level of the Algorithm 96
4.4 Contributions 99
Chapter 5 Design of the DNA Vernam Cipher at the Molecular Level 101
5.1 Principle of the Algorithm 102
5.2 Design and Implementation Details 105
5.2.1 Design of DNA Molecule Structure 105
5.2.2 DNA Molecule Structure Implementation 108
5.2.3 Laboratory Utility 110
5.3 Performance Evaluation 111
5.4 Contributions 111

Chapter 6 Conclusions **113**
6.1 Overview of the Contributions 114
6.2 Future Works 116
Scientific Activity **117**
Bibliography **119**
Appendix ..... **a**

List of Figures

2.1 Symmetric Encryption Model 7
2.2 a) Block Cipher b) Stream Cipher 8
2.3 Vernam Cipher 10
2.4 Cryptosystem for Duplex Transmission 14
2.5 Cryptosystem for Storage 15
2.6 General scheme of compression process 16
2.7 Chain of compression encoding – decoding processes 17
2.8 Rate – Distortion Function 19
2.9 Schemes of Complete and Partial-Selective Encryption 22
2.10 Encryption of the DC and AC coefficients of a DCT block 23
2.11 Cell and its Genetic Material 25
2.12 Nucleotide Structure 26
2.13 DNA Strands formed by nucleotide and hydrogen bonds 26
2.14 DNA Structure 26
2.15 Hybridization between 2 Complementary Strands of DNA a) beginning of the process b) final product of hybridization 28
2.16 Few Wang tiles 28
2.17 DNA Triple-Helix Tile [Bor11] 28
2.18 Hybridization between DNA Tiles that can be used for Arithmetic Operations 30
2.19 Histograms of the Plaintexts and Ciphertexts 33
3.1 Chromosomal Sequences from a Genetic Database 38
3.2 Encryption Process of the DNA Indexing Algorithm 39
3.3 Use Case Diagram 41
3.4 Class Diagram 42
3.5 Object Diagram 43
3.6 Sequence Diagram 43
3.7 Growing Rate of the Key Table Computation Runtime 47
3.8 Growing Rate of the Encryption Runtime 47

3.9 Growing Rate of the Decryption Runtime 47
3.10 Examples of Plaintext, Ciphertext and Their Statistical Measurements 49
3.11 Key Dictionary: for each byte there is a range of integer substitution values; their positions may be used as new substitution values 54
3.12 Distribution of the Encoded Data Bitstream a) tiff, b) jpeg 55
3.13 Number of Substitution Values for each Byte of the Plaintext 56
3.14 Distributions of the a) Plaintext and b) Ciphertext Values 56
3.15 Example of Encryption with LCS in DNA Sequences 57
3.16 Joint Compression – Encryption Scheme 58
3.17 Scheme of the Joint Compression – Encryption Method 59
3.18 Rate – Distortion Curve According to the Quantization Step 60
3.19 Lagrange Multiplier and Quantization Steps Values 62
3.20 D(R) Curves according to the Number of Substitutions in Key Dictionary 62
3.21 Huffman Coding of the Key Dictionary 63
3.22 Change in the Bitrate due to the number of substitutions in the KD 64
3.23 Histogram and Approximation of the Generalized Gaussian Distribution 65
3.24 Transform of the Test Image; Numeration represents Subband Images 65
3.25 R(D) Curves of the Subband Image 0 66
3.26 R(D) Curves of the Subband Image 2 67
3.27 R(D) Curves of the Subband Image 4 67
3.28 Number of Substitutions per Value allocated empirically for each Image Subband 69
3.29 PSNR –R Curves of the 3 Compression – Encryption Algorithm's Versions 70
4.1 Requesting Genome Information for Zea Mays (corn) Species 76
4.2 Genome Information about Zea Mays Species 76
4.3 Chromosomes of the Zea Mays 77
4.4 DNA Sequence in FASTA Format and the Download Options 77
4.5 Format of the Secret Key a) 2 bits specifing the method a – 00, b – 01, c – 10, d – 11, b) 3 bits specifing the number of transmitted IDs, c) the actual secret key formed of 1 – 8 IDs, each of 8 bytes 81
4.6 Examples of DNA Sequences in GenBank Format and their IDs 82

4.7 Example of Secret Key Format 82
4.8 Scheme of OTP Cryptosystem for Duplex Transmission based on DNA-RS Key..... 83
4.9 OTP Cryptosystem for Storage, based on DNA-RS Key a) Encryption Side of Writing Unit b) Decryption Side of Reading Unit 86
4.10 Growing Rate of the File Reading Runtime..... 90
4.11 Growing Rate of Shifting Procedure Runtime 91
4.12 Growing Rate of Multiplexing Procedure Runtime 92
4.13 Growing Rate of the Time Spent to Transform from DNA Alphabet to Binary 93
4.14 Growing Rate of the Time Spent to Perform Binary XOR 94
4.15 Comparison between Symmetric Ciphers Computational Time 95
4.16 Histograms of the Plaintext Bitmap Image and Corresponding Ciphertext 96
4.17 Histograms of the Plaintext Tiff Image and Corresponding Ciphertext..... 96
5.1 Amounts of DNA Tiles for Plaintext Bitstream Assembling 102
5.2 Plaintext Bitstream of DNA Molecules obtained by Hybridization 103
5.3 DNA Structures designed for Key Bitstream 103
5.4 XOR Operation between the Plaintext and the Key obtained by Hybridization 104
5.5 Hybridization between Plaintext and Key DNA Molecules..... 104
5.6 Ciphertext composed of the DNA Molecules 105
5.7 Holliday Junction 106
5.8 DNA Double Crossover Molecules 106
5.9 DNA Origami..... 107
5.10 a) DAE Molecule b) modified DAE Molecule c) Schematic View..... 108
5.11 Schematic DNA Structure for Binary “1” 108
5.12 Atomic Force Microscope..... 110

List of Tables

2.1 Conversion Table from Binary to DNA 29
3.1 Pseudo code for basic operations of the DNA Indexing cipher 45
3.2 Measurements of the key-table computation runtime 45
3.3 Measurements of the encryption runtime 46
3.4 Measurements of the decryption runtime 46
3.5 Plaintext and ciphertext entropy measurements 48
3.6 Plaintext and ciphertext correlation coefficients 49
3.7 Rate – Distortion Values of Subband Image 0 66
3.8 Rate – Distortion Values of Subband Image 2 67
3.9 Rate – Distortion Values of Subband Image 4 67
3.10 PSNR – R Values of the 3 Compression – Encryption Algorithm's Versions 69
4.1 Conversion table from DNA to binary 75
4.2 DNA sequences lengths corresponding to Zea mays (corn) chromosomes 78
4.3 DNA sequences lengths corresponding to cat chromosomes 78
4.4 DNA sequences lengths corresponding to human chromosomes 78
4.5 Examples of different input data (obtained by measuring real files) 79
4.6 Example of different cleartext messages, the corresponding DNA key length (bp), and respectively the secret key information 80
4.7 Time measurements of the DNA sequence reading from the file 89
4.8 Time measurements of the DNA sequence shifting 90
4.9 Time measurements of the DNA sequences multiplexing 91
4.10 Time measurements of the DNA to binary transformation 93
4.11. Time measurements of the binary XOR operation 94
4.12 Time measurements of the symmetric encryption algorithms 95
4.13 Plaintext and ciphertext entropy measurements 97
4.14 Plaintext and ciphertext correlation coefficients 97

List of Abbreviations

AES - **A**dvanced **E**ncryption **S**tandard

AFM - **A**tomic **F**orce **M**icroscope

CBC - **C**ipher **B**lock **C**haining

CC - **C**orrelation **C**oefficient

CCR - **C**hanges in **C**ompression **R**atio

CFB - **C**ipher **F**eed**b**ack Mode

CR - **C**ompression **R**atio

DCT - **D**iscrete **C**osine **T**ransform

DES - **D**ata **E**ncryption **S**tandard

DFT - **D**iscrete **F**ourier **T**ransform

DNA - **D**eoxyribose **N**ucleic **A**cid

DPCM - **D**ifferential **P**ulse **C**ode **M**odulation

DWT - **D**iscrete **W**avelet **T**ransform

EZW - **E**mbedded **Z**erotree **W**avelet

FIPS - **F**ederal **I**nformation **P**rocessing **S**tandard

GIF - **G**raphics **I**nterchange **F**ormat

HGP - **H**uman **G**enome **P**roject

IDEA - **I**nternational **D**ata **E**ncryption **A**lgorithm

JPEG - **J**oint **P**hotographic **E**xperts **G**roup

LCS - **L**ongest **C**ommon **S**ubsequence

MHT - **M**ultiple **H**uffman **T**ables

MPEG - **M**oving **P**icture **E**xperts **G**roup

MSE - **M**ean **S**quared **E**rror

NIST - **N**ational **I**nstitute of **S**tandards and **T**echnology

OFB - **O**utput **F**eed**b**ack Mode

OTP – **O**ne **T**ime **P**ad

PCK - **P**ublic **K**ey **C**ryptography

PGP - **P**retty **G**ood **P**rivacy

PNG - **P**ortable **N**etwork **G**raphics

RSA - Ron **R**ivest, Adi **S**hamir, and Leonard **A**dleman

PSNR - **P**eak **S**ignal to **N**oise **R**atio

SFM - **S**canning **F**orce **M**icroscope

UML - **U**nified **M**odeling **L**anguage

Chapter 1

Introduction

1.1 Motivation and Objectives of the Thesis

Interest in information security existed since ancient times and it is present in our modern life. Techniques to protect information are evolving together with the progress in information technology. Secret information was hidden in books or paintings; it appeared in form of the unintelligible text. Some of the first ciphers based on substitution of letters in written text were Polybius and Cesar ciphers. There are two directions in information protection: cryptography and steganography. These two sciences manipulate information in order to change its meaning or hide its existence. Computer age brought a different interpretation of information and new directions in development of ciphers and cryptographic protocols. Computational power offered the possibility to build new and strong algorithms in cryptography, but it was also a strong tool used by cryptanalysts to break the cryptosystems. This means that the subject of finding new and powerful ciphers is always of interest and new directions in cryptography are explored.

Cryptography provides a range of features for information security. The main aspects treated by cryptography are: confidentiality, data integrity, authentication, and non-repudiation. The objectives of this thesis were to concentrate on the confidentiality part and to find new methods (ciphers) to ensure privacy through the use of DNA.

DNA cryptography consists in the use of genetics and biomolecular computation and it is one of the newest directions in cryptography. Genetic material such as DNA can be used as a vast storage space. This idea is inspired from the fact that DNA is a natural carrier of information which is encoded by a 4-letter alphabet: A, C, G, and T. This alphabet can be easily transposed into the binary alphabet (A – 00, C – 01, G – 10, T - 11). Therefore DNA can be used as a storage media for any kind of information. The property of hybridization between complementary DNA nucleotides bases (A-T, C-G) is exploited in the biomolecular computing field as a central process of computations. It is a natural process that appears between complementary DNA strands of nucleotides and that's why it is

named a self-assembling process. DNA computing started with Adleman's research [Adl94], while some basic directions of DNA cryptography are described in [GLR04].

Considering the fact that DNA cryptography is a novel domain, one of the objectives of the thesis was to find and define different ways in which it can be applied in information security. Three main directions of using DNA in cryptography were found: storage space, computational power, generation of cryptographic keys from its long sequences. There can be two working environments with DNA: at molecular level, in a laboratory, with biological DNA and with digital DNA using available genetic databases. Different techniques to manipulate DNA at molecular level were analyzed. A conversion of its alphabet to binary was established. Three novel encryption methods based on DNA were designed, developed, tested and improved in this work.

Genomes sequencing and their appearance in the form of electronic databases was a big step for the growth in the genomic research domain [Lan11]. The benefits of the digital genomic databases can be extended also to the information security domain. For example, these databases can be used for the practical application of the one-time pad (OTP) encryption scheme. The OTP properties correspond to the characteristics of the unbreakable encryption system defined by Claude Shannon [Sha49].

A variety of possible genes and chromosomes from different organisms are good materials for creation of random, non-repeating and for only one use pads. Nowadays there are electronic databases of whole sequenced genomes from different organisms including human, dog, mouse, frog, fruit fly, social amoeba and many others. These sequences can be accessed from public genetic databases [wncbi] in different formats.

A research of how compression is combined with encryption was performed. Compression and protection of the multimedia data have always been topics of interest because, as technology advances, the amount of data that is desired to be transmitted, stored, or used in real time applications is becoming greater. Multimedia files: text, audio, image, or video can be very large and there are always concerns about their secure and fast transmission. One of the designed DNA ciphers was modified into a joint compression – encryption scheme. This modification was performed because the usual direct encryption was doubling the size of the data.

This thesis approaches the problem of OTP encryption scheme. It offers a strong security, but is not used in practical applications because of the difficulty to generate and distribute a pure random and very long key bitstream. DNA properties were explored to solve these drawbacks. In one of the designed ciphers it was proposed to use the hybridization

process to generate a random binary sequence for the secret key. In the other two ciphers genetic databases were used to create a secret key and also to easily distribute it.

1.2 Thesis Structure

This thesis is organized in 6 chapters. First chapter is an introduction to the thesis subject and objectives. Second chapter offers a brief theoretical background and state of the art for the scientific fields involved in this work. The next three chapters describe contributions: design, development, and analysis of the proposed DNA ciphers. The final chapter, conclusions, makes an overview of the thesis contributions and presents possible future works.

Chapter 2 presents state of the art of the involved fields: cryptography, compression, molecular biology, and bioinformatics. Due to the diversity of the involved fields, a brief theoretical background is provided for them along with the important references of books and papers relevant to the described domains.

Chapter 3 presents one of the thesis contributions: DNA Indexing cipher. It is a symmetric encryption algorithm that uses DNA sequences from genetic databases for the secret key. In the first part of the chapter the principle of the algorithm is described. Details of its design and implementation are presented next. Performances of the algorithm were tested and analyzed using different metrics. Results of computational time, security level, comparison with another algorithm, and compression efficiency are presented in this chapter. It was observed that the bitrate increases twice after encryption. In order to solve this drawback of the algorithm certain improvements were proposed and tested. Direct encryption was compared to the joint compression – encryption scheme. A smaller bitrate was obtained in the case of joint compression – encryption for the same distortion of the data.

Chapter 4 describes another contribution of OTP cryptosystems based on genetic databases. The method to generate random binary sequences from DNA described in this chapter solves the problem of generation and distribution of the random long secret keys of OTP ciphers. Schematic views and protocols for OTP encryption systems for transmission and storage are presented next. The encryption method of the cryptosystem is Vernam cipher and the secret key is a bitstream generated from the DNA sequences.

Advantage of the DNA based random sequence generation is that a binary random sequence of any length can easily be generated from the public or private genetic databases. An unlimited number of distinct random sequences can be obtained by

multiplexing, shifting, or concatenating sequences from different species. The major drawback of the OTP cryptosystem is the key transmission. This problem doesn't occur in the DNA-RS generation because the secret key can be communicated through the short identification numbers of the DNA sequences used at its construction. The advantage of the OTP cryptosystem is its high security level [Sha49], [Ver26]. Storage of the encryption keys implies keeping the DNA identification numbers and method of key construction. The benefit is that only a smaller number of bits need to be stored instead of the entire encryption key.

Chapter 5 presents contributions to DNA Cryptography at molecular level. Design of Vernam cipher with DNA structures is proposed. A research of different DNA structures and proposal of a structure for this cipher is presented. Details of algorithm steps and structure implementations are given. The second part of this algorithm is intended for laboratories with special equipment for biomolecular experiments.

Chapter 6 concludes the thesis work. An overview of contributions is presented here. Parts of the work that can be continued are presented in future works. After the conclusion chapter a bibliography is given and a list of publications of this work. In appendix some of the most relevant papers are presented.

Chapter 2

State of the Art

Contents in Brief

2.1 Elements of Cryptography 5
2.2 Data Compression 15
2.3 DNA Cryptography 24
2.4 Performance Evaluation Aspects 31

In this chapter the basic notions of cryptography, compression, and use of genetic material in information technology are presented. Basic contributions to each field are introduced along with the significant and relevant bibliographical references.

Subchapter 2.1 presents the central elements of cryptography with emphasis to the most relevant aspects for this thesis. Subchapter 2.2 is about data compression key concepts and basic knowledge of joint compression – encryption. Elements of biology and two main types of DNA cryptography are introduced in subchapter 2.3. In subchapter 2.4 three main metrics used to evaluate the performance of the proposed encryption and compression methods are presented.

2.1 Elements of Cryptography

Cryptography is the science of techniques and protocols intended to ensure information security. *Cryptanalysis* is the art of analyzing and breaking secure communication; it consists in attacking cryptographic methods. Cryptography is practiced by *cryptographers* and cryptanalysis is practiced by *cryptanalysts (attackers)*. *Cryptology* encompasses the branches of cryptography and cryptanalysis [Sch96].

The purpose of cryptography is to provide a range of features for information security. The most important of them are [Den07]:

- *Confidentiality (privacy)* provides the secrecy of information content. It transforms the meaningful data into senseless message. *Ciphers* are the cryptographic algorithms used to guarantee this characteristic.
- *Data integrity* means its protection from unauthorized access and alteration. Possible changes in the original data are insertion and deletion. Integrity is achieved through cryptographic hash functions.
- *Authentication* is the identification of the transmitted information and of the parts engaged into a communication.
- *Non-repudiation* means to respect the obligations of a contract. This property can be obtained by using signatures.

Data that has perceptual meaning for us is called *clear text* or *plain text*. Transformation of the plaintext into an unreadable file is called *encryption*. Encrypted plaintext becomes *ciphertext*. In order to obtain the original cleartext, the process of *decryption* is applied on the ciphertext data. These two processes of encryption and decryption are compositional parts of a *cryptographic algorithm* named *cipher*. A cipher is applied on the data together with a *secret key*. *Cryptosystem* includes encryption, decryption, *key generation* algorithm and all the necessary protocols to ensure secure communication [PGP04]. The *key space* is the number of elements in the alphabet raised to the power of the key length. Key space is important for cryptanalysis; it gives the number of all the possible keys that can be as well the secret key.

History of cryptography and steganography is presented in [Kah96], [Wat01] and [PF07]. Introduction notions and fundamental knowledge of information security is well covered in [Sch96] and [Bor11]. A comprehensive survey about security of data communications and networks is W. Stallings book [Sta11]. Another reference which presents techniques and algorithms of greatest interest is [MOV96]. A broad overview of computer security is presented in [NIST95].

2.1.1Conventional Cryptography (Symmetric)

Symmetric cryptography is also named conventional or classical because it was the first and only type of encryption until the 1970s, when the public key cryptography was introduced. Symmetric ciphers use the same key for encryption and decryption. This key must be secret and communicated through a secure channel to all parties involved in a communication (Fig. 2.1).

The most common operations used in this type of cryptography are *substitution* and *transposition*. Substitution consists in giving a different value to each plaintext word (one to

many bits). The transformed plaintext words become ciphertext values. Transposition, also named permutation, means interchanging the positions of the plaintext words.

First ciphers appeared in the years BC in ancient Egypt, Greece, and Rome. They were designed to secure written text and based on substitutions and transpositions of the alphabet letters. Well known examples for the earliest cryptography are Caesar and Polybius ciphers. In the XV century the polyalphabetic ciphers, like Vigenère, appeared. They were resistant to the frequency analysis techniques to which the ancient ciphers where vulnerable [Kah96].

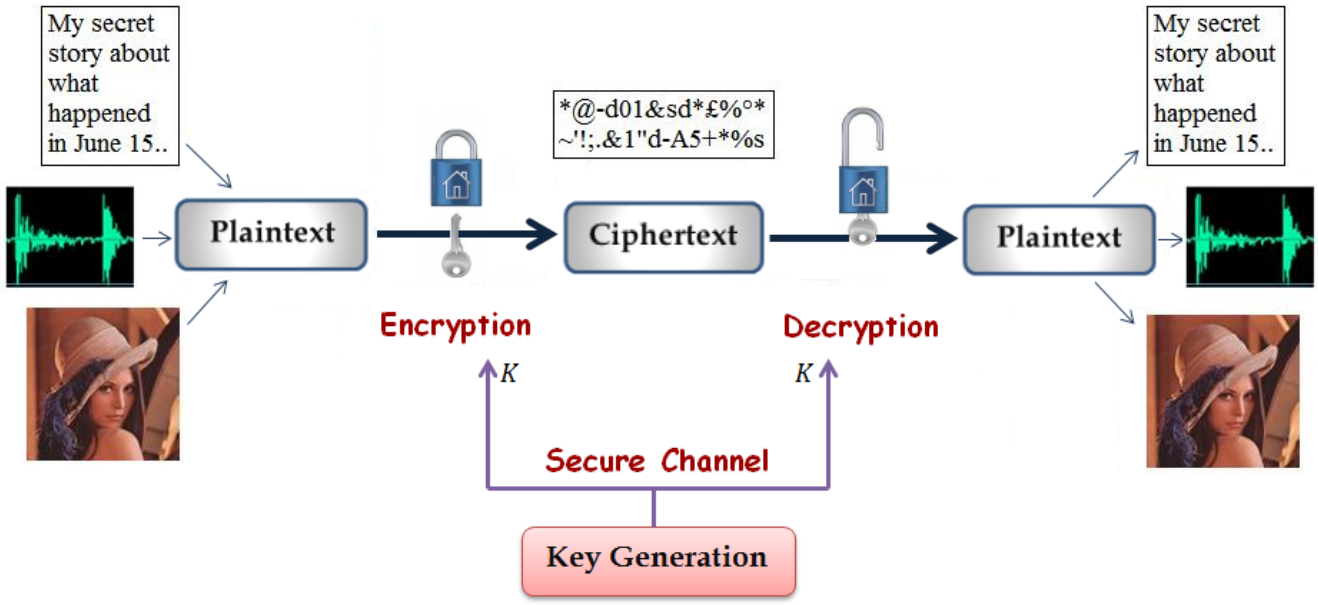


Figure 2.1 Symmetric Encryption Model

$$K_E = K_D = K \tag{2.1}$$

$$E_K(P) = C \tag{2.2}$$

$$D_K(C) = D_K\big(E_K(P)\big) = M \tag{2.3}$$

where

C - ciphertext, P – plaintext, E – encryption, D – decryption, K - key

Evolution in computer science brought a different level of cryptography. Security started to be applied on the binary stream, rather than on alphabet letters and consequently on any type of data: text, image, video, etc. Cryptanalysis and breaking of earliest ciphers became faster and easier with a new computational force. In 1972 NIST launched a

computer security program to introduce a tested, certified cryptographic algorithm and use it as standard in information security [NIST01]. In 1976 the Data Encryption Standard (DES), an algorithm developed by IBM in 1970, was adopted as a federal standard by NIST. Due to the increase in computing power in the 1990s, the vulnerability of the DES algorithm became its short encryption key. It was substituted by 3DES, which is the DES algorithm applied 3 times. Security of the 3DES was strong, but it was considered slow in implementation. In 2002 the AES encryption algorithm was adopted as federal standard by NIST and it is still in use.

2.1.1.1 Stream and Block Ciphers

Symmetric algorithms can be of 2 types: *stream ciphers* and *block ciphers*. In stream ciphers encryption is performed on small units of plaintext, like one bit or one byte at a time. Block ciphers operate on larger message units, like 64-bit blocks in DES.

The basic distinction between these two types of ciphers is time dependency or "memory". Let's consider the plaintext message $M = (m_1, m_2, \ldots m_n)$ as a sequence of blocks and corresponding to it ciphertext $C = (c_1, c_2, \ldots c_n)$ a sequence of blocks after encryption. Block cipher is memoryless; it transforms independently each message unit into the ciphertext unit using an encryption function and a secret key (Fig. 2.2, a). Stream cipher transforms a plaintext unit m_j into the ciphertext unit c_j using a time dependent key k_j and an encryption function (Fig. 2.2, b). Vernam cipher based on the one time pad principle is one of the most known stream ciphers and it will be presented in subsection 2.1.3.

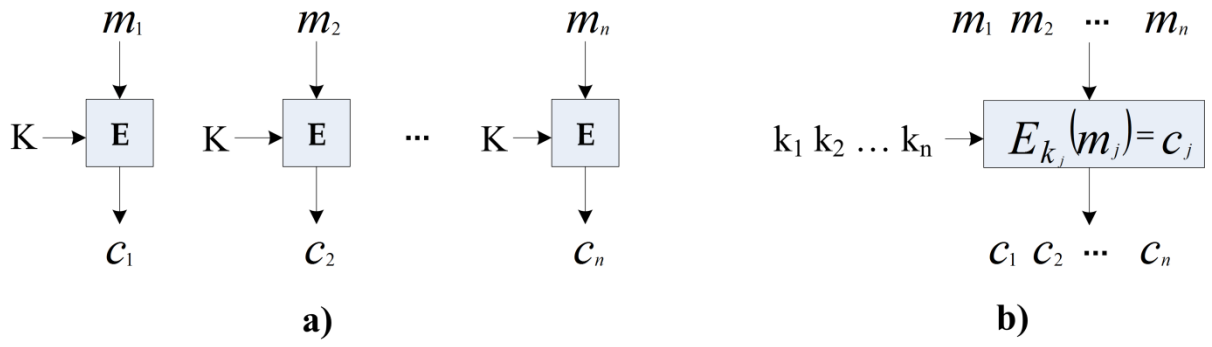


Figure 2.2 a) Block Cipher b) Stream Cipher

In general, two identical plaintext units encrypted with stream cipher don't have the same ciphertext output because the key sequence is not the same at different moments of encryption. In case of block cipher two identical plaintext blocks may result after encryption in same ciphertext which brings vulnerability to the known plaintext attack and vulnerability to insertion or deletion of some ciphertext (subchapter 2.1.4). In order to

solve this problem a certain chaining or feedback can be introduced between the block encryptions. Some of the possible modes to do this are: *Cipher Block Chaining (CBC)*, *Cipher Feedback Mode (CFB)*, *Output Feedback (OFB)*, etc. [Bor11].

2.1.2 Public Key Cryptography (PCK)

The basic principle of public key (asymmetric) cryptography is to use a pair of public and secret keys. In such systems encryption is performed with a public key and decryption with its private key pair. Encryption (public) key is different from the decryption (private) key, but they are related.

$$K_E \neq K_D \tag{2.4}$$

This pair of keys is generated using mathematical functions and in some algorithms, like RSA, one of these two keys can be made public and used for encryption, which means that the other one must be kept secret and used for decryption. It is computationally unfeasible to derive the secret key knowing its public pair and encryption algorithm.

Appearance of PCK solved the key management problem which is transmission of secure information. Symmetric ciphers are fast, they are used to encrypt large amounts of data, but before using them, transmission of the secret key through a secure channel needs to be done. This part can be solved by asymmetric algorithms which are not so fast, but they can encrypt a secret key of a symmetric algorithm without previous key exchange.

The concept of public key cryptography was first introduced by Diffie and Hellman [DH76]. One of the most known public key algorithms is RSA invented by R. Rivest, A. Shamir, and L. Adleman [RSA78]. The strength of this cipher is given by the computational complexity of factoring large numbers. Another strong asymmetric cipher is ElGamal created by T. ElGamal [ElG85]. This algorithm, like Diffie-Hellman, is based on the difficulty to solve the problem of discrete logarithms. Digital Signature Algorithm (DSA) was declared a standard (DSS) by NIST, specified in FIPS and attributed as invention to David Kravitz [FIPS94].

Public-key cryptography is based on mathematical functions and much of its theory on number theory [Sta11]. The security of asymmetric ciphers relay on finding large prime numbers, at least 100 decimal digits. Multiplication of large prime numbers is computationally simple, but finding the original numbers given the final product is a time and resource consuming task.

The following mathematical formulas describe the principle of the RSA cipher. Let's consider two large prime numbers p and q and their product to be n.

$$n = p * q \tag{2.5}$$

Next is computation of the Euler's totient, finding a coprime integer e to it and integer d such that:

$$\varphi(n) = (p-1)(q-1) \tag{2.6}$$

$$d * e \bmod \varphi(n) = 1 \tag{2.7}$$

$$C = P^e \bmod n \tag{2.8}$$

$$P = C^d \bmod n \tag{2.9}$$

This algorithm offers the public key *(n, e)* for encryption and the private key *(n, d)* for decryption (*d* is secret). Integer number *n* is the product of two primes (2.5), while *e* and *d* mathematically derive from *n* (2.7). *C* is the ciphertext and *P* is the plaintext. Encryption (2.8) and decryption (2.9) are computationally simple processes, knowing the keys (*n, e* and *n, d*) [BB01].

2.1.3 One-Time-Pad

One-Time-Pad (OTP) is a principle of key generation applied on the stream ciphering method which offers a perfect secrecy [Sch96], [Sta11], [Sha49], if all the requirements are fulfilled. It is also considered that this scheme is unbreakable in theory, but difficult to realize in practical applications [Sch02], [Sta11].

A part of the OTP encryption method appeared in 1917, in Vernam teletype cipher [Ver26]. Vernam defined a stream cipher where a key, stored on a punched tape, was combined, applying XOR operation, character by character with the plaintext message producing a ciphertext. J. Mauborgne added that the key string of bits should be truly random and used only at one single encryption-decryption process and then Vernam cipher became OTP co-invented by G. Vernam and J. Mauborgne. The principle of this cipher is shown in Fig. 2.3 and it can be expressed by the formula (2.10):

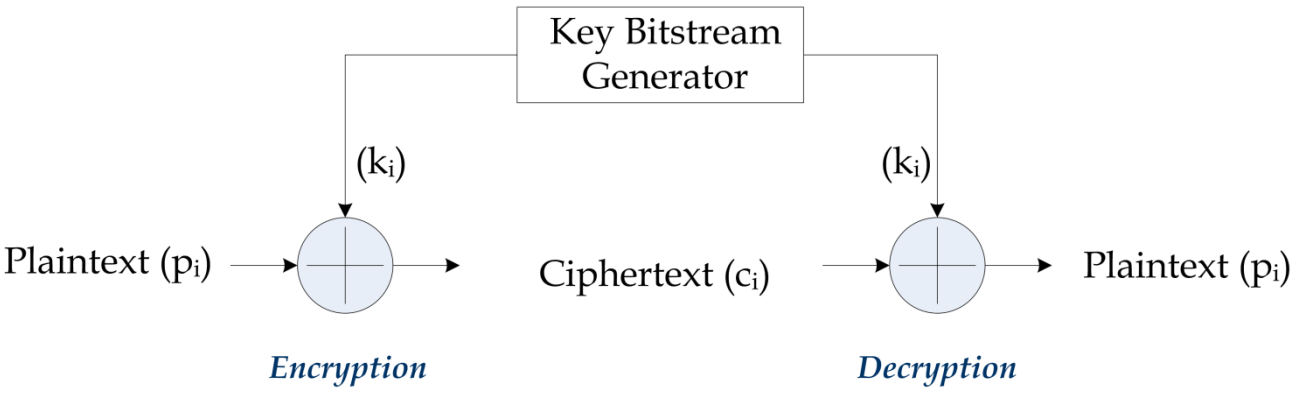


Figure 2.3 Vernam Cipher

$$c_i = p_i \oplus k_i \tag{2.10}$$

where

p_i =is the ith bit of the plaintext bitstream

k_i = is the ith bit of the key bitstream

c_i = is the ith bit of the ciphertext bitstream

$\oplus$ = is the exclusive-or (XOR) operation

Claude Shannon described in his work the principles for perfect secrecy [Sha49]. These characteristics for the unbreakable encryption system are the same with the OTP properties. They can be summarized as the following constrains on the encryption key:

- it must be truly random
- at least as large as the plain-text
- never reused in whole or part
- kept secret

In fact there are contradictory debates about OTP encryption scheme. For example, Bruce Schneier, a famous cryptographer and writer, said about OTP in 1996 the following:

"Believe it or not, there is a perfect encryption scheme."

"Assuming an eavesdropper can't get access to the one-time pad used to encrypt the message, this scheme is perfectly secure."

"A random key sequence added to a nonrandom plaintext message produces a completely random ciphertext message and no amount of computing power can change that." [Sch96]

In 2002 he wrote about OTP from a quite different perspective:

"One-time pads are useless for all but very specialized applications, primarily historical and non-computer."

"They replace a cryptographic problem that we know a lot about solving -- how to design secure algorithms -- with an implementation problem we have very little hope of solving. They're not the future. And you should look at anyone who says otherwise with deep and profound suspicion." [Sch02]

What is sure is that this encryption scheme received a lot of attention and in this work (Chapter 4) a method will be presented that solves the key generation and distribution problems, the biggest drawbacks of the OTP method.

2.1.4 Cryptanalysis

The goal of the cryptanalysis is to break the cryptosystem by exploiting its weaknesses. Revealing the secret key is equivalent to a total break of the cipher. Finding a way to recover the plaintext from the ciphertext without knowing the secret key is considered a partial break [Knu94].

The straightforward way to find the secret key is through a *brute-force attack* which consists in trying all the possible keys. By Kerckhoff's principle and Shannon's maxim the encryption algorithm is supposed to be known and the strength of the cryptosystem is in the key. Thus the key space should be large enough making the brute force attack infeasible. Cryptanalysis techniques are used to find the key in a less number of tries by finding some flaws in cryptosystem. Cryptanalytic attacks can be classified by the amount of information known by the attacker [Koh08], [Sta11]:

- *Ciphertext-only attack*: the encryption algorithm and ciphertext are available for cryptanalysis.
- *Known plaintext attack*: the encryption algorithm, ciphertext, and a portion of plaintext corresponding to the ciphertext are known to cryptanalyst.
- *Chosen-plaintext (chosen-ciphertext) attack*: the attacker is able to encrypt or decrypt any information of his choice and obtain significant plaintext - ciphertext pairs. It happens when the encryption system with the embedded secret key is available to the cryptanalyst.
- *Adaptive chosen plaintext or ciphertext attack*: is the adaptive version of the attack presented above. The attacker can adapt the plaintext based on the results obtained from previous encryptions.
- *Related key attack:* is based on the relation between different keys known to the attacker.

When the cryptanalyst has only the ciphertext, in order to reduce the number of attempts through the brute-force attack, the ciphertext data is analyzed and usually some statistical tests are applied on it. Knowledge about data types, such as English or other language text, image, source code, etc. gives the possibility to perform known plaintext attack. Certain patterns in the text, headers that appear at the beginning of the file, etc. offers portions of plaintext – ciphertext pairs corresponding to the secret key.

2.1.5 Key Management

Key management comprises a set of methods to support aspects related to keys in cryptographic purposes. Important features related to key management are: generation, distribution, backup, and storage of the keys involved in secure communication. Key management is an essential part of cryptography due to the fact that message secrecy and all the secure communication are compromised if some information about the key is revealed [MOV96].

Key management in public – key cryptography is easier than in symmetric – key systems. In order to encrypt data, only public key is required. Authentication of the public key user is necessary, but not the key secrecy. Key exchange for symmetric – key systems can be performed in different ways as described below.

1. *A* and *B* meet to exchange the key(s)
2. Having first secret key exchanged, *A* and *B* can securely transmit other keys using the first one
3. Third party *C* can send secret keys through a secure channel to *A* and *B*
4. Symmetric key exchange through a symmetric encryption [Gor11]

The first case with manual delivery is useful in communications requiring high security level, but it is slow and difficult for other applications. The second case is risky because if the adversary discovers the first secret key, it gives him access to all the other keys encrypted with it. Case 3 presume that A and B initially exchange and share a secret key with C. Exchange of symmetric secret keys through asymmetric encryption requires authentication of the public key users [MOV96].

2.1.6 Cryptosystems for Duplex Transmission and Storage

The Design of a cryptosystem imposes choosing appropriate techniques and methods to ensure message confidentiality, integrity, and authenticity. It also includes authentication of parties involved in secure communication, non-repudiation, authorization, etc. [Bor11].

Duplex communication is an exchange of information between two parties in both directions. It can be *half-duplex* or *full-duplex* transmission. Half-duplex transmission is not simultaneous; meaning that while one entity in a communication is transmitting the recipient must wait and can't transmit information at the same time. In full-duplex information transmission is continuous; it can be performed in both directions at the same time [BTT+13].

Fig. 2.4 presents a scheme of cryptosystem for secure transmission of the cryptograms and secret keys. In this system each new secure communication of the messages is performed with a new key, as part of the OTP principle. This scheme is presented in more details on the application in chapter 4. It shows a duplex communication between two parties: A and B. Continuous lines show transmission from part A to B and dashed lines in the opposite direction. When part A wants to transmit encrypted information to part B, it first generates the key (K_A), then encrypt the key using a symmetric or public algorithm (block K_AE), transmit the key to part B, and then encrypt the message using this key ($E_{K_A}(M_A) = C_A$). Side B decrypts the secret key and then uses it for message decryption. Part B will follow the same steps in order to transmit a message.

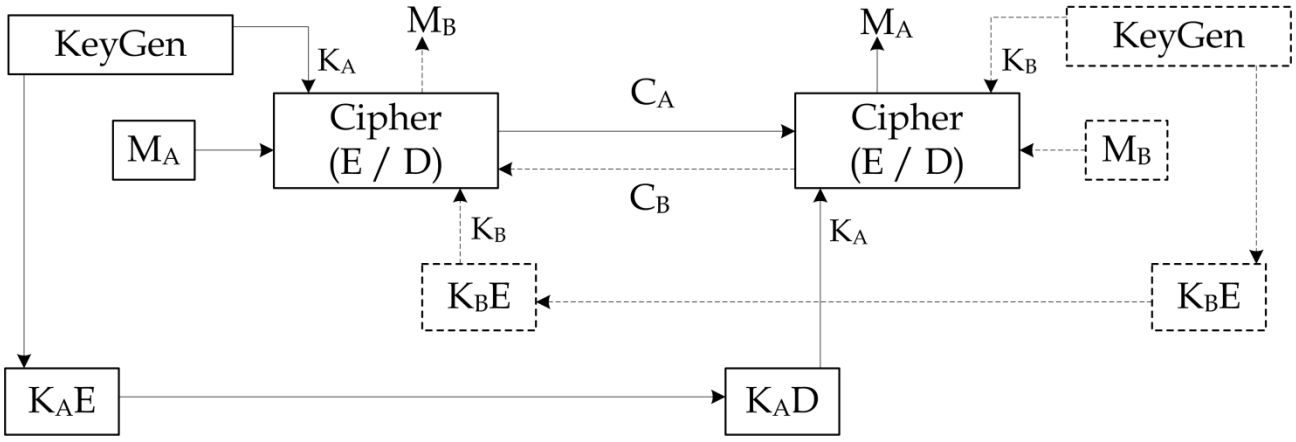


Figure 2.4 Cryptosystem for Duplex Transmission

Legend of Figure 2.4:

M_A – cleartext message of user A

M_B – cleartext message of user B

KeyGen – block of secret key generation(K_A, K_B)

K_AE – block of secret key encryption, using a symmetric or public algorithm for user A

K_BE – block of secret key encryption, using a symmetric or public algorithm for user B

K_AD– block of secret key decryption (at side B), using the same algorithm as at K_AE

K_BD – block of secret key decryption (at side A), using the same algorithm as at K_BE

Cipher (E / D) – is a symmetric cipher used for encryption and decryption of the messages M_A and M_B

Fig 2.5 demonstrates a cryptosystem for storage of the cryptograms and secret keys. Encryption side is the writing unit. After the secret key (K) generation and its encryption (block E*) it is placed on the storage media. After the data encryption: $E_K(M) = C$, the cryptogram (C) is also placed on the storage media. Decryption side is the reading unit. It decrypts the secret key (block D*) and then uses it to decrypt the cryptogram: $M = D_K(C)$.

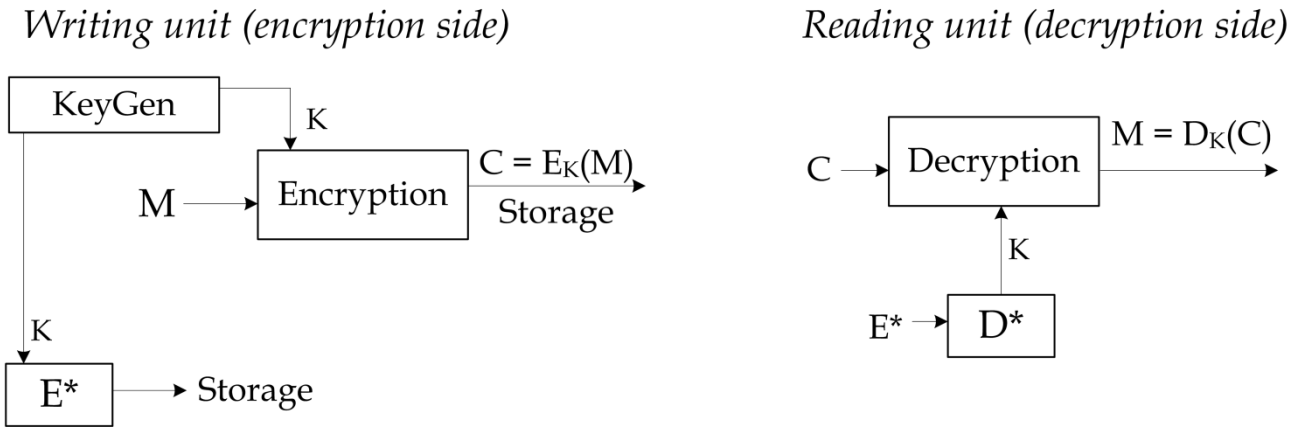


Figure 2.5 Cryptosystem for Storage

2.2 Data Compression

Data compression techniques are based on the elimination of certain redundancy (patterns) present in the data and removal of the fine details in multimedia data that are not important for human perception. Redundancy can be of different types according to the nature of the data. Neighboring pixels in the image usually have close values; in any language certain letters appear more often than others; there can be a certain periodicity in the audio signal. High frequency details in the multimedia data that are not important for human perception can be eliminated. Compression algorithms exploit these aspects of data in order to obtain a compact size [Say03].

The process of compression is named *encoding* and the reconstruction stage is named *decoding*. The input signal to the encoder may be any type of data like: audio, video, image signals (Fig. 2.6). Through encoding the format and size of the original input data X is transformed and its compressed representation X_C is obtained. Considering that data is not changed by channel transmission, the input to the decoder is X_C. The source decoder takes the compressed representation and reconstructs the original signal. The reconstructed

signal X_R is identical in lossless compression and in lossy compression it is an approximation of X.

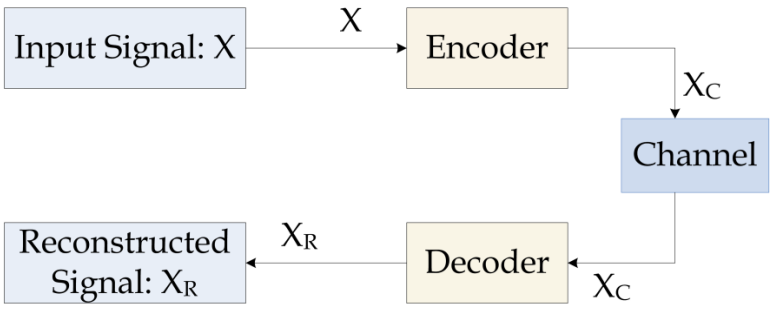


Figure 2.6 General scheme of compression process

The goal of data compression is to reduce the size, number of bits used to represent a certain information. *Bit rate* is the measure used to estimate the number of bits per value in the compressed data. In case of an image file bit rate is measured in bits per pixel.

Next section describes the steps of the compression process. In subchapter 2.2.2 elements of rate distortion theory are presented as it is relevant for compression effectiveness. In the next subchapters 2.2.3 – 2.2.4 different types of compression are explained, followed by methods to combine compression with encryption.

2.2.1 Data Compression Chain

In Fig. 2.6 a general scheme of compression was shown. This section presents the steps and schematic view of the encoding and decoding processes. As shown in figure 2.7 the encoding process is composed of 3 major operations. The first is to transform the original data in order to obtain a representation that is more convenient for the encoder. This is done by applying transformations like DCT or DWT on the data. The probability distribution of the obtained coefficients can be approximated by the generalized Gaussian function (GGF) or Laplacian probability density function (PDF) [Par03]. This new representation of the data is well suited for a good coding performance.

Next operation is the quantization of the data. It can be included or not in the compression scheme according to the compression type (subchapter 2.2.3). Quantization introduces loss of information. It can be used to eliminate very fine details, like high frequency information in an image, or audio file, without perceptual loss for our visual and audio system. It is a block of compression that introduces tradeoff between distortion and bit rate (subchapter 2.2.2). The final encoding operation is entropy coding. It attributes codes of different lengths to the signal values according to their probability (Fig. 2.7) [Ant11].

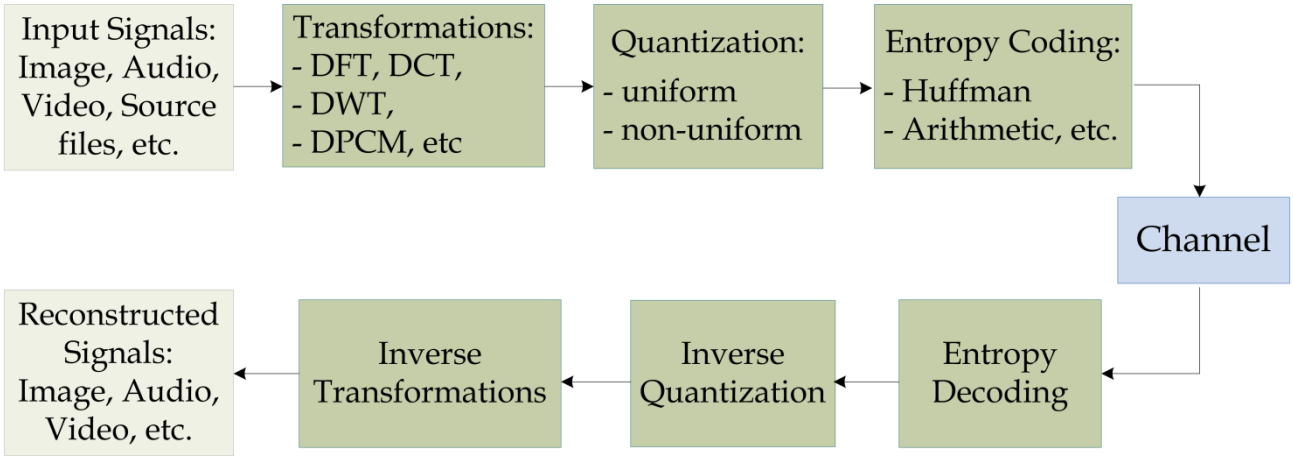


Figure 2.7 Chain of compression encoding – decoding processes

Transformations:

Prediction methods like DPCM (Differential Pulse Code Modulation) use correlations between successive samples. For example in case of images a combination of neighboring, previously encoded pixels is used as a prediction for the current pixel. In lossless JPEG the difference between the value of certain pixel and its prediction is encoded with Huffman coding. In lossy compression the quantized difference is encoded.

Frequency transformations like DFT (Discrete Fourier Transform) and DCT (Discrete Cosine Transform) are used to represent data in a more efficient form for compression. The high frequency coefficients represent some details of the multimedia data that are less perceptual for human visual and auditory systems. This characteristic is exploited in lossy compression for images, audio and video files. Representation of the files in the frequency transformed domain facilitates the extraction and elimination of the coefficients corresponding to high frequencies. By this transformation the spatial highly correlated

data is transformed into uncorrelated coefficients. In DCT transform the most important coefficients for perceptibility are in the high left corner of the transformed image (the low-level coefficients).

Decomposition like DWT (Discrete Wavelet Transform) is widely used in image compression. Wavelet decomposition is a successive application of high and low pass filters which results in sub-bands of high frequency details in diagonal, horizontal and vertical directions respectively and the image approximation of remaining low frequencies. Coefficients corresponding to the high frequencies are almost all of them close to zero and just few of them have higher values. Using a threshold all the coefficients close to zero can be set to zero and the rest of high frequency coefficients above the threshold can be encoded [Ble10], [Say03].

Quantization:

Quantization is the approximation method used in lossy compression. The goal of quantization is to obtain a smaller bitrate through representation of each group of close values by one single value. This mapping is irreversible and at decoding a group of originally different close values have the same value. The scale of approximation is measured by the quantization step.

Quantization process can be realized by a scalar or vector quantizer according to the data representation that can be a set of scalars or a set of vectors. A scalar quantizer realizes a mapping between the data values and a range of possible encoding values. For example, considering a uniformly distributed source in the interval $[-A,\ A]$ and M possible code words for encoding, the quantization step (Q_s) is $\frac{2A}{M}$. The original source interval: $[-A,\ A]$ is divided in uniform intervals of width Q_s. The middle value of each interval is used to represent all the range of source values from this interval. A vector quantizer groups source output into vectors of close values and then encode them by finding the closest code-vector. Decoder of vector quantizer has a look-up table for reconstruction of the vectors [GG91].

Entropy Coding:

Entropy coding is composed from a model of the data and a coder. A model of the data is a map of probabilities for each of its elements; their probability distribution [Ble10]. The coder uses this probability distribution and creates codes. A longer code is corresponding to the less probable symbol and a shorter code corresponds to the most probable symbol. Entropy encoding is prefix-free, meaning that none of the codewords is a prefix to the other codewords. This prefix-free property is necessary for the correct decoding of the

bitstream composed of variable-length codewords. Huffman and arithmetic entropy encoding techniques are the most commonly used algorithms.

2.2.2 Rate-Distortion Theory

Rate distortion theory makes part of information theory and it is a measure of compression performance. Given a certain level of data quality, hereby its distortion, it is desired to obtain the smallest number of bits per value, meaning the best possible bit rate [Boc09].

Distortion (D) of the data is measured as a difference from its original values $\{x_i\}$ and those approximated after compression, at reconstruction $\{\hat{x}_i\}$. It is usually measured by mean square error:

$$D(x, \hat{x}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{2.11}$$

Bit rate (R) is the average number of bits per symbol in the data. Considering that there are n symbols (S) in the data, bit rate can be expressed as follows:

$$R = \frac{1}{n}\sum_{i=1}^{n} length(S) \tag{2.12}$$

Given a source and a distortion measure, according to rate distortion theory there is a rate (R) – distortion (D) function: *R(D)* which has usually a shape like in Fig. 2.8[GG91].

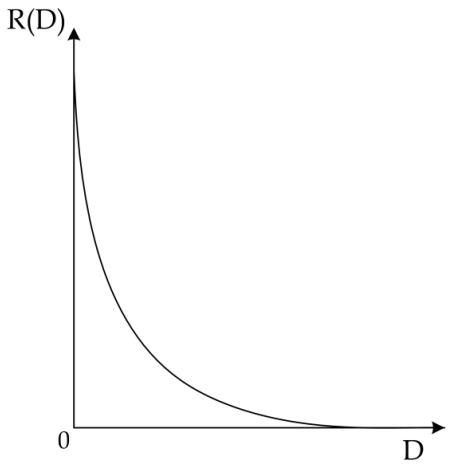


Figure 2.8 Rate – Distortion Function

The quantization block is part of compression that introduces distortion, but it is also the one that reduces the bit rate. Different quantization methods can be compared through

R(D) function. The R(D) function specifies the best tradeoff that can be obtained at the lowest possible bit rate and the minimal signal distortion. Rate distortion theory addressed the optimization problem that can be expressed in two forms:

1. Given a maximum bit rate R, to minimize the distortion D. It is a constrained R problem, hard to solve.

$$\min\{D\}, given\ R \tag{2.13}$$

2. Given a maximum distortion D, to minimize the bit rate R. It is a constrained D problem, also hard to solve[CT91]:

$$\min\{R\}, given\ D \tag{2.14}$$

Lagrange approach reformulates this optimization problem into the unconstrained form:

$$\min\{D + \lambda * R\} \tag{2.15}$$

Using Lagrange formulation the optimization problems from (2.13) and (2.14) can be expressed correspondingly as follows:

$$\mathrm{J}(\mathrm{R},\lambda) = D + \lambda R \tag{2.16}$$

$$\mathrm{J}(\mathrm{R},\lambda) = R + \lambda D \tag{2.17}$$

Minimum D at a given R, or minimum R at a given D can be found by setting the derivative of the expressions from (2.16) and (2.17) to zero.

$$\frac{\mathrm{J}(\mathrm{R},\lambda)}{dR} = \frac{dD}{dR} + \lambda = 0 \tag{2.16}$$

$$\frac{\mathrm{J}(\mathrm{R},\lambda)}{dR} = 1 + \lambda\frac{dD}{dR} = 0 \tag{2.17}$$

The Lagrange multiplier λ is a constant. It provides proportion of bit rate and distortion. In subchapter 3.4.2 the minimization of the Lagrangian function is used in the joint compression scheme.

2.2.3 Types of Data Compression

There are two types of compression: *lossy* and *lossless*. The objective of the lossless compression is to ensure data fidelity. In this case the decompressed data correspond

exactly to the original one. Only the statistical redundancy is used for compression [RJ91]. Lossless compression is applied to the data where the approximation is not acceptable, like: text documents, executable programs, source code. The amount of compression is limited to the entropy of the source. The well-known method used in lossless data compression is Lempel-Ziv, invented in 1977 and named by their authors [ZL78]. It is used in image formats like GIF and PNG.

The goal of lossy compression is to ensure the best tradeoff between data quality and its efficiency in storage. The decompressed data is an approximation of the original information. Common techniques that introduce loss of information are quantization and rounding of numbers [Rum09]. Once a certain value was rounded or represented by the index of quantization level it can't be recovered. The well-known and widely used lossy image and video compression standards are JPEG, MPEG, H264, and HEVC. A more recent standard for image compression is JPEG2000. It has a better compression performance then JPEG, but it is not yet commonly used on the internet.

2.2.4 Classification of the Multimedia Compression – Encryption methods

During encryption data passes through a series of transpositions and substitutions. If the security of the algorithm is strong, the redundancy of the plaintext will not be transferred on the ciphertext. If the redundancy of the data is high, such as in image, audio, or video files then there will be a high probability that the encrypted files will keep a part of the pattern from the plaintext. This is one of the reasons why the compression of the multimedia data is applied first and then the encryption. This model was proposed in a strong, hybrid cryptosystem named Pretty Good Privacy (PGP) [PGP04].

Encryption process is never applied before compression because of the practical issues. The encryption process randomizes the original data trying to achieve an equal probability of data appearance; therefore, there will not be remaining information that can be compressed.

The classical way is to perform compression of the data and then to perform encryption of the whole bitstream. This process is named *complete* or *direct encryption* (Fig. 2.9); it is time and space consuming and therefore may sometimes not be suitable for the real time applications. Full encryption is used when a high level of security is required and mostly for storage. Multimedia data is usually involved in the real-time interactions where the transmission must be fast, and it has already large volume without encryption, which can increase the size of the data. In order to solve this security issue the *partial* or *selective encryption* was proposed [CL00], [PC08]. The idea of selective encryption (SE) is to encrypt

only a part of the compressed data (Fig. 2.9). In this way the volume of the data will be reduced and the speed of transmission increased.

Another method to avoid the computation and storage cost of encryption is to integrate it inside the entropy coding [SK05]. This method is a *compression-combined encryption* also named *joint compression - encryption*. In this case security is integrated inside the process of compression.

Complete multimedia data encryption means that the whole encoded bitstream is encrypted without considering the format of the data. What can change is the cipher chosen for data protection. The most known and used symmetric ciphers are Data Encryption Standard (DES) [FIPS93], Advanced Encryption Standard (AES) [FIPS01], International Data Encryption Algorithm (IDEA) [LM91], etc.

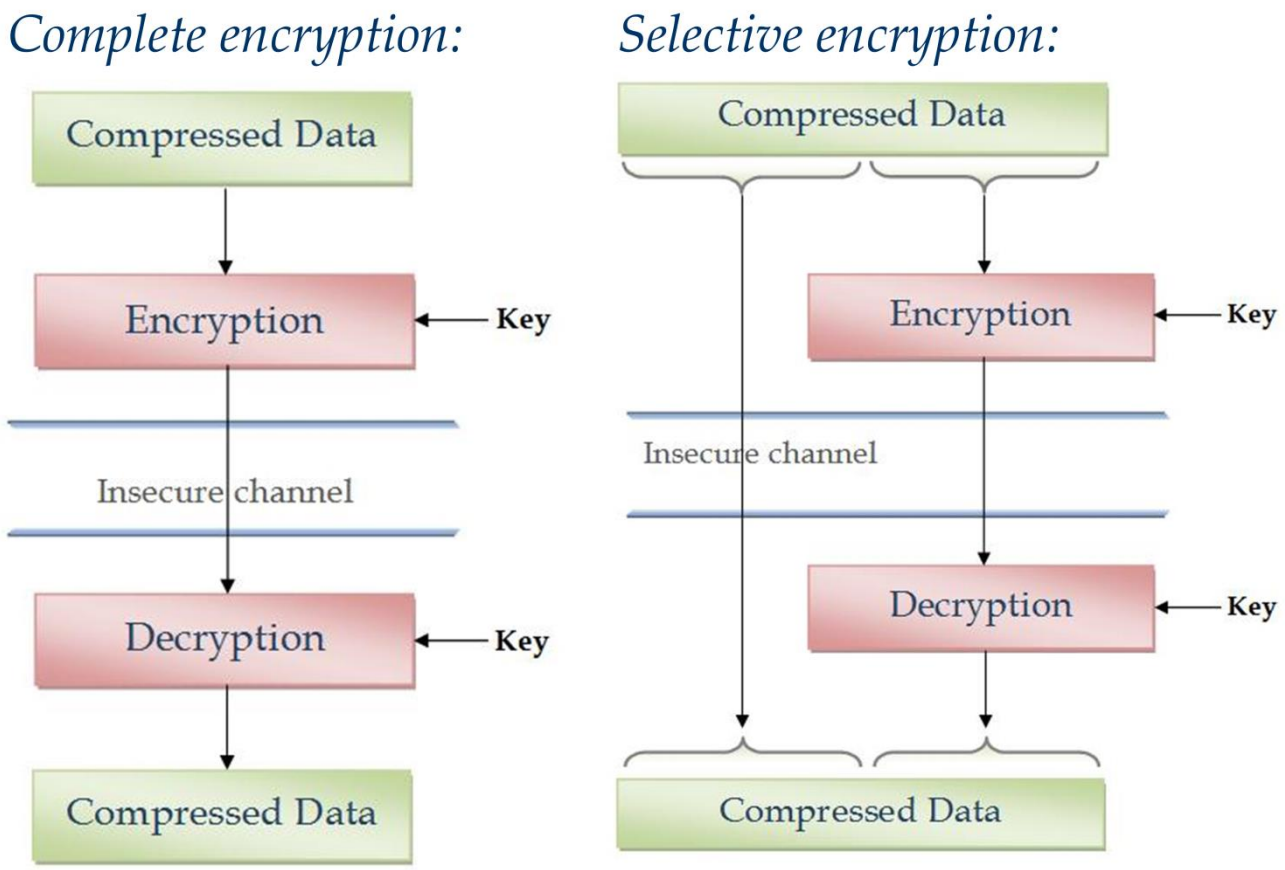


Figure 2.9 Schemes of Complete and Partial-Selective Encryption

Selective encryption can be performed in different ways; it depends on the part of the data selected for encryption. One of the SE techniques is based on the selection of few DCT

coefficients that are the most important for human perception of the data. DCT (Discrete Cosine Transform) represent data in frequency transformed domain and is used in compression in order to eliminate the high frequency coefficients which are not perceptible by human auditory and visual systems. This transform makes part of the JPEG and MPEG compression standards. There are 3 basic SE methods that relies on DCT coefficients: sign bit encryption of each DCT block [SB98], encryption of the first and single DC coefficient (the average value of the block data values) of each DCT block [LKCV06], and the last one is encryption of the DC coefficient with few AC coefficients of each DCT block [PR05], [RRM+99] (Fig. 2.10). The number of selected AC coefficient gives the tradeoff between compression and security.

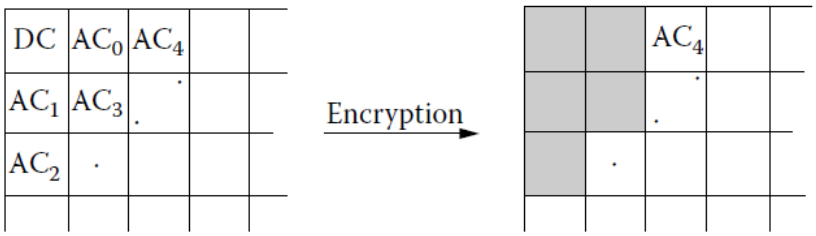


Figure 2.10 Encryption of the DC and AC coefficients of a DCT block

Another method of SE is to encrypt some of the wavelet transform coefficients [BU98]. Reference [SK05] presents a SE technique based on Embedded Zerotree Wavelet (EZW) algorithm of compression [Sha93]. EZW is an entropy coding algorithm based on the wavelet transform of the image (or other signal) and representation in a quad-tree of the coefficients. The EZW algorithm performs a progressive encoding finding the most important coefficients. The output of the EZW algorithm is a bit stream with increasing accuracy. [SK05] exploits the fact that this ordered representation of the bit-stream is suitable for the partial encryption. Depending on the desired accuracy a certain number of first bits from the encoded bit-stream are encrypted.

Format-compliant encryption can also be categorized as a SE method. The principle is to encrypt the compressed code-stream with respect to the format of the compressed file which means keeping the headers and markers unchanged [ESU07]. In [WSZ+01] it was proposed to concatenate bits chosen for encryption, to encrypt them with a standard symmetric cipher (like DES or AES) and place the encrypted bits back to the code-stream. If the result is not compliant then a fixed length index is assigned to each code-word and then the indexes are encrypted and placed back to the code-stream. This method decreases

significantly the compression ratio, because all the code-words will have the same length after indexation. In [WD04] a compliant encryption of the JPEG2000 code-stream was proposed. The basic idea is to consider the code-stream of interest as a plaintext (P), to generate a secret key stream (K) of the same length as the P, to compute the ciphertext (C) from P and K, if the C is non-compliant, then C becomes P (P = C) and then a new C is computed. This method can bring some additional computations, but on the other hand it offers a simple method of compliant encryption without increasing the size of the compressed file.

Compression-combined encryption can be successfully realized by integrating encryption inside the entropy coding technique. The basic idea of the entropy coding encryption is to convert classical entropy coders into encryption ciphers. An example of such transformation is Multiple Huffman Tables (MHT) designed by [WK01]. Their work was enhanced in [XK04] and then in [VF11]. The principle from the initial work [WK01] is to replace a single statistical model by multiple statistical models randomly chosen from the pre-stored models. In [VF11] the output of the MHT is XORed with another key in order to increase the security level. Thus two keys are used: the one with the order of the chosen Huffman tables and another with a binary sequence to XOR the chosen table. The length of the code-words remains unchanged because the pre-stored Huffman trees have the same topology. Therefore there is no loss of the compression efficiency and the complexity of integrating compression along with encryption is much reduced.

Another entropy coder modified to include the encryption is presented in [GMO06]. A randomized arithmetic coding is used to ensure encryption integrated inside the compression. The secret key that introduces security is a random swap of the probability intervals of the symbol. The magnitude of the interval corresponding to a certain symbol remains unchanged. The only thing that changes is the position of that interval. The interval is swapped as position, but the probability interval length remains the same.

2.3 DNA Cryptography

DNA cryptography is a new broad scientific branch, which includes a variety of core scientific areas: information security (cryptography, steganography, key management), molecular biology, bioinformatics, biomolecular computation. It is a new and promising field in information security. It combines the classical solutions in cryptography with the strength of the genetic material. Biological DNA can be used in steganography and cryptography as the storage material. Molecular computations can be performed with

biological DNA structures and then applied on the classical ciphers. Several projects in genome sequensing offer the possibility to exploit digital DNA databases for the criptographic purposes.

Some notions of molecular biology and DNA are given in subchapter 2.3.1. Aspects of using the biological DNA for cryptography are described in section 2.3.2. Importance of genetic databases and digital DNA sequence for criptography is presented in subchapter 2.3.3.

2.3.1 Molecular Biology Principles

A cell is the fundamental functional unit in biological organisms. Most of the cells contain a nucleus and *chromosomes* inside of it. Genetic information (DNA) that controls cell functionality is divided into chromosomes [CDLT04]. Each chromosome is composed of a single DNA molecule which carries *genes* (Fig. 2.11). The *genome* is the amount of all the genetic, hereditary information of an organism. It contains information from all the chromosomes. Complex organisms contain billions of cells. Each cell holds in its nucleus the same copy of chromosomes, but depending on the cell type it activates only a specific part of the whole genetic material (gene expression). The cell has the ability to store, retrieve and translate genetic instructions providing life to the organism [AJLR+08].

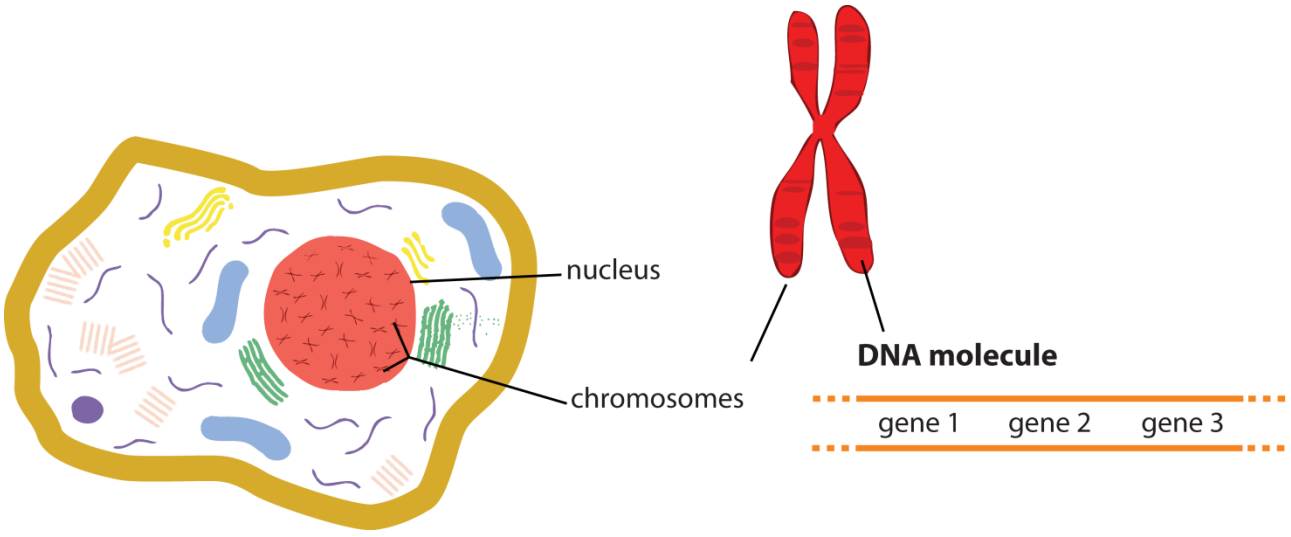


Figure 2.11 Cell and its Genetic Material

2.3.1.1 DNA Structure

Deoxyribose Nucleic Acid (DNA) has a helical shape, comprised of two long strands of nucleotides. A nucleotide has one of 4 bases: A – adenine, G – guanine, C – cytosine, or T – thymine, a deoxyribose sugar and a phosphate group (Fig. 2.12) [Sch03].

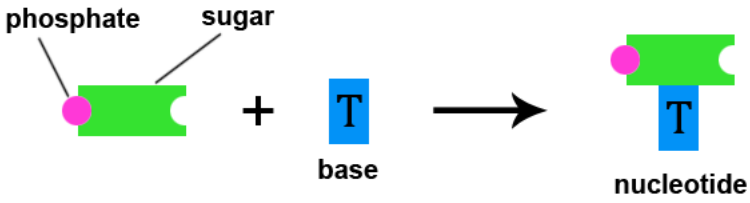


Figure 2.12 Nucleotide Structure

The sugars and phosphates make nucleotides to bind in a single DNA strand. The hydrogen bonds hold 2 strands together and create a double-stranded DNA (Fig. 2.13). Hydrogen bonds last only between complementary pairs: A-T and C-G [AJLR+08].

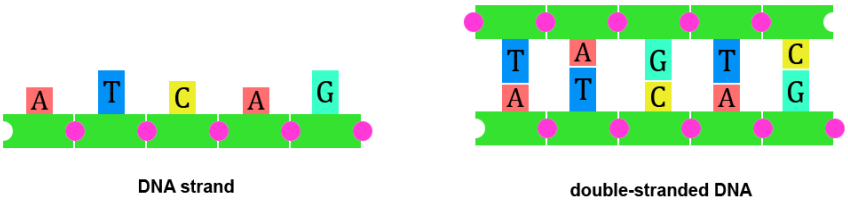


Figure 2.13 DNA Strands formed by nucleotide and hydrogen bonds

The three-dimensional structure of DNA was discovered in 1953 by Watson and Crick [WC53]. DNA strands twist around each other forming a helix (Fig. 2.14).

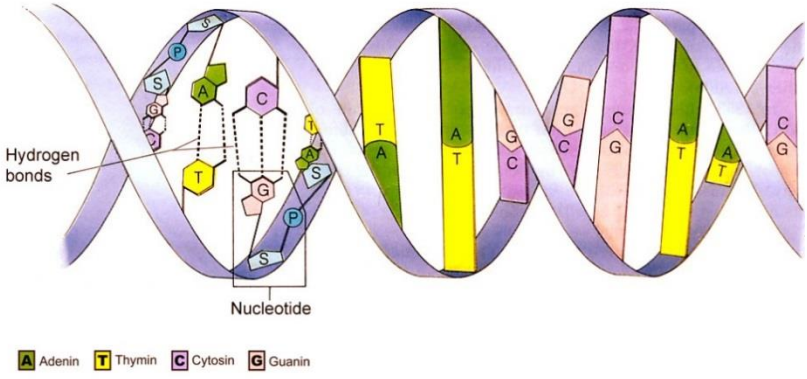


Figure 2.14 DNA Structure

Central dogma of molecular biology explains how genetic information is expressed in a protein. Genetic code is a correlation between the codons (3 nucleotide bases of DNA) and

amino acids. Amino acids are structural units of proteins, the same way as nucleotides are in DNA [CDLT04]. DNA genes "tell" the cell in what order to assemble the sequence of amino acids in protein.

2.3.2 Cryptography at the Molecular Level

This part describes operations that can be performed with DNA sequences in a laboratory. Considering the available techniques to manipulate molecules of DNA, the possibilities of storage and computations at molecular level are presented next.

2.3.2.1 Techniques for DNA Analysis

Over the last 60 years important discoveries was made in molecular biology: chemical structure of DNA [WC53], separation of DNA polymerase (A. Kornberg, 1956), mechanism of the biological synthesis (Kornberg and Ochoa, 1959), DNA sequencing since the 1970s. DNA analysis and manipulations was possible due to the important techniques that were designed and developed for this purpose.

DNA sequencing encompasses several techniques to determine the order of nucleotide's bases in a DNA sequence. One of the methods is to use fluorescent tags for the nucleotide bases and obtaining a fluorescent complementary DNA strand to the one of interest. Analysis of genes and a better understanding of their role in organisms' lives are possible due to this process [Sch03].

DNA recombination is a technique to manipulate genes. It is also named "gene splicing" or "genetic engineering". In this method certain proteins - enzymes are used to cut and paste parts of the DNA spiral. When a double strand of DNA is cut by this technique it will have certain terminations that can be used to glue to other pieces of DNA [HJ04].

Hybridization is a natural process in DNA molecules. It happens when two complementary strands of DNA come together to form a double-strand. Process of hybridization is slow at the beginning, until a region of two complementary strands binding appears; the rest of the matching process is fast if the strands are all complementary (Fig. 2.15) [AJLR+08].



Figure 2.15 Hybridization between 2 Complementary Strands of DNA a) beginning of the process b) final product of hybridization

DNA synthesis is creation of synthetic DNA molecules named oligonucleotides. Synthetic oligonucleotides represent DNA strands usually10-100 nucleotides long. This technique can be used to replace a damaged portion of DNA or in technology for storage and transmission of the information.

DNA synthesis together with hybridization can be used to create different molecular shapes. One of such structures being used extensively in molecular computation is *DNA tile* designed by Wang tiles [Wan61]. H. Wang designed equal-sized squares with a color on each side. These squares can be combined together respecting the same color for neighbouring sides (Fig. 2.16).

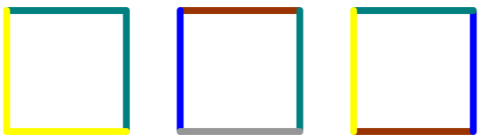


Figure 2.16 Few Wang tiles

The principle of Wang tiles was used in DNA computing field to create DNA structures with similar functionality. Oligonucleotide strands are designed to cross-over and hybridize in different helixes (Fig. 2.17).

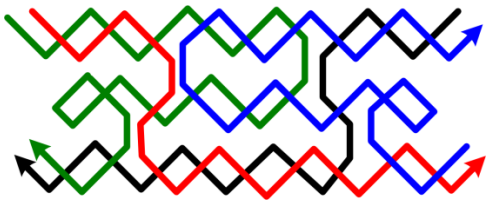


Figure 2.17 DNA Triple-Helix Tile [Bor11]

The final structure contains few helixes bound to each other. Helixes are formed from different DNA strands [GLR04]. Terminations of a tile are named "sticky ends" because they are formed of DNA nucleotides that can hybridize with other complementary terminations and in this way 2 tiles stick to each other.

2.3.2.2 Molecular Storage and Computations

The biological medium of DNA molecules can be viewed from the perspective of storage material or computational force. The human genome is around 0.72 GB of data in just around 3.5 picograms [DBV+03]. This means that a large amount of information can be stored in a compact, invisible for us, space.

Different concealing techniques have been proposed to explore a hardly noticeable molecular medium. A successful experiment of DNA steganography is presented in [TRB99]. It consists in hiding a DNA-encoded message in a background of other DNA sequences. Laboratory techniques and the starting point of the message is required in order to read it.

There is a variety of techniques proposing to introduce watermarks in DNA of a living organism. A significant work in this domain is presented in [BDH+04]. Some other methods in this area are described in: [SFP02], [KL10], [SNF+10], [HB07]. A watermark can be introduced in coding regions of DNA using codons redundancy and in this way no functional changes will appear [BDH+04]. Watermark information can be also introduced in non-coding regions of DNA [SFP03].

In order to store data in DNA molecules it must be encoded in DNA alphabet. In [TRB99] a mapping method is presented where each letter from the English alphabet (A - Y), numbers (0 - 9), and some punctuation marks where encoded in 3 DNA letters, like: Q – AAC, or 9 – GCG. Considering that all of the digital information is encoded in binary, a straightforward method is to make a mapping table between DNA and binary alphabets (Table 2.1) [TB09]. Using this mapping method any digital information can be transformed easily in DNA sequence.

Binary	DNA
00	**A**
01	**C**
10	**G**
11	**T**

Table 2.1 Conversion Table from Binary to DNA

The other usage of DNA molecules is their computational capability. Biomolecular computation is based on the existing techniques in DNA analysis and manipulation (subchapter 2.3.2.1). The DNA computing field has emerged due to the experiment "in vitro" performed on DNA molecules by Leonard Adleman in 1994 [Adl94]. He showed

that biomolecular computing can be used to solve a problem like finding a Hamiltonian path in a graph. His achievements brought scientific interest to this domain, followed by many other experiments and discoveries.

In 1980 Nadrian Seeman introduced design of the DNA controllable structures [See81]. The role of these structures is to be building blocks of more complex molecular structures. The hybridization process is applied on these basic units to obtain a variety of DNA nanostructures.

In order to perform a DNA bio-chemical arithmetic operation the following ingredients are required: laboratory utilities, structures from DNA material with sticky terminations (Fig. 2.17), solution in which a natural hybridization can occur between these terminations (Fig. 2.18). DNA tiles must be properly encoded in binary symbols in order to obtain the desired binary arithmetic operation.

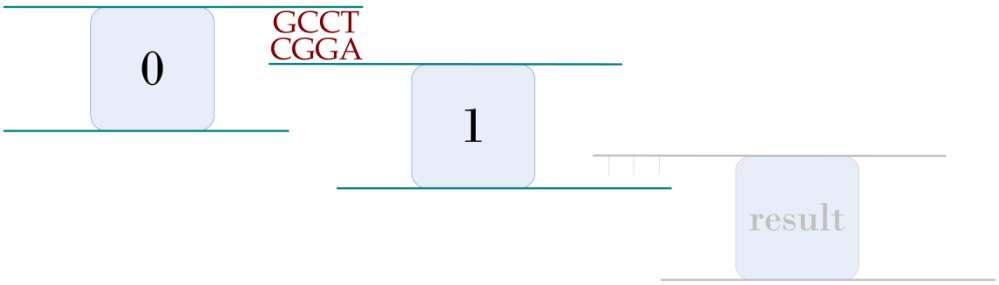


Figure 2.18 Hybridization between DNA Tiles that can be used for Arithmetic Operations

DNA computing can be used to implement existing cryptographic algorithms at molecular level. Vernam cipher, based on binary XOR between plaintext and ciphertext bitstreams can be performed using DNA tiles by encoding them in binary ones and zeros. In [MLRS00] a method is described how to obtain a binary XOR operation using self-assembling DNA tiles. In [LWR99] different binary operations were performed using DNA tiles. The [GLR04] presents Vernam cipher [Ver26] implemented with DNA tiles.

2.3.3 Digital DNA in Cryptography

The interest to find out the exact sequence of DNA inside different living organisms brought an enormous amount of financial, technological and human resources to this subject. In 1985 the Human Genome Project (HGP) started. It was carried out in 2003 through the effort of a variety of international organisations. The basic procedure is DNA sequencing (subchapter 2.3.2.1). Data sharing was the core principle of this project: "The

adoption of free release and data sharing has been among the major achievements of the Human Genome Project" [Sanger].

Therefore there is at this moment a great variety of online, with free access, genetic databases and tools for analysis. A link to a comprehensive list of genetic databases, organizations and tools is given in [IHGM]. There is a great amount of available genomic and protein sequences belonging to different organisms.

Two of the ciphers proposed in this thesis use genetic databases in cryptography. The first method uses DNA sequence for substitution of plaintext values. The ciphertext will contain the positions in the genetic sequence from where to read the plaintext values. In the other method a DNA sequence is transformed into a binary stream. Then a bitwise XOR operation is performed with it and a plaintext binary stream. Both methods use DNA sequences as the secret key in symmetric encryption. Randomness of the key, how it can be obtained and transmitted are discussed in the following chapters.

2.4 Performance Evaluation Aspects

In this thesis three cryptographic algorithms were designed and developed. Certain metrics were used to evaluate their efficiency. Performance was evaluated from different perspectives: computational time, security level, and compression ratio. These measurements are discussed in more details in what follows.

2.4.1 Computational Complexity

Computational complexity estimates the amount of resources required for solving a certain problem. Efficiency of an algorithm can be evaluated by a theoretical analysis, where complexity theory is involved. Another way to test method's efficiency is to implement it and take measurements of the elapsed time during its performance.

In complexity theory the amount of resources used by an algorithm is estimated and computed based on the input parameter which is the number of operations. According to complexity theory, computational time is a sum of all the performed operations. The number of operations can be represented by a fixed or variable number:

- fixed (parameter independent)

  Example: k = 5; a = 9; (No of operations is 2, O(2))
- variable (parameter dependent)

Example: for $i = 1 \rightarrow n$ {k = k + i; } (No of operations is n, O(n))

Also computational time in complexity theory can be expressed by:$O(n)$, $\Omega(n)$, or $\theta(n)$. They represent the upper, lower and exact bounds of the necessary resources, and n is the variable number of operations. In most analysis the O(n) notation is used.

The execution time of an algorithm grows with the input size and its function can be: logarithmic – $O(\log_x n)$, linear – $O(n)$, quadratic – $O(n^2)$, cubic – $O(n^3)$, or exponential – $O(2^n)$. Logarithmically growing rate of the runtime is the most optimal and the exponential time is preferably to avoid.

The other method to evaluate execution time is to measure it during the algorithm performance. Depending on the development environment and programming language the time measurement functions can vary, but the general principle is the following:

Time Function StartVar;

<*Algorithm code*>

Time Function StopVar;

Spent Time length = stop – start;

In order to compare algorithms with this method, they must be tested on the same computer, developed in the same programming language and environment. The growing rate of the computational time can be visualized on a graph by computing runtime at progressively growing number of operations for the tested procedure.

2.4.2 Security Level

Security strength of a cipher can be evaluated by different techniques like: statistical measurements, cryptanalytic attacks, analysis of the key space and its randomness. In subchapter 2.1.4 basic cryptanalytic attacks were presented. Here more details are discussed on statistical measurements of the ciphertext in comparison with the plaintext, the key space, and random numbers.

Statistical measurements like histogram, correlation coefficient, and entropy gives the knowledge about patterns in the analysed information. The presence of patterns in the ciphertext gives the opportunity for the attackers to define a rule by which they can retrieve useful information without using the key. Statistical techniques are important in case of a ciphertext-only attack, where an attacker has the access to the ciphertext, but not to the key or related plaintext.

Histogram is a graphical representation of the signal statistical probability distribution. Given a discrete range of values in a signal, its histogram shows the occurrence of each

value in it. Each value has a bar in the histogram and it is as high as the frequency of appearance of that value in the signal. Statistical distribution of the ciphertext can be analysed in comparison with the plaintext distribution or by itself. Plaintext distribution usually has certain forms, patterns, describing information inside it. Ciphertext histogram, if the encryption is security strong, has a uniform random distribution, which means that ciphertext values were generated at random from a uniform distribution (Fig. 2.19) [MOV96].

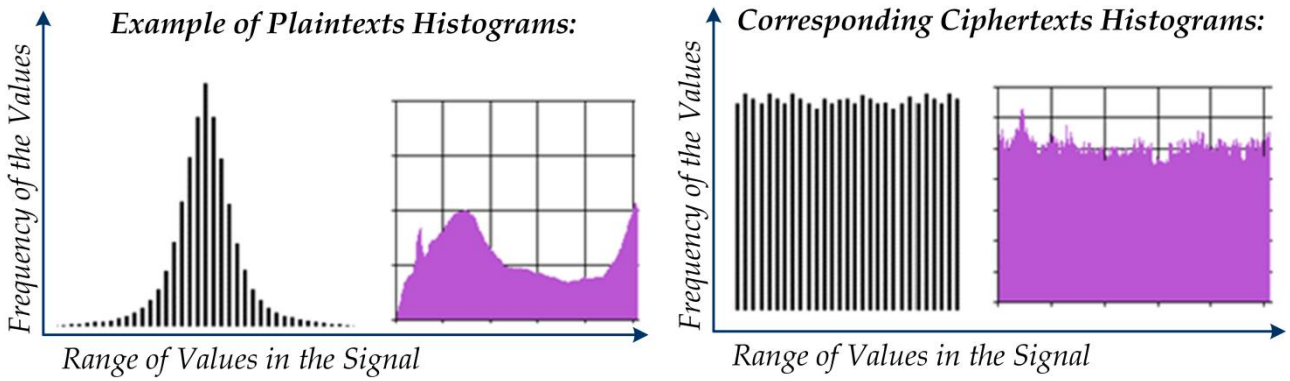


Figure 2.19 Histograms of the Plaintexts and Ciphertexts

Entropy measures the uncertainty or the randomness of the signal. Considering a certain variable X defined by a set of values $\{x_1,\ x_2, \dots ,\ x_n\}$ with its probability

$$P(X = x_i) = p_i \tag{2.16}$$

where

$$0 \leq p_i \leq 1, \forall\, i \in \{1 \dots n\} \tag{2.17}$$

$$\sum_{i=1}^{n} p_i = 1$$

Considering variable X a binary expressed signal, its self-information is

$$I(x_i) = -log_2 p_i (\text{bits/symbol}) \tag{2.18}$$

Entropy of X is the average amount of this information. It is defined and measured as:

$$H(X) = -\sum_{i=1}^{n} p_i log_2 p_i \text{ (bits/symbol)} \tag{2.19}$$

The range of entropy values for variable X is

$$0 \leq H(X) \leq log_2 n$$

$$H(X) = 0 \iff \exists!\ p_i \in P: p_i = 1 \wedge p_j = 0, \forall\, j \neq i \tag{2.20}$$

$$H(X) = log_2 n \iff p_i = \frac{1}{n}, \forall\, i \in \{1 \,..\, n\}$$

When entropy is equal to zero it means there is no uncertainty about the signal prediction. When entropy has its maximum value it means that all of the symbols in the signal are equally likely to occur [Sha48]. For this reason the entropy of the ciphertext is desired to be as high as possible.

In the plaintext a correlation between different groups of bits may appear, depending on the type of the data. For example in an image there is often a strong correlation between the neighbouring pixels. When an image is encrypted it is desirable to avoid the presence of such correlation in the ciphertext. Correlation between the ciphertext values can be measured through the correlation coefficient [RN88].

Pearson's Product Moment Correlation Coefficient (r_{xy}) is the most used formula to determine linear correlation between two variables: X and Y. Considering normally distributed and linear to each other X and Y variables defined by sets of values $\{x_1,\ x_2, \dots,\ x_n\}$ and respectively $\{y_1,\ y_2, \dots,\ y_n\}$ their Pearson's correlation coefficient can be expressed as [You06]:

$$r_{xy} = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.21}$$

where $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the mean value of the X and $\bar{y}$ is the mean value of the Y.

Correlation coefficient values close to zero mean there is a weak linear relation between compared values and when they are close to one this relation is strong. In the analysis of the ciphertext it is desired to obtain small values of the correlation coefficient.

The key space of a cipher is defined as a set of all the possible keys that can be formed for that cipher. The amount of all the possible keys is given by the alphabet in which the key is represented raised to the power of its length. A larger key space means better resistance to the brute-force attack, hence better security.

Example:
Key: 101
Alphabet: 2 letters (0 and 1)
Length of the key: 3
Key Space (2^3): 000, 001, 010, 011, 100, 101, 110, 111

Random ciphertext can be obtained from a non-random plaintext if it is combined with a random key sequence [Sch96]. Generation of true random numbers requires a naturally occurring source of randomness [MOV96]. Truly randomness cannot be achieved by mathematical functions. Only certain physical processes can guarantee really random numbers [Sch09]. The use of mathematical formulas will lead to pseudo-random numbers: numbers that appear random, but are predetermined in reality. True random numbers, instead, come from measurements of random physical phenomena like atmospheric noise or radioactive sources. DNA material offers the possibility to generate random numbers in two ways: through its hybridization process and its sequence. These aspects are presented in more detail in the following chapters.

2.4.3 Compression Ratio

Compression algorithms reduce the size of the data whereas some of the ciphers may increase it. Estimations of size changes in the data can be performed through the bitrate (R) (subchapter 2.2.2) or compression ratio. The bitrate gives an average number of bits per symbol used to represent information while compression ratio estimates changes in size of the entire file. Compression ratio (CR) is defined as amount of original data divided by amount of compressed data.

$$\boldsymbol{CR = \frac{Original\ Size}{Compressed\ Size}} \tag{2.22}$$

It is desirable that after encryption the compression ratio of the data will not change, or the changes will vary in a small range. Compression efficiency after applying a certain cipher can be measured by changes in compression ratio (CCR), a metric defined as:

$$CCR = \frac{C_s - P_s}{C_s} * 100\% \tag{2.23}$$

where P_s is the size of the plaintext and C_s is the size of the ciphertext.

Ideally the compression ratio is unchanged and CCR = 0, if for example CCR > 10% then the compression ratio is changed greatly, more than 10% [Lia28].

Chapter 3

DNA Indexing Cipher

Contents in Brief

3.1 Principle of the Algorithm 38

3.2 Design and Implementation Details 40

3.3 Performance evaluation.....44

3.4 Modifications of DNA Indexing Cipher for Better Compression Ratio and Security...52

3.5 Contributions 70

DNA Indexing is a symmetric stream cipher. It uses genetic databases in order to access long DNA sequences. These sequences are used as a place with pointers (indexes) to the real message. The principle of indexing used in this algorithm is presented in [ASE06]. The sequence used at encryption is communicated to the receiver through unique identification number of the sequence in the database.

A variety of possible genes and chromosomes from different organisms are good sources for creation of random, non-repeating and for only one use pads. Nowadays there are electronic databases of entirely sequenced genomes from different organisms including human, dog, mouse, frog, fruit fly, social amoeba and many others. These sequences can be accessed from public genetic databases [wncbi] in different formats.

3.1 Principle of the Algorithm

DNA Indexing is a stream cipher where information is processed one byte at a time. The principle is to transform one plaintext byte into a sequence of 4 DNA letters. The next step is to search this short sequence through the chromosomal sequence, which was chosen as the key for the encryption (Fig. 3.1). Each time a plaintext byte sequence is retrieved in the chromosomal sequence, the position of this place is memorized in a vector as one of the possible values for encryption by substitution for this byte. Vectors of substitutions for all the bytes are memorized in a key dictionary. Therefore for each byte from the plaintext there is a range of possible values from which one is chosen randomly for encryption by substitution. In order to obtain a substantial number of substitutions for each byte, the key-sequence needs to be sufficiently long, for example 30 000 bases. The steps of encryption are presented below. An example of encryption is given in Fig. 3.2.

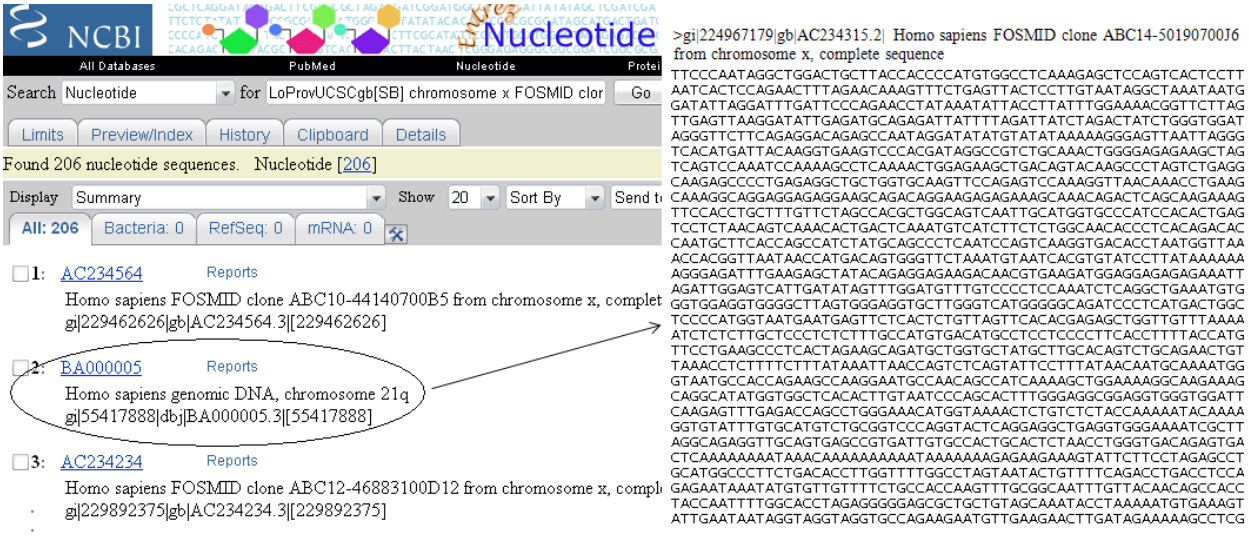


Figure 3.1 Chromosomal Sequences from a Genetic Database

1) Key dictionary computation:
   a) Each byte of 256 possible values is transformed in a sequence of 4 letters by the following principle: 10 00 11 01 (141) → GATC.
   b) A search is performed for all the bytes through the key, a long chromosomal sequence composed from letters: A, C, G, and T.
   c) Each time the byte sequence is retrieved in the key sequence, the index of that position is memorized in a vector dedicated for that byte.

d) The result of these operations is a key table of size 256xN, where N is a variable length because each byte can have a different number of corresponding values in this table.

2) Encryption is performed one byte at a time. It consists in substitution of the byte with a value randomly retrieved from its vector in the key table.
3) The final ciphertext is an array composed of the integer values.

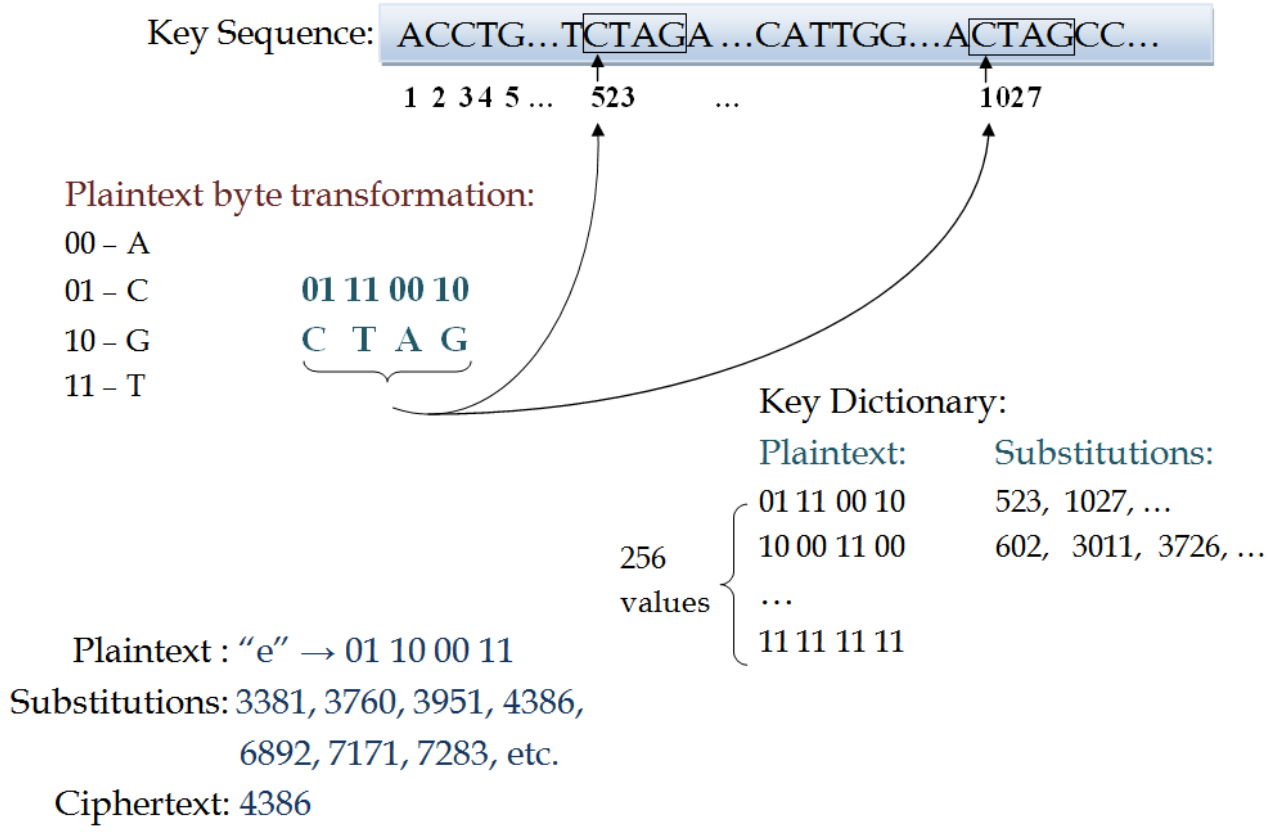


Figure 3.2 Encryption Process of the DNA Indexing Algorithm

Since this is a symmetric key algorithm, the same chromosomal sequence is used during the decryption process:

- Each integer from the ciphertext is used as pointer into the key sequence.
- The Receiver reads 4 letters from the indicated position and transforms them to binary representation using the same reversible principle presented at the key table computation.

- The plaintext is reconstructed when all the bytes are retrieved from the indicated positions

If we look back in history, we'll find that the principle of this algorithm is similar to that of a *book cipher*. The idea of using books in cryptographic purposes dates from 1526 [RS10]. In those times Jacobus Silvestri proposed in his work a code book cipher for the secret communications. Any encryption algorithm that has a book or a long piece of text as the key can be named a book cipher. An example of such algorithm is a substitution of each word from the plaintext with its position in a certain book, where the position is given by counting each word from the book. A genome sequence can be considered as such book and the genomic databases as a digital library available for this cipher.

DNA Indexing encryption algorithm can be considered partially a *homophonic substitution cipher*. The principle of homophonic substitution is to create a table where each letter of the alphabet has a certain number of substitution values. The number of substitution values corresponds to the frequency with which a letter appears in the language.

The substitution operation of DNA Indexing cipher is simple to achieve, but the reverse process is complex without knowing the key, because the distribution pattern of the ciphertext is completely different from that of the plaintext. For each byte of the plaintext there will be a certain number of substitution values. The number of substitution values for a byte will depend on its appearance in the chromosome. If someone wants to reuse the same key for many encryptions then it will be useful to transform this cipher into a homophonic one.

3.2 Design and Implementation Details

DNA Indexing cipher was designed and developed as a software program. Its design captures two kinds of views: the user point of view and the implementation point of view. It was accomplish through diagrams created in Violet UML Editor. UML (Unified Modeling Language) provides graphic notations for creation of the visual models for the software application. It offers architectural structure and behavior description for the project. Diagrams created with this tool helps to avoid ambiguities in system definition.

The first diagram describes the capabilities expected from the system. For this purpose use-cases were used, which show typical interactions between the user and the system under development. The purpose was to capture each possible task that a user can perform with the system in a use-case. All the use-cases together should describe the full system functionality [BS04]. Fig. 3.3 presents the use-case diagram for the DNA Indexing cipher program.

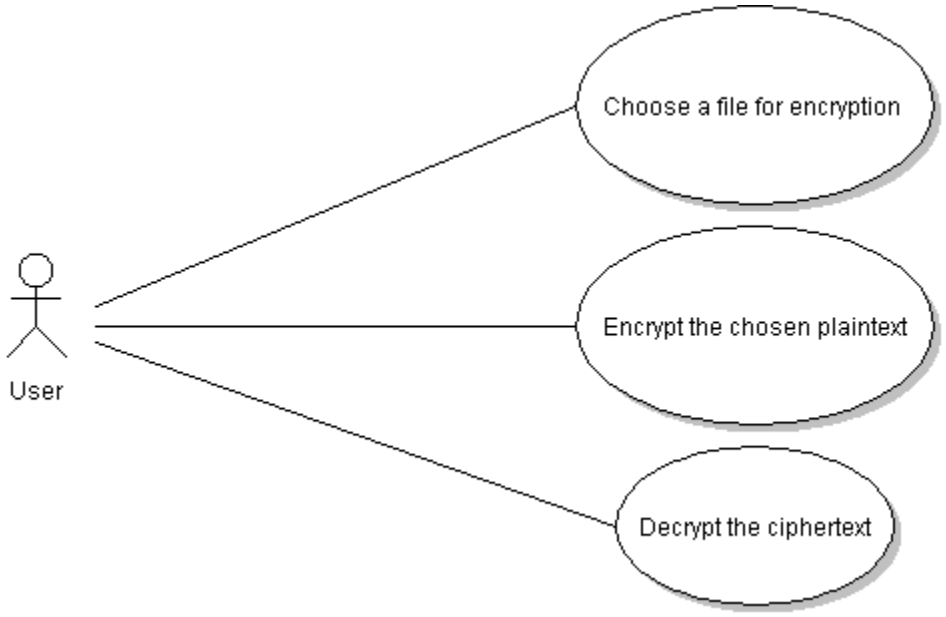


Figure 3.3 Use Case Diagram

Now, having accomplished the user perspective, the next step is to consider how each use-case can be achieved. For this purpose a class diagram, object diagram, and interaction diagram were used. Each of them provides more details for the development.

From the class diagram the initial program code of classes can be generated and then populated with variables and methods [BS04]. This initial program reveals architectural relations that must exist between classes (Fig. 3.4).

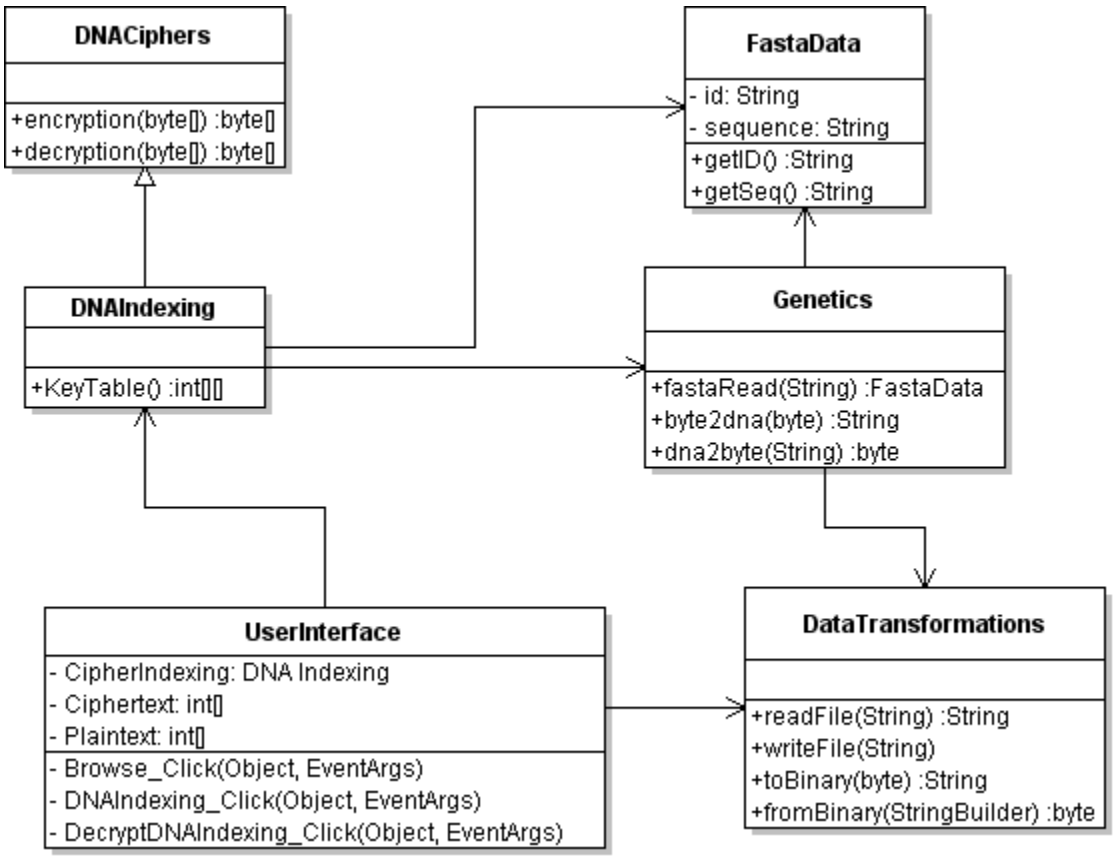


Figure 3.4 Class Diagram

Class diagram of Fig. 3.4 shows a structure view of all the classes involved in the DNA Indexing cipher development. The DNAIndexing class inherits encryption and decryption functions from the DNACiphers class and contains its own function: KeyTable(). In the KeyTable() function objects of types FastaData and Genetics are created. These objects are used to call functions that operate on the DNA sequences in FASTA format files. The UserInterface class contains functions that allow choosing a file, its encryption and decryption with DNA Indexing cipher. In the encryption – decryption functions objects of DNAIndexing class are created.

The object diagram gives the architecture of objects in the system (Fig. 3.5). It shows the state of the classes when the program is running and is a complementary view to the class diagram. The sequence diagram describes the behavior expected from the objects (Fig. 3.6). It shows a complete chain of actions during program execution. This diagram is linked to the use case diagram. It starts with one of the actions described in Fig 3.3. It is a detailed description of how the application works.

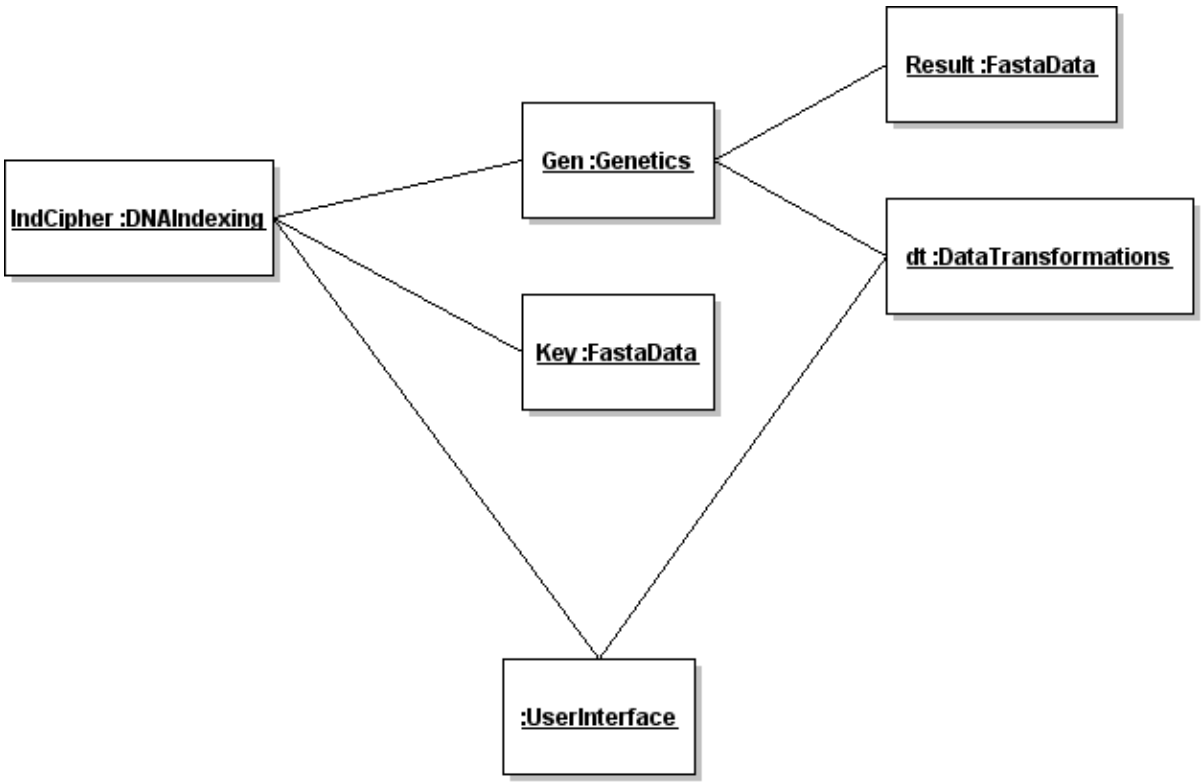


Figure 3.5 Object Diagram

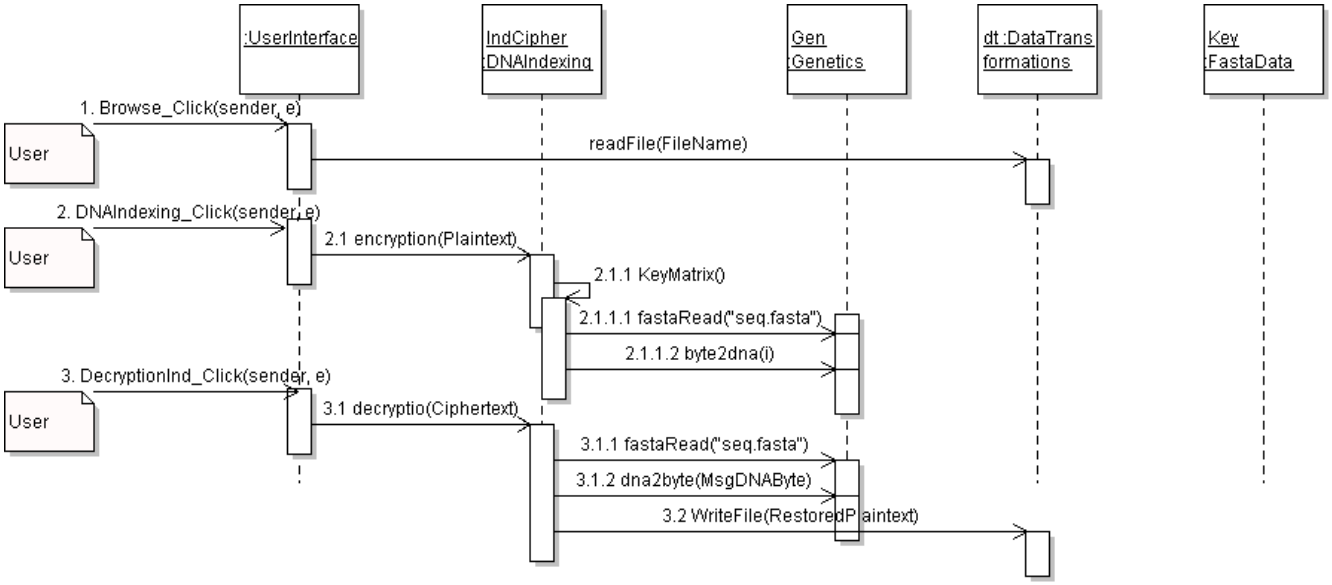


Figure 3.6 Sequence Diagram

The development of the DNA Indexing cipher was performed using Microsoft Visual C# 2010 Express. It was designed to operate on text and image files. In order to browse the system for an image file a dialog box was displayed using the *ShowDialog()* method of the *OpenFileDialog* class in the *System.Windows.Forms* namespace. The *Filter* property of this class was used to specify which image formats can be browsed. The *FileName* property was used to memorize in a string variable the path of the chosen image file. The *File.ReadAllBytes* method from *System.IO* namespace was used to get the bytes of the encoded bitstream of the image. All the functional code for realizing encryption and decryption was written in classes and methods described in the diagrams (Fig. 3.3- 3.6).

3.3 Performance Evaluation

3.3.1 Computational Complexity of the Algorithm

Complexity analysis of an algorithm is important because it reveals its efficiency for the real time applications. In this work the computational time of the algorithm was analyzed using complexity theory methods. The obtained conclusions were tested to be true through the implementation results.

The execution time of an algorithm is considered to be the sum of all the operations. The number of operations can be either constant or variable and depend on input parameters. According to the approximations from complexity theory, the smallest possible class of functions is used to express the growing rate of the algorithm's runtime [Has88]. Therefore, if the number of operations is for example 1 + 2n, then the complexity would be $O(n)$; if the number of operations is $4 + n + n^3$, then the complexity would be $O(n^3)$.

In this work complexity was analyzed for 3 important operations of the DNA Indexing algorithm: key dictionary computation, encryption, and decryption. The key dictionary is computed in *2*256*n* operations, where *256* is the number of possible values for a byte, and *n* is the length of the key sequence. Encryption and decryption are performed in *2*m* operations, where *m* is the number of plaintext and ciphertext words. Ciphertext has the same number of words as the plaintext. Taking the smallest class of functions, complexity for the key dictionary computation is *O(n)* and for the encryption-decryption process is *O(m)*. This means that the growing rate of the computational time is linear according to the input size. In Table 3.1 is presented the pseudo code of these operations.

Key Dictionary Computation:	Encryption:	Decryption:

m = length(SeqDNA)
next = 0
FOR X = 1 to 256
FOR Y = 1 to m
IF X = SeqDNA(y:y+3)
KeyDic[x][next] = y
next++;
END IF
END FOR
END FOR
	n = length(Plaintext)
	FOR X = 1 to n
	Index = RandomNo(1,
	length(KeyDic[x]))
	Ciphertext[x] =
	KeyDic[x][Index]
	END FOR
		n = length(Ciphertext)
		FOR X = 1 to n
		Index = Ciphertext[x]
		ByteOfPlaintext =
		SeqDNA(Index:Index+3)
		END FOR

Table 3.1 Pseudo code for basic operations of the DNA Indexing cipher

The experimental results have proved the correctness of the estimated complexity. In order to see the progression of the runtime, the program was executed at different, progressively increasing values of *n* and *m*. Fig. 3.7–3.9 presents graphics of the runtime growing rate for the key table computation, encryption and decryption processes. Some of the execution time measurements are presented in Tables 3.2– 3.4.

Key Length (nucleotides)	Runtime (ms)
1000	62
5000	359
10000	702
15000	1029
25000	1763
37000	2543

Table 3.2 Measurements of the key-table computation runtime

Plaintext Size (KB)	Runtime (ms)
74.6	5
214	16
419.9	31
720.4	49
957.4	63
1169.5	76

Table 3.3 Measurements of the encryption runtime

Ciphertext Size (KB)	Runtime (ms)
24.4	14
54.3	31
74.6	43
214	123
419.9	248
720.4	421

Table 3.4 Measurements of the decryption runtime

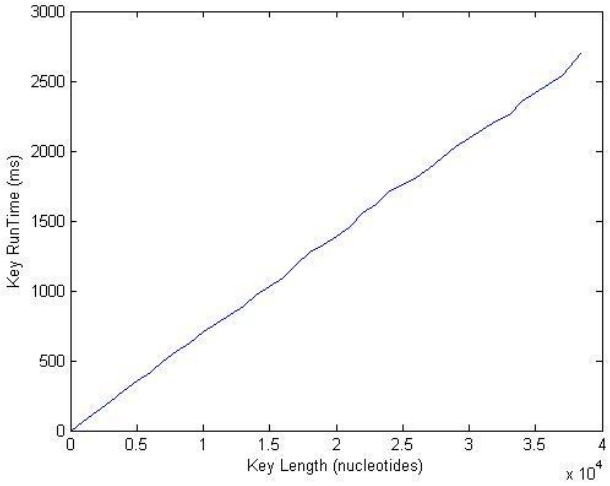


Figure 3.7 Growing Rate of the Key Table Computation Runtime

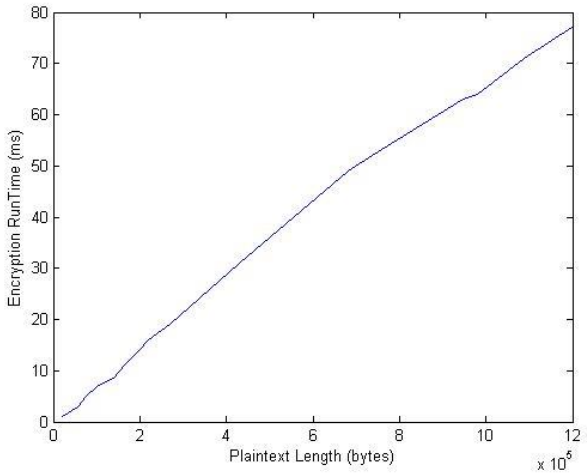


Figure 3.8 Growing Rate of the Encryption Runtime

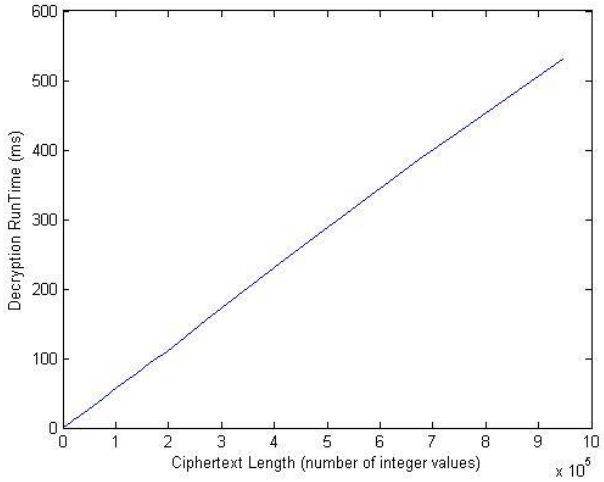


Figure 3.9 Growing Rate of the Decryption Runtime

3.3.2 Security Level of the Algorithm

Security of the algorithm was analyzed using different approaches, like: statistical measurements, cryptanalytic attacks, key space analysis and secure transmission of the ID number.

A. Statistical Measurements

Histogram was used to analyze the statistical probability distribution of the ciphertext. It was compared to the histogram of the plaintext. In cryptography it is important that the distribution of the ciphertext will not contain patterns of the plaintext distribution and a more uniform distribution of the ciphertext offers a better security. Implementation results (Fig. 3.10) shows that the distribution of the ciphertext is random and it doesn't contain patterns of the plaintext distribution.

Other statistical measurements used in this work to measure security strength of the ciphertext are *entropy* and *correlation coefficient* (CC). Entropy measures the uncertainty and randomness of the signal. Thus, entropy of the ciphertext is desired to be high. CC indicates the degree of correlation between neighboring values like pixels or letters; its value is intended to be small for the ciphertext. From our measurements (Tables 3.5–3.6) the ciphertext entropy is almost twice as high as the plaintext entropy and CC of the data is in average three times smaller after encryption.

Plaintext	Entropy of the	Entropy of the
Images		
Mandrill.png	5.81	11.8
Lena.jpeg	6.72	12.7
Lena.tiff	7.42	13.8
Geometry.png	3.7	8.3
Text files		
Text 1	4.61	8.39
Text 2	4.42	8.21

Table 3.5 Plaintext and ciphertext entropy measurements

Images	Correlation Coefficient	Correlation Coefficient
Images		
Cameramen.tiff	0.99	0.36
Lena.jpeg	0.99	0.29
Lena.tiff	0.97	0.32
Geometry.png	0.35	0.13

Table 3.6 Plaintext and ciphertext correlation coefficients

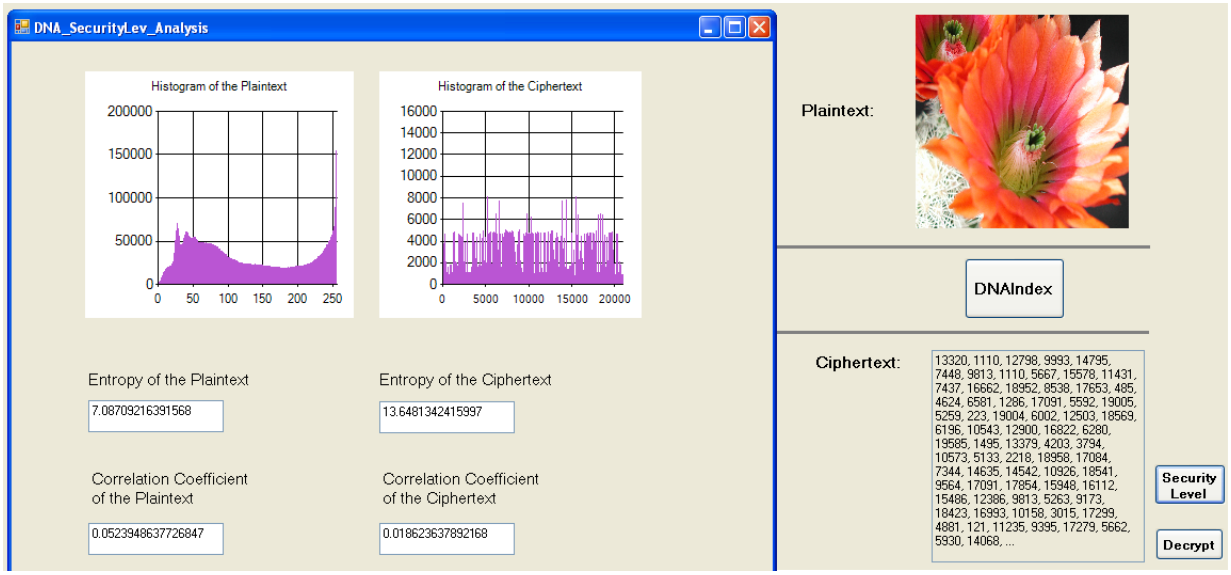


Figure 3.10 Examples of Plaintext, Ciphertext and Their Statistical Measurements

B. *Cryptanalytic Attacks*

DNA Indexing algorithm is based on the OTP principle which protects it from vulnerability to most of the classic attacks. Resistance to the known-plaintext attack is given by the fact that each plaintext byte has not just one, but a range of corresponding ciphertext values. Sample of known plaintext-ciphertext pairs needs to be very large to make this attack more successful. Ciphertext-only attack exploits patterns in the current ciphertext, disposing a set of previous ciphertexts. Due to the OTP principle, plaintext bytes will always have a different range of substitution values from one encryption to another, thus a set of ciphertexts or plaintext-ciphertext pairs from previous encryptions are not useful for breaking the current ciphertext.

An interesting study, as future work, would be a related-key attack on the DNA Indexing algorithm. The keys for this encryption method are DNA sequences from different databases. Similarities between different sequences can be analyzed using algorithms based on string kernels [MMP09]. According to the amount and length of the repeating intervals between two random key sequences, a certain level of vulnerability can be established to this kind of attack.

C. Key Space Analysis

The key space of an encryption algorithm is desired to be as large as possible in order to resist brute-force attacks. If the key is a sequence of bits then, its space will be $2^{length(Key)}$. Trying all the possible keys will give $2^{length(Key)}$ number of attempts to obtain a successful break. On average the correct answer can be found in half this number of tries.

In case of DNA Indexing the key is composed of two parts: genetic sequence and its ID number. The key for encryption-decryption is a genetic sequence. In this case, to try of all the possible keys means trying all the genetic sequences from a database. One of the databases can be the NIH genetic sequence database named GenBank. It is a collection of all publicly available DNA sequences [wncbi]. It contains approximately 135 440 924 DNA sequence records. Trying all these sequences will be equivalent to 2^{27}, which means a key of 27 bits. On the other hand, genetic sequence is long, as mentioned in section 2, it should be at least 30 000 bases. The alphabet of the key-sequence is composed of 4 letters: A, C, G, and T. Thus, in case of using a private database, trying all the possible keys becomes a number of $4^{30,000}$. There is also a possibility to create a large publicly available database; in this case the key space is equivalent to the number of sequences in the database.

D. Secure Transmission of the ID Number

Secure transmission of the sequence ID number is important when a public database is used. If the database is public, access to the ID number means a direct access to the key-sequence. The length of the ID number in GenBank is 6 – 8 characters, which means up to 64 bits. A block of 64 bits can be encrypted with a traditional symmetric algorithm, like DES, and then sent through an existing encryption channel, using a previously exchanged key. The accession number can also be sent using public-key cryptography. Anyone who has a copy of a public key can easily encrypt information that only can be read with the private key. Communication involves only public keys, and no private key is ever transmitted or shared. This type of key distribution is used in PGP [PGP04] and in many other systems, due to the facility of public key distribution.

3.3.3 Comparison with another cipher of similar principle

The DNA Indexing encryption algorithm [TBH+10] analyzed in this work was developed based on the principle presented in [ASE06]. This principle consists in using the chromosomal sequence as the key and performing a substitution procedure of the plaintext character with the index position retrieved from the key. The innovations brought forward in [TBH+10] were to consider the benefits of genomic databases for the OTP method and computation of the key table. In what follows the time performance of the key table computation is compared to the method used in [ASE06], where in order to find a ciphertext value to replace a byte of the plaintext a search was used each time through the chromosomal sequence starting at a random position.

The number of operations required for key table computation of the DNA Indexing algorithm was computed earlier to be: 256*2*n. The number of operations used in previous algorithm in order to perform one search, which means to encrypt one byte, is equivalent to the average appearance of that byte in the key. Based on the experimental results the following assumptions were made: for a genetic sequence of length 38000 bases the number of appearances of a byte varies between 15 and 220 and in average it would be 100. Thus one search can be roughly equivalent to about (38000/100)*2 = 380*2 operations; where the number 2 represents the same operations as for DNA Indexing algorithm. Depending on the size of the message this method can be faster or slower than the key table computation proposed in the DNA Indexing algorithm. An example is presented below, where DNA Indexing is named "Alg. X" and the previously designed: "Alg. Y".

Key length: 38000 bases

Plaintext size: 1 KB
*Alg. X: No of operations = 256*2*38000 = 19 456 000*
*Alg. Y: No of operations = 380*2*1024 = 778 240*

Plaintext size 30 KB
*Alg. X: No of operations = 256*2*38000 = 19 456 000*
*Alg. Y: No of operations = 380*2*30*1024 = 23 347 200*

Therefore DNA Indexing algorithm is preferred in case of large messages to encrypt and the previously designed algorithm for shorter messages. In case of real time encryption and transmission DNA Indexing has an advantage because the table of all the possible substitution values is computed before the actual encryption. In this way the number of operations during encryption is equal to the number of bytes in the plaintext.

3.4 Modifications of DNA Indexing Cipher for Better Compression Ratio and Security

DNA Indexing cipher was analyzed from the compression point of view. Encryption was applied on the compressed bitstream of the plaintext in the form of direct encryption (subchapter 2.2.4), (Fig. 2.9). Then, compression ratio of the ciphertext was compared to that of the plaintext. Because the change in compression ratio was significant, one of the major concerns was to optimize the algorithm in order to obtain a better compression of the data. Some improvements were introduced as modifications to the original algorithm of encryption. Also a scheme of joint compression - encryption processes was designed and developed. The rate – distortion function was computed and compared for different number of substitutions per value. The original, joint and distributed joint compression – encryption schemes where visually compared through PSNR – R function.

3.4.1 Improvements of Direct Encryption

Complete encryption of the image data was performed on the encoded bitstream, using the scheme described in subchapter 2.2.4 and shown in Fig. 2.9. When compression and encryption need to be applied on the multimedia data, usually encryption is performed as a second process because it changes the probability distribution of the data samples, making their appearance as equal and uniform as possible. If all of the data samples are equally probable then there is no information that can be compressed by entropy encoding. Thus, before encryption, the first step was to obtain the compressed bitstream of the image.

Encryption was applied on the bit array of the compressed plaintext and performed by transforming each byte of the plaintext into an integer value. The output of the encryption process, the ciphertext, is an array of integer values, as was exemplified in Fig. 3.2 – 3.10.

The bitstream of the ciphertext can be represented in two forms: using a fixed number of bits for each integer value, or using the exact number of bits for each codeword and then memorizing their flags which specify the number of bits of each word. For the first method, where a fixed number of bits is used for each codeword, the number of bits assigned to a codeword is given by the maximal value of a codeword. Thus, if the length of the secret key (DNA sequence) is, for example, 37839; then the maximal possible value of a codeword is that value, so the number of bits needed to represent a value from the ciphertext is 16. Considering the fact that each plaintext byte is transformed in an integer number of 16 bits, the ciphertext bitstream will be twice as long as the plaintext bitstream.

What follows is an example where the original bitmap image is compressed to jpeg files using different compression ratios (CR). The aim of this example is to show the reduction in size of an image due to compression. The change in compression ratio (CCR) metric (subchapter 2.4.3) is used to estimate the increase in size because of the encryption with DNA Indexing cipher.

Jpeg images:

Bitmap image:



Size: 257 KB (263 222 bytes)

Dimension: 512 x 512 pixels

$$R = \frac{Size}{Dimension} = \frac{263\ 222}{262\ 144} \approx 1\ ^{byte}/_{pixel}$$

Compression :

1



Size: 33.1 KB

Dim: 512 x 512 pixels

$$R_1 = \frac{33\ 951}{262\ 144} \approx 1\ ^{bit}/_{pixel}$$

$$CR_1 = \frac{263\ 222}{33\ 951} \approx 7.75$$

2



Size: 7 KB

Dim: 512 x 512

$$R_2 = \frac{7\ 167}{262\ 144} \approx 0.03\ ^{bit}/_{pixel}$$

$$CR_2 = \frac{263\ 222}{7\ 167} \approx 36.7$$

CCR after encryption:

Image 1: $\frac{Size\ Plaintext}{Size\ Ciphertext} = \frac{33\ 951}{60\ 263} \approx 0.56$ $\quad CCR_1 = \frac{60\ 263 - 33\ 951}{60\ 263} * 100\% = 43.6\%$

Image 2: $\frac{Size\ Plaintext}{Size\ Ciphertext} = \frac{7\ 167}{12\ 542} \approx 0.57$ $\quad CCR_2 = \frac{12\ 542 - 7\ 167}{12\ 542} * 100\% = 42.8\%$

From the point of view of compression efficiency the increase in data size after encryption is of major interest. The value of a codeword from the ciphertext can vary between one and the length of the key and its number of bits accordingly. In this work a DNA sequence of length 37839 was used, so the number of bits of a codeword can be in the range: 1 – 16. By running the encryption algorithm a few times for different image files and computing the average number of bits for a ciphertext codeword showed that it is around 14 bits. The most probable number of bits for a codeword is 15. Here are some probabilities of

codeword lengths: P(15) ≈ 0.45, P(14) ≈ 0.2, P(16) ≈ 0.1. Considering all the lengths of the codewords the average number was found to be 14 bits. This means that 8 bits of the plaintext are transformed, approximately, in 14 bits of the ciphertext. The increase in size in this case is 1.75 times. Considering the probabilities of codeword lengths, a fixed number, of 16 bits, is more suitable for representation of the integer values.

3.4.1.1 Change of Indexes in Key Dictionary

The number of bits for a ciphertext codeword can be optimized. The dictionary of substitution values is computed before actual encryption. In this key-table all the bytes (256) have a vector of corresponding integer values. These integer values are given by the positions where the byte value sequence (like "ACTT") is equal to the same sequence in the key. This integer values are used as ciphertext words for the substitution of the byte from the plaintext.

As transmitter and receiver must compute the same dictionary for the key, then the position of each integer value inside the key table is known. Thus, there is no need to transmit large integer values of 16 bits. Their position number can be sent instead. This will result in sending fewer bits for each ciphertext word (Fig. 3.11).

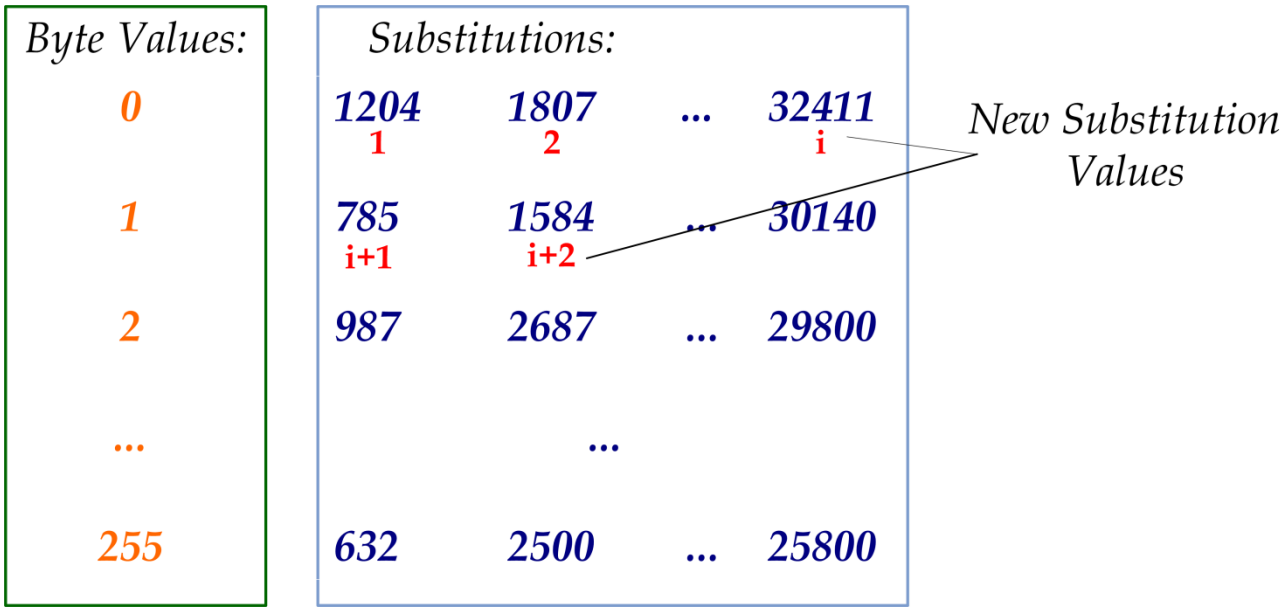


Figure 3.11 Key Dictionary: for each byte there is a range of integer substitution values; their positions may be used as new substitution values

In this case the length of a codeword will depend on the total number of substitutions in the dictionary. Limiting the number of substitutions will lead to a better compression ratio. In the previous section, where direct encryption was described, it was concluded that one codeword has the size of 16 bits. Considering that 32 different substitution values for 1 byte are enough, the total number of substitution values in the dictionary will be 8192 (256*32). This means that new substitution values (Fig. 3.11) can be represented on 13 bits each ($8192 = 2^{13}$). Making a tradeoff between security and compression, each codeword can be represented in less number of bits by reducing the total number of substitutions in the dictionary. For example, with a 2048 (2^{11}) number of substitutions in the dictionary, which is 8 different substitutions per 1 byte, the codeword length in bits is 11.

3.4.1.2 Homophonic Substitution

Another improvement for this cipher is to attribute a different number of substitution values for each byte according to the frequency with which they appear in the plaintext.

Distribution of the encoded (compressed) data bitstream is quite random and with less patterns then in the decoded or original signal bitstream, but there may still be some bytes appearing more often than others (Fig. 3.12).

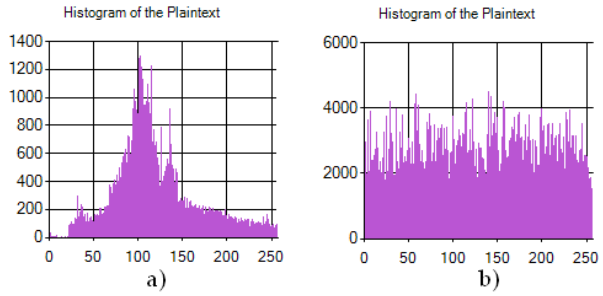


Figure 3.12 Distribution of the Encoded Data Bitstream a) tiff, b) jpeg

This means that for a more uniform ciphertext, number of substitutions attributed to a byte must vary according to its appearance in the plaintext. Originally in DNA Indexing cipher the number of indexes for substitution of each byte was random. It depended on how often a certain byte sequence was retrieved in the key sequence (Fig. 3.13). Now for each byte there is a certain number of corresponding indexes for substitution; this number corresponds to the appearance probability of that byte in the data. A highly probable value in the compressed data bitstream has more corresponding substitution values than a less probable value (Fig. 3.13). The number of substitutions for each byte in the dictionary is equal to the product between the total number of substitution in the dictionary and the

probability of each byte. This is the principle of homophonic substitution and the result of its application on the dictionary is a more uniform distribution of the ciphertext (Fig. 3.14).

KD before:		KD after:		
Plaintext bytes:	Substitutions:	Plaintext bytes:	Prob.:	Substitutions:
1	201, 314, 578, 874, 922, … 30587	X_1	0.5	161, 455,621, 1023, 4710, 5287, … 34584
2	1456,3015, … 28445	X_2	0.2	7065, 4052 … 29478
3	161, 455,621, 1023, 4710, 5287 … 34584	X_3	0.15	1456, … 28445
…	…	…		…
256	932, 7065, 4052 … 29478	X_N	0.001	28445

Figure 3.13 Number of Substitution Values for each Byte of the Plaintext

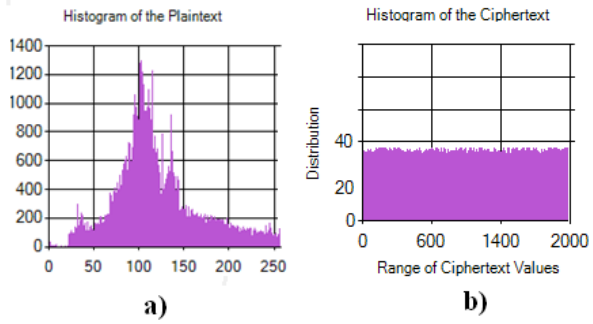


Figure 3.14 Distributions of the a) Plaintext and b) Ciphertext Values

3.4.1.3 Encryption and Compression through Longest Common Subsequences in DNA Sequence

A modification was considered on the DNA Indexing direct encryption. Instead of substituting each byte of the plaintext with an index in the key DNA sequence, the Longest Common Subsequence (LCS) principle was used. The steps of this method are the following:

- The compressed bitstream of the plaintext data is transformed in DNA sequence using conversion from Table 2.1.
- The longest common sequences between the plaintext and key DNA sequences are searched.
- The starting points of these plaintext sequences and their length were used to substitute them in the ciphertext.

In Fig. 3.15 an example of the method is given. It shows matches between 2 sequences. The numbers in the table show how long the common sequence is. The encryption starts from the last column of the table: 16. In this column the maximum number is searched which is 3 and it shows the longest sequence in this place between the key and the plaintext. The number 3 is used to compute the starting point of this match (14 – 3 + 1) and to indicate the length. The process of finding the max number in the column repeats in the 13th (16 – 3) column for the next longest common sequence.

Data → / Key ↓		1 C	2 G	3 A	4 C	6 T	7 C	8 A	9 G	10 G	11 C	12 A	13 T	14 C	15 G	16 A
1	A	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
2	C	1	0	0	2	0	1	0	0	0	1	0	0	1	0	0
3	T	0	0	0	0	3	0	0	0	0	0	0	1	0	0	0
4	C	1	0	0	1	0	4	0	0	0	1	0	0	2	0	0
5	G	0	2	0	0	0	0	0	1	1	0	0	0	0	3	0
6	G	0	1	0	0	0	0	0	1	2	0	0	0	0	1	0
7	C	1	0	0	1	0	1	0	0	0	3	0	0	1	0	0
8	A	0	0	1	0	0	0	2	0	0	0	4	0	0	0	1
9	T	0	0	0	0	1	0	0	0	0	0	0	5	0	0	0
10	A	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
11	C	1	0	0	2	0	1	0	0	0	1	0	0	1	0	0
12	C	1	0	0	1	0	1	0	0	0	1	0	0	1	0	0
13	G	0	2	0	0	0	0	0	1	1	0	0	1	0	2	0
14	A	0	0	1	0	0	0	1	0	0	0	1	0	0	0	3

Start

Figure 3.15 Example of Encryption with LCS in DNA Sequences

Following the example in Fig. 3.15, encrypted data is represented in pairs of:

- (Position in the key, number of elements to read from that place)

  (4, 2); (1, 3); (7, 2); (5, 5); (12, 3)

For the experiment a key of 16 000 letters was used; meaning that the position in the key was represented with 14 bits and the average length of common subsequences was around 7-8 letters which is 3 bits. This means that 6 - 8 letters of data (12 - 16 bits) after encryption became 17 bits. This is an improvement from point of view of compression for the DNA Indexing algorithm, but the computational time is very high.

3.4.2 Joint Compression – Encryption

Joint Compression – Encryption is a concept that unifies two independent processes: compression and encryption in one single process. The general purpose of this approach is to reduce the amount of resources and computational time needed for the secure transmission of the multimedia data. The state of the art techniques for the joint compression – encryption are presented in subchapter 2.2.4. These methods are based on the integration of the ciphering process inside the entropy coder. In this work the DNA Indexing cipher was modified to be included in the compression process (Fig. 3.16).

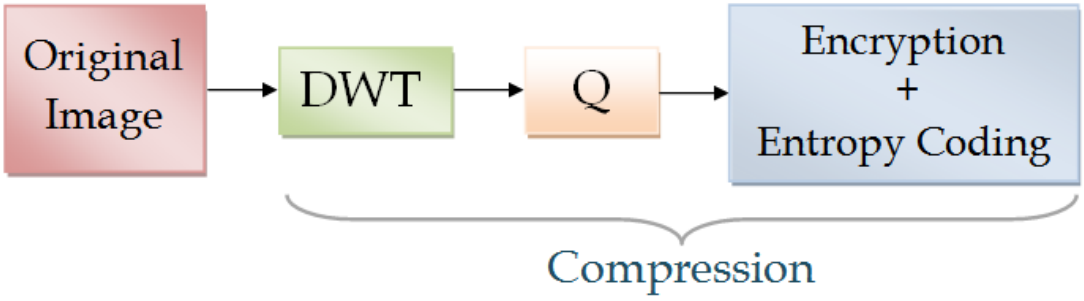


Figure 3.16 Joint Compression – Encryption Scheme

An analysis block was introduced to determine the desired level of quality and security (A(λ,p), Fig. 3.17). This block performs bit allocation (subchapter 3.4.2.1) by choosing Lagrange multiplier (λ) values and it sets a security level by choosing number of substitutions per value (p). Uniform scalar quantization is then performed according to the chosen parameter of quality. Huffman entropy coding technique is applied on the key dictionary of substitutions for the DNA Indexing cipher. Finally encryption is performed using the encoded key dictionary (Fig. 3.17).

At the reconstruction side, decoding of the key dictionary is the first step. Probability distribution is modeled using the shape parameter (α) and standard deviation (σ) calculated during encoding on the quantized data values (Formula 3.1). The same number of substitution values is used for the key dictionary reconstruction. Next, using the obtained key, the decryption of the data is performed. Obtained decrypted data goes through the rest of the decoding reconstruction stages of the quantization and wavelet transform (Fig. 3.17).

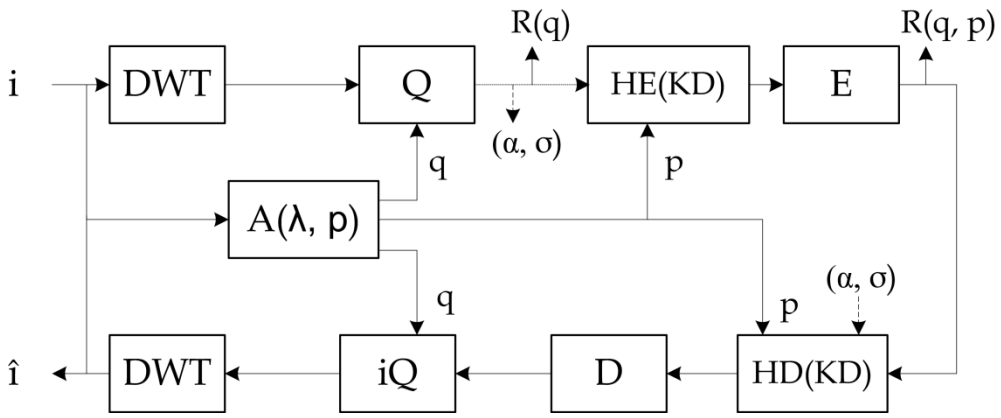


Figure 3.17 Scheme of the Joint Compression – Encryption Method

Legend of Figure 3.17:

i - original image

$\hat{\imath}$ – reconstructed image

DWT – discrete wavelet transform

Q – (mid-tread) uniform quantization

iQ – inverse quantization

q – quantization step

p – number of substitutions

(α, σ) - shape parameter and standard deviation describes the probability distribution

R(q) – bitrate depending on the quantization step

R(q, p) – bitrate depending on quantization step and number of substitutions in the key dictionary

A(λ, p) – analysis block used to determine quality and security level

HE(KD) – Huffman encoding of the key dictionary

HD(KD) – Huffman decoding of the key dictionary

E - block of encryption

D - block of decryption

3.4.2.1 Analysis Block - Bit Allocation

The mathematical model and processes inside the analysis block $A(\lambda, p)$ from the Fig. 3.17 are described in this subchapter. This block is used in the context of this joint compression – encryption method to determine the optimal bitrate according to the desired quality of the image and the level of security. Secure encoding – decoding process described in Fig. 3.17 was performed many times in order to obtain necessary values used as input to this block.

For the quality tradeoff the goal is to determine the quantization level that should be used for the quantization of the wavelet transformed subband images. For each quantization value there is a corresponding pair of Distortion – Bitrate (D - R) values.

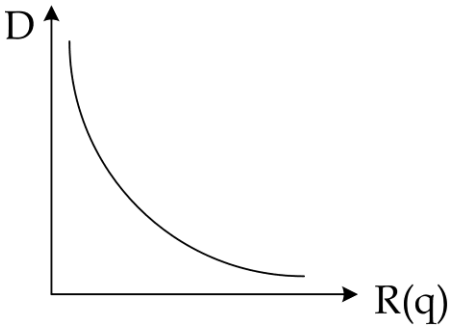


Figure 3.18 Rate – Distortion Curve According to the Quantization Step

The distortion (D) introduced by quantization was measured here as the mean squared error (MSE) between the original wavelet coefficients and those reconstructed at decoding after the inverse quantization. The distortion for a certain subband image k can be described in discrete domain as follows:

$$D_k(Q_k) = \frac{1}{n}\sum_{i=1}^{n}(X_i - Q_k(X)_i)^2 \tag{3.4}$$

where X_i are the values of the original wavelet transformed image coefficients, the $Q(X)_i$ are the reconstructed values, and n is the number of the values in the image.

In case of the orthogonal and biorthogonal wavelet transform, the total distortion is measured by the MSE between the original and reconstructed signal:

$$D = \sum_{k=1}^{N} \Delta_k \pi_k D_k\,(Q_k) \tag{3.5}$$

where N is the number of subbands, $D_k(Q_k)$ is the distortion of one subband, $\{\pi_k\}$ are the weight coefficients introduced by the filters [Use96]. For the orthogonal, non-redundant

wavelet transform, where the filters are normalized to the value of 2, there is $\pi_k = 1$ for all the subbands. The weight coefficients $\{\Delta_k\}$ are optional. They can be used to impose preferred frequency [ABR+93], or for certain performance optimizations in quality measurements [PTA+01], [PTAB+01].

The bitrate (R) value is affected by two parameters: the quantization step (q) used at encoding and the number of substitutions per value (p) used at encryption. Small values of the quantization step offer a better quality of the image, but increase also the R. When each byte from the plaintext can be mapped to many different values in the ciphertext, in order to represent binary all of that ciphertext values, a bigger number of bits per value is normally required. A higher security level is obtained when there are more substitution values, but it also increases the R. Considering the fact that at the end of the compression – encryption process the bitrate is affected by these two parameters, for a certain subband image *k* the bitrate is the following:

$$R_k(q_k, p_k) = \frac{1}{n}\sum_{i=1}^{n} length(c_i) \tag{3.6}$$

where $length(c_i)$ is the number of bits in a ciphertext codeword.

The average bitrate can be obtained by the weighted sum of the bitrates from each image:

$$R = \sum_{i=1}^{N} a_i \, R_i(q_i, p_i) \tag{3.7}$$

where $R_i(q_i, p_i)$ represents the average bitrate for the subband image i and $a_i = \frac{1}{2^{2i}}$ is the weight of this image according to the proportion of its size to the size of the whole image.

The optimization problem of rate distortion theory was described in subchapter 2.2.2. In this work the constrained R problem (min$\{D\}$) was substituted by the unconstrained Lagrangian cost function:

$$\min\{J\}$$

$$J = \sum_{i\,=\,1}^{N} \Delta_i \pi_i D_i(q_i) + \lambda \sum_{i\,=\,1}^{N} a_i \, R_i \, (q_i, p_i) \tag{3.8}$$

In order to determine the optimal quantization level the Lagrange multiplier (λ) selection is used. Its value is determined from the minimization of the Lagrangian cost function (J).

Supposing that *R* and *D* of an image are differentiable everywhere, the minimum of the J is given by setting its partial derivative with respect to R_i to zero:

$$\frac{dJ}{dR_i(q_i, p_i)} = \Delta_i \pi_i \frac{dD_i(q_i)}{dR_i(q_i, p_i)} + \lambda a_i = 0 \tag{3.9}$$

$$\frac{dD_i(q_i)}{dR_i(q_i, p_i)} = -\lambda \frac{a_i}{\Delta_i \pi_i}$$

Setting Lagrange multiplier (λ) to different values will give different bitrate – distortion values corresponding to certain quantization levels (Fig. 3.19).

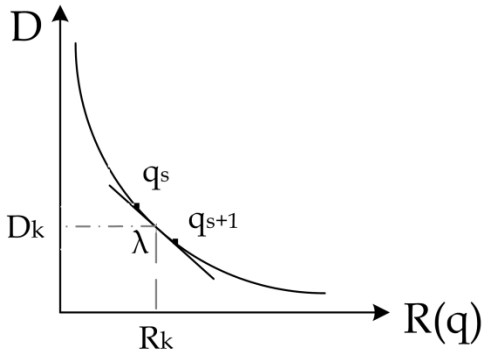


Figure 3.19 Lagrange Multiplier and Quantization Steps Values

According to the encryption algorithm used in this work, a higher number of substitution values in the dictionary increase the security level, but also the bitrate. Thus, the D – R curve will depend on the additional parameter: number of substitution (P) (Fig. 3.20).

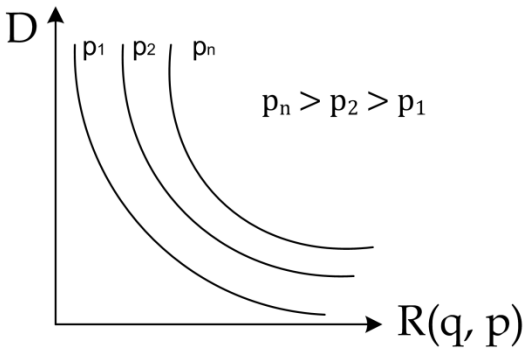


Figure 3.20 D(R) Curves according to the Number of Substitutions in the Key Dictionary

The range of values for the λ was obtained by performing the compression – encryption process for different values of the quantization step and number of substitutions.

Considering a fixed number of substitutions (p_i) the D and R values where calculated for a range of the quantization steps (q_s = *{1, 2, 4, 8, 16, 32}*). Next, a different number of substitutions was considered (p_{i+1}) and the D, R values were computed again for the same range of the quantization steps. Experimental results of these operations are presented in subchapter 3.4.2.3.

3.4.2.2 Entropy Coding of the Key Dictionary

The key dictionary (KD) of this cipher is a two dimensional array. Each row of it represents a vector of substitution values for a one single value in the plaintext. Values of the plaintext are quantized wavelet coefficients. The principle of finding substitution values and constructing the key dictionary was presented in subchapter 3.1. Change in representation of the substitution values was presented in Fig. 3.11. In this subchapter is presented how the entropy coding is applied on the key dictionary and the point where enciphering is included in the compression process.

The substitution values from the two dimensional array of the KD are assigned to the values of the plaintext. Quantized wavelet coefficients are the values of the plaintext. Entropy coding is applied considering the probability distribution model of the source. As there is a connection between the plaintext values and the key dictionary, the substitution values can be encoded using the encoder of its source.

Huffman coding technique was used for entropy coding. The general principle of encoding is that the substitution values for the most probable coefficients of the source are attributed shorter codes and accordingly the less probable coefficients are encoded with more bits (Fig. 3.21). During encoding

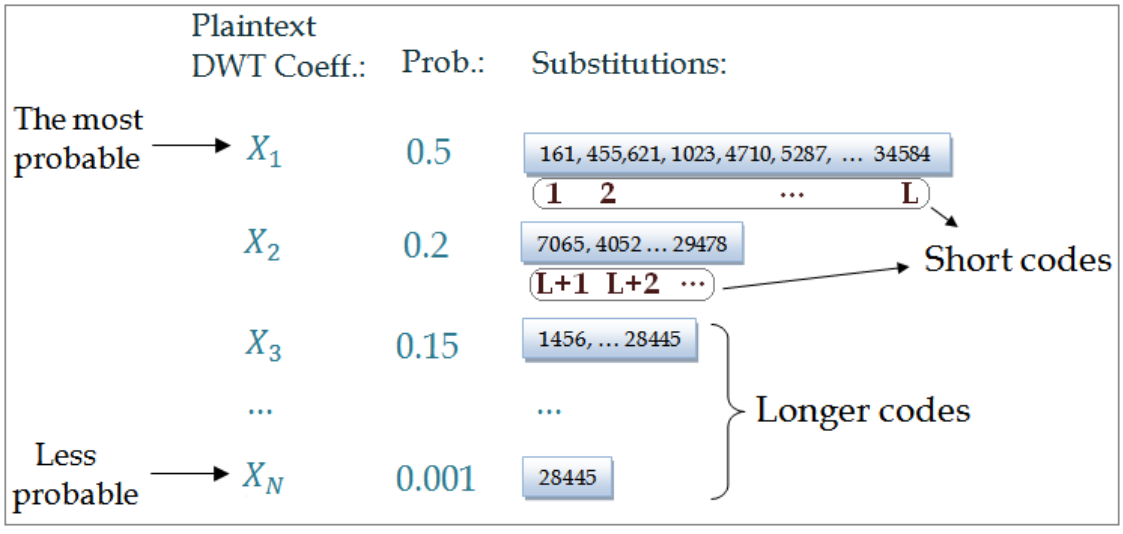


Figure 3.21 Huffman Coding of the Key Dictionary

The precise number of substitutions for each coefficient is computed according to its probability distribution and total number of substitutions in the dictionary.

$$NrS_i = NrS_{total} * P_i$$

where NrS_i is the number of substitution for the coefficient i; NrS_{total} is the total number of substitution in the KD; P_i is the probability distribution of the coefficient i

Change in bitrate from R(q) to R(q, p) (Fig. 3.22) appears due to the fact that there are more substitution values in the KD then the number of coefficients in the plaintext. The greater is the number of substitutions in the KD the more is the increase in the bitrate.

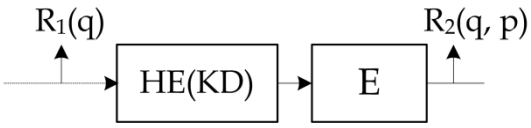


Figure 3.22 Change in the Bitrate due to the number of substitutions in the KD

The key dictionary was computed considering the model of the source. It was shown that the probability distribution of the wavelet coefficients can be approximated by the generalized Gaussian distribution [ABM+92], [Par03]:

$$p_{\sigma,\alpha} = \frac{A(\alpha)}{\sigma} e^{-\left|B(\alpha)\frac{x}{\sigma}\right|^{\alpha}} \tag{3.1}$$

where $(\alpha) = \sqrt{\frac{\Gamma(3/\alpha)}{\Gamma(1/\alpha)}}$, $A(\alpha) = \frac{\alpha B(\alpha)}{2\Gamma(1/\alpha)}$, $\Gamma(n) = (n-1)!$ is Gamma function

The important parameters in probability distribution of the coefficients are: the shape parameter (α) and standard deviation (σ):

$$\alpha \cong \frac{1.447}{lnK - 0.345}, \quad \sigma = \sqrt{E(X^2)} \tag{3.2}$$

where $K = \frac{E(X^4)}{E(X^2)^2}$ is the Kurtosis and $E(\cdot)$ is the expected value:

$$E(X^r) = \sum_{i=1}^{n} x_i^r P(X = x_i) \tag{3.3}$$

The shape parameter and standard deviation are sufficient information to compute the probability distribution of the source $X = x_i, \forall\, i \in [1, n]$ without knowing its actual values. Because the key dictionary is computed with consideration to this distribution, the receiver needs the α and σ parameters in order to compute the same key. The receiver of

the ciphertext can compute the same dictionary knowing the model (in this case generalized Gaussian distribution) (Fig. 3.23).

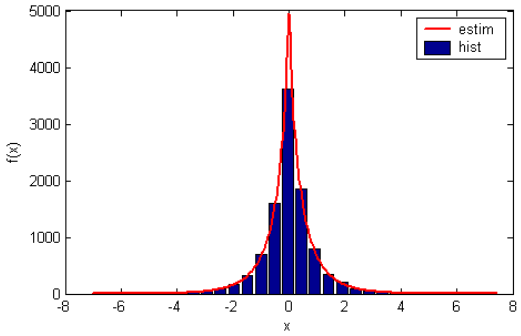


Figure 3.23 Histogram and Approximation of the Generalized Gaussian Distribution

3.4.2.3 Experimental Results of R(D) Functions

The experimental results were taken for the 512x512 grayscale lena.bmp image, taken from the [UCLA]. The bitrate and distortion were calculated for some of the wavelet transformed image subbands according to a range of quantization step and number of substitutions per symbol. A fix number of substitutions per value was considered for these experimental results. The following input values were used:

- Quantization steps: 1, 2, 4, 8, 16, 32
- Number of substitutions per symbol: 1, 2, 3, 5, 10
- Number of wavelet decompositions: 2
- Subband images: 0, 2, 4 (Fig. 3.24)

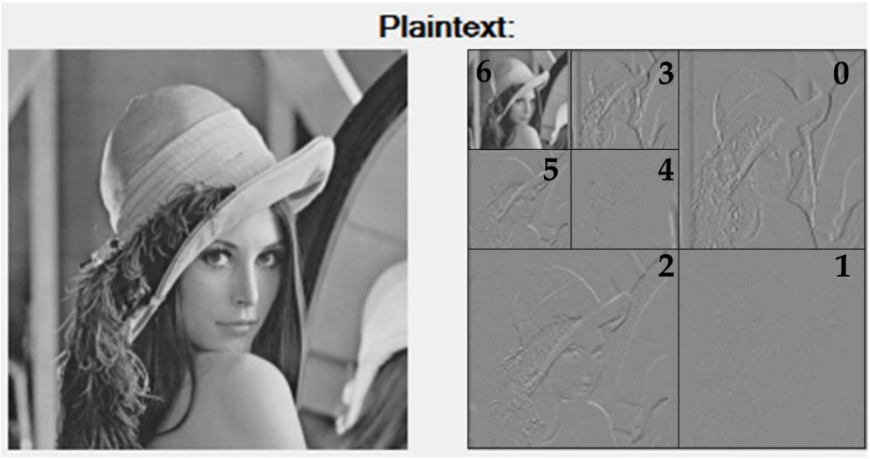


Figure 3.24 Wavelet Transform of the Test Image; Numeration represents Subband Images

Experimental results:

Subband Image 0: Qs is the quantization step; D is the distortion; $R(Q_s, 1)$, $R(Q_s, 3)$, $R(Q_s, 5)$, $R(Q_s, 7)$, and $R(Q_s, 9)$ correspond to the bitrate values depending on a range of Qs and number of substitutions which are: 1, 3, 5, 7, or 9.

Qs	**D**	$\mathbf{R(Q_s, 1)}$	$\mathbf{R(Q_s, 3)}$	$\mathbf{R(Q_s, 5)}$	$\mathbf{R(Q_s, 7)}$	$\mathbf{R(Q_s, 9)}$
1	0	4.4	5.77	6.23	6.99	7.12
2	0.42	3.07	4.92	5.61	5.97	6.32
4	1.12	2.07	3.7	4.57	5.04	5.37
8	2.96	1.43	2.72	3.69	3.97	4.55
16	6.97	1.15	2.28	3.1	3.36	3.82
32	14.5	1.04	2.07	2.84	3.1	3.5

Table 3.7 Rate – Distortion Values of Subband Image 0

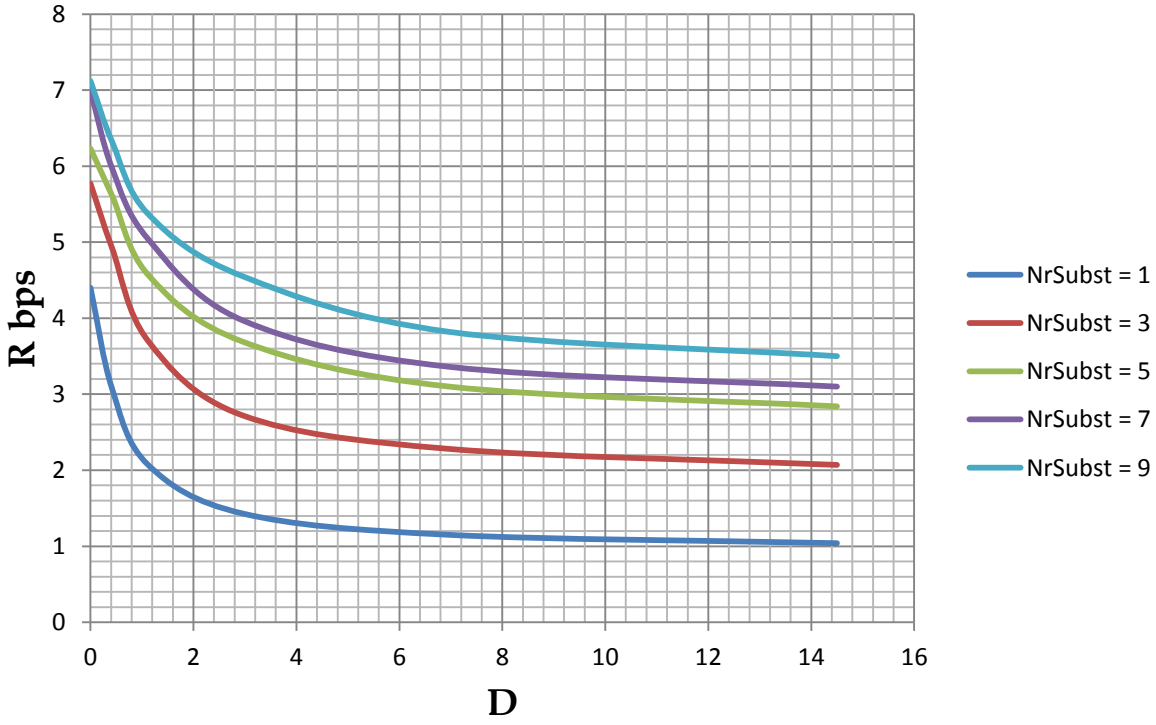


Figure 3.25 R(D) Curves of the Subband Image 0

Subband Image 2:

Qs	D	$R(Q_s, 1)$	$R(Q_s, 3)$	$R(Q_s, 5)$	$R(Q_s, 7)$	$R(Q_s, 9)$
1	0	3.89	5.54	6.41	6.6	7.19
2	0.42	2.75	4.58	5.26	5.75	5.92
4	1.09	1.76	3.29	4.42	4.73	5.04
8	2.56	1.25	2.48	3.35	3.62	4.14
16	5.3	1.08	2.15	2.93	3.19	3.62
32	9.44	1.02	2.03	2.78	3.04	3.42

Table 3.8 Rate – Distortion Values of Subband Image 2

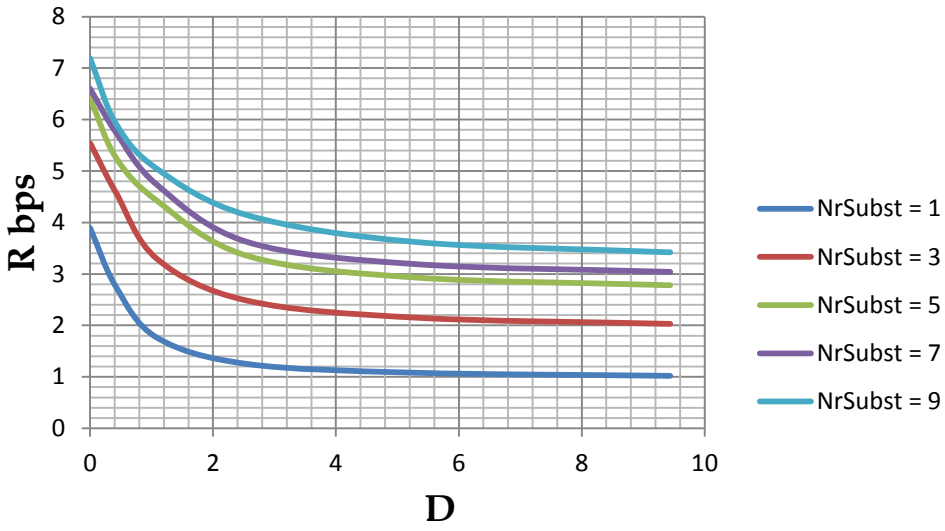


Figure 3.26 R(D) Curves of the Subband Image 2

Subband Image 4:

Qs	D	$R(Q_s, 1)$	$R(Q_s, 3)$	$R(Q_s, 5)$	$R(Q_s, 7)$	$R(Q_s, 9)$
1	0	3.57	5.51	6.35	6.49	6.79
2	0.31	2.24	3.98	4.57	5.16	5.5
4	0.7	1.52	2.85	3.83	4.13	4.73
8	1.73	1.19	2.36	3.2	3.46	3.95
16	3.84	1.06	2.11	2.89	3.15	3.57
32	7.19	1.01	2.02	2.77	3.02	3.4

Table 3.9 Rate – Distortion Values of Subband Image 4

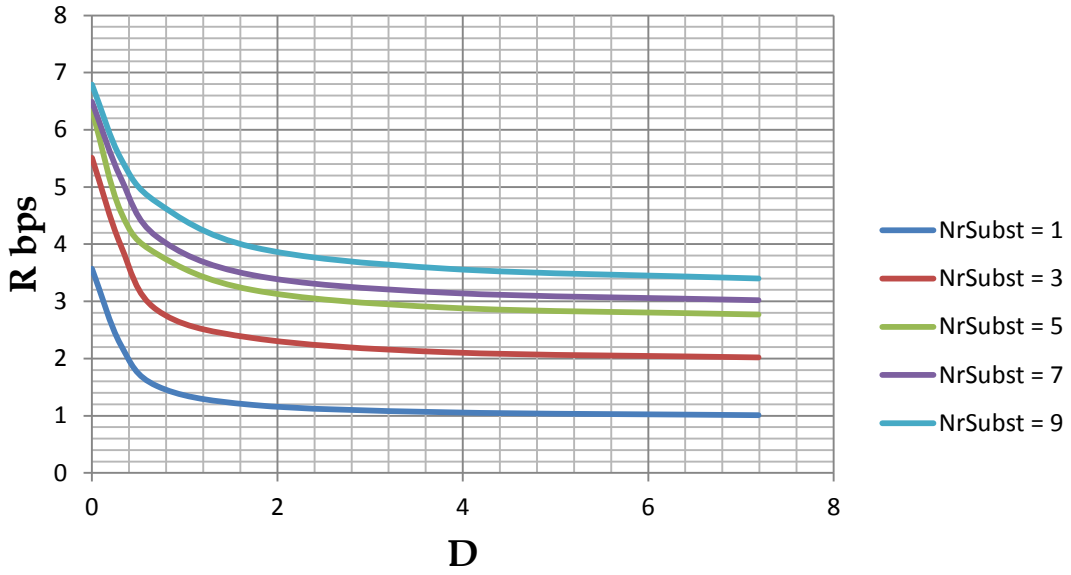


Figure 3.27 R(D) Curves of the Subband Image 4

3.4.2.4 Comparison of Compression Efficiency between different Versions of the Algorithm

According to the bit allocation described in subchapter 3.4.2.1 and bitrate (R) – distortion (D) values computed and presented in subchapter 3.4.2.2, Lagrange multiplier (λ) values were computed for the analyzed image. A range of possible λ values was established by computing the λ_{min} and λ_{max} values valid for each image subband. According to the selected value of $\lambda = -\frac{dD}{dR}$ an appropriate quantization step was chosen for each image subband. A vector of obtained quantization steps was further used to perform compression – encryption process on the image and compute its R, D and PSNR. Value of PSNR for the grayscale image was computed according to the following formula:

$$PSNR = 10log_{10}\left(\frac{255^2}{D}\right) \tag{3.10}$$

Curves of PSNR – R were computed for 3 different cases:

1. The original version of the algorithm with direct encryption. Number of substitutions in this case is variable according to the number of matches between the DNA sequence used as the key and the bytes values. In average each byte may have a 100 of substitutions (subchapter 3.3.3).

2. its modification in joint compression – encryption with 2 substitutions per value in each image subband
3. joint compression – encryption with a distributed number of substitutions per value in each image subband (Fig. 3.28)

In the case 3, number of substitutions per value in each image subband was chosen empirically according to importance of each image subband for the perception. Diagonal image subbands contain less important details about the image and thus were protected less then for example the subband image of low frequencies (Fig. 3.28). In order to determine the optimal number of bits per value in each image subband a separate study with more tests and analyses may be performed as future works.

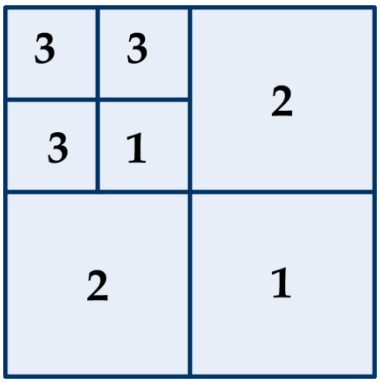


Figure 3.28 Number of Substitutions per Value allocated empirically for each Image Subband

Joint C-E with Fix Nr. of Substitutions		Separate C-E		Joint C-E with Distributed Nr. of Subst.	
R	**PSNR**	**R**	**PSNR**	**R**	**PSNR**
3.5	41.18	3.71	38.79	3.31	41.92
3.4	40.45	3.4	37.95	3.17	40.93
2.99	38.5	3.28	37.55	2.72	38.62
2.92	38.2	3.13	36.4	2.59	37.95
2.8	37.55	2.91	34.91	2.53	37.55
2.65	35.65	2.69	33.69	2.42	36.4

Table 3.10 PSNR – R Values of the 3 Compression – Encryption Algorithm's Versions

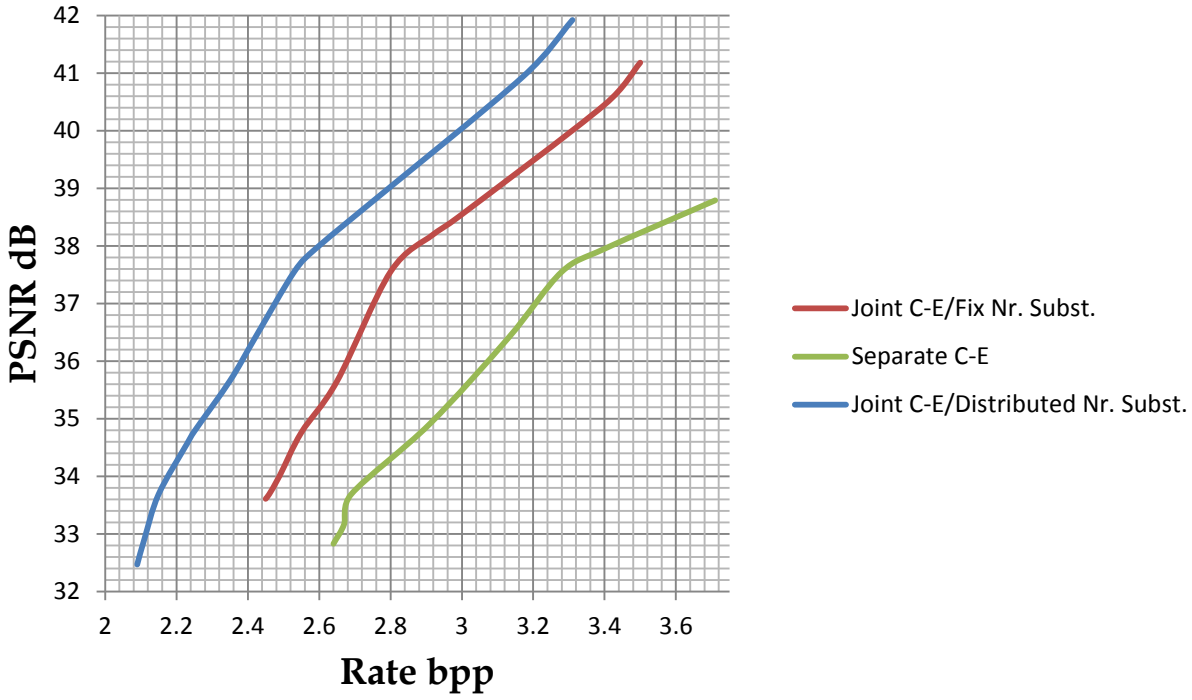


Figure 3.29 PSNR –R Curves of the 3 Compression – Encryption Algorithm's Versions

In Fig. 3.29 it can be observed that joint compression encryption scheme has a better compression at the same level of distortion than the separate, direct compression encryption version of the algorithm. Experimental result shown in Fig. 3.29 was obtained for one image, based on this result more tests and analyses may be performed on different images as future works.

Different number of substitutions per value was allocated empirically for each image subband, according to the importance of the information in the subband (Fig 3.28). It has been shown that this distribution of substitutions offers even better compression at the same level of distortion (Fig 3.29).

3.5 Contributions

Symmetric encryption method based on DNA sequences was presented here. It uses digital biological DNA from public genetic databases in order to retrieve sequences and use them for encryption. The DNA Indexing algorithm represents a practical application of DNA cryptography and brings a great advantage of using genomic databases; a feature that was not previously exploited in cryptography.

Computational time, security level and compression efficiency were analyzed for this algorithm. Computational time was found to be linear and statistical measurements showed a good security of the ciphertext and OTP principle protect it from most of the cryptanalytic attacks. Direct encryption of compressed image data bitstream showed that the size of the data increases twice after encryption. Certain optimizations were proposed and implemented to reduce the size by changing the algorithm. A tradeoff between security and compression can be established by the users due to this optimization. More substitution values in the dictionary lead to a higher security and less number of substitutions to a higher compression ratio.

This algorithm was developed using .NET(C#, WinForms). There is a graphical interface allowing the user to select text or image data, perform the encryption/decryption processes, and visualize the result. The developed program also offers visualization of statistical measurements for plaintext and ciphertext data intended for measurement of the security level. The software application was designed in Violet UML Editor. User, class, and sequence diagrams show its architecture from different points of view.

Contributions related to this cipher are described in the following publications: [TBH+10], [BT10], [VTT10], [TB13], and [TAB13]. Paper [TBH+10] presents a detailed description of the algorithm principle and exemplification of its application. In paper [BT10] it is integrated as part of the existing methods in DNA cryptography. In [TB13] results of its evaluation for the computational time and security level are presented. Paper [TAB13] shows results of compression efficiency evaluation and a part of optimizations.

Chapter 4

Generation of Random Sequences using Genetic Databases and proposal of OTP Cryptosystems

Contents in Brief

4.1 DNA Based Random Sequence Generation Method 74
4.2 OTP Encryption Systems for Transmission and Storage.....83
4.3 Performance Evaluation 87
4.4 Contributions 99

Pure random sequences are widely used in cryptographic applications for cryptographic keys [Sch96], [Sta11]. The difficulty is to generate such pure random sequences as well as the management of these keys (to transmit and store them securely). Instead of pure random sequences, in cryptography, pseudo-random numbers are used (PRN), generated from a seed. The adversary must not know the seed used for PRN generation. A more secure seed is the one that was generated only once. Thus, the use of OTP (one time pad) is of great interest in cryptography, where a key is only used once in a confidential communication and the length of the key is at least as long as the message in clear. Such a system was proved to be unbreakable [Sch96], [Sha49]. The main problems which occur in OTP implementation are: the great number of very long keys required and their management (transmission on trusty channels).

The randomness of DNA sequences, proved by the fact that they practically cannot be compressed [NW99], [CL+09], [FL+11], [DR+10], [RA11], can be used to generate long random binary sequences. The method of generating random binary sequences out of DNA sequences is presented in subchapter 4.2. Four different ways of obtaining the

desired length are proposed and also the format of the secret key which need to be shared between the users.

Subchapter 4.3 presents the block-schemes of OTP encryption systems for transmission and storage based on DNA random sequences. In subchapter 4.4 results of the performance evaluation are presented. Computational time was measured and computed for all of the componential processes of the encryption and respectively decryption, which is composed of the same processes. Statistical measurements were taken and cryptographic attacks were analyzed in order to determine security of the system. The last and 5th section of the chapter is dedicated to illustrate the advantages of the proposed DNA based method for random sequence generation, respectively OTP cryptosystems for transmission and storage.

4.1 DNA Based Random Sequence Generation Method

Random binary sequences based on DNA sequences (chromosomes, genes, etc.) can be provided by biological databases, the greatest being [ddbj], [webi], [wncbi']. Appearance of letters (A, C, G, ant T) in the DNA sequence belonging to a living organism (human, animal, plant) is random. The randomness of DNA sequence can be proved by the fact that it can hardly be compressed [NW99], [CL+09], [FL+11], [DR+10], [RA11].

4.1.1 Distribution of the DNA Sequence

Distribution of the DNA sequence can be observed by analyzing the existing compression techniques for these sequences. A highly compressible file will contain significant patterns; contrary, the smaller the compression ratio, the more uniform is the signal distribution. A signal with elements that have equal probability appearance cannot be compressed.

In [NW99] it is claimed that the protein sequences, which are constructed out of DNA sequences, are not compressible due to their highly compact representation and mutations. The general purpose compression algorithms have been proved to be unsuitable for protein and DNA [CL+09]. Compression of protein and DNA need to relay on their biological properties and even based on them, the results in [NW99] shows that the protein sequences are not compressible.

Organisms with short evolutionary distance may have significant similarities between them. In [CL+09] the idea of only storing differences to a reference sequence is presented. In [FL+11] and [DR+10] this idea with some improvements gave 0.35 bits per base storage

rate. In [RA11] it is sustained that they obtained the best DNA sequence compression of 1.58 bits/base. The DNA alphabet has 4 letters, so naturally one base can be stored in 2 bits.

Results from a variety of research papers indicate that DNA sequence is poorly compressible. A better compression ratio is obtained by comparing certain sequences of similar organisms. This means that a stand-alone sequence of DNA can be considered quite random.

4.1.2 Transformation from DNA to Binary

The genetic code has four bases: A – adenine, C – cytosine, G – guanine, T – thymine [CD+04] which can be converted into binary, using a uniform encoding:

Binary	DNA
00	**A**
01	**C**
10	**G**
11	**T**

Table 4.1 Conversion table from DNA to binary

These substitutions can be easily realized using switch-case selection of a programming language. This is how the obtained DNA sequence is transformed into a binary one. Before this step the actual DNA sequence need to be provided. One of the possible ways to access a DNA sequence is exemplified in the following subchapter.

4.1.3 Access to the DNA Sequence

DNA sequences can be obtained from the publically available genetic databases like [ddbj], [webi], or [wncbi']. What follows is one of the possible ways to access them. NCBI website gives the possibility to inspect genome information for a variety of species. Fig. 4.1 is presenting the request for the Zea mays (corn) genome. Fig. 4.2 shows the result to this request.

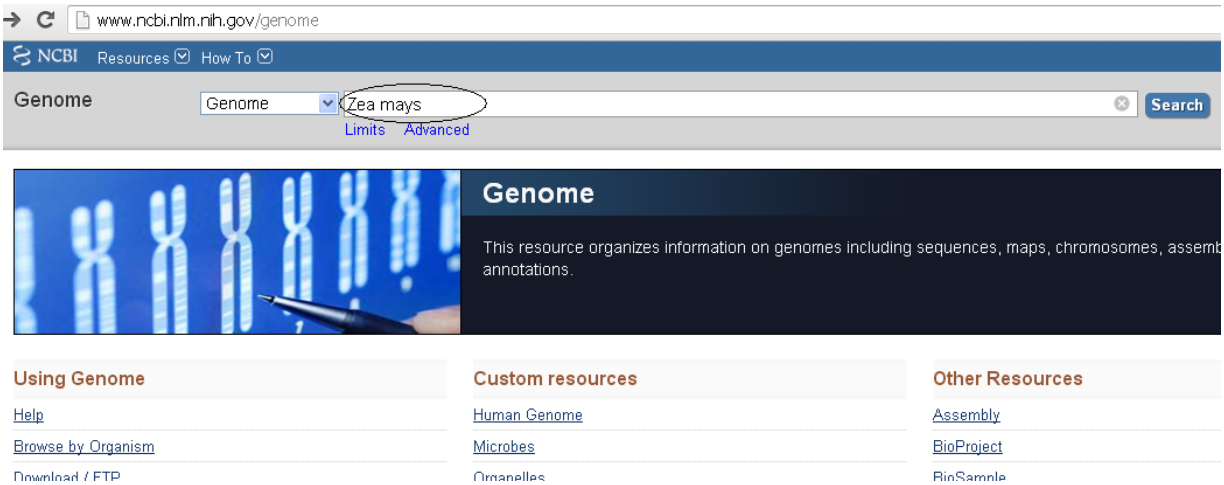


Figure 4.1 Requesting Genome Information for Zea Mays (corn) Species

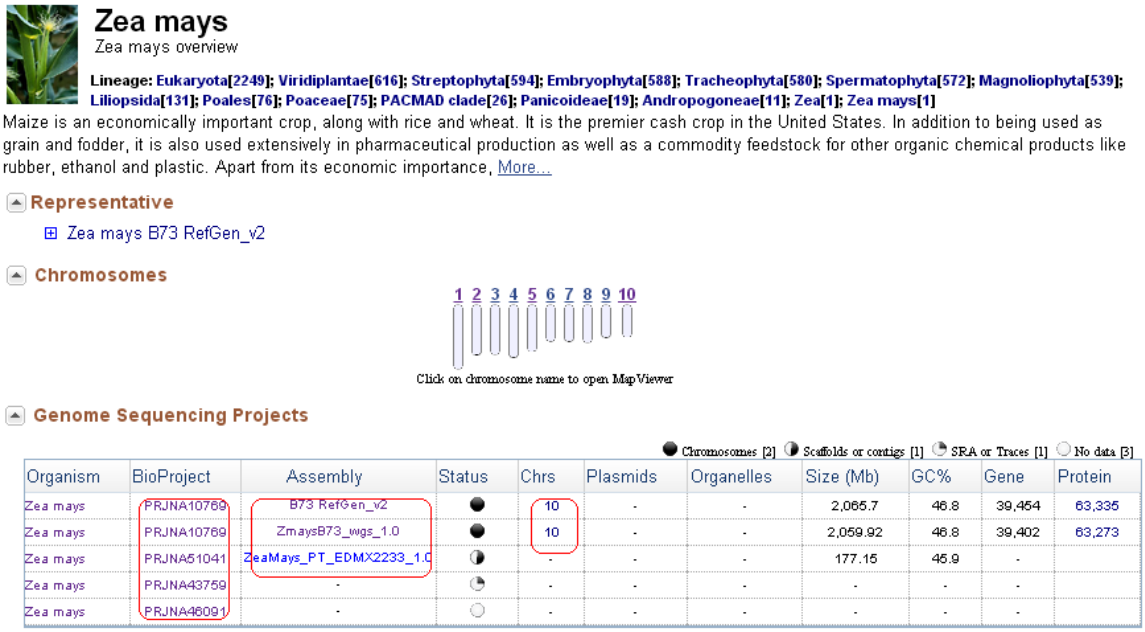


Figure 4.2 Genome Information about Zea Mays Species

Zea mays has 10 chromosomes, they can be seen by accessing links from BioProject, Assembly, or Chrs columns (Fig. 4.2). The result is shown in Fig. 4.3.

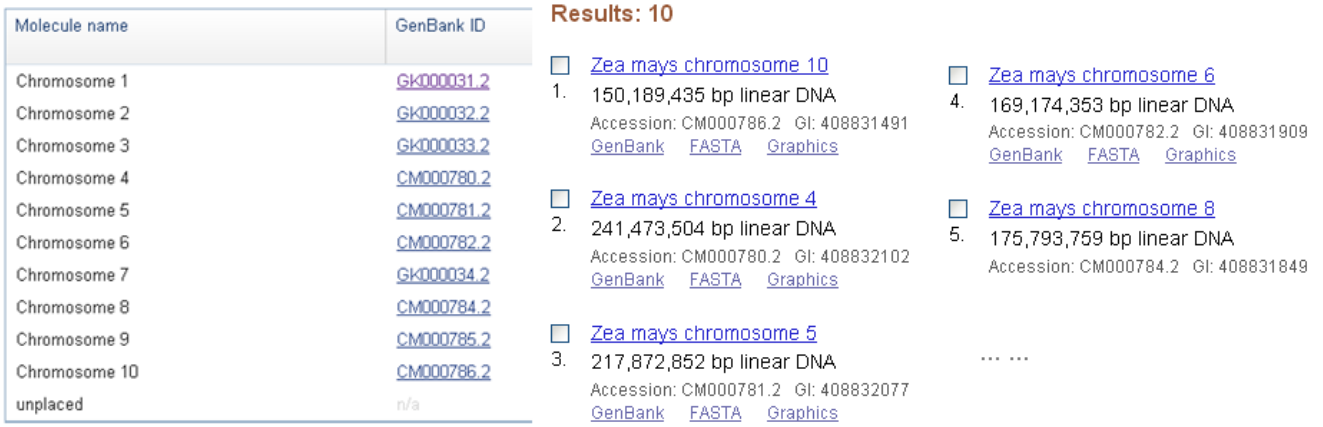


Figure 4.3 Chromosomes of the Zea Mays

Information about a certain chromosome sequence can be viewed in GenBank, FASTA, or Graphics format. In the send option the desired file format can be chosen (Fig. 4.4) and then the sequence can be saved for further processing.

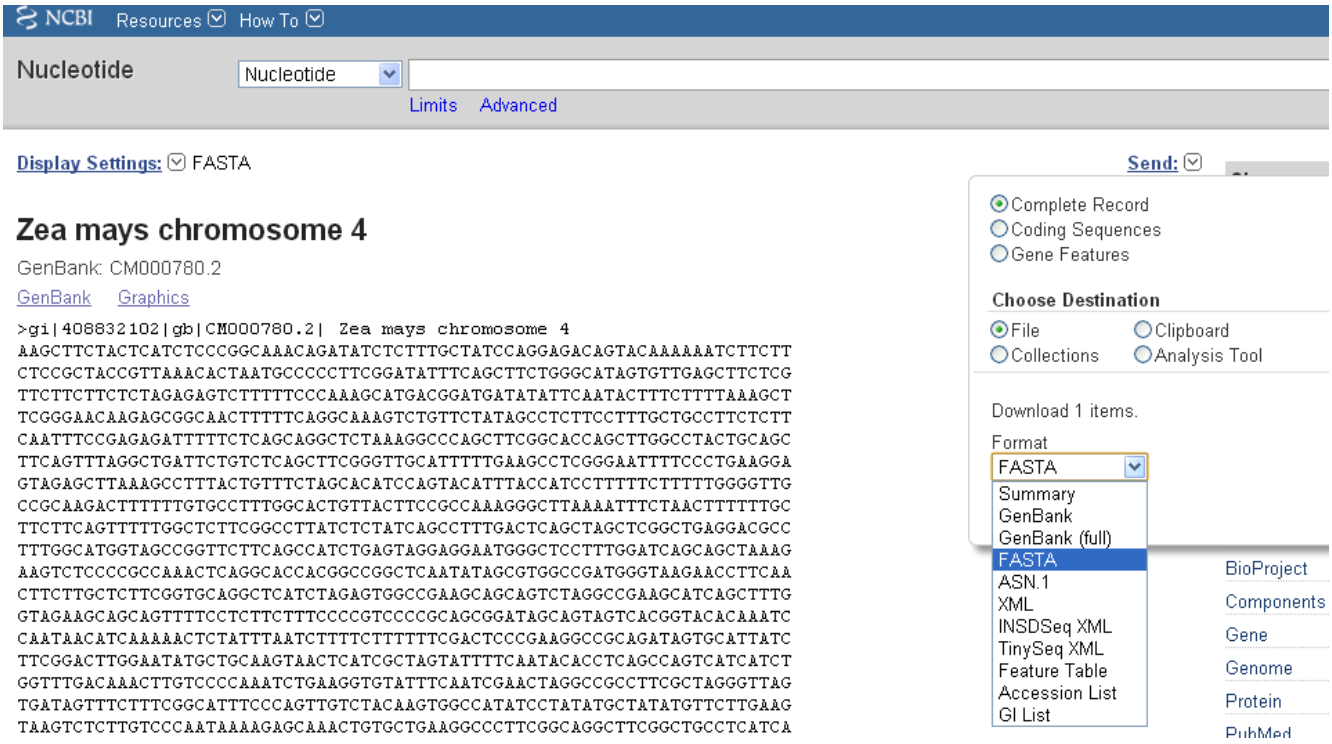


Figure 4.4 DNA Sequence in FASTA Format and the Download Options

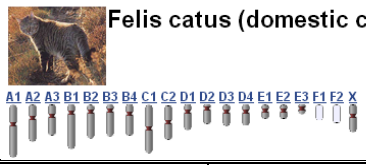
4.1.4 Length of the DNA Sequence

The lengths of DNA sequences in genetic databases are variable: from tens (a gene) to hundreds of millions (a chromosome) bases. Tables 4.2, 4.3, and 4.4 are giving lengths of DNA sequences (in base pairs - bp) for some living organisms.



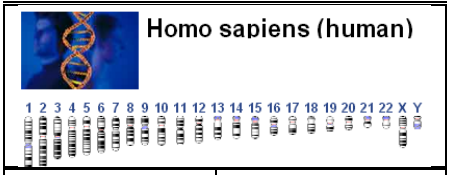
Chromosome	Length in bp
1	301,354,135
2	237,068,873
3	232,140,174
4	241,473,504
5	217,872,852
6	169,174,353
7	176,764,762
8	175,793,759
9	156,750,706
10	150,189,435

Table 4.2 DNA sequences lengths corresponding to Zea mays (corn) chromosomes



Chromosome	Length in bp
A1	239,302,903
A2	169,043,629
A3	142,459,683
B1	205,241,052
B2	154,261,789
B3	148,491,654
B4	144,259,557
C1	221,441,202
C2	157,659,299
D1	116,869,131
D2	89,822,065
D3	95,741,729
D4	96,020,406
E1	63,002,102
E2	64,039,838
E3	43,024,555
F1	68,669,167
F2	82,763,536
X	126,427,096

Table 4.3 DNA sequences lengths corresponding to cat chromosomes



Chromosome	Length in bp
1	249,250,621
2	243,199,373
3	198,022,430
4	191,154,276
5	180,915,260
6	171,115,067
7	159,138,663
8	146,364,022
9	141,213,431
10	135,534,747
11	135,006,516
12	133,851,895
13	115,169,878
14	107,349,540
15	102,531,392
16	90,354,753
17	81,195,210
18	78,077,248
19	59,128,983
20	63,025,520
21	48,129,895
22	51,304,566
X	155,270,560
Y	59,373,566

Table 4.4 DNA sequences lengths corresponding to human chromosomes

In order to obtain random sequences (RS) for cryptographic OTP applications, the sequence need to have a length of at least the same as of that of the cleartext message and to be used only once. For this reason, the length of the generated sequence needs to match the message length, which is defined by:

- Message type: - Text
  - Image
  - Sound (bitrate)
  - Video (bitrate)
- Transmission time for sound and video

These input data will determine the required length of the random sequence used as the key. Table 4.5 illustrates some examples of possible input data.

Text	Image	Sound	Video
15KB	402KB	5.34MB/278s (160kbps)	113 MB/1140s (audio 111 kbps, video 704 kbps)
639KB	573KB	4.74MB/248s (160kbps)	329MB/3359s (audio 112 kbps, video 695 kbps)
1.14MB	1.31MB	8.48MB/222s (320kbps)	349MB/2640s (audio 153 kbps, video 934 kbps)
207MB	3.44MB	3.16MB/195s (128kbps)	699MB/5708s (audio 99 kbps, video 909 kbps)

Table 4.5 Examples of different input data (obtained by measuring real files)

In genetic databases there are DNA sequences of very great length (chromosomes), acceptable for many applications. Table 4.6 exemplifies different cleartext messages, the length of the corresponding required DNA key (in bp), and the dimension of the corresponding secret key information [BTT+13]. The secret key will contain the identification numbers (IDs) of the DNA sequences used to create the final DNA sequence for the key.

Cleartext message	DNA sequence (key) length	Secret key information
Text - 15KB	61,440 bp	1 ID of DNA sequence (8 bytes)
Image - 402KB	1,646,592 bp	1 ID of DNA sequence (8 bytes)
Sound (160kbps) 5.34MB/278s	22,397,583 bp	1 ID of DNA sequence (8 bytes)
Video - 329MB/3359s (audio 112 kbps, video 695 kbps)	1,384,120,320 bp	~ 6 IDs of DNA sequences (48 bytes)

Table 4.6 Example of different cleartext messages, the corresponding DNA key length (bp), and respectively the secret key information

In order to obtain a high number of DNA sequences of great lengths, there are more possibilities:

a) one chromosome taken from a given database (see Tables 4.2 –4.4)
b) multiplexing, cycling (shifting) and concatenation of more sequences obtained from the same chromosome (Fig. 4.1)

Example:

Original sequence: AATAGCACAATAATCACATTCTTGGCTTCTACTCATCT

Modified sequence: GCTTCTACTCATCTAATAGCACAATAATCACATTCTTG

c) multiplexing sequences from different chromosomes belonging to the same species

Example:

Zea mays Cr. 4: AAGCTTCTACTCATCTCCCGGCAAACAGATAT...
Zea mays Cr. 7: GGAATAGCACAATAAGTGCGCAAAATCGAAG...
Zea mays Cr. 9: GATCACATTCTTGGATTTTTGGTGGAGACCAT...

MUX(Zea mays{Cr. 4, Cr. 7, Cr.9}) = AGGAGAGATCACTTATAC...

d) multiplexing sequences from different species

Example:

Homo Sapiens Cr. 5 → l_1 = 180,915,260 bp = 361,830,520 bits

Zea mays Cr. 8 → l_2 = 175,793,759 bp = 351,587,518 bits

Felis catus Cr. C1 → l_3 = 221,441,202 bp = 442,882,404 bits

→ $l_{mux} = 3*l_{min}(l_1, l_2, l_3)$ = 1,054,762,554 bits

4.1.5 Format of the Secret Key

In order to obtain a symmetric key for decryption, the receiver need to know what method was used at encryption to generate the key, how many DNA sequences were used, and the IDs of these sequences. All this information must be included in a secret key and transmitted confidentially. The format of the secret key is given in Fig. 4.5.

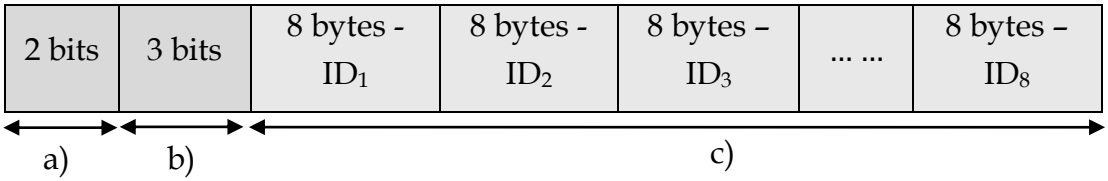


Figure 4.5 Format of the Secret Key a) 2 bits specifing the method a – 00, b – 01, c – 10, d – 11, b) 3 bits specifing the number of transmitted IDs, c) the actual secret key formed of 1 – 8 IDs, each of 8 bytes

Methods (a, b, c, and d) to generate the key were described above.

Example:

For a cleartext message corresponding to a video file of 329 MB (Table 4.6), a 1,384,120,320 bp DNA sequence is required. Such a sequence can be obtained by multiplexing (method d) 5 – 6 different chromosomal DNA sequences (Tables 4.2 –4.4). Consequently, the secret key will transmit the corresponding IDs. Fig. 4.6 is illustrating a possible combination of sequences in GenBank format and their IDs. Fig. 4.7 is illustrating the secret key format.

```
LOCUS       CM000663           249250621 bp    DNA     linear   CON 29-JUN-2009
DEFINITION  Homo sapiens chromosome 1, GRCh37 primary reference assembly.
ACCESSION   CM000663
VERSION     CM000663.1  GI:224384768
DBLINK      BioProject: PRJNA31257
KEYWORDS    .
SOURCE      Homo sapiens (human)
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
```

a) ID_1 – CM000663

```
LOCUS       CM000664           243199373 bp    DNA     linear   CON 29-JUN-2009
DEFINITION  Homo sapiens chromosome 2, GRCh37 primary reference assembly.
ACCESSION   CM000664
VERSION     CM000664.1  GI:224384767
DBLINK      BioProject: PRJNA31257
KEYWORDS    .
SOURCE      Homo sapiens (human)
  ORGANISM  Homo sapiens
```

b) ID_2 – CM000664

```
LOCUS       CM001378          239302903 bp    DNA     linear   CON 14-DEC-2011
DEFINITION  Felis catus breed Abyssinian chromosome A1, whole genome shotgun
            sequence.
ACCESSION   CM001378
VERSION     CM001378.1  GI:362110686
DBLINK      BioProject: PRJNA16726
KEYWORDS    WGS.
SOURCE      Felis catus (domestic cat)
  ORGANISM  Felis catus
```

c) ID_3 – CM001378

Figure 4.6 Examples of DNA Sequences in GenBank Format and their IDs

10	101	ID_1 CM000663	ID_2CM000664	ID_3CM001378	... ...

Figure 4.7 Example of Secret Key Format

4.1.6 Overview of the Method and its Advantages

DNA based random sequences can be obtained following the steps:

1. Determining the length of the desired sequence according to the length of the cleartext.
2. Selecting the desired method for sequence construction
3. Accessing suitable sequences from the database
4. Generation of the secret key
5. Generation of the final DNA sequence

The final DNA sequence is next transformed in binary in order to be used for encryption or decryption.

This method for DNA-RS generation has the following advantages:

- generation of binary random sequences of any length using DNA structures from public or private databases
- the number of distinct random sequences is practically unlimited, taken into consideration the versatility of the possible ways (a ÷ d)
- the key does not have to be transmitted entirely, only the secret key (Fig. 4.5) of 9 - 65 bytes length requiring confidentiality in transmission and storage

4.2 OTP Encryption Systems for Transmission and Storage

4.2.1 OTP Cryptosystem for Duplex Transmission Based on DNA-RS

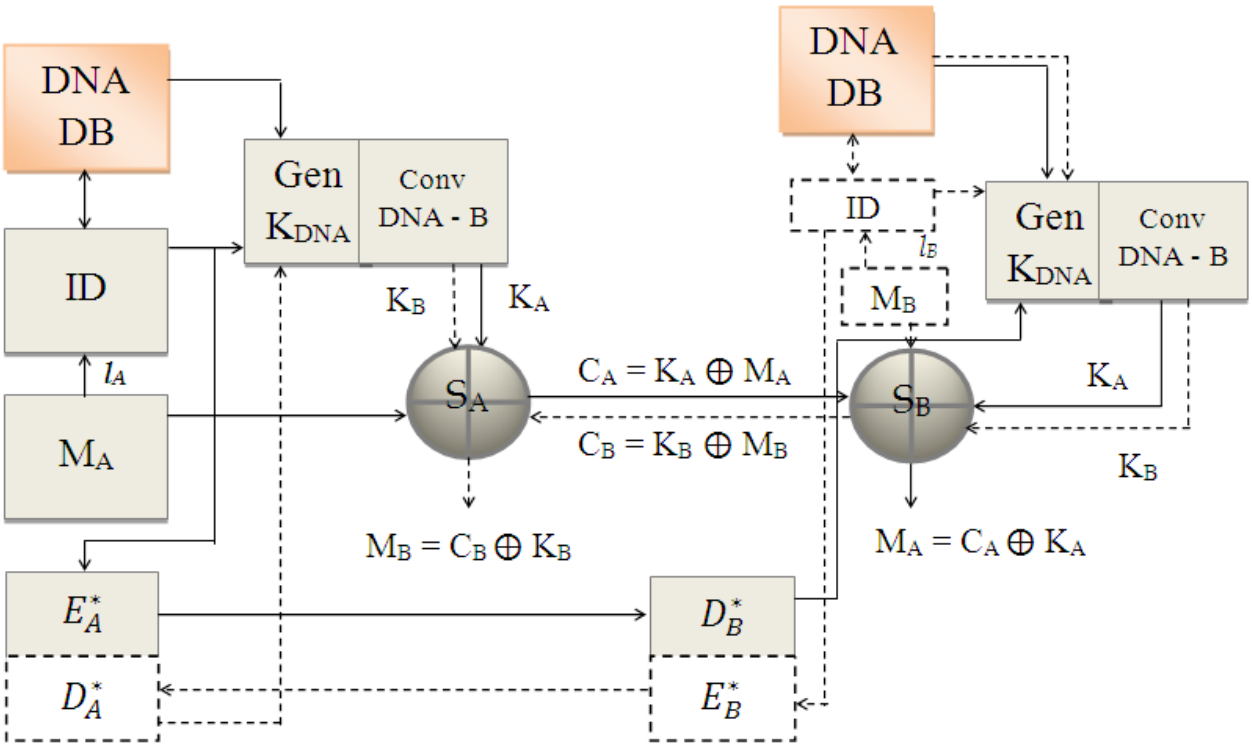


Figure 4.8 Scheme of OTP Cryptosystem for Duplex Transmission based on DNA-RS Key

Legend of Figure 4.8:

DNA DB – a database of DNA sequences (identical for both parts: A and B), which can be public or private

ID – block of input data containing: required length for the final DNA sequence according to the size of the input data; IDs of the sequences to be used at the key generation; method of key computation (information used to generate secret key)

M_A – cleartext message of user A

M_B – cleartext message of user B

Gen K_{DNA} – block of DNA sequence generation, used as the key (K_{DNA})

Conv DNA-B – transformation from DNA alphabet to binary (K_A, K_B)

$E^*_{A,B}$ – block of input data (ID) encryption, using a symmetric or public algorithm for users A and B

$D^*_{A,B}$ – block of input data (ID) decryption, using the algorithm chosen at $E^*_{A,B}$

S_A – modulo-2 adder used by A for OTP encryption and decryption

S_B – modulo-2 adder used by B for OTP encryption and decryption

Protocol of using duplex OTP Cryptosystem:

I. Transmission $A \rightarrow B$

a. *A* side encryption

(1) Providing the input data: length of the cleartext message, identification numbers (IDs) of chosen DNA sequences and key generation method.

(2) DNA key generation using input data and genetic database: K_{DNA} of user A.

(3) Generation of OTP key (K_A) using conversion DNA → binary

(4) Encryption of input data using a symmetric or public algorithm (block $\boldsymbol{E^*_A}$) and its transmission to part B

(5) OTP encryption of cleartext message M_A: $C_A = K_A \oplus M_A$ and transmission of the obtained cryptogram (C_A) to part B

b. *B* side decryption

(6) Decryption of input data (ID) realized in block $\boldsymbol{D^*_B}$

(7) Generation of K_A key using input data obtained at (6), DNA database identical with part A, and the same key generator (Gen K_{DNA} and Conv DNA-B)

(8) OTP decryption of C_A cryptogram using K_A key obtained at (7): $C_A \oplus K_A = M_A$

II. Transmission $B \rightarrow A$

a. *B* side encryption

(1) Providing the input data: length of the cleartext message, identification numbers (IDs) of chosen DNA sequences and key generation method.

(2) DNA key generation using input data and genetic database: KDNA of user B.

(3) Generation of OTP key (K_B) using conversion DNA → binary

(4) Encryption of input data using a symmetric or public algorithm (block $\boldsymbol{E_B^*}$) and its transmission to part A

(5) OTP encryption of cleartext message M_B: $C_B = K_B \oplus M_B$ and transmission of the obtained cryptogram (C_B) to part A

b. *A* side decryption

(6) Decryption of ID realized in block $\boldsymbol{D_A^*}$

(7) Generation of K_B key using input data obtained at (6), DNA database identical with part B, and the same key generator (Gen K_{DNA} and Conv DNA-B)

(8) OTP decryption of C_B cryptogram using K_B key obtained at (7): $C_B \oplus K_B = M_B$

The OTP cryptosystem for duplex transmission based on DNA-RS key has the following advantages:

- The OTP system (a secret key used only once and having the length at least equal to that of the cleartext message) was shown to be unbreakable [Sha49], [Ver26]
- The OTP key doesn't have to be transmitted entirely, the recipient can easily generate the key using confidential transmission ($\boldsymbol{E_A^*}$, $\boldsymbol{D_B^*}$) of ID and by using the same DNA database, which provides an easy key management, thus removing the main drawback of symmetric cryptography: difficult key management, especially when the number of users grows.
- Security of the cryptosystem is given by the security of the algorithm used at input data encryption (E^*, D^*) [BTT+13]

4.2.2 OTP Cryptosystem for Storage, based on DNA-RS key

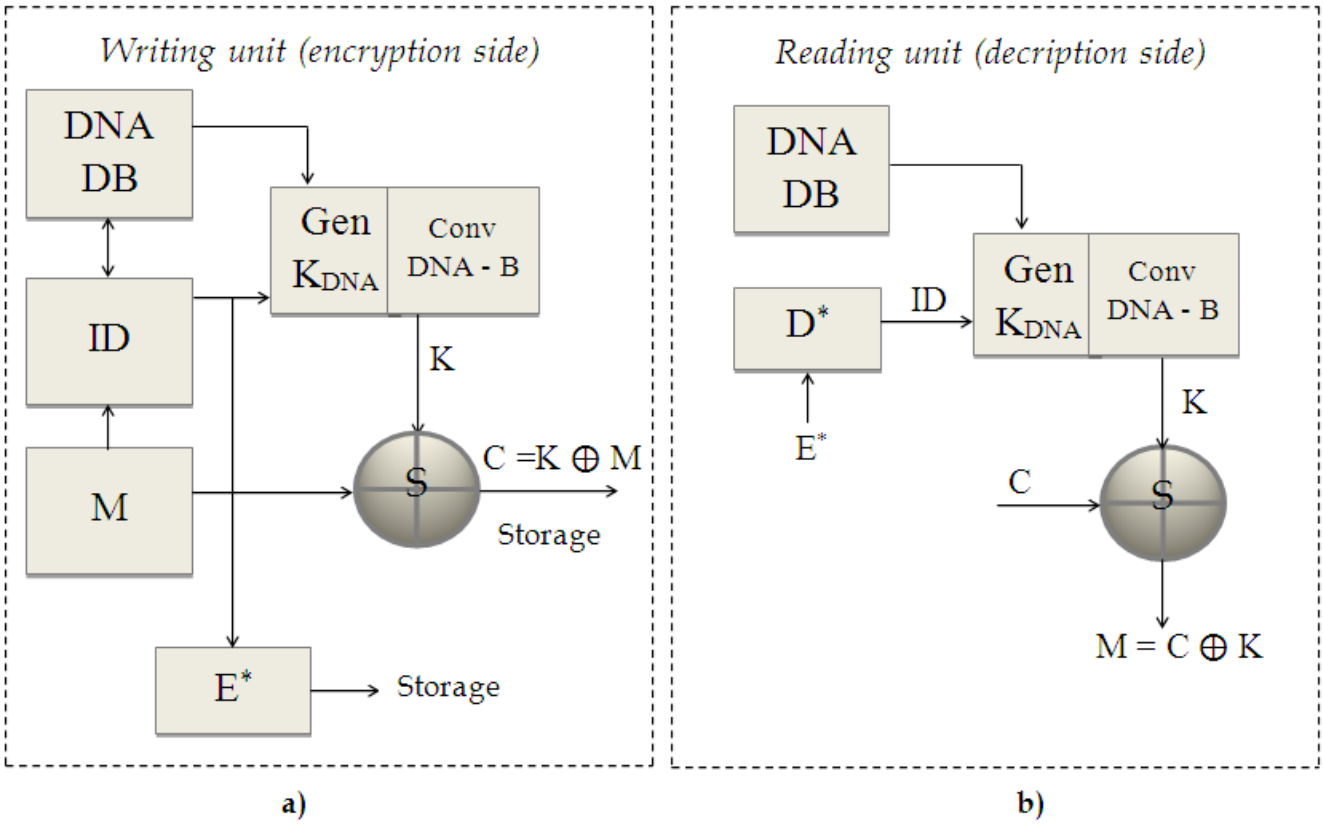


Figure 4.9 OTP Cryptosystem for Storage, based on DNA-RS Key a) Encryption Side of Writing Unit b) Decryption Side of Reading Unit

Obs.: block notations used in Fig. 4.9 are identical to the notations used in Fig. 4.8.

Protocol of writing – reading using OTP cryptosystem based on DNA-RS key:

a. Encryption (writing unit)

(1) Providing the input data: length of the cleartext message, identification numbers (IDs) of chosen DNA sequencesand key generation method.

(2) DNA key generation using input data and genetic database: K_{DNA}.

(3) Generation of OTP key (K) using conversion DNA → binary

(4) Encryption of input data using a symmetric or public algorithm (block E^*) and depositing on the storage media

(5) OTP encryption of data M: $C = K \oplus M$ and depositing of the cryptogram (C) on the storage media (CD, DVD, etc.)

b. Decryption (reading unit)

(6) Decryption of input data (ID) realized in block D^*

(7) Generation of key (K) using input data obtained at (6), DNA database identical with the one used at writing unit, and the same key generator (Gen K_{DNA} and Conv DNA-B)

(8) OTP decryption of cryptogram (C) using key (K) obtained at (7): $C \oplus K = M$

OTP cryptosystem for storage, based on DNA-RS key has the following advantage:

- Storage of encryption keys is easier than in the classical system because it consists only of input data (ID), so it has a much shorter length than the encryption key (K)[BTT+13]

4.3 Performance Evaluation

4.3.1 Computational Complexity of the Algorithm

The complexity analysis of an algorithm is important because it reveals its efficiency for the real time applications. In this work the computational time of the algorithm was analyzed through the experimental tests using software functions to measure the time. This new method was compared to the existing symmetric encryption algorithms: DES, AES, 3DES.

In order to obtain fair and stable results different software functions, that measure elapsed time, was tested and the most suitable was chosen for further testing. This part is described in subchapter 4.3.1.1. In subchapter 4.3.1.2 the experimental measurements are shown. Computational time was measured separately for all the parts of the encryption – decryption process and then the total encryption time was calculated. Decryption is composed of the same operations and thus is performed in the same time as encryption. At the end the results of the comparison between different symmetric cryptographic algorithms are presented.

4.3.1.1 Time Evaluation Functions

Implementations were performed in Visual Studio 2010, C# on the computer with the following properties: Genuine Intel(R) CPU Dual Core 1.79 GHz, 1 GB RAM. In VS10 there are different functions available to measure execution time. What follows is an overview of these functions as well as the one chosen for time measurements in this work.

1) *DataTime*

```
DateTime begin = DateTime.Now;
//Some code
DateTime end = DateTime.Now;
Double time = (stop - start).TotalMilliseconds;
```

It is a fast function with resolution of 10 milliseconds. The speed of time measurememnt functions is not important for this work because these functions are not used during the actual encryption. This function can be unreliable if the system-time changes. This happens in very rare situations, like through synchronization with a time server [Kri08]. In this work, measuring the same operation with the same parameters, using this functions gave high variations in the results.

2) *Stopwatch*

```
Stopwatch RunTime = Stopwatch.StartNew();
//Some code
RunTime.Stop();
long w = RunTime.ElapsedMilliseconds;
```

In this work, using the stopwatch gave unstable results. According to the [Kri08] this function can be unreliable on PC with multiple processors and also on on processors that do not have a constant clock speed.

3) *Process.TotalProcessorTime*

```
TimeSpan start = Process.GetCurrentProcess().TotalProcessorTime;
…..
TimeSpan stop = Process.GetCurrentProcess().TotalProcessorTime;
Double time = (stop - start).TotalMilliseconds;
```

This procedure measures how long the process placed between the start and stop kept the CPU busy. It offers more stable results because it doesn't measure how much time has passed, the time while the code is waiting (sleeping) is not counted and the timings are not distorted by other processes that consume CPU [Kri08].
The results of this work were the most stable using this procedure and therefore the measurements presented in the following subchapter (and in any other) were taken in this way. The resolution in time measurement is of 15.6 ms.

4.3.1.2 Experimental Measurements of the Computational Time

The encryption and decryption processes of this cryptographic method are composed of the same parts. The operations performed in one process are the following: reading the file(s) containing DNA sequence; performing one of the operations: multiplexing, concatenation, or shifting on the obtained sequence(s); transformation of the obtained sequence from DNA alphabet to binary; and finally bitwise XOR operation between the data bitstream and the obtained OTP binary sequence. Computational time was measured separately for each process using the third procedure described in subchapter 4.3.1.1 and then the total time was calculated. In order to observe the growing rate of the execution time, the measurements were taken considering different sizes of the data.

1) DNA sequence file reading

Plaintext Length (KB)	File Reading (ms)
43.9	4.5
105	5.85
199	7.2
475	16.6
768	31.25
1510	47.5
3110	106.1
6210	223.9
9660	307.3

Table 4.7 Time measurements of the DNA sequence reading from the file

The DNA sequence needed for encryption or decryption is half of the data (plaintext or ciphertext) bitstream size (one DNA letter represents 2 bits). Thus, the file reading stops when the sequence reaches half of the data size. The growing rate is linear (Fig. 4.10) and it can be described approximately by the function:

$$\boldsymbol{y = 0.035 * x} \tag{4.1}$$

where y is file reading time in ms and x is the size of the plaintext in KB (Fig. 4.10).

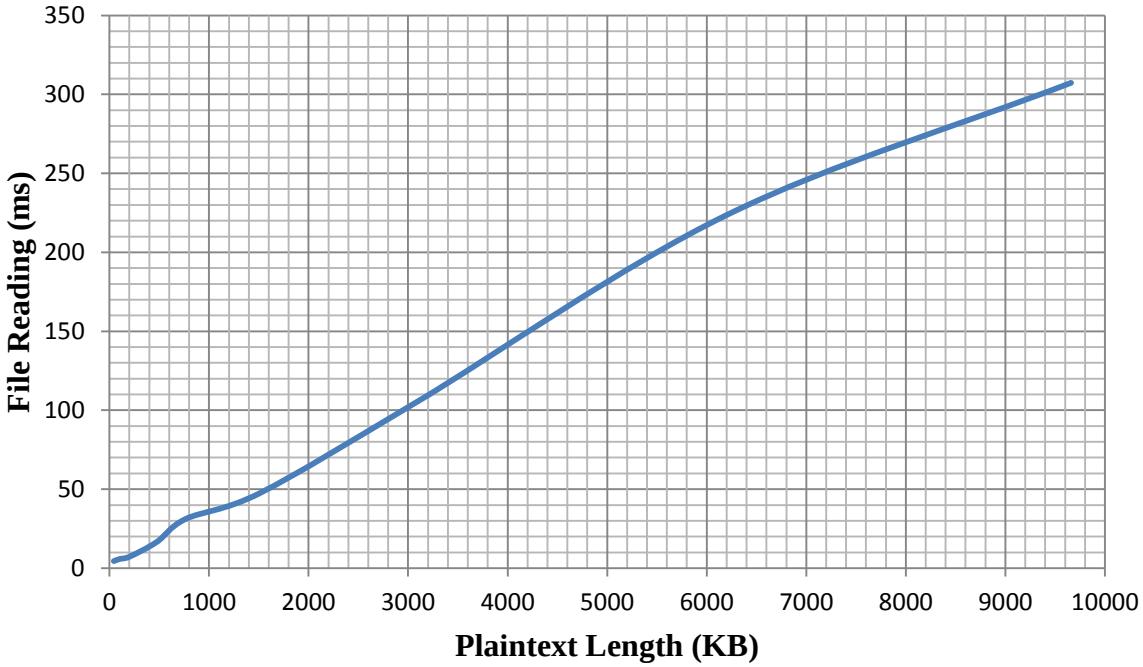


Figure 4.10 Growing Rate of the File Reading Runtime

2) *Concatenation of the DNA sequences*

Concatenation is the time it takes to append all the DNA sequences read during the previous step. Considering the sizes of the plaintext from Table 4.7, and even twice larger, the concatenation time is 0 ms. This is explained by the fact that DNA sequences are very large and their concatenation is required only for very large data to encrypt.

3) *Shifting of the DNA sequences*

Execution time for shifting procedure depends on the amount of shifted letters. The measurement results are presented in Table 4.8.

Shift Value (Number of characters*10^3)	Shifting (ms)
180	0
1000	5.85
2500	17.56
5000	31.2
8000	55.5
10000	72.7

Table 4.8 Time measurements of the DNA sequence shifting

The growing rate is linear (Fig. 4.11) and it can be described approximately by the function:

$$y = 0.0065 * 10^{-3} * x \quad (4.2)$$

where y is the time of shifting in ms and x is the number of shifted characters (Fig. 4.11).

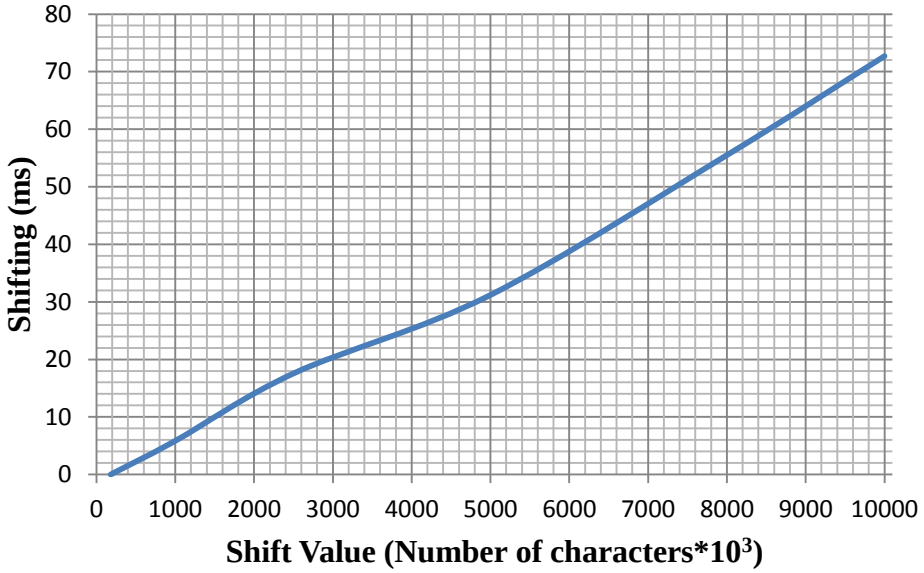


Figure 4.11 Growing Rate of Shifting Procedure Runtime

4) *Multiplexing of the DNA sequences*

Multiplexing of the DNA sequences is performed until the final sequence reaches half of the plaintext bitstream size. Thus, the time to perform this operation was computed according to the size of the plaintext data (Table 4.9).

Plaintext Length (KB)	Multiplexing (ms)
43.9	0
105	3.1
199	3.9
475	13
768	15.6
1510	39.2
3110	88.5
6210	140.6
9660	240.4

Table 4.9 Time measurements of the DNA sequences multiplexing

The growing rate of the multiplexing time is linear (Fig. 4.12) and it can be described approximately by the function:

$$\boldsymbol{y = 0.12 * x} \tag{4.3}$$

where y is the multiplexing time in ms and x is the size of the plaintext in KB (Fig. 4.12).

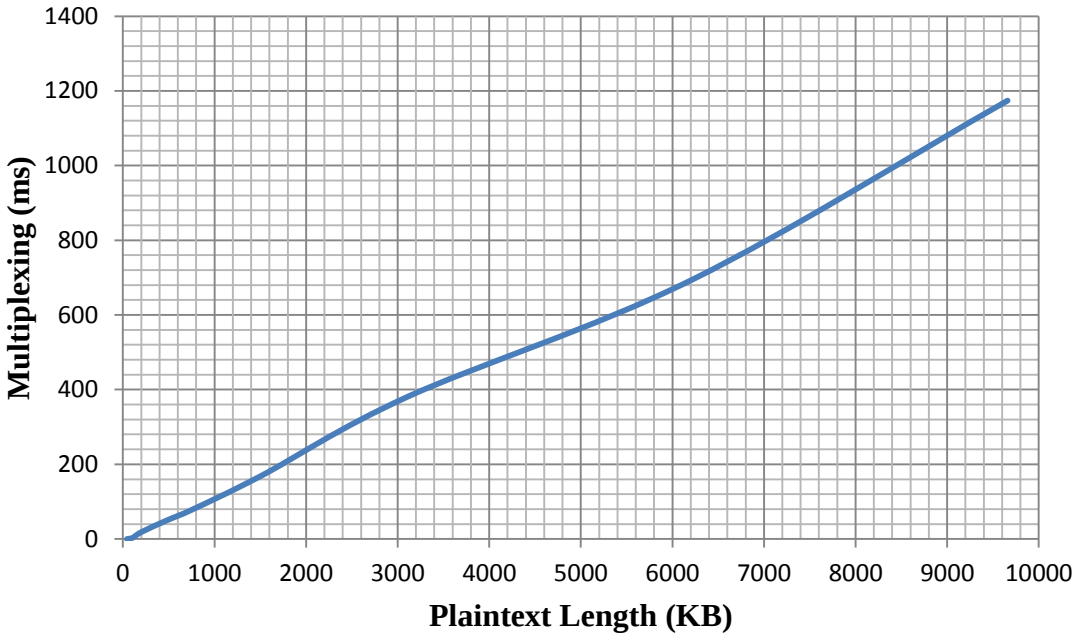


Figure 4.12 Growing Rate of Multiplexing Procedure Runtime

5) *Transformation from DNA alphabet to binary*
   After that one of the methods: multiplexing, concatenation, or shifting was performed on the DNA sequence(s) the obtained sequence was transformed in binary by switch – case mapping procedure: A – 00, C – 01, G – 10, and T – 11. The time spent for this operation is proportional to the length of the final DNA sequence (Table 4.10).

Plaintext Length (KB)	DNA to Binary Transformation (ms)
43.9	10
105	23.4
199	38.6
475	91
768	152
1510	320
3110	637
6210	1253
9660	1978

Table 4.10 Time measurements of the DNA to binary transformation

The growing rate is linear (Fig. 4.13) and it can be described by the function:

$$y = \frac{1}{5}x \qquad (4.4)$$

where y is the DNA to binary mapping time and x is the size of the plaintext (Fig. 4.13).

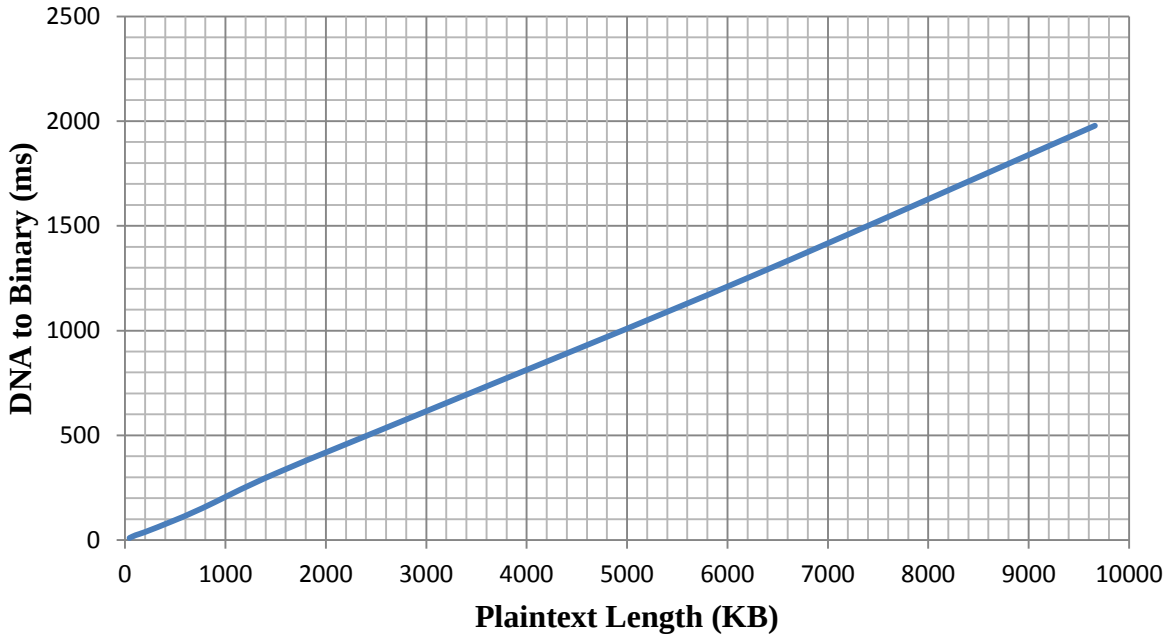


Figure 4.13 Growing Rate of the Time Spent to Transform from DNA Alphabet to Binary

6) *Bitwise XOR operation*

The final procedure in encryption – decryption processes is the binary XOR between the data bitstream and the key bitstream obtained from DNA sequence. The time taken to perform this operation is proportional to the number of bits in the plaintext (Table 4.11).

Plaintext Length (KB)	XOR (ms)
43.9	0
105	0
199	0
475	0
768	0
1510	5.6
3110	5.85
6210	20
9660	31.24

Table 4.11. Time measurements of the binary XOR operation

The growing rate is linear (Fig. 4.14) and it can be described approximately by the function:

$$\boldsymbol{y = 0.003 * x} \tag{4.5}$$

where y is the XOR operation time in ms and x is the size of the plaintext in KB (Fig. 4.14).

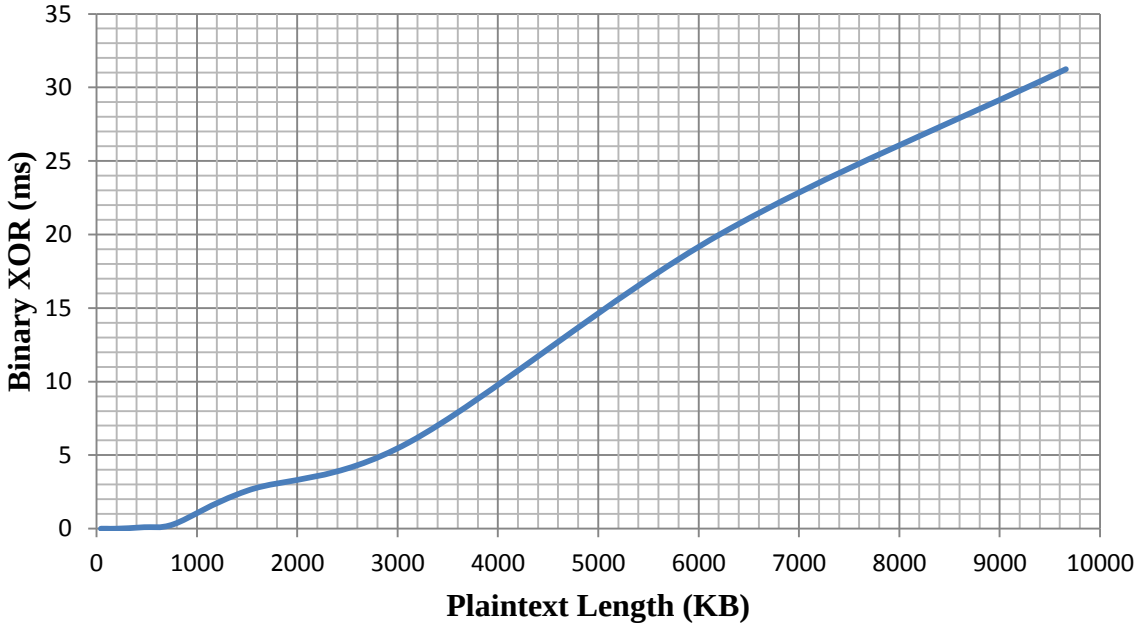


Figure 4.14 Growing Rate of the Time Spent to Perform Binary XOR

The new DNA XOR method was compared to other existing symmetric encryption algorithms. The overall encryption time for this method was obtained by summating the milliseconds of all the composing operations. The AES algorithm was tested with a 128 bit encryption key. The computational time comparison results are presented in Table 4.12.

Plaintext Length (KB)	DES (ms)	AES (ms)	3DES (ms)	DNA XOR (ms)
105	15.6	15.6	15.6	29.25
199	17.8	17.8	31.2	45.8
475	31.2	46.8	62.5	107.6
768	62.5	62.5	85.5	183
1510	105	140.6	171.8	373
3110	203	328	359	748.8
6210	390	617	718.7	1497
9660	609	953	1093	2316

Table 4.12 Time measurements of the symmetric encryption algorithms

In Fig. 4.15 a comparison between the growing rates of the symmetric encryption algorithms is presented. The fastest algorithm is DES and the DNA XOR is slower than the others. AES and 3DES appeared as improvements to DES, as the higher security was required. As can be seen in the Fig. 4.15 the higher security also comes with a higher computational complexity. The DNA XOR algorithm is offering an OTP key and thus the highest security. The rise in computational time is acceptable considering the ensured level of security.

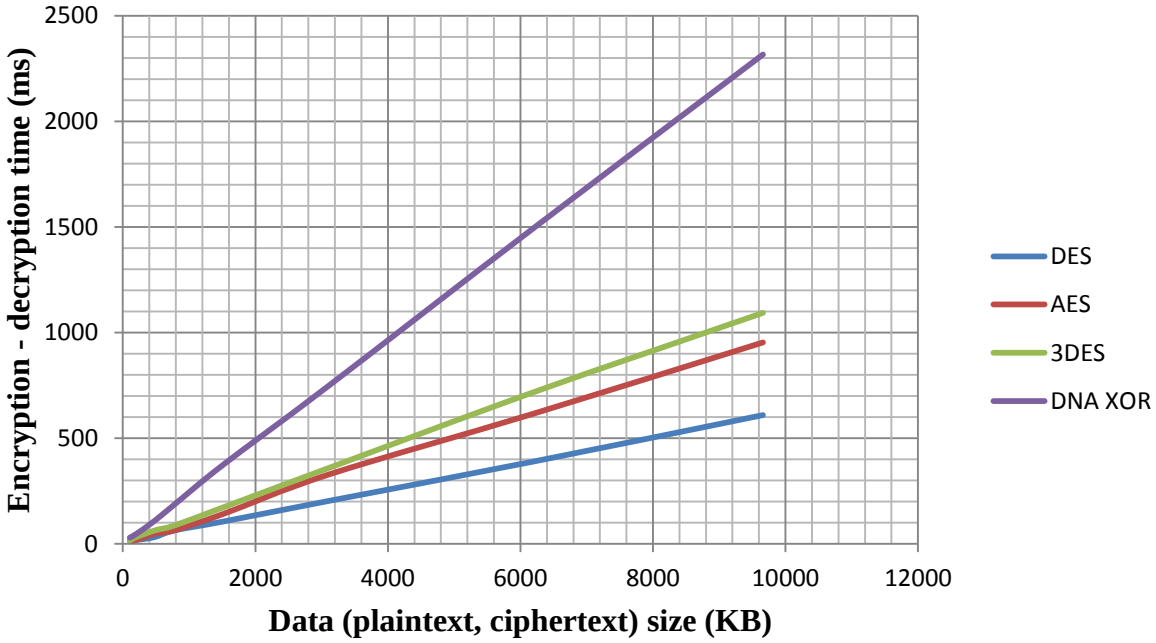


Figure 4.15 Comparison between Symmetric Ciphers Computational Time

4.3.2 Security Level of the Algorithm

The security of the algorithm was analyzed using different approaches, like: statistical measurements, cryptanalytic attacks, key space analysis and secure transmission of the ID numbers.

A. Statistical Measurements

Statistical measurements were used in this work in order to measure security strength of the ciphertext, to observe its randomness and dissimilation from the plaintext. The following statistical evaluations were made: *histogram*, *entropy* and *correlation coefficient* (CC). Histogram offers the possibility to visualize the distribution of the data. Entropy measures numerically the uncertainty and randomness of the signal. A large value for entropy means a less predictable signal. CC indicates the degree of correlation between neighboring values like pixels or letters; its value is intended to be small for the ciphertext.

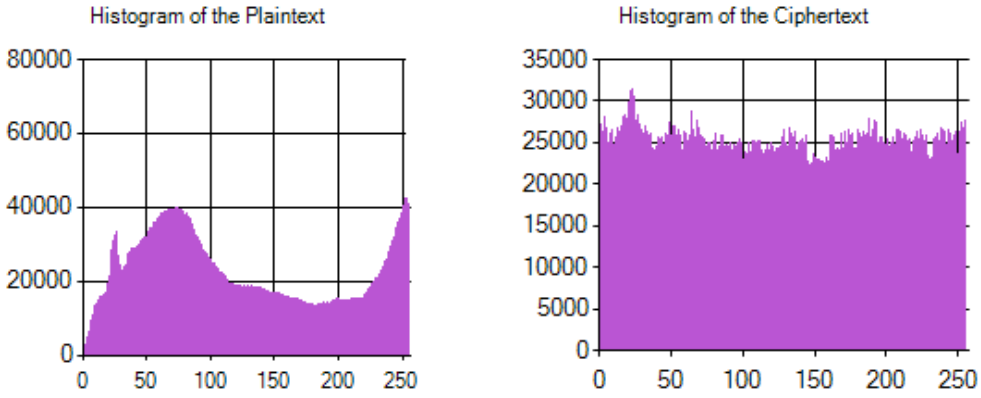


Figure4.16 Histograms of the Plaintext Bitmap Image and Corresponding Ciphertext

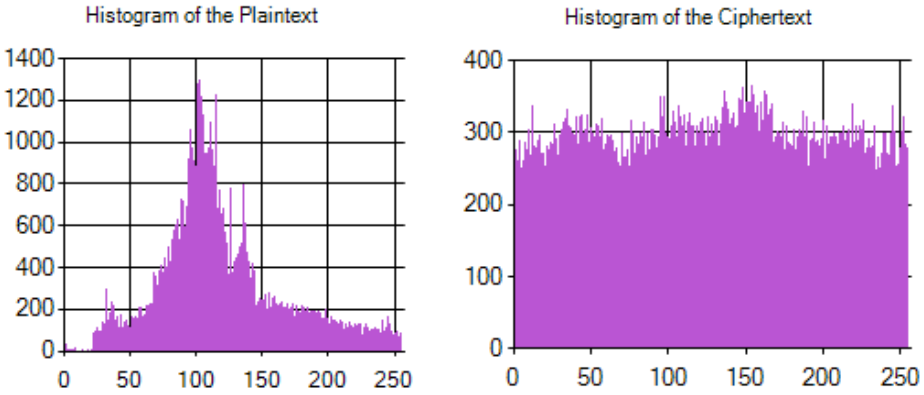


Figure 4.17 Histograms of the Plaintext Tiff Image and Corresponding Ciphertext

Data in the plaintext is highly correlated (Fig. 4.16) and appearance of its values has a wide range, like between 100 – 1200 in Fig. 4.17. Distribution of the ciphertext, however, is more uniform and the neighborhood values are not correlated (Fig. 4.16 –4.17).

Entropy of the	Entropy of the
7.24	7.91
7.42	7.86
7.16	7.84
7.82	7.92
7.47	7.93

Table 4.13 Plaintext and ciphertext entropy measurements

Correlation Coefficient of the	Correlation Coefficient of the
0,11	0.002
0.031	0.007
0.62	0.013
0.016	0.002
0.175	0.009

Table 4.14 Plaintext and ciphertext correlation coefficients

As expected the ciphertext entropy values are higher than the plaintext entropy (Table 4.13) and the correlation coefficients of the ciphertexts are smaller than in case of the plaintext (Table 4.14).

B. *Cryptanalytic Attacks*

Resistance to the classic cryptanalytic attacks is given by the OTP principle [Sha49], [Ver26]. Given a random key obtained from DNA sequence which is used only once, a ciphertext may correspond to any plaintext of the same length, thus all the results are equally likely.

Given the perfect secrecy [Sha49], OTP is resistant to all the attacks including brute-force attacks. Trying all keys will result in plaintexts, which are all equally likely to represent the original plaintext. In case of the known plaintext (part of the plaintext data being known) brute-force attacks is still unusable because the attacker will not be able to use this information to find out the rest of the key and as the key changes for each encryption the previous information is also useless.

All the practical problems related to the use of the OTP are solved with the genetic databases and the method presented in chapters 2 and 3 to use them.

C. Key Space Analysis

The key space of an encryption algorithm is desired to be as large as possible in order to resist to the brute-force attack. The encryption – decryption key is a sequence of bits and thus its space is $2^{length(Key)}$. Trying all the possible keys will give $2^{length(Key)}$ number of attempts to obtain a successful break. On the average the correct answer can be found in half of this number of tries.

Considering the fact that the provenience of the random binary sequence is from the genetic databases, trying all the sequences from the database is also studied here. One of the databases can be the NIH genetic sequence database named GenBank. It is a collection of all publicly available DNA sequences [wncbi]. It contains approximately 135 440 924 DNA sequence records. Trying each of these sequences in the key construction will be equivalent to a start number 2^{27}. Considering all the combinations that can be made when the sequences are concatenated, multiplexed, or shifted in order to obtain the final sequence, this number of tries tends to infinity.

D. Secure Transmission of the ID Number

Secure transmission of the sequence ID numbers and the method by which they were combined is important when a public database is used. If the database is public, access to the ID numbers and combination method means a direct access to the key-sequence. The length of the ID number in GenBank is 6 – 8 characters, which means up to 8 bytes. The secret key is of 9 - 65 bytes long composed of a header and IDs (subchapter 4.1.5). This secret information can be encrypted in block of 64 bits using a traditional symmetric algorithm, like DES, and then sent through an existing encryption channel, using a previously exchanged key. Accession number can be sent also using public-key cryptography. Anyone who has a copy of a public key can easily encrypt information that can be read only with private key. Communication involves only public keys, and no private key is ever transmitted or shared. This type of key distribution is used in PGP [PGP04], due to the facility of public key distribution.

4.4 Contributions

A new method for random sequences generation based on DNA structures and two OTP systems for transmission, respectively for storage, along with their protocols were presented [BTT+13]. The main advantage of the method for random sequence generation is the possibility to generate practically an infinite number of such sequences starting from DNA structures and as long desired. If these sequences are used in cryptographic applications as keys, they require confidential transmission of only 9 – 65 bytes of information and not of the entire key as normally in OTP, meaning that the key management becomes very easy.

The two OTP systems for transmission and storage have the advantage of the OTP: greatest security based on using a key once and the lengths of the key being at least equal with that of the cleartext message. The major drawback of OTP systems is the key management, but this is overcome by the advantage of the key generation previously described.

Performance evaluation was made for computational time and security level. The new method was shown to be fast, it takes 2 ms to encrypt or decrypt 9.66 MB of data. In comparison to other symmetric encryption algorithms it is slower, but the security is much higher. Its OTP principle makes it strong against existing cryptanalytic attacks. Statistical measurements showed that the distribution of the ciphertext is random and uniform.

Chapter 5

Design of the DNA Vernam Cipher at the Molecular Level

Contents in Brief

5.1 Principle of the Algorithm 102
5.2 Design and Implementation Details 105
5.3 Performance Evaluation 111
5.4 Contributions 111

DNA Vernam cipher represents a way to introduce cryptography at the molecular level. It shows different techniques to represent binary information in genetic structures and proposes hybridization as a central process for computations.

This encryption method is based on a truly random key used only once in logical XOR operation with the plaintext. Binary XOR was designed to be performed using biomolecular computation and artificially designed DNA motifs. A study of most suitable DNA structures was performed and the isomer of double crossover molecule (DX) named DAE molecule was chosen as a building block of mathematical XOR operation. These structures also named DNA tiles can perform self-assembling operations. Another important property of these structures is the natural hybridization process between them which gives a truly random symmetric key.

5.1 Principle of the Algorithm

The principle of generating random numbers from natural events (subchapter 2.4.2) is used in this cipher and the physical phenomenon is DNA hybridization. It is the basic procedure for the encryption-decryption process and of the key generation. One of the important steps was to find a DNA structure to represent an information unit. For this purpose specialized literature was consulted [Rot09], [CP09] and a suitable structure was chosen.

This encryption method combines the power of biomolecular computation and classical XOR OTP cipher. This principle was inspired by [GLR04] where DNA triple-helix molecules were used to perform XOR OTP cipher. The experimental feasibility of biomolecular computing was demonstrated in 1994 by Leonard Adleman [Adl94]. In his experiment single stranded DNA molecules were used and a series of procedures adapted from molecular biology. This experiment was a starting point for carrying out computations at the molecular level followed by the series of works in DNA computing field.

Binary data can be encoded using a single DNA molecule for each bit. The structure of the molecule will be described in detail in the next section. The difference between "0" and "1" is given by the group of nucleotides at the molecule terminations, like "ACTG" and "TGCA"(Fig. 5.1). Bit-wise XOR operation is performed using the DNA hybridization process and by assembling the plaintext binary message, the key and the resulted encrypted message in strings. The string with the result is the only one that remains at the end; other strings (plaintext and OTP) are removed [TB09].

The first step of the algorithm is encoding the information we want to encrypt in a string of DNA molecules. For this purpose two stacks with DNA molecules can be used, designed to represent the binary "1" and "0" (Fig. 5.1).

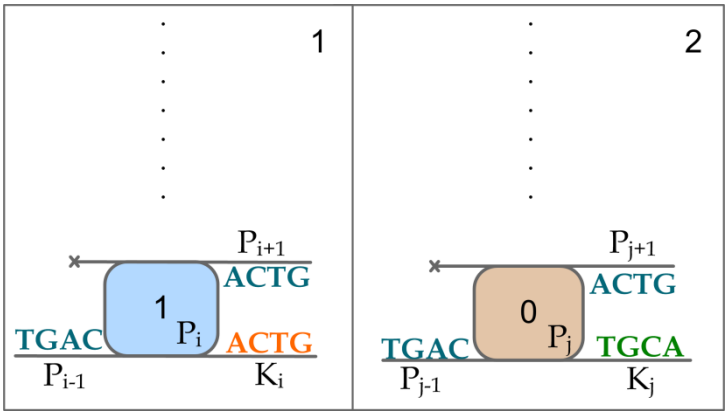


Figure 5.1 Amounts of DNA Tiles for Plaintext Bitstream Assembling

These stacks can be operated by a microcontroller which acknowledges the beginning of encoding, makes a decision concerning the next bit, and verifies when the encoding stops. Sequences "TGAC" and "ACTG" are complementary, they are designed to bind to each other and form a bitstream of DNA structures (Fig. 5.2).

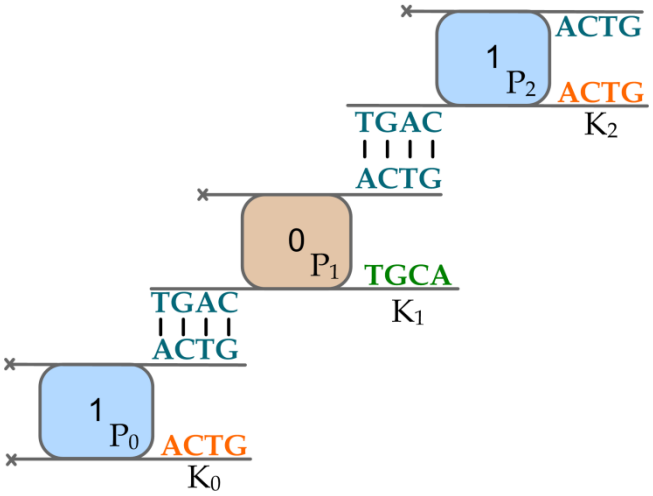


Figure 5.2 Plaintext Bitstream of DNA Molecules obtained by Hybridization

The next step of the algorithm is an XOR operation between the DNA molecules of the encoded information and the key. The key is also selected from a great set of labeled DNA molecules encoded as binary "1" and "0" (Fig. 5.3).

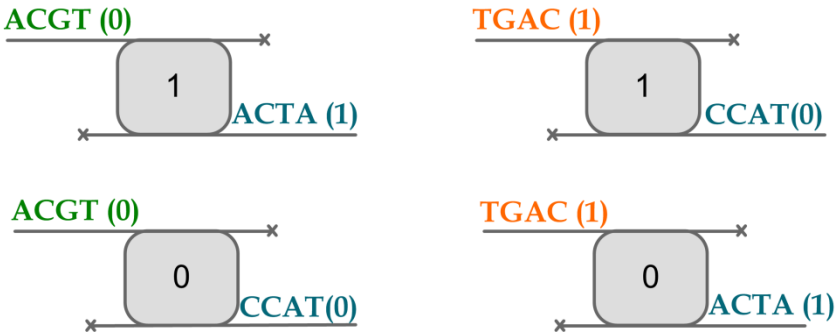


Figure 5.3 DNA Structures designed for Key Bitstream

There are 4 types of DNA structures for the key. Two of them bind to the plaintext bit "0" and the other two bind to the plaintext bit "1". A part of these molecules will randomly bind through the process of hybridization to the molecules representing the plaintext bits

(Fig. 5.4). Those molecules from the key set which hybridize with the plaintext information become the encryption key.

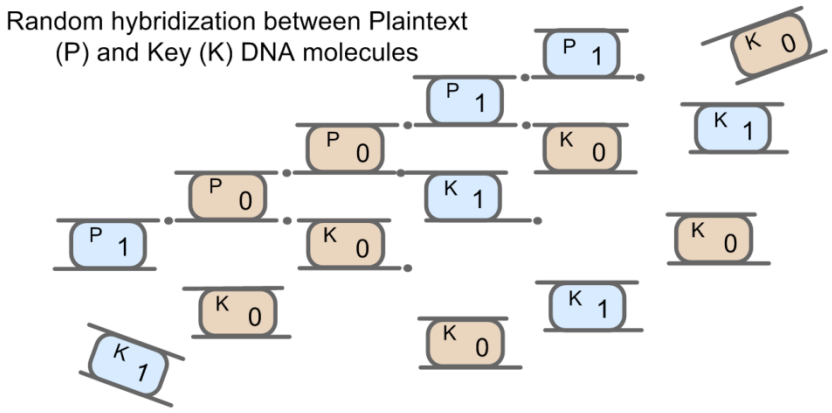


Figure 5.4 XOR Operation between the Plaintext and the Key obtained by Hybridization

The randomness of the key is ensured by the process of hybridization. There is an equal probability for a binary "1" or "0" of the key tile binding to the same bit of the plaintext (Fig. 5.5).

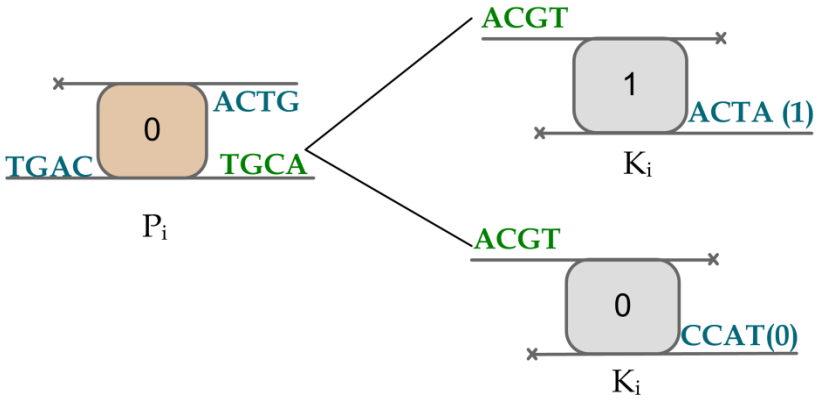


Figure 5.5 Hybridization between Plaintext and Key DNA Molecules

The ciphertext is composed of the DNA molecules which bind to the key molecules at terminations "ACTA" and "CCAT" as the result of XOR operation (Fig. 5.6). The ciphertext DNA molecules can be separated from the rest of the DNA material by using restriction enzymes. Next, ciphertext chain of molecules is sent to the destination with the string of labels for the key.

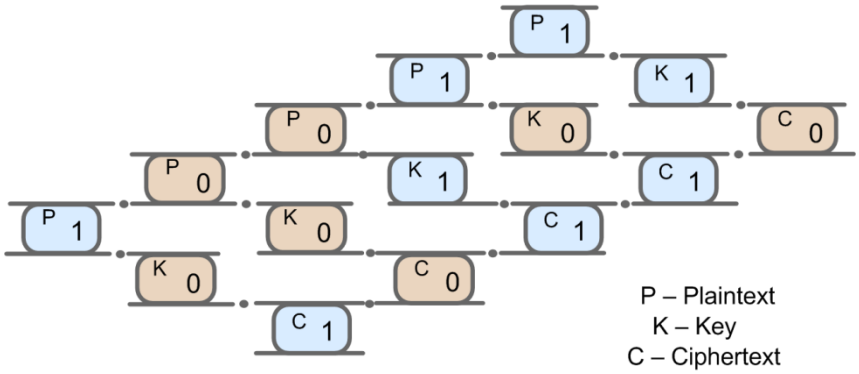


Figure 5.6 Ciphertext composed of the DNA Molecules

Decryption is made using the same key and operations. The receiver has an identical pad and will use the received string of labels in order to select from this pad the encryption key. Performing XOR operation between the ciphertext and the key will result in the original plaintext.

5.2 Design and Implementation Details

5.2.1 Design of DNA Molecule Structure

Design of controllable DNA structures was first introduced by Nadrian Seeman in 1980s [See81]. He invented DNA nanotechnology - the science and technology of building devices using DNA molecules. In nature DNA plays the role of genetic information carrier, but in this branch of technology it is used just as a structural material. Attraction of complementary DNA strands is used for building different nanoscale structures. DNA structure for this encryption method was chosen according to the aspects of different shapes described below.

Holliday junction is one of the simplest forms of DNA building blocks. It is a junction between 4 complementary to each other DNA strands (Fig. 5.7). It is not recommended to be used in the nanoengineering because of its instability induced by strong electrostatic repulsion [CP09], [FS93].

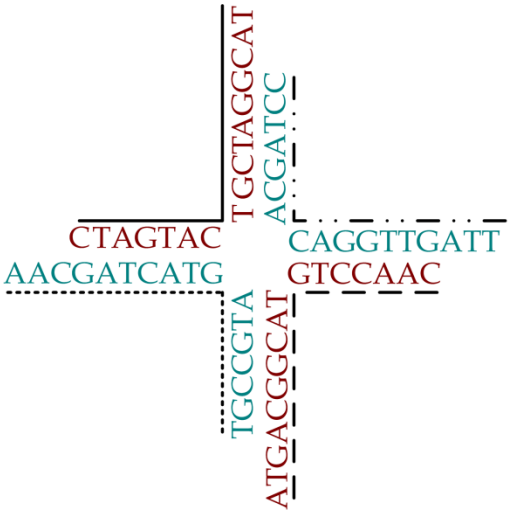


Figure 5.7 Holliday Junction

Double crossover molecule (DX) is a more stable structure which consists of two DNA helixes connected by two Holliday junctions. There are five different structures of DX molecules [FS93] (Fig. 5.8). Three of them have parallel helical domains: DPE, DPOW, DPON and the other two antiparallel helical domains: DAE, DAO.

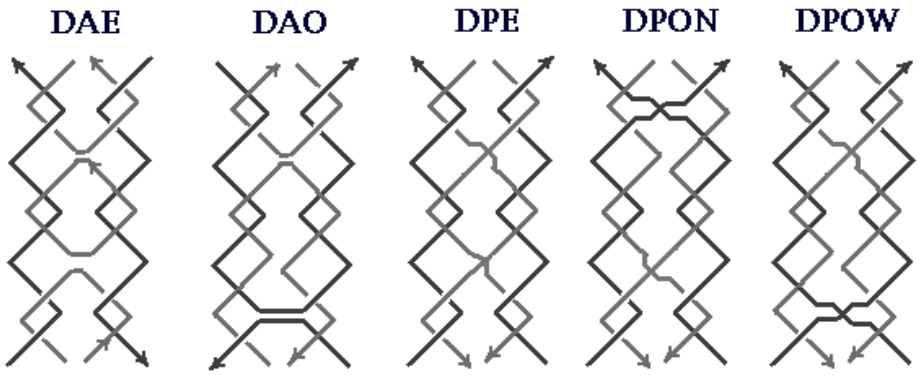


Figure 5.8 DNA Double Crossover Molecules

DX molecules with antiparallel domains are more stable than those with parallel domains [SWY+98]. Therefore the DAE and DAO structures present more interest. The difference between them is that DAE molecule has an even number of double helical half-turns between crossovers and DAO molecule has odd number of half-turns. For this encryption method a DAE structure was selected because it is stable and allows the use of one long DNA strand and several short strands in its construction.

DNA Origami was developed by Paul Rothemund [Rot06]. This method consists of using one long single DNA strand, in order to create a basic structure, and many short DNA sequences that come as complementary parts to the basic structure forming in this way double stranded DNA in desired shape (Fig. 5.9). The same principle was used in [SQJ04].

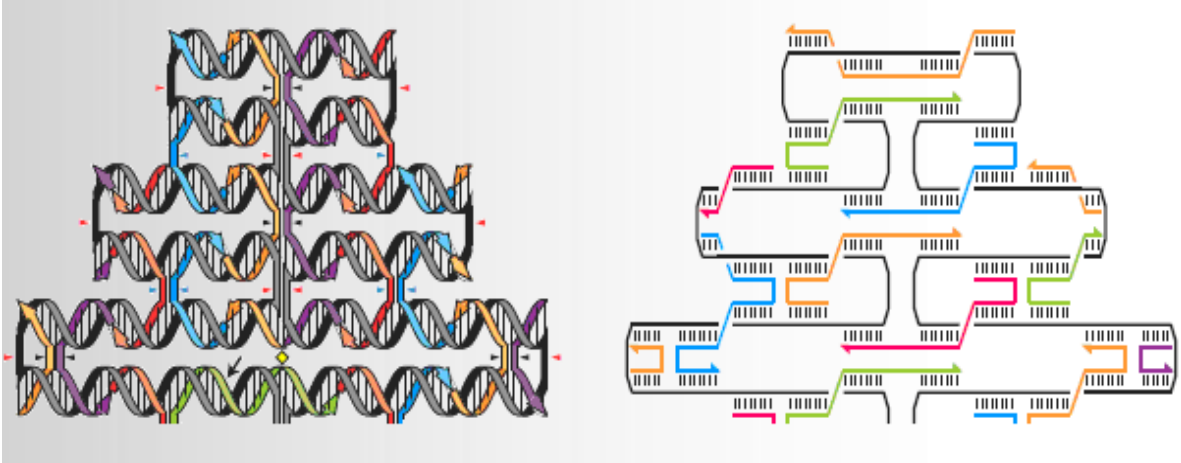


Figure 5.9 DNA Origami

Crossovers that appear between DNA helixes are incorporated for stability of the structure. In [Rot06] crossovers appear at every 1.5 helical turns which is equivalent to 16 bases space between crossovers.

An important aspect is finding a long single-stranded DNA. Certain synthesizers allow the synthesis of long oligonucleotides up to 250 bases [wgen]. Longer single-stranded DNA can be founded in viruses like M13mp18 used in [Rot06].

Considering all the aspects presented above, for this cipher a stable DAE structure was selected with one long DNA strand and several short strands (Fig. 5.10). Long DNA strands can be synthesized as oligonucleotides of 210 bases in length and short strands as oligonucleotides of 42, 46, 50, or 54 bases in length. The length of short oligonucleotides depends on the number of single-stranded (sticky) terminations of the structure. These terminations are used for binding other structures with complementary ends. Crossovers were selected at every helical period (2 turns). One helical turn has $10.67 \approx 10.5$ bases [Lev98] which mean that distance between crossovers will be 21 base pairs (bp).

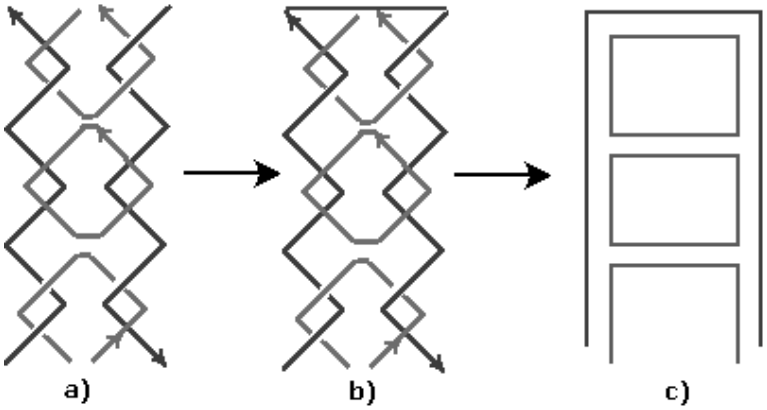


Figure 5.10 a) DAE Molecule b) modified DAE Molecule c) Schematic View

5.2.2 DNA Molecule Structure Implementation

Desired shape for DNA molecule was established in Fig. 5.10. It was chosen to be used as an information unit in Vernam cipher. Here the implementation of the selected structure is presented.

From the initial long DNA strand of 210 bases offsets were calculated for the complementary strands in order to create the desired shape (Fig. 5.11). Arrows from the figure represents opposite polarities between strands (5′ to 3′ directions). The longest DNA strand (210 bases) was sequenced in intervals of 21 bases which is equivalent to a DNA period assuming that per turn there are 10.5 base pairs (bp). In this case a DNA period is the distance between crossovers.

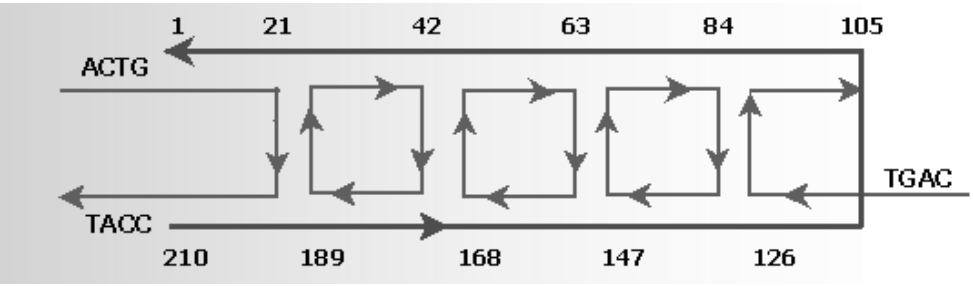


Figure 5.11 Schematic DNA Structure for Binary “1”

Steps of the implementation for this structure are the following:

1) Generation of random sequence from DNA alphabet (A, C, T, G) of length 210 bases.

2) Generation of complementary sequences to the first one. For example for the first middle circular sequence (Fig. 5.11) complementary sequences were generated in intervals: 22-42 and 169-189.

3) At the edges complementary sequences can have different lengths depending on the number of "sticky end" of the structure. For such sequences concatenation is performed of the terminations like: "ACTG", "TACC", or "TGAC" (Fig. 5.11).

In ordered to encrypt the message "secret" which is composed of 42 bits we need 42 structures described above. For each structure the necessary sequences were calculated. Terminations of marginal short sequences are different in case of binary "1" or "0" and if it is a start bit. For the first bit "1" of the message the following sequences were calculated:

1. TAGTCAGTGTATCCATGGTATCTGTACGTTTGGTATCAGAAGCCCCCGAAACTAGGTTCCGCTTGCGACCGTTT
ATGTTTCAGAACAGGGCTGTTCGCGTATGATACGTCGTAAACCTAGATAGTCAACCATGTCTAGATTTGTCACGT
CCTGTATGACCGTATTCCGGGTCCTGTCGGTAGGATTCTAGCCTACCCTTAAGCTTGTCG
2. ACTGATCAGTCACATAGGTACCATATCGGATGGGAATTCGAACAGCTACC
3. GACATGCAAACCATAGTCTTCCCCAGGACAGCCATCCTAAGA
4. GGGGCTTTGATCCAAGGCGAGCAGGACATACTGGCATAAGG
5. ACGCTGGCAAATCTACAAAGTTTGGTACAGATCATAAACAGT
6. TGACTGCAGCATTTGGATCTATCAGCTTGTCCCGACAAGCGCACTA

For the next bit also "1" the last sequence is different. Sequence 6 starts with additional "TGAC" in order to bind to the first bit with termination "ACTG". The first bit does not have termination "TGAC" in order to restrict binding in one single direction. In case of binary "0" in sequence 2 instead of "TACC", "GTCA" appear. These terminations make a difference between binary "1" and "0".

From these computations files were obtained with sequences for all the DNA structures involved in the encryption process. These files can be used for synthesizing of structures in solution, in a laboratory. The next subchapter describes a laboratory utility that can be used for implementation of the cipher.

5.2.3 Laboratory Utility

The experiment of data encryption using biomolecular computation can be realized using the atomic force microscope (AFM) also named scanning force microscope (SFM). This tool is used for imaging, measuring and manipulating nanoscale matters (Fig. 5.12).

AFM provides highly accurate images of DNA at nanometer scale resolution. DNA samples are placed on a flat, smooth substrate, immobilized and then scanned by AFM. The scanning process results in the topography of the surface.

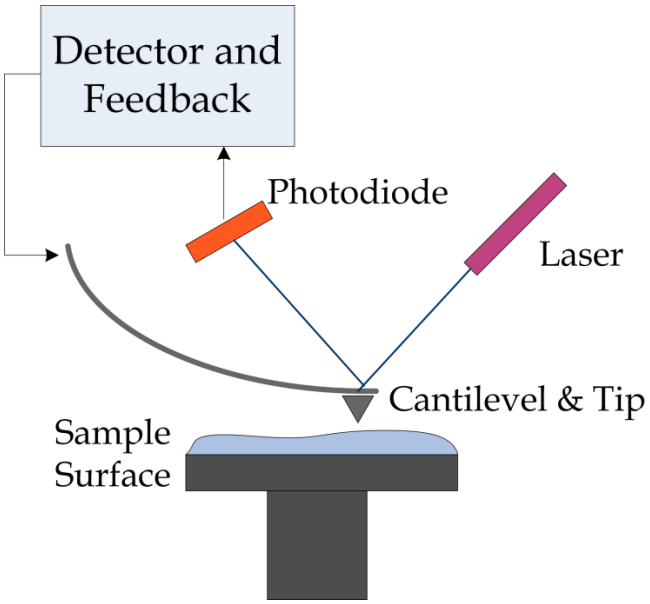


Figure 5.12 Atomic Force Microscope

AFM have been further developed into a operative tool. Therefore manipulation of distinct atoms and molecules becomes possible. Isolation of biological material was performed by AFM tip [LLA+04]. Single DNA molecule can be cut, pushed and folded in desired shape using this microscope [JHH+05]. Therefore AFM is an appropriate technique for this method.

The AFM structure consists of a cantilever and a thin tip at the end. The tip is brought close to the surface with the analyzed samples. As the surface is not perfectly smooth the tip cannot be at the same height all the time during scanning process. For this reason a feedback mechanism is used in AFM. This mechanism adjusts the distance between the tip and the analyzed surface. According to the elastic Hooke's law, the force that appears between tip and surface creates a deflection of the cantilever. The deflection is measured by a laser spot reflected from the top of the cantilever. The scanning process results in a topography of the surface sample [TBP11].

5.3 Performance Evaluation

In the absence of a standard for the implementation of the encryption algorithms in biomolecular environment, the process is slow and involves complex resources. Hybridization time depends on many factors like the amount of sequences, their length and techniques used for the implementation. It can vary from several minutes to a few hours. At this moment the execution time is not the strong point of the algorithm, but with the standardization of the whole process it might not be an issue in the future.

Important resource for the implementation of this algorithm is a specialized laboratory were the biomolecular data is processed and stored. AFM is a useful tool which facilitates imaging, measuring and manipulating nanoscale matters.

The strength of this algorithm is the level of security that it offers. The first aspect is the biomolecular environment which is more difficult to access. Due to the dimension of the biomolecular data, it can be easily hidden using steganographic techniques. Also an attacker would need an expensive laboratory in order to read the message. Another strong point of this algorithm is the key which is as long as the plaintext and is a truly random number generated through a natural process of hybridization. Therefore the algorithm satisfies the conditions of an OTP which offers a strong security. The self-assembly is a strong aspect of DNA nanotechnology which brings the parallel computation and automation of the implementation process [TBP+11].

5.4 Contributions

The symmetric encryption method based on DNA was presented in this chapter. It was designed to be implemented and used at molecular level. The DNA Vernam cipher is based on biomolecular computation, OTP principle, and special designed structures of DNA molecules.

OTP ensures a secure encryption, but at the same time requires a truly random number for the key which can be achieved by physical phenomena like DNA hybridization, in this case. In order to use this encryption method 2 parts must be provided: a software program and a laboratory experiment.

The main contributions for DNA Vernam cipher are: design and development of DNA molecule structure; a software program in Matlab that gives files with DNA sequences

necessary for creation of the DNA structures; and description of the useful tool (AFM) for the laboratory experiment.

Contributions to this part of cryptography at the molecular level are described and presented in the following papers: [TB09], [BT10], [TBP+11], and [TBP11]. [TB09] gives introduction notions about DNA structures and their origin from Wang tiles. Next, a thorough description of the DNA Vernam Cipher is presented. Each step of the algorithm is described in detail.

In paper [BT10] principles of biomolecular computation are shown. It describes DNA structure and techniques for DNA analysis. An overview of several algorithms for DNA steganography and cryptography is presented. The steganography technique is based on the PCR procedure for hiding and recovery of the messages and DNA hybridization between strands of DNA for encoding in binary.

Papers [TBP+11] and [TBP11] presents the research on the existing DNA structures used in nanotechnology. This is followed by the design of a new DNA structure for the Vernam cipher. After that a description of how DNA sequences for this structure can be generated is given. Finally laboratory utilities are presented with which the last phase of the encryption should be performed.

Chapter 6

Conclusions

This work is a scientific research on the DNA cryptography domain. It is an emerging and promising direction in cryptography. There are a few potential directions for using DNA in cryptography. One of them is biomolecular computing. The first experiment to solve a mathematical problem through the DNA molecules interactions was [Adl94]. It was followed by an experiment of a microprocessor based on these computations [RWB+96]. A variety of DNA shapes were designed for this purpose [See81], [FS93], [Rot06]. In this work biomolecular shapes and computations for the Vernam cipher were explored. A great interest for the biomolecular computations is the self-assembling structures and parallel computations. Structures of DNA molecules are self-assembling due to the process of hybridization that appear naturally between the complementary nucleotide bases. Parallel computations appear when hybridization of DNA structures happens at the same time in a solution where they are placed.

Another usage of DNA is its storage capabilities. There is an alphabet of 4 letters and any type of information can be rewritten in DNA using this alphabet. A mapping scheme between the DNA 4-letter alphabet and binary 2-letter alphabet was established as the most practical conversion table in this work and it was used extensively in the proposed ciphers. Data stored in DNA can easily be hidden due to its microscopic size. A sequence of DNA with encoded message can be placed in a great amount of meaningless sequences and then retrieved using laboratory specific techniques [TRB99].

Another usage of DNA for cryptography was discovered during the scientific research work of this thesis [BTT+13]. One of the most significant aspects in cryptography is generation of random numbers for the key and the secret communication between parties. Mathematical functions can generate pseudo-random numbers starting from a seed. Generation of pure random numbers is a difficult task for computers; some natural events are used as sources for that. This seed must be securely communicated for the reconstruction of the secret key. The longer a non-repeating random sequence of bits used for the secret key is, the longer the seed from which this sequence was generated. Studies

on DNA sequences showed that they hardly can be compressed. This means that they can be interpreted as random binary sequences using the transformation between these two alphabets. Availability of DNA sequences is given by the genetic databases with public access. To generate a secret key either the original sequence can be used, or some simple operations can be performed for the alteration of the DNA sequence, such as shifting, multiplexing, etc. Transmission of the secret key is ensured by a short identification number that each sequence has in a genetic database.

6.1 Overview of the Contributions

A study of existing techniques in DNA cryptography was performed and based on the ideas presented in literature new usages of DNA was established in this work. What follows is an overview of the thesis contributions along with the related publications of the work.

In chapter 3 the DNA Indexing symmetric cipher is presented. It was designed to use genetic databases in order to retrieve DNA sequences and use them for the substitution operations and as a secret key. The principle of the algorithm is described in [TBH+10]. A few diagrams were drawn in UML Violet editor in order to have a schematic view and functional framework of the software application for the algorithm. A windows form application was developed using VS10 C# in which it is possible to choose a text or image file for encryption – decryption. The results are visualized in a graphical interface.

Computational time was estimated through a theoretical analysis and practical time measurements on the basic operations of the cipher. Based on the obtained measurements a graphical view of the computational time growing rate was established for each operation. Security of the algorithm was estimated through statistical measurements. Experimental results of the histogram, correlation coefficient, and entropy were obtained for the plaintext and corresponding ciphertext. A theoretical analysis of the cryptanalytic attacks and key space was performed. A comparison to another algorithm of a similar principle was done using complexity theory. Results of this performance analysis were published in paper [TB13].

It was observed that the bitrate increases twice after the encryption with the DNA Indexing cipher. Different ways of how compression can be combined with encryption was analyzed. Certain improvements of direct encryption was proposed and presented in

[TAB13]. Next these changes were extended to a scheme where compression was combined with the DNA Indexing algorithm in the same process. New joint compression – encryption method was implemented in VS10 C# as a windows form application. A mathematical model of the bit allocation was provided and integrated in the practical application. Experimental results for the rate distortion function were obtained and then different versions of the algorithm were tested and compared for their compression efficiency.

In chapter 4 a method for obtaining random binary sequences from genetic and OTP cryptosystems for transmission and storage is presented [BTT+13]. Distribution and compression capacity of the DNA sequence were discussed in order to determine its randomness. Conversion table between the DNA and binary alphabet was established. A detailed description with visual examples of how a certain DNA sequence can be accessed was presented next. A study of the DNA sequences length was performed due to its importance for the cryptographic purposes.

Schematic views and protocols were given for the OTP cryptosystems based on the bitwise XOR operation for the encryption – decryption and generation of random binary sequences from DNA databases. Generation of random sequences starting from DNA sequences and XOR operation for the ciphering method were implemented using VS10 C#. After selection of text or image file for encryption, DNA sequences and method for the secret key generation, the results are visualized. Experimental measurements were taken for the computational time of each process of the algorithm and then an overall computational time was estimated. Obtained results were used to compare novel cipher to the well-known existing algorithms, like DES, AES, etc. Statistical measurements were taken to estimate security level of the algorithm.

Chapter 5 describes the DNA Vernam cipher that was designed to be implemented at molecular level. Principles of the algorithm and introduction notions about DNA structures are described in paper [TB09]. Biomolecular computation, techniques to operate on DNA sequences, and an overview of the algorithm is given in [BT10]. A research on the existing DNA structures used in biomolecular computations was performed and proposal of a new structure for the bitwise XOR operation of the Vernam cipher was given. This part of the work was described in papers [TBP+11] and [TBP11].

6.2 Future Works

One of the future works is to perform a laboratory experiment for the method described in chapter 5. Special equipment is required for the experiment of the biomolecular computation. During the measurements of the computational time for the procedures used in the proposed DNA ciphers it was observed that the switch case operation for the conversion between the DNA and binary alphabets can be performed faster. The switch case can be substituted by a faster procedure: access of a vector element. A vector can be filled with zeros except for the positions of the DNA letters ASCII codes. At these four places the values will be of the corresponding binary conversion representation. Analysis of the security level for the novel scheme of the joint compression – encryption can be performed using statistical measurements or other techniques. Other future works would be practical experiments of the cryptanalytic attacks that can be performed on the proposed ciphers. Based on the promising results from the first experimental tests on the joint compression-encryption scheme, a dedicated study may be performed to test its performances.

Scientific Activity

List of Publications

[1] Monica E. Borda, **Olga Tornea**, Tatiana Hodorogea, "Secret writing by DNA hybridization", *Acta Technica Napocensis*, Electronics and Communications, Vol. 50, No. 2, pp. 21 – 24, 2009.

[2] **O. Tornea**, M. Borda, "DNA Cryptographic Algorithms", *International Conference on Advancements of Medicine and Health Care Trough Technology Proceedings*, Vol. 26, pp. 223-226, Cluj-Napoca, 2009.

[3] **Olga Tornea**, Monica Borda, Tatiana Hodorogea, Mircea-Florin Vaida, "Encryption System with Indexing DNA Chromosomes Cryptographic Algorithm", *IASTED International Conference on Biomedical Engineering (BioMed 2010)*, Innsbruck, Austria, paper 680-099, pp. 12-15, ISBN: 978-0-88986-825-0, 975-0-88986-826-7, Acta Press, 15-18 February 2010.

[4] M. Borda, **O. Tornea**, "DNA Secret Writing Techniques", Proceedings of 8th International Conference on Communications, invited paper, Vol. 2, *IEEE Explore*, pp. 451-460, Bucharest, 2010.

[5] M.F. Vaida, R. Terec, **O. Tornea**, L. Chiorean, A. Vanea, "DNA Alternative Security", *Advances in Intelligent Systems and Technologies Proceedings ECIT2010 – 6th European Conference on Intelligent Systems and Technologies*, pp. 1-4, Iasi, Romania, October 2010.

[6] **O. Tornea**, M.E. Borda, V. Pileczki, R. Malutan, "DNA Vernam cipher", *E-Health and Bioengineering Conference (EHB)*, pp. 1-4, 2011.

[7] **O. Tornea**, M. E. Borda, V. Pileczki, "Cryptographic Algorithm based on DNA Nanotechnology", *The 6th International Conference on Interdisciplinarity in Education ICIE'11*, pp. 237 – 241, Karabuk/Safranbolu, Turkey, April 2011.

[8] **Olga Tornea**, Monica E. Borda, "Security and Complexity of a DNA-Based Cipher", 11th RoEduNet International Conference "Networking in Education and Research", Sinaia, Romania, ARNIEC/RoEduNet Agency, IEEE Romanian Section, ISSN-L 2068-1038, pp. 182-186, January 17-19 2013.

[9] **Olga Tornea**, Marc Antonini, Monica Borda, "Multimedia Data Compression and Encryption using DNA Cipher", *SSET 2013* winner of the Doctor ETTI (2nd Prize), Cluj-Napoca, Romania, May 24 2013.

[10] Monica E. Borda, **Olga Tornea**, Romulus Terebes, Raul Malutan, "New DNA Based Random Sequence Generation and OTP Encryption Systems for Transmission and Storage", *The 6th International Conference on Security for Information Technology and Communications*, June 25, 2013.

Honors & Awards

- Excellence Prize and Gold Medal at 11th PROINVENTICA 2013

  Professor Monica Borda, Ph.D. Student Olga Tornea, Associate Professor Romulus Terebes, Assistant Professor Raul Malutan for "OTP Encryption Method and System based on Random Sequences Determined from DNA Structures".
- SSET 2013 winner of the Doctor ETTI, 2nd Prize, with the paper: "Multimedia Data Compression and Encryption using DNA Cipher".

Citations

[I] M. Danziger, M.A. Henriques, "Computational Intelligence Applied on Cryptology: a Brief Review", IEEE Latin America Transactions, Vol. 10, No. 3, April 2012 (***cites publication [2]***)

[II] C. Popovici, "Aspects of DNA Cryptography", Annals of the University of Craiova, Mathematics and Computer Science Series, Vol. 37(3), pp. 147-151, 2010 (***cites publication [2]***)

[III] R. Soni, V. Soni, K. Mathariya, "Innovative field of cryptography: DNA cryptography", International Conference on Information Technology Convergence and Services, January 2012 (***cites publications [1], [2], [3], [5]***)

[IV] M. Sabry, M. Hashem, T. Nazmy, "Three Reversible Data Encoding Algorithms based on DNA and Amino Acids' Structure", International Journal of Computer Applications, Vol. 54 No.8, September 2012 (***cites publication [2]***)

[V] T. Mandge, V. Choudhary, "A Review on Emerging Cryptography Technique: DNA Cryptography", IJCA Proceedings on International Conference on Recent Trends in Information Technology and Computer Science 2012 ICRTITCS(13):9-13, February 2013 (***cites publications [1], [4]***)

[VI] Y. Zhang, D. Zhou, L. He, Y.H. Karanfil, B. Fu, "A New DNA Cryptogram Scheme Based on PCR Technology", Journal of Theoretical and Applied Information Technology, Vol. 45, No.1, November 2012 (***cites publication [4]***)

[VII] G. Jacob, A. Murugan, "DNA based Cryptography: An Overview and Analysis", International Journal of Emerging Sciences, 3(1), 36-42, March 2013 (***cites publication [4]***)

[VIII] A. Agrawal, A. Bhopale, J. Sharma, M.A. Shizan, D. Gautam, "Implementation of DNA algorithm for secure voice communication", International Journal of Scientific & Engineering Research, Vol. 3, June 2012 (***cites publication [2]***)

Bibliography

[ABR+93] M. Antonini, M. Barlaud, B. Rougé, and C. Lambert-Nebout, "Weighted Optimum Bit Allocation for Multiresolution Satellite Image Coding", in *CCECE*, Vancouver, Canada, pp. 14-17, September 1993.

[ABM+92] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform", *IEEE Transactions on Image Processing*, Vol. 1, No. 2, pp. 205 – 220, 1992.

[Adl94] L.M. Adleman, "Molecular computation of solution to combinatorial problems", *Science*, Vol. 266, pp. 1021-1024, 1994.

[Ant11] M. Antonini, "Techniques de compression pour le codage des images et des vidéos", lecture notes, 2011.

[ASE06] S. T. Amin, M. Saeb, S. El-Gindi, "A DNA-based Implementation of YAEA Encryption Algorithm", *IASTED International Conference*, pp. 120-125, 2006.

[BDH+04] S. Block, D. Donoho, T. Hwa, et al., "DNA Barcodes and Watermarks", *MITRE Corporation*, 2004.

[Ble10] G. E. Blelloch, "Introduction to Data Compression", Carnegie Mellon University, 2010.

[Boc09] I. Bocharova, "Compression for Multimedia", *Cambridge University Press*, 2009.

[Bor11] M. Borda, "Fundamentals in Information Theory and Coding", *Springer*, May 2011.

[BS04] K. Barclay, J. Savage, "Object-Oriented Design with UML and Java", *Elsevier*, 2004.

[BT10] M. Borda, O. Tornea, "DNA Secrete Writing Techniques", *Proceedings of 8th International Conference on Communications,IEEE Explore*,Vol. 2, pp. 451-460, Bucharest, 2010.

[BTT+13] M. Borda, O. Tornea, R. Terebes, R. Malutan, "Method and cryptographic OTP system based on DNA random sequences", Gold Medal for Patent request at ProInvent 2013, No. of patent application: A10003/14.02.2013.

[BU98] A. Bruckmann, A. Uhl, "Selective Medical Image Compression using Wavelet Techniques", *Journal of Computing and Information Technology (Special Issue on Biomedical Image Processing and Analysis)* 2:6, pp. 203-213, 1998.

[CD+04]C.R. Calladine, H.R. Drew, B.F. Luisi, A.A. Travers, Understanding DNA The Molecule & How It Works, Academic Press, 2004.

[CL00] H. Cheng, X. Li, "Partial encryption of compressed images and videos", IEEE Transactions on Signal Processing, Vol. 48(8), pp. 2439–2451, 2000.

[CL+09] S. Christley, Y. Lu, C. Li, X. Xie, "Human genomes as email attachments", *Bioinformatics*, Vol. 25, pp. 274–275, 2009.

[CP09] D.P. Clark, N.J. Pazdernik, "Biotechnology: Applying the Genetic Revolution", *Elsevier*, 2009.

[CT91] T.M. Cover, J.A. Thomas, "Elements of Information Theory", *John Wiley & Sons, Inc.*, 1991.

[Den07] Tom St Denis, "Cryptography for Developers", *Syngress Publishing, Inc.*, 2007.

[DBV+03] J. Dolezel, J. Bartos, H. Voglmayr, J. Greilhuber, "Nuclear DNA content and genome size of trout and human", *Cytometry, Wiley-Liss, Inc.*, Vol. 51, No. 2,pp. 127–128, 2003.

[ddbj] http://www.ddbj.nig.ac.jp/Welcome-e.html

[DH76] W. Diffie, and M. Hellman, "New Directions in Cryptography." *Proceedings of the AFIPS National Computer Conference*, June 1976.

[DR+10] K. Daily, P. Rigor, S. Christley, X. Xie, P. Baldi, "Data structures and compression algorithms for high-throughput sequencing technologies", *BMC Bioinformatics*, Vol. 11, No. 514, 2010.

[ElG85] T. ElGamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *IEEE Transactions on Information Theory*, Vol. 31 (4), pp. 469–472. 1985.

[ESU07] D. Engel, T. Stütz, A. Uhl, "Format-compliant JPEG2000 Encryption in JPSEC: Security, Applicability and the Impact of Compression Parameters", *EURASIP Journal on Information Security*, Volume 2007, Article ID 94565, pp. 20 pages, 2007.

[FIPS94] "Digital Signature Standard (DSS)", *Federal Information Processing Standards Publication 186*, May 1994. (http://www.itl.nist.gov/fipspubs/fip186.htm)

[FIPS93] FIPS 46-2, Data Encryption Standard, 1993.

[FIPS01] FIPS 197, Advanced Encryption Standard (AES), 2001.

[FL+11] M.H.Y. Fritz, R. Leinonen, G. Cochrane, E. Birney, "Efficient storage of high throughput DNA sequencing data using reference-based compression", *Genome Res*, Vol. 21, pp. 734-740, 2011.

[FS93] T.J. Fu, N.C. Seeman, "DNA double crossover molecules", *Biochemistry*, Vol. 32, pp. 3211–3220, 1993.

[GG91] A. Gersho, R.M. Gray, "Vector Quantization and Signal Compression", *Kluwer Academic Publishers*, 1991.

[GLR04] A. Gehani, T. LaBean, J. Reif, "DNA-Based Cryptography", *Springer*, Vol. 2950, pp. 167-188, 2004.

[GMO06] M. Grangetto, E. Magli, G. Olmo, "Multimedia selective encryption by means of randomized arithmetic coding", *IEEE Transactions on Multimedia*, Vol. 8, pp. 905-917, 2006.

[Gor11] Steven Gordon, "Key Management and Distribution", courses on Security and Cryptography, Sirindhorn International Institute of Technology, January 2011.

[HB07] D. Heider, A. Barnekow, "DNA-based watermarks using the DNA-Crypt algorithm", *BMCBioinformatics*, 8:176, 2007.

[Has88]J. Hastad, "Lower bounds in computational complexity theory", *Notices of the AMS*, Vol. 35, No 5, pp. 677–683, 1988.

[IHGM] Institute of Human Genetics, Munich - http://ihg.gsf.de/ihg/databases.html

[JHH+05] L. Jun-Hong, L. Hai-Kuo, A. Hong-Jie, W. Guo-Hua, et al., "Nano-manipulation of single DNA molecules based on atomic force microscopy", *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, Vol. 7, pp. 7478-81, 2005.

[KL10] L. Kencla, M. Loeblb, "DNA-inspired information concealing: A survey", *Elsevier Inc.*, pp. 251-262,2010.

[Knu94] L.R. Knudsen, "Block Ciphers - Analysis, Design, Applications," Ph.D. dissertation, Aarhus University, Nov 1994.

[Koh08] David R. Kohel, "Cryptography" 11th July, 2008.

[Kri08]http://kristofverbiest.blogspot.ro/2008/10/beware-of-stopwatch.html

[Lan11] E.S. Lander, "Initial impact of the sequencing of the human genome", *Nature*, Vol. 470, pp. 187-197, 2011.

[Lev98] M. Levitt, "How many base-pairs per turn does DNA have in solution and in chromatin.Some theoretical calculations", *Biochemistry*, Vol. 75, No. 2, pp. 640-644, 1998.

[Lia08]S. Lian, "Multimedia Content Encryption: techniques and applications ", *CRC Press*, 2008.

[LKCV06] A. N. Lemma, S. Katzenbeisser, M. U. Celik, and M. V. Veen, "Secure watermark embedding through partial encryption", *Proceedings of International Workshop on Digital Watermarking* (IWDW), Lecture Notes in Computer Science, Vol. 4283, pp. 433–445, 2006.

[LLA+04] J. Lu, H. Li, H. An, G. Wang, et al., "Positioning Isolation and Biochemical Analysis of Single DNA Molecules Based on Nanomanipulation and Single-Molecule PCR", *American Chemical Society*, Vol. 126, No. 36, pp. 11136-11137, 2004.

[LM91] X. Lai, and J. L. Massey, "A Proposal for a New Block Encryption Standard", *Springer-Verlag, Lecture Notes on Computer Science (LNCS)*, Vol. 473, pp. 389–404, 1991.

[MMP09] A. Mohapatra, P.M. Mishra, S. Padhy, "Discriminative DNA classification and motif prediction using weighted degree string kernels with shift and mismatch", *Advances in Computing, Communication and Control*, pp. 56-61, *ACM*, 2009.

[MOV96] A. Menezes, P. Van Oorschot, S. Vanstone, "Handbook of applied cryptography", *CRC Press*, 1996.

[NIST01] D.P. Leech, M.W. Chinworth, "The Economic Impacts of NIST's Data Encryption Standard (DES) Program", *TASC, Inc.*, October 2001.

[NIST95] National Institute of Standards and Technology, "An Introduction to Computer Security: The NIST Handbook", *NIST Special Publication 800–12*, October 1995.

[NW99] C.G. Nevill-Manning, I.H. Witten, "Protein is incompressible", *In Proceedings of the Conference on Data Compression (DCC '99)*, pp. 257, 1999.

[Par03] C. Parisot, "Model-Based allocations and scan-based discrete wavelet transform for image and video coding", PhD Thesis, *University of Nice-Sophia Antipolis*, 2003.

[PC08] W. Puech, G. Coatrieux, "Compression of Biomedical Images and Signals, Chapter 10: Hybrid Coding: Encryption-watermarking-compression for Medical Information Security", *ISTE-John Wiley & Sons, serie: Digital Signal Processing*, pp. 247-275, 2008.

[PGP04] PGP Corporation, Phil Zimmermann, "An introduction to cryptography", 2004.

[PR05] W. Puech, J.M. Rodrigues, "Crypto-Compression of Medical Images by Selective Encryption of DCT", *13th European Signal Processing Conference, EUSIPCO'05*, Antalya, Turkey, 2005.

[PTA+01] C. Parisot, S. Tramini, M. Antonini, M. Barlaud, C. Latry, C. Lambert – Nebout, "Optimisationd'uneChaîne Image de Télédétection : de la Compression Embarquée aux Post-Traitements Sol", *colloque GRETSI*, Toulouse, 2001.

[PTAB+01] C. Parisot, S. Tramini, M. Antonini, M. Barlaud, et al., "Optimization of the joint coding/decoding structure", *International Conference on Image Processing*, Grèce, 2001.

[RA11] P. Rajarajeswari, A. Apparao, "DNABIT Compress – Genome compression algorithm", *Bioinformation*, Vol. 5, No. 8, pp. 350–360, 2011.

[RJ91] M. Rabbani, P. W. Jones, "Digital Image Compression Techniques", Tutorial texts in optical engineering, Vol TT7, *SPIE Press*, 1991.

[RN88] J.L. Rodgers, W.A. Nicewander,"Thirteen ways to look at the correlation coefficient",*The American Statistician*, Vol. 42, No. 1, pp. 59 – 66, 1988.

[Rot06] P.W. Rothemund, "Folding DNA to create nanoscale shapes and patterns", *Nature*, Vol. 440, pp. 297-302, 2006.

[RRM+99] A. Romeo, G. Romdotti, M. Mattavelli, and D. Mlynek, "Cryptosystem architectures for very high throughput multimedia encryption: The RPK solution", In *Proceedings 6th IEEE International Conference on Electronics, Circuits and Systems* (*ICECS '99)*, Vol. 1, 5–8, pp. 261–264, 1999.

[RS10] O.S. Rao, S.P. Setty, "Efficient mapping methods for Elliptic Curve Cryptosystems", *International Journal of engineering Science and Technology*.Vol. 2, No. 8, pp. 3651-3656, 2010.

[RSA78] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM*, Vol. 21 Nr. 2 February 1978.

[Rum09] A. H. Rumpf, "A Brief Introduction to Data Compression and Information Theory", *Ripon College Summation*, pp. 15-18, 2009.

[RWB+96] S. Roweis, E. Winfree, R. Burgoyne, N. V. Chelyapov, et al., "A sticker based architecture for DNA computation", *In Proceedings of the Second Annual Meeting on DNA Based Computers*, Vol. 44 of *DIMACS*, pp. 1-30, 1996.

[Sanger] www.sanger.ac.uk

[Say03] K. Sayood, "Data Compression", Encyclopedia of Information Systems, Vol. 1, pp. 423 – 444, *Elsevier Science*, 2003.

[SB98] C. Shi, B. Bhargava, "A fast MPEG video encryption algorithm", In *Proceedings 6th ACM International Multimedia Conference*, pp. 81–88, 1998.

[Sch96] B. Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C", *John Wiley & Sons Inc.*, 1996.

[Sch02] B. Schneier, "Crypto-Gram Newsletter", *Counterpane Internet Security, Inc.*, October 2002.

[Sch09] B. Schneier, "The History of One-Time Pads and the Origins of Sigaba", Blog post, 2009.

[See81]N.C. Seeman, "Nucleic Acid Junctions: Building Blocks for Genetic Engineering in Three Dimensions",*R.H. Sarma, Adenine Press*, pp. 269-277, 1981.

[SFP02] B. Shimanovsky, J. Feng, M. Potkonjak, "Hiding Data in DNA", *Proceeding IH '02 International Workshop onInformation Hiding*, pp. 373-386, 2002.

[Sha49] C.E. Shannon, "Communication Theory of Secrecy Systems", *Bell System Technical Journal*, Vol. 28, No. 4, pp. 656-715, 1949.

[Sha48] C.E. Shannon, "A Mathematical Theory of Communication",*Bell System Technical Journal*, Vol. 27, No. 3,pp. 379–423, 1948.

[Sha93] M. Shapiro, "Embedded Image Coding using Zerotrees of Wavelet Coefficients", *IEEE Trans. Signal Processing*, Vol. 41, pp. 3445-3462, 1993.

[SK05] P. Salama, B. King, "Efficient secure image transmission: compression integrated with encryption", *Proc. SPIE*, Vol. 5681, pp. 47-58, 2005.

[SNF10] H. Shiu, K. Ng, J.F. Fang, et al., "Data hiding methods based upon DNA sequences", *Elsevier Inc.*, pp. 2196–2208,2010.

[Sta11] W. Stallings, "Cryptography and Network Security: Principles and Practice", (5th Ed.), *Prentice Hall*, 2011.

[SQJ04]W.M. Shih, J.D. Quispe, G.F. Joyce, "1.7-kilobase single-stranded DNA that folds into a nanoscale octahedron", *Nature*, Vol. 427, pp. 618-621, 2004.

[SWY+98] N.C. Seeman, H. Wang, X. Yang, and J. Chen, "New Motifs in DNA Nanotechnology", *Nanotechnology*, Vol. 9, No 3, pp. 269-277, 1998.

[TAB13] O. Tornea, M. Antonini, M. Borda, "Multimedia Data Compression and Encryption using DNA Cipher", *Communications Department, Technical University of Cluj-Napoca*, winner of 2nd Prize, album *SSET 2013*.

[TB09] O. Tornea and M. E. Borda, "DNA Cryptographic Algorithms", *IFMBE Proc.*, Vol. 26, pp. 223-226, 2009.

[TB13]O.Tornea, M.E. Borda, "Security and Complexity of a DNA-Based Cipher", *11th RoEduNet International Conference "Networking in Education and Research"*, pp. 182-186, 2013.

[TBH+10] O. Tornea, M.E. Borda, T. Hodorogea, M. Vaida, "Encryption system with Indexing DNA Chromosomes Cryptographic Algorithm", *IASTED International Conference*, 680-099, pp. 12-15, 2010.

[TBP11]O. Tornea, M.E. Borda, V. Pileczki, "Cryptographic Algorithm based on DNA Nanotechnology", *The 6th International Conference on Interdisciplinary in Education ICIE'11*, April 2011.

[TBP+11] O. Tornea, M.E. Borda, V. Pileczki, R. Malutan, "DNA Vernam cipher", *E-Health and Bioengineering Conference (EHB)*, 2011.

[TRB99] C. T. Taylor, V. Risca, and C. Bancroft, "Hiding messages in DNA microdots", *Nature*, Vol. 399, pp. 533-534, 1999.

[UCLA] Image Communications Lab at UCLA - http://johnvillasenor.com/

[Use96] B. Usevitch, "Optimal Bit Allocation for biorthogonal wavelet coding", *Proceedings of Data Comperssion Conference*, Snowbird, USA, March 1996.

[Ver26] G. S. Vernam, "Cipher Printing Telegraph Systems", *Journal of* the *American Institute of Electrical Engineers*, Vol. XI.V, pp. 109-115, 1926.

[VF11] L. M. Varalakshmi, S. G. Florence, "An enhanced encryption algorithm for video based on multiple Huffman tables", *Springer*, 2011.

[VTT10] M.F Vaida, R. Terec, O. Tornea, L. Chiorean, A. Vanea, "DNA Alternative Security", *Advances in Intelligent Systems and Technologies Proceedings ECIT2010 – 6th European Conference on Intelligent Systems and Technologies*, pp. 1-4, October 2010.

[wacnr] http://www.dsimb.inserm.fr/~fuchs/M2BI/AnalSeq/Annexes/Sequences/

Accession_Numbers.htm

[WC53] J. Watson, F. Crick, "Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid", *Nature*, Vol. 171, No. 4356, pp. 737–738, 1953.

[WD04] Y. Wu, R. H. Deng, "Compliant encryption of JPEG2000 code-streams", *Image Processing ICIP*, Vol. 5, pp. 3439 – 3442, 2004.

[webi] http://www.ebi.ac.uk/ena/

[wgen]www.genelink.com

[wncbi] http://www.ncbi.nlm.nih.gov/genbank

[wncbi'] http://www.ncbi.nlm.nih.gov/pubmed/21071399

[WK01] C. Wu and C.-C. J. Kuo, "Efficient multimedia encryption via entropy codec design", *SPIE international symposium on electronic imaging*, Vol. 4314, pp. 128-138, 2001.

[WSZ+01] J. Wen, M. Severa, W. Zeng, M. Luttrell, W. Jin, "A Format-Compliant Configurable Encryption Framework for Access Control of Multimedia", *IEEE Workshop on Multimedia Signal Proc.*, pp. 435-440, 2001.

[XK04] D. Xie, C.-C. J. Kuo, "Enhanced multiple Huffman table (MHT) encryption scheme using key hopping", *In Proceedings IEEE International Symposium on Circuits and Systems (ISCAS)*, Vol. 23-26, pp. 568–571, 2004.

[You06] W.R. Yount, "Research Design and Statistical Analysisin Christian Ministry" (chapter 22: Correlation Coefficients), 4th ed.*Napce Organisation*, 2006.

[ZL78] J. Ziv and A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding", *IEEE Transactions on Information Theory*, Vol. 24, No. 5, pp. 530–536, 1978.

Appendix

The most relevant papers for the thesis research activity:

[1] **Olga Tornea**, Monica Borda, Tatiana Hodorogea, Mircea-Florin Vaida, "Encryption System with Indexing DNA Chromosomes Cryptographic Algorithm", *IASTED International Conference on Biomedical Engineering (BioMed 2010)*, Innsbruck, Austria, paper 680-099, pp. 12-15, ISBN: 978-0-88986-825-0, 975-0-88986-826-7, Acta Press, 15-18 February 2010.

[2] **O. Tornea**, M.E. Borda, V. Pileczki, R. Malutan, "DNA Vernam cipher", *E-Health and Bioengineering Conference (EHB)*, 2011.

[3] **Olga Tornea**, Marc Antonini, Monica Borda, "Multimedia Data Compression and Encryption using DNA Cipher", *SSET 2013* Doctor ETTI, Cluj-Napoca, Romania, May 24 2013.

[4] Monica E. Borda, **Olga Tornea**, Romulus Terebes, Raul Malutan, "New DNA Based Random Sequence Generation and OTP Encryption Systems for Transmission and Storage", *The 6th International Conference on Security for Information Technology and Communications*, June 25, 2013.

Proceedings of the 7th IASTED International Conference
Biomedical Engineering (BioMED 2010)
February 17 - 19, 2010 Innsbruck, Austria

ENCRYPTION SYSTEM WITH INDEXING DNA CHROMOSOMES CRYPTOGRAPHIC ALGORITHM

Olga Tornea, Monica Borda, Tatiana Hodorogea, Mircea-Florin Vaida
The Faculty of Electronics, Telecommunications and Information Technologies,
Technical University of Cluj-Napoca, 26-28 George Baritiu Street
Cluj-Napoca, 400027, Romania
tornea.olga@yahoo.com, Monica.Borda@com.utcluj.ro, Tatiana.Hodorogea@com.utcluj.ro, Mircea.Vaida@com.utcluj.ro

ABSTRACT
DNA has a great cryptographic strength, its binding properties between nucleotides bases (A—T, C—G) offer the possibility to create self-assembly structures which are an efficient means of executing parallel molecular computations. Our work is based on the development of the new encryption system with indexing DNA chromosomes cryptographic algorithm using MATLAB Bioinformatics Toolbox. This toolbox offers support for developing DNA cryptographic operations and providing more secure cryptographic algorithms. The algorithm is based on the idea to use DNA chromosomes as one-time-pad structures and index them in order to encrypt the plaintext message.
Our work is based on the complexity of the development of the new encryption system with indexing DNA chromosomes cryptographic algorithm.

KEY WORDS
DNA Encryption (DNAE) System, Central Dogma of Molecular Biology (CDMB), One-Time-Pad, Bio-nanotechnology, Indexing Chromosomes

1. Introduction

Why we need data security is already well-known. Do we need to find alternative, more secure encryption techniques for protecting sensitive data? With current network, Internet, and distributed systems, cryptography has become a key technology to ensure the security of today's information infrastructure. A cryptographic system that an attacker is unable to penetrate even with access to infinite computing power is called *unconditionally secure*. The mathematics of such a system is based on information theory and probability theory. When an attacker is theoretically able to intrude, but it is computationally infeasible with available resources, the cryptographic system is said to be *conditionally secure.* The mathematics in such systems is based on computational complexity theory. To design a secure cryptographic system is a very challenging. A cryptographic system has one or more algorithms which implement a computational procedure by taking a variable input and generating a corresponding output. DNA cryptography is based on Adleman's research of DNA computing [1]. Basic procedures of DNA OTP encryption schemes are given by [2]. DNA consists of two complementary strands. Each strand is made of a series of units called "nucleotides". In this algorithm a DNA strand is meant as a series of such bases. DNA chromosomes consist of many different genes, hundreds of bases long. Chromosomes are used in this cryptosystem as OTP where information is not properly stored, but they are used as a place with indexes to the real message. The principle of indexing used in this algorithm is presented in [3]. If an algorithm's behaviour is completely determined by the input, it is called *deterministic,* and if its behaviour is not determined completely by input and generates different output each time executed with the same input, it is *probabilistic*. A distributed algorithm in which two or more entities take part is defined as a protocol including a set of communicational and computational steps. Each communicational step requires data to be transferred from one side to the other and each computational step may occur only on one side of the protocol. The goal of every cryptographer is to reduce the probability of a successful attack against the security of an encryption system – to zero. Probability theory provides the answer for this goal. Our work is based on the complexity of the development of the new encryption system with indexing DNA chromosomes cryptographic algorithm, an unconditionally secure and probabilistic DNAE System. DNAE is designed to have applications in textual and image information security.

2. The Encryption Protocol

Adleman began the new field of bio-molecular computing research. His idea was to use DNA biochemistry for solving problems that are impossible to solve by

conventional computers, or that require an enormous number of computation steps. The DNAE technique simulates the CDMB steps: transcription, splicing, and translation process. The time complexity of an attack on a message of length *n*, is $O(2^n)$. DNA computing takes advantages of combinatorial properties of DNA for massively-parallel computation.

In our work we used a cryptosystem with symmetric key named One-Time-Pad (OTP). It is an algorithm where each key is used just once where from the name of One-Time-Pad. OTP encryption uses a large non-repeating set of truly random key letters. Each pad is used exactly once, for only one message. The sender encrypts the message and then destroys the used pad. As it is a symmetric key cryptosystem, the receiver has an identical pad and uses it for decryption. The receiver destroys the same pad after decrypting the message. New message means new key letters. A ciphertext message is equally likely to correspond to any possible plaintext message. Cryptosystems which use a secret random OTP are known to be perfectly secure [4].

Introducing DNA into the common symmetric key cryptography, it is possible to follow the pattern of symmetric key cryptosystem, while also exploiting the inherent massively-parallel computing properties and storage capacity of DNA in order to perform the encryption and decryption using OTP keys [5]. The resulting encryption algorithm which uses DNA medium is much more complex than the one used by conventional encryption methods.

We developed an encryption algorithm which uses OTP as symmetric key and real chromosomal sequence as OTP. We extracted chromosomal sequence from public available data bases [6] and used it for implementation of this algorithm (Figure 1 and 2).

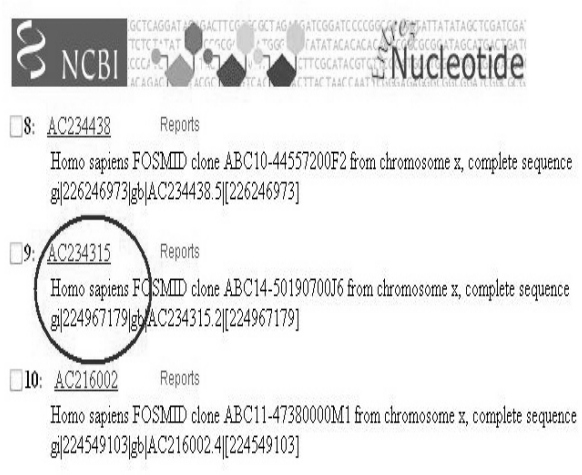


Figure 1 Chromosome extracted for implementation from public available data base

```
>gi|224967179|gb|AC234315.2| Homo sapiens FOSMID clone ABC14-50190700J6 from
chromosome x, complete sequence
TTCCCAATAGGCTGGACTGCTTACCACCCCATGTGGCCTCAAAGAGCTCCAGTCACTCCTTTACGAACCC
AATCACTCCAGAACTTTAGAACAAAGTTTCTGAGTTACTCCTTGTAATAGGCTAAATAATGGCTCCCAAA
GATATTAGGATTTGATTCCCAGAACCTATAAATATTACCTTATTTGGAAAACGGTTCTTAGCAGATGTGA
TTGAGTTAAGGATATTGAGATGCAGAGATTATTTTAGATTATCTAGACTATCTGGGTGGATGTATTGGTC
AGGGTTCTTCAGAGGACAGAGCCAATAGGATATATGTATATAAAAAGGGAGTTAATTAGGGAGAATTGGC
TCACATGATTACAAGGTGAAGTCCCACGATAGGCCGTCTGCAAACTGGGGAGAGAAGCTAGTTGTGTGGC
TCAGTCCAAATCCAAAAGCCTCAAAACTGGAGAAGCTGACAGTACAAGCCCTAGTCTGAGGCCAAAGGTC
CAAGAGCCCCTGAGAGGCTGCTGGTGCAAGTTCCAGAGTCCAAAGGTTAACAAACCTGAAGTCTGGTGTC
CAAAGGCAGGAGGAGAGGAAGCAGACAGGAAGAGAGAAAGCAAACAGACTCAGCAAGAAAGCTGCTGTTC
TTCCACCTGCTTTGTTCTAGCCACGCTGGCAGTCAATTGCATGGTGCCCATCCACACTGAGGGTGGATCT
TCCTCTAACAGTCAAACACTGACTCAAATGTCATCTTCTCTGGCAACACCCTCACAGACACACCCAGAAA
CAATGCTTCACCAGCCATCTATGCAGCCCTCAATCCAGTCAAGGTGACACCTAATGGTTAATGGTTATTA
ACCACGGTTAATAACCATGACAGTGGGTTCTAAATGTAATCACGTGTATCCTTATAAAAAAAAGAGGCAG
AGGGAGATTTGAAGAGCTATACAGAGGAGAAGACAACGTGAAGATGGAGGAGAGAGAAATTTGGCCATCA
```

Figure 2 Fragment from sequence file in FASTA format

3. Message Encryption

Plaintext message was transformed in bits and after that in DNA format. We used a text message for encryption so an encryption unit was a character and in ASCII cod it was represented on 7 bits, or in case it was an image a pixel was an encryption unit and it could be represented on 8 bits at least. Transformation from a 2-letter (0, 1) alphabet to a 4-letter (A, C, G and T) alphabet was done using 2 bits to represent a letter:

A – 00
C – 01
G – 10
T – 11

Using this substitution a character was represented on 4 letters which is equivalent to a byte.

Using Matlab functions we obtained decimal ASCII cods of the plaintext message, and transformed them in binary form, each character on 8 bits. After that, using functions from Bioinformatics Toolbox we transformed our message from binary to DNA alphabet.

Each character was transformed in a 4-letter DNA sequence and searched in the chromosomal sequence, used as OTP. OTP sequence was scanned from bases 1 to 37839. At each step was analyzed a segment of 4 bases from the OTP sequence and compared to the characters DNA sequence. If 4-letter sequence representing a character from the message was retrieved in the chromosomal sequence then the starting point (index in chromosome) of identical 4-letters was memorized in an array. At the next step was analyzed another 4 bases from OTP where first 3 of them are the last bases from previous step (Figure 3).

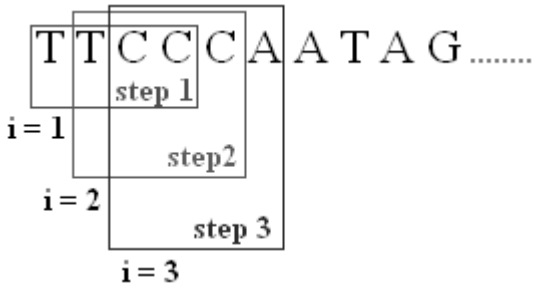


Figure 3 Exemplification of the OTP scanning process for message encryption

For each character was obtained an array of indexes in chromosomal sequence. Number of indexes for a character depends on how often the character's DNA sequence retrieved in the chromosomal sequence. For each character was chosen a random index from its array of indexes. We obtained the final encrypted message: an array of random indexes, one for each character. Below is presented example of the message encryption with implementation results.

Example of implementation results:

Message: "secret"
ASCII cods: 115 101 99 114 101 116

s → 115 → 01110011 → CTAT → indexes:

166	258	789	927	1295	2954
3045	3098	3181	3207	3361	3763
4436	4559	5242	5443	5794	5938
5966	7392	7698	7762	7789	7832
8128	8627	9918	11871	12240	12332
12383	12581	13107	13128	13324	14919
15169	15177	15494	15602	15844	16073
16369	16829	16891	16939	17227	17342
17718	17818	18564	19530	20022	20437
20619	21145	21411	21419	21725	22030
22051	23157	23180	23231	23311	23367
23430	23434	23556	23811	24005	24038
24182	24568	25871	27176	27208	27896
29321	29642	29848	30087	30097	30110
30438	30472	31090	31487	33204	33226
33321	33378	33612	35520	35530	35646
35768					

e → 101 → 01100101 → CGCC → indexes:

3381	3760	3951	4386	6892	7171
7283	7306	7440	8016	8176	8221
8493	8877	10047	10747	10751	11132
13405	13676	13681	13758	13769	13893
13941	13981	14057	14083	14097	14134
14137	14162	14173	14176	14253	14282
14322	14333	14381	14418	14496	14531
14623	14658	15985	16148	21816	22334
23049	23113	26603	26712	27049	29764
30184	31147	32741	33015	33027	33117
34420	34638	34906	35066	36156	36215
36227	36315	37436			

For each character was chosen a random index from its array of indexes using Matlab function. Below are established positions of random indexes inside character's arrays:

115 → 70th index 23811
101 → 26th index 13981
99 → 7th index 8011
114 → 57th index 21195
101 → 57th index 32741
116 → 158th index 25264

Final encrypted message is:

23811 13981 8011 21195 32741 25264

4. Message Decryption

At message decryption is used the same OTP as at encryption, because it is a symmetric key algorithm. The key is Homo sapiens FOSMID clone ABC14-50190700J6, from chromosome x complete sequence. First we read this sequence using functions from Bioinformatics Toolbox:

FASTAData = fastaread('homo_sapiensFosmid_clone.fasta')

Each index from received encrypted message was used to point in chromosomal sequence:

SeqNT=FASTAData.Sequence(i:i+3)

Using these pointers we extracted for each character a 4-bases DNA sequence. This variable was transformed in numerical value, using transformation offered by Matlab Bioinformatics Toolbox (A-1, C-2, G-3, T-4). As transformation starts with 1, at encryption to each digit was added a unit and after that it was transformed in base (example, "00" binary → 0 digit → 0+1 → A). At decryption from each obtained digit was subtracted a unit and after that transformed in 2 bits:

Example:

CCCA (bases) → 2221(digits) → 2-1, 2-1, 2-1, 1-1 → 1110 (digits) → 01 01 01 00 (bits)

Obtained binary numbers are the ASCII cods of the recovered message characters.

5. Complexity of Secure DNA Encryption System

With an OTP, an adversary has no information about how to cryptanalyze the ciphertext, since every key is equally likely to correspond. Cryptosystems which use a secret random OTP are known to be perfectly secure and are applicable primarily for transmission of ultra-secure information. The problem of this cryptosystem is that the key letters have to be truly random and the key sequence can not be reused ever again.

Here is where the advantages of using DNA in cryptography came. We used at implementation as the key a human chromosome, but any genetic sequences, of any living matter can be used as a secret key for encryption. We can exploit great storing capabilities and variety of DNA sequences for the usual OTP cryptosystem.

This encryption algorithm treats also the problem of the vulnerability to frequency attack. For the same character from the plaintext message we obtained a number of different indexes which are used as values in the ciphertext by a random choice. This solve the problem of frequency distribution of letters in a ciphered message [7] .The same character, for instance "e" will appear in ciphertext as 13981, or 32741, or any other index which was found for this character in the chromosome. Another advantage of this algorithm is that at encryption of another message, indexes for each character will be different from the previous encryptions values. The same character will appear in ciphertext under different values at encryption of two different messages. Chromosome or any DNA sequence which was chosen to be the encryption key dictates what kind and how many indexes for ciphertext will have a character.

6. Conclusion

Based on the ideas presented in [1], [2], and [3] an original DNA cryptographic algorithm was performed. We use one-time-pad principle and DNA chromosomes storing capabilities for message encryption. Implementation was performed in Matlab using Bioinformatics Toolbox and genetic database maintained by NCBI. One single chromosome from any species is composed from thousand of bases and it is perfect to be used in this algorithm to address the characters from the plaintext. Each character from the plaintext message was transformed into a unique sequence of 4 bases and searched in the chromosome, used as OTP. A random index of a character in chromosome becomes part of the ciphertext. The strength of this algorithm is based on the secrecy of the OTP and protection from frequency attack. The aim of this paper is to find useful and practical DNA cryptographic algorithm and to study its applicability in DNA technology. Laboratory implementations are possible (microarray technology [8]), but are still expensive and time consuming. Despite of this, simple and effective algorithms are necessary in order to bring DNA computing on digital level and use it on a large scale.

Acknowledgements

This paper was funded by the Romanian Agency UEFISCU within the PN II, IDEI no. 909/2007 research grant.

Research Project PN2, "Alternative Security Technologies for Network Applications"- supported by the National Authority of Romania.

References

[1] L. M. Adleman, Molecular computation of solution to combinatorial problems, *Science, 266*, 1994, 1021-1024.

[2] A. Gehani, T. LaBean, and J. Reif, DNA-Based Cryptography, *Lecture Notes in Computer Science, Springer*, 2004.

[3] S. T. Amin, M. Saeb, S. El-Gindi, A DNA-based Implementation of YAEA Encryption Algorithm, *IASTED International Conference on Computational Intelligence*, San Francisco, 2006, 120-125

[4] B. Schneier, *Applied cryptography: protocols, algorithms, and source code in C* (John Wiley & Sons Inc, 1996.)

[5] Tatiana Hodorogea, Mircea-Florin Vaida, Alternate Cryptography Techniques, ICCC05, Miskolc-Lillafured, Hungary, 24-27, 2005, Vol. 1, 513-518

[6] R. K. Wilson, The sequence of Homo sapiens FOSMID clone ABC14-50190700J6, submitted to http://www.ncbi.nlm.nih.gov, 2009.

[7] Kahn D., *The codebrakers* (McMillan, New York, 1967)

[8] M. Schena, *Microarray analysis* (Wiley-Liss, July 2003.)

DNA Vernam Cipher

O. Tornea[1], M. E. Borda[1], V. Pileczki[2] and B. Belean[1]

[1]Communications Department, Technical University of Cluj-Napoca, Romania
[2] Biology Department, Babes-Bolyai University, Cluj-Napoca, Romania
tornea.olga@yahoo.com, Monica.Borda@com.utcluj.ro, valentinapilecki@yahoo.com, Bogdan.BELEAN@com.utcluj.ro

***Abstract* - This encryption method is based on a truly random key used only once in logical XOR operation with the plaintext. Binary XOR was performed using biomolecular computation and artificially designed DNA motifs. A study of most suitable DNA structures was performed and the isomer of double crossover molecule (DX) named DAE molecule was chosen as a building block of mathematical XOR operation. These structures named also DNA tiles can perform self-assembling operations. Another important property of these structures is the natural hybridization process between them which gives a truly random symmetric key.**

Keywords: DNA nanotechnology, DNA self-assembly, one-time-pad, biomolecular computation, symmetric encryption.

I. INTRODUCTION

Vernam cipher was invented by Gilbert Vernam in 1917 as a teletype cipher [1]. He defined a stream cipher where a key, stored on a punched tape, was combined, applying XOR operation, character by character with the plaintext message producing a ciphertext. If the key string of bits is truly random and used only at one single encryption-decryption process, then Vernam cipher become one-time-pad which was co-invented by G. Vernam and J. Mauborgne.

One-time-pad (OTP) encryption method has been proved to be unbreakable by Claude Shannon at Bell Labs. He defined characteristics for the unbreakable encryption system which are the same with the OTP properties: the key must be truly random, at least as large as the plain-text, never reused in whole or part, and kept secret [2]. Bruce Schneier [3] called OTP a "perfect encryption scheme" in case the key meet the requirements to be truly random and to never be reused. He also said that truly randomness can not be achieved by mathematical functions. Only certain physical processes can guarantee really random numbers. The use of mathematical formula will lead as to the pseudo-random numbers: numbers that appear random, but are predetermined in reality. True random numbers, instead, come from measurements of random physical phenomena like atmospheric noise or radioactive source. This principle is used in this study and the physical phenomenon is DNA hybridization.

DNA hybridization was used here for the encryption-decryption process and for the key generation. One of the important steps was to find a DNA structure to represent an information unit. For this purpose specialized literature was consulted [4, 5] and a suitable structure was chosen.

This encryption method combines the power of biomolecular computation and classical XOR one-time-pad cipher. This principle was inspired from [6] where DNA triple-helix molecules were used to perform XOR OTP cipher. The experimental feasibility of biomolecular computing was demonstrated in 1994 by Leonard Adleman [7]. His experiment, computed in vitro was to solve Hamiltonian Path Problem using single stranded DNA molecules and a series of procedures adapted from molecular biology. This experiment was a starting point for carrying out computations at the molecular level followed by the series of works in DNA computing field.

Details about implementation steps of DNA Vernam cipher are presented as follows: section 2 describes design of different DNA structures and their stability; section 3 is about implementation of DNA structure and section 4 describes the algorithm which needs to be performed in laboratory.

II. DESIGN OF DNA STRUCTURE

Design of controllable DNA structures was first introduced by Nadrian Seeman in 1980s [8]. He invented DNA nanotechnology - the science and technology of building devices using DNA molecules. In nature DNA plays the role of genetic information carrier, but in this branch of technology it is used just as a structural material. Attraction of complementary DNA strands is used for building different nanoscale structures which are described below. At the end of this section is presented a DNA structure chosen for this project.

Holliday junction is one of the simplest forms of DNA building blocks. It is a junction between 4 complementary to each other DNA strands (Fig. 1). It is not recommended to be used in the nanoengineering because of its instability induced by strong electrostatic repulsion [5, 9].

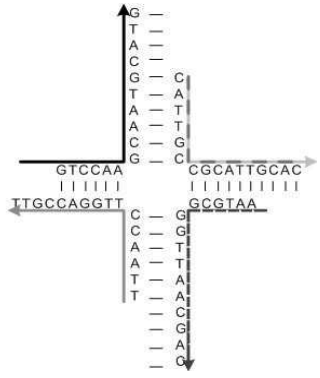


Fig. 1. Holliday junction

Double crossover molecule (DX) is a more stable structure which consists of two DNA helixes connected by two Holliday junctions. There are five different structures of DX molecules [9] (Fig. 2). Three of them have parallel helical domains: DPE, DPOW, DPON and the other two antiparallel helical domains: DAE, DAO.

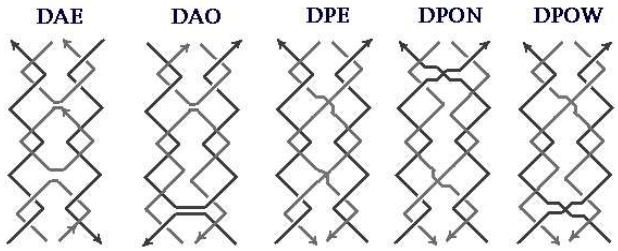


Fig. 2. DNA Double Crossover Molecules

DX molecules with antiparallel domains are more stable than those with parallel domains [10]. Therefore the DAE and DAO structures present more interest. The difference between them is that DAE molecule has even number of double helical half-turns between crossovers and DAO molecule has odd number of half-turns. Here a DAE structure was selected because it is stable and allows the use of one long DNA strand and several short strands in its construction.

DNA Origami was developed by Paul Rothemund [4]. This method consists of using one long single DNA strand, in order to create a basic structure, and many short DNA sequences that come as complementary parts to the basic structure forming in this way double stranded DNA in desired shape (Fig. 3). The same principle was used in [11].

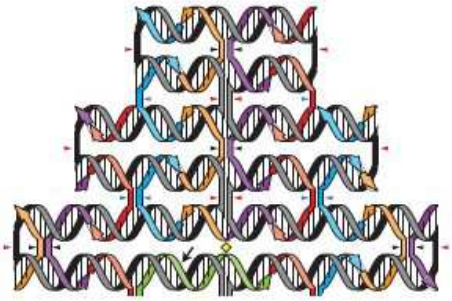


Fig. 3. DNA Orgami

Crossovers that appear between DNA helixes are incorporated for stability of the structure. In [4] crossovers appear at every 1.5 helical turns which is equivalent to 16 bases space between crossovers.

An important aspect is finding a long single-stranded DNA. Certain synthesizers allow the synthesis of long oligonucleotides up to 250 bases [12]. Longer single-stranded DNA can be founded in viruses like M13mp18 used in [4].

Considering all the aspects presented above, for this study a stable DAE structure was selected with one long DNA strand and several short strands (Fig. 4). Long DNA strands can be synthesized as oligonucleotides of 210 bases in length and short strands as oligonucleotides of 42, 46, 50, or 54 bases in length. The length of short oligonucleotides depends on the number of single-stranded (sticky) terminations of the structure. These terminations are used for binding other structures with complementary ends. Crossovers were selected at every helical period (2 turns). One helical turn has 10.67 ≈ 10.5 bases [13] which mean that distance between crossovers will be 21 base pairs (bp).

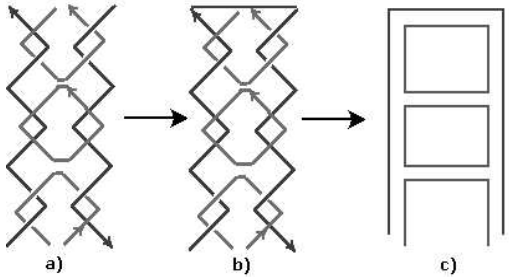


Fig. 4. DAE molecule b) modified DAE molecule c) schematic view

III. Implementation of DNA Structure

In the previous section the desired DNA shape was established in order to be used as an information unit in Vernam cipher. Here is presented the implementation of the selected structure.

From the initial long DNA strand of 210 bases were calculated offsets for the complementary strands in order to create the desired shape (Fig. 5). Arrows in the figure 5 represents opposite polarities between strands (5' to 3' directions). The longest DNA strand (210 bases) was sequenced in intervals of 21 bases which is equivalent to a DNA period assuming that per turn there are 10.5 base pairs (bp). In this case a DNA period is the distance between crossovers.

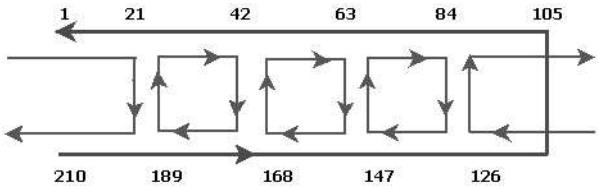


Fig. 5. Schematic DNA structure with offsets

Steps of the implementation for this structure are the following:

1) Generation of random sequence from DNA alphabet (A, C, T, G) of length 210 bases.

2) Generation of complementary sequences to the first one. For example for the first middle circular sequence (Fig. 5) complementary sequences were generated in intervals: 22-42 and 169-189.

3) At the edges complementary sequences can have different lengths depending on the number of "sticky end" of the structure. For such sequences is performed concatenation of the terminations like: "ACTG", "TACC", or "TGAC" (Fig. 6).

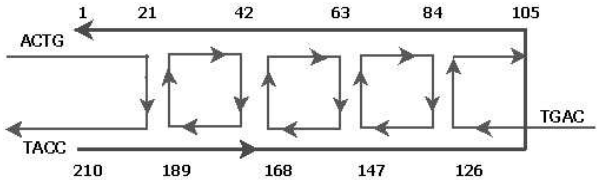


Fig. 6. DNA structure for a message binary "1"

In ordered to encrypt message "secret" which is composed of 42 bits we need 42 structures described above. For each structure were calculated necessary sequences. Terminations of marginal short sequences are different in case of binary "1" or "0" and if it is a start bit. For the first bit "1" of the message was calculated following sequences:

1. TAGTCAGTGTATCCATGGTATCTGTACGTTTGGTATCAGAAG CCCCCGAAACTAGGTTCCGCTTGCGACCGTTTAGATGTTTCAGA ACAGGGCTGTTCGCGTGATACGTCGTAAACCTAGATAGTCAAC CATGTCTAGTATTTGTCACGTCCTGTATGACCGTATTCCGGGTC CTGTCGGTAGGATTCTAGCCTACCCTTAAGCTTGTCG
2. ACTGATCAGTCACATAGGTACCATATCGGATGGGAATTCGAA CAGCTACC
3. GACATGCAAACCATAGTCTTCCCCAGGACAGCCATCCTAAGA
4. GGGGCTTTGATCCAAGGCGAGCAGGACATACTGGCATAAGG
5. ACGCTGGCAAATCTACAAAGTTTGGTACAGATCATAAACAGT
6. TGCAGCATTTGGATCTATCAGCTTGTCCCGACAAGCGCACTA

For the next bit also "1" the last sequence is different. Sequence 6 starts with additional "TGAC" in order to bind to the first bit with termination "ACTG". First bit does not have such termination "TGAC" in order to restrict binding in one single direction. In case of binary "0" in sequence 2 instead of "TACC" appear "GTCA". These terminations make a difference between binary "1" and "0".

IV. ENCRYPTION ALGORITHM

From previous computations were obtained files with sequences for all the DNA structures involved in encryption process. These files can be used for synthesizing of structures in solution, in a laboratory. First step is synthesizing all necessary units and the second step is performing encryption algorithm. In the next two subsections are described laboratory utilities for implementation and steps of the encryption algorithm.

1. Laboratory utilities

The experiment of data encryption using biomolecular computation can be realized using the atomic force microscope (AFM) named also scanning force microscope (SFM). This tool is used for imaging, measuring and manipulating nanoscale matters (Fig. 7).

AFM provides highly accurate images of DNA at nanometer scale resolution. DNA samples are placed on a flat, smooth substrate, immobilized and then scanned by AFM. Scanning process results in topography of the surface.

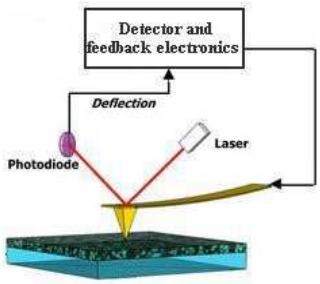


Fig. 7. Schematic view of AFM

AFM have been further developed into operative tool. Therefore manipulation of distinct atoms and molecules becomes possible. Isolation of biological material was performed by AFM tip [14]. Single DNA molecule can be cut, pushed and folded in desired shape using AFM [15]. Therefore AFM is an appropriate technique for this study.

2. Encryption algorithm

Binary data can be encoded using a single DNA molecule for each bit as described before. Difference between "0" and "1" is given by the group of nucleotides at the sticky ends (Fig. 6). Bit-wise XOR operation is performed using DNA hybridization process and by assembling the plaintext binary message, the one-time-pad and the resulted encrypted message in strings (Fig. 8). The string with the result is the only one that remains at the end; other strings (plaintext and OTP) are removed [16].

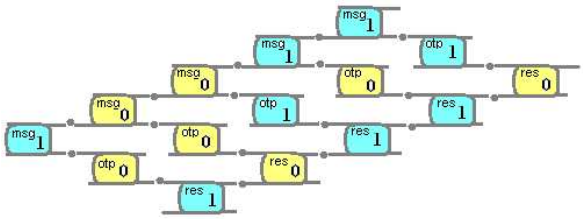


Fig. 8. Binding process of DNA Molecules

The first step of the algorithm is encoding the information we want to encrypt in a string of DNA molecules. For this purpose can be used two stacks with DNA molecules designed to represent the binary "1" and "0". These stacks can be operated by a microcontroller which acknowledges the beginning of encoding, makes a decision concerning the next bit, and verifies when the encoding stops.

The next step of the algorithm is an XOR operation between the DNA molecules of the encoded information and the OTP key. The OTP in this case is a great set of labeled DNA molecules encoded as binary "1" and "0". A part of these molecules will randomly bind to the encoded information. The randomness is ensured by the process of hybridization. Those molecules from OTP which hybridize with the plaintext information become the encryption key.

The ciphertext is composed of the DNA molecules which bind to the other side of the OTP as the result of XOR operation (Fig. 8). The ciphertext DNA molecules can be separated from the rest of the DNA material by using restriction enzymes and sent to the destination with the string of labels for the key.

Decryption is made using the same key and operations. The receiver has an identical pad and will use the received string of labels in order to select from this pad the encryption key. Performing the XOR operation between the ciphertext and the key will result in plaintext.

V. Complexity of the Algorithm

In this section is analyzed the complexity of the algorithm in terms of execution time and resources. In the absence of a standard for the implementation of the encryption algorithms in biomolecular environment, the process is slow and involves complex resources.

Hybridization time depends on many factors like the amount of sequences, their length and techniques used for the implementation. It can vary from several minutes to a few hours. At this moment the execution time is not the strong point of the algorithm, but with the standardization of the whole process it might not be an issue in the future.

Necessary resources for the implementation of this algorithm are a specialized laboratory were the biomolecular data is processed and stored. A useful tool is AFM which facilitates imaging, measuring and manipulating nanoscale matters.

The strength of this algorithm is the level of security that it offers. The first aspect is biomolecular environment which is more difficult to access. Due to the dimension of the biomolecular data, it can be easily hidden using steganographic techniques. Also an attacker would need an expensive laboratory in order to read the message. Another strength point of this algorithm is the key which is as long as the plaintext and is a truly random number generated through a natural process of hybridization. Therefore the algorithm satisfies the conditions of an OTP which offers a strong security. The self-assembly is a strong aspect of DNA nanotechnology which brings the parallel computation and automation of the implementation process.

VI. Conclusions

In this paper a symmetric encryption method was presented, based on biomolecular computation, one- time-pad principle, and special designed structure of DNA molecule. OTP ensures a perfect encryption, but at the same time requires a truly random number for the key which can be achieved by physical phenomena like DNA hybridization, in this case. Implementation of this encryption method contains 2 parts: a software program and a laboratory experiment. The main contributions of this work are: development of DNA molecule structure; a software program that gives files with DNA sequences necessary for creation of the DNA structures; and description of the useful tool (AFM) for the laboratory experiment. Laboratory experiment is intended to be a future work of this study.

Acknowledgment

This paper was supported by the project "Improvement of the doctoral studies quality in engineering science for development of the knowledge based society-QDOC" contract no. POSDRU/107/1.5/S/78534, project co-funded by the European Social Fund through the Sectorial Operational Program Human Resources 2007-2013.

References

[1] G. S. Vernam, "Cipher Printing Telegraph Systems", Journal of the American Institute of Electrical Engineers, Vol. XI.V, pp. 109-115, 1926.
[2] C. E. Shannon, "Communication Theory of Secrecy Systems", Bell System Technical Journal, Vol. 28, No. 4, pp. 656-715, 1949.
[3] B. Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C", John Wiley & Sons Inc., 1996.
[4] P. W. K. Rothemund, "Folding DNA to create nanoscale shapes and patterns", Nature, Vol. 440, pp. 297-302, 2006.
[5] D. P. Clark, N. J. Pazdernik, "Biotechnology: Applying the Genetic Revolution", Elsevier, 2009.
[6] A. Gehani, T. LaBean, J. Reif, "DNA-Based Cryptography", Springer, Vol. 2950, pp 167-188, 2004.
[7] L. M. Adleman, "Molecular computation of solution to combinatorial problems", Science, Vol. 266, pp. 1021-1024, 1994.
[8] N.C. Seeman, "Nucleic Acid Junctions: Building Blocks for Genetic Engineering in Three Dimensions", Adenine Press, pp. 269-277, 1981.
[9] T.J. Fu, N.C. Seeman, "DNA double crossover molecules", Biochemistry, Vol. 32, pp. 3211–3220, 1993.
[10] N.C. Seeman, H. Wang, X. Yang, and J. Chen, "New Motifs in DNA Nanotechnology", Nanotechnology, Vol. 9, No 3, pp. 269-277, 1998.
[11] W.M. Shih, J. D. Quispe, G.F.A Joyce, "1.7-kilobase single-stranded DNA that folds into a nanoscale octahedron", Nature, Vol. 427, pp. 618-621, 2004.
[12] www.genelink.com
[13] M. Levitt, "How many base-pairs per turn does DNA have in solution and in chromatin? Some theoretical calculations", Biochemistry, Vol. 75, No. 2, pp. 640-644, 1998.
[14] J. Lu, H. Li, H. An, G. Wang, Y. Wang, M. Li, Y. Zhang, and J. Hu, "Positioning Isolation and Biochemical Analysis of Single DNA Molecules Based on Nanomanipulation and Single-Molecule PCR", American Chemical Society, Vol. 126, No. 36, pp. 11136-11137, 2004.
[15] L. Jun-Hong, L. Hai-Kuo, A. Hong-Jie, W. Guo-Hua, W. Ying, L. Min-Qian, Z. Yi, L. Bin, H. Jun, "Nano-manipulation of single DNA molecules based on atomic force microscopy", Conf. Proc. IEEE Eng. Med. Biol. Soc., Vol. 7, pp. 7478-81, 2005.
[16] O. Tornea and M. E. Borda, "DNA Cryptographic Algorithms", IFMBE Proc., Vol. 26, pp. 223-226, 2009.

Multimedia Data Compression and Encryption using DNA Cipher

Olga TORNEA, Marc ANTONINI, Monica BORDA

Abstract

In this paper are presented different methods of combining compression with encryption. Their efficiency and purpose are also discussed. A DNA cipher was chosen to perform complete encryption. Experimental results were analyzed and 2 optimizations were proposed to obtain higher compression ratio and stronger security of the data.

1 Introduction

Compression and protection of the multimedia data have been always topics of interest because, as technology advances, the amount of private data that is desired to be transmitted, stored, or used in real time applications increases.

The requirements for information transmission have always been to fit a limited bandwidth, to achieve a high bitrate, and to fit a small storage space. Compression is the technique that achieves these requirements; it reduces the data volume as much as possible while maintaining an adequate level of its quality.

The security issues are to ensure confidentiality, data integrity and authentication. The high level of security is ensured by full data encryption, with significant amount of computations, and the size of the encrypted data can be larger than the size of the original data. Additional computations reduce significantly the speed of real-time applications. Growth in the data volume creates problems with the storage space.

Application of the security techniques can significantly reduce the benefits of compression results. Therefore those two processes need to be combined carefully. In this paper will be presented the order in which to apply them, how they can be combined in one single process and the experimental results of direct encryption with DNA cipher.

In section 2 are discussed different ways of applying compression and encryption. The order in which these two processes are used is crucial for both compression and security efficiency. The algorithm that was chosen for the image encryption is described in section 3.1. In section 3.2 are described results of applying this algorithm in a direct encryption. In section 3.3 are proposed 2 optimizations of the algorithm: one for compression ratio and another one for security.

2 Classification of the Multimedia Compression – Encryption methods

In most cases the information transmitted through the network need to be both encrypted and compressed. The goals of these two processes are different. The compression objectives are to obtain high compression ratio at desired quality of the data and at low computational complexity. The goal of cryptography is to ensure a high security level which usually means additional computations and sometimes the increase in data volume. Both processes reduce the redundancy of the data. Thus these tow processes need to be applied in such order that one of them will not cancel the advantages obtained from another.

During the encryption data passes through a series of transpositions and substitutions. If the security of the algorithm is strong, the redundancy of the plaintext will not be transferred on the ciphertext. If the redundancy of the data is high, such as in image, audio, or video files then there will be a high probability that the encrypted files will keep a part of the pattern from the plaintext. This is one of the reasons why the compression of the multimedia data is applied first and then the encryption. This model was proposed in a strong, hybrid cryptosystem named Pretty Good Privacy (PGP) [1].

Encryption process is never applied before compression because of the practical issues. The encryption process randomizes the original data trying to achieve an equal probability of data appearance; therefore, there will not remain information that can be compressed.

The classical way is to perform compression of the data and then to perform encryption of the whole bitstream. This process is named *complete* or *direct encryption* (Fig. 1 a); it is time and space consuming and therefore sometimes it may not be suitable for the real time applications. Full encryption is used when a high level of security is required and mostly for storage. Multimedia data is usually involved in the real-time interactions where the transmission must be fast, and it has already large volume without encryption, which can increase the size of the data. In order to solve this security issue the *partial* or *selective encryption* (Fig. 1 b) was proposed in [2, 3]. The idea of selective encryption (SE) is to encrypt only a part of the compressed data. In this way the volume of the data will be reduced and the speed of transmission increased.

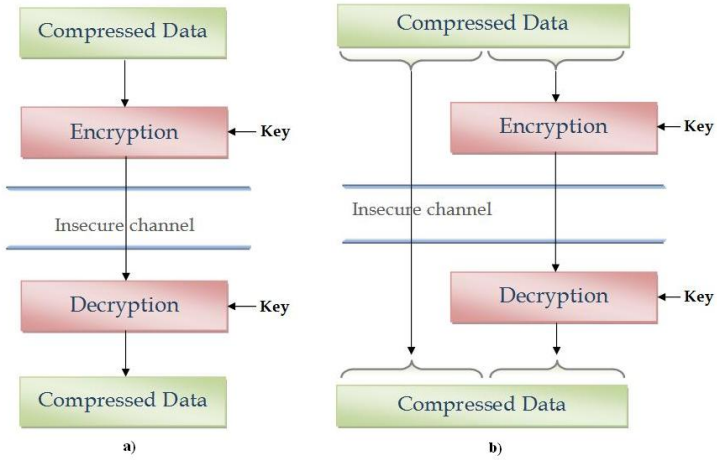


Figure 1: a) Complete Encryption b) Selective Encryption

Another method to avoid the computation and storage cost of encryption is to integrate it inside the entropy coding [4]. This method is a compression-combined encryption. In this case security is integrated inside the process of compression. The basic idea of the entropy coding encryption is to convert classical entropy coders into encryption ciphers [5], [6]. The principle is to replace a single statistical model by multiple statistical models randomly chosen from the pre-stored models.

In this paper a DNA cipher was used to encrypt the compressed data using direct encryption. Considering the experimental results of this operation, an improvement of direct encryption was proposed and implemented.

3 Experimental Results

3.1 DNA Cipher

DNA cipher used in this paper is a symmetric key encryption algorithm [7]. It uses genomic databases to retrieve the secret key in the form of a chromosomal sequence. This sequence is represented in a 4-letter alphabet (A, C, G, T) and it can be downloaded from any genomic database like: GenBank, DDBJ, etc. Each DNA sequence from the database has its unique identification number composed of 6 – 8 characters [8]. In a symmetric cryptosystem encryption and decryption are performed with the same key. Receiver needs to know the identification number of the sequence used as the key in order to find it in the established database.

A. Encryption process

DNA Indexing is a stream cipher where information is processed one byte at a time. The principle is to transform one plaintext byte into a sequence of 4 DNA letters. The next step is to search this short sequence through the chromosomal sequence, which was chosen as the key for the encryption. Each time a plaintext byte sequence is retrieved in the chromosomal sequence, the position of this place is memorized in a vector as one of the possible values for encryption by substitution for this byte. Vectors of substitutions for all the bytes are memorized in a key dictionary. Therefore for each byte from the plaintext there is a range of possible values from which one is chosen randomly for encryption by substitution. In order to obtain a substantial number of substitutions for each byte, the key-sequence needs to be sufficiently long, for example 30 000 bases. Below are presented steps of encryption and an example of encryption in Fig. 2.

1. Key dictionary computation:
   a) Each byte of 256 possible values is transformed in a sequence of 4 letters by the following principle: 10 00 11 01 (141) → GATC.
   b) For all the bytes is performed a search through the key, a long chromosomal sequence composed from letters: A, C, G, and T.
   c) Each time the byte sequence is retrieved in the key sequence, the index of that position is memorized in a vector dedicated for that byte.
   d) The result of these operations is a key table of size 256xN, where N is a variable length because each byte can have a different number of corresponding values in this table.
2. Encryption is performed one byte at a time. It consists in substitution of the byte with a value randomly retrieved from its vector in the key table.
3. The final ciphertext is an array composed of the integer values.

B. Decryption process

Decryption is performed with the same key – sequence. Each number from the ciphertext is used as a pointer to the DNA sequence, indicating where to read the plaintext byte sequence.

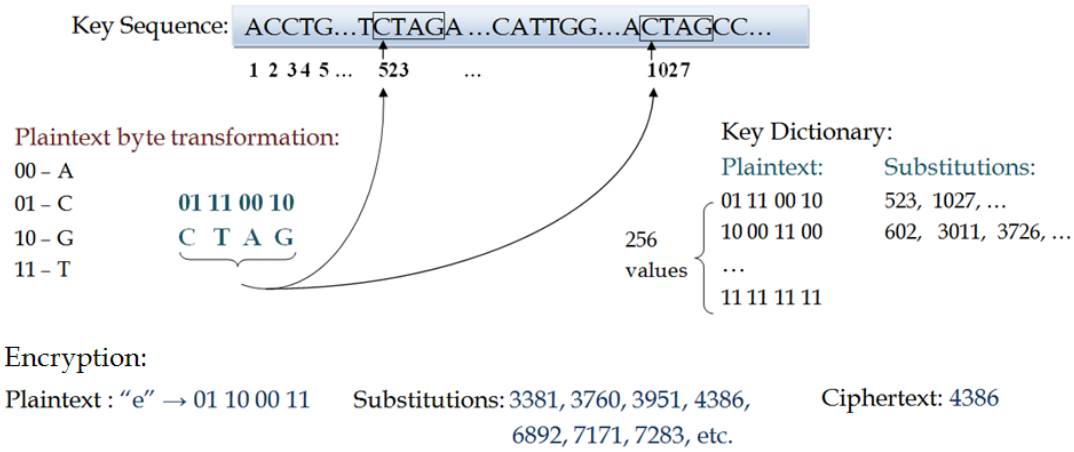


Figure 2: Encryption process of the DNA Cipher

3.2 Direct Encryption

Complete encryption of the image data was performed on the encoded bitstream, using the scheme described in section 2 and shown in Fig. 1, a. When compression and encryption need to be applied on the multimedia data, encryption is performed as a second process because it changes the probability distribution of the data samples, making their appearance as equal and uniform as possible. If all of the data samples are equally probable then there is no information that can be compressed by entropy encoding. Thus, before encryption, the first step was to obtain the compressed bitstream of the image.

Encryption was applied on the bit array of the compressed plaintext and performed by transforming each byte of the plaintext into an integer value. The output of the encryption process, the ciphertext, is an array of integer values, as was exemplified in Fig. 2.

The bitstream of the ciphertext can be represented in two forms: using a fixed number of bits for each integer value, or using the exact number of bits for each codeword and then memorizing their flags which specify the number of bits of each word. For the first method, where is used a fixed number of bits for each codeword, the number of bits assigned to a codeword is given by the maximal value of a codeword. Thus, if the length of the secret key (DNA sequence) is, for example, 37839; then the maximal possible value of a codeword is that value, so the number of bits needed to represent a value from the ciphertext is 16. Considering the fact that each plaintext byte is transformed in an integer number of 16 bits, the ciphertext bitstream will be twice longer than the plaintext bitstream.

From the point of view of compression efficiency the increase in data size after encryption is of major interest. The value of a codeword from the ciphertext can vary between one and the

length of the key and its number of bits accordingly. In this work was used a DNA sequence of length 37839, so the number of bits of a codeword can be in the range: 1 – 16. Running the encryption algorithm few times for different image files and computing the average number of bits for a ciphertext codeword gave that it is around 14 bits. The most probable number of bits for a codeword is 15. Here are some probabilities of codeword lengths: $P(15) \approx 0.45$, $P(14) \approx 0.2$, $P(16) \approx 0.1$. Considering all the lengths of the codewords the average number was found to be 14 bits. This means that 8 bits of the plaintext are transformed, approximately, in 14 bits of the ciphertext. The increase in size in this case is of 1.75 times. Considering the probabilities of codeword lengths, a fixed number, of 16 bits, is more suitable for representation of the integer values.

3.2 Improvement of Direct Encryption

Number of bits for a ciphertext codeword can be optimized. The dictionary of substitution values is computed before actual encryption. In this key-table all the bytes (256) have a vector of corresponding integer values. This integer values are given by the positions where the byte value sequence (like "ACTT") is equal to the same sequence in the key. This integer values are used as ciphertext words for the substitution of the byte from the plaintext.

As transmitter and receiver must compute the same dictionary for the key, then the position of each integer value inside the key table is known. Thus, there is no need to transmit large integer values of 16 bits. Their position number can be sent instead. This will result in sending fewer bits for each ciphertext word (Fig. 3).

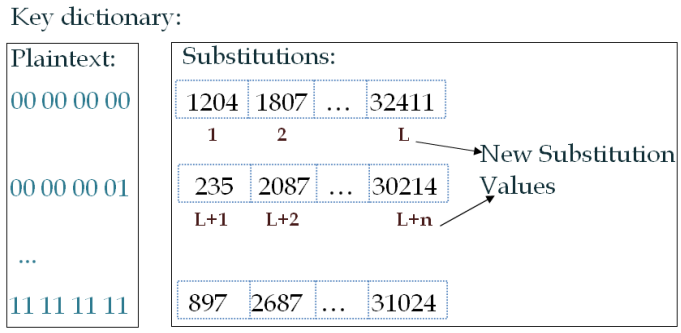


Figure 3: Substitution of the plaintext bytes by positions of the codewords in the key dictionary

In this case the length of a codeword will depend on the total number of substitutions in the dictionary. Limiting the number of substitutions will lead to a better compression ratio. From previous section, where direct encryption was described, resulted that one codeword has the size of 16 bits. Considering that 32 different substitution values for 1 byte are enough, the total number of substitution values in the dictionary will be 8192 (256*32). This means that new substitution values (Fig. 3) can be represented on 13 bits each ($8192 = 2^{13}$). Making a tradeoff between security and compression, each codeword can be represented in less number of bits by reducing the total number of substitutions in the dictionary. For example, with a 2048 (2^{11}) number of substitutions in the dictionary, which is 8 different substitutions per 1 byte, the codeword length in bits is 11.

Another improvement proposed in this paper is to attribute a different number of substitution values for each byte. Distribution of the compressed data bitstream is usually uniform, but there still can be some of the bytes appearing more often than the others (Fig. 4). This means that for a more uniform ciphertext, number of substitutions attributed to a byte must vary according to its appearance in the plaintext.

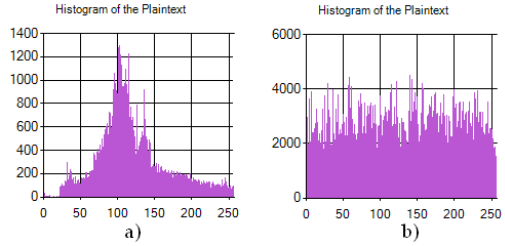


Figure 4: Distribution of the compressed data bitstream a) tiff, b) jpeg

Attribution of varying number of substitution values can be very useful, like for a tiff image in Fig. 4 a. At the beginning the number of indexes for substitution of each byte was random; it depended on how often a certain byte sequence was retrieved in the key sequence (Fig. 5 a). Now for each byte there is a certain number of corresponding indexes for substitution; this number corresponds to the appearance probability of that byte in the data. A vary probable value in the compressed data bitstream has more corresponding substitution values than a less probable value (Fig. 5 b). Number of substitutions for each byte in the dictionary is equal to the product of total number of substitution in the dictionary and the probability of each byte. This is the principle of homophonic substitution and the result of its application on the dictionary is a more uniform distribution of the ciphertext (Fig. 6).

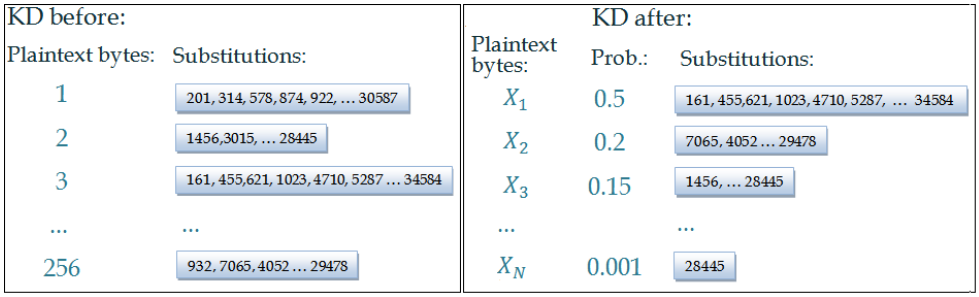


Figure 5: Number of substitution values for each byte of the plaintext

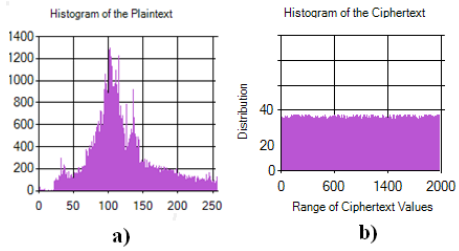


Fig. 6: Distribution of the a) plaintext and b) ciphertext values

4 Conclusions

This paper presents existing methods for application of compression along with security on multimedia data. The most optimal options were highlighted in section 2. It was pointed out that there can be complete or selective encryption and those two processes can be combined in one by different techniques.

For a practical application was chosen a DNA cipher and a compete encryption of compressed image data bitstream was performed with it. It has been found that the size of the data increases significantly (twice) after encryption. An optimization was proposed to reduce the size of each codeword and thus of the whole ciphertext. A tradeoff between security and compression can be established by the users due to this optimization. More substitution values in the dictionary lead to a higher security and less number of substitutions to a higher compression ratio.

In order to improve security, the homophonic substitution principle was introduced in computation of substitution values. Number of substitution values in the dictionary was computed to be equivalent to the appearance probability of the value to be substituted in order to obtain a more uniform distribution of the ciphertext.

References

[1] PGP Corporation, Zimmermann P., "An Introduction to Cryptography", 2004.

[2] Cheng H., Li X., "Partial encryption of compressed images and videos", *IEEE Transactions on Signal Processing*, Vol. 48(8), 2000, pp. 2439–2451.

[3] Lian S., Liu Z., Ren Z., and Wang Z., "Selective video encryption based on advanced video coding", In *Proceedings of 2005 Pacific-Rim Conference on Multimedia (PCM2005)*, Part II, Lecture Notes in Computer Science, Vol. 3768, 2005, pp. 281–290.

[4] Salama P., King B., "Efficient secure image transmission: compression integrated with encryption", *Proc. SPIE*, Vol. 5681, 2005, pp. 47-58.

[5] Wu C., Kuo C.J., "Efficient multimedia encryption via entropy codec design", *SPIE international symposium on electronic imaging*, Vol. 4314, 2001, pp. 128-138.

[6] Grangetto M., Magli E., Olmo G., "Multimedia selective encryption by means of randomized arithmetic coding", *IEEE Transactions on Multimedia*, Vol. 8, pp. 905-917, 2006.

[7] Tornea O., Borda M.E., Hodorogea T., Vaida M., "Encryption system with Indexing DNA chromosomes cryptographic algorithm", *IASTED International Conference*, Vol. 680-099, 2010, pp. 12-15.

[8] http://www.dsimb.inserm.fr/~fuchs/M2BI/AnalSeq/Annexes/Sequences/Accession_Numbers.htm

Acknowledgment

This paper was supported by the project "Improvement of the doctoral studies quality in engineering science for development of the knowledge based society-QDOC" contract no. POSDRU/107/1.5/S/78534, project co-funded by the European Social Fund through the Sectorial Operational Program Human Resources 2007-2013.

Biography

Olga Tornea graduated Technical University of Cluj-Napoca (TUCN) in 2009 and obtained Bachelor's degree in Telecommunications. In 2012 she obtained Master's degree in Image and Signal Processing from (TUCN) and University of Nice Sophia-Antipolis France (UNS). Currently she is a PhD student jointly at TUCN and UNS. Her interests are in cryptography, genetic databases and compression.

Olga TORNEA, PhD Student
Technical University of Cluj-Napoca
Communications Department
Str. Dorobantilor Nr. 71-73, 400609, Cluj-Napoca, ROMANIA
E-mail: Olga.Tornea@com.utcluj.ro

New DNA Based Random Sequence Generation and OTP Encryption Systems for Transmission and Storage

Monica E. Borda, Olga Tornea, Romulus Terebes, Raul Malutan
Communications Department
Technical University of Cluj-Napoca
26 – 28 Gh. Baritiu St., 400027, Cluj-Napoca, Romania
Monica.Borda@com.utcluj.ro, Olga.Tornea@com.utcluj.ro,
Romulus.Terebes@com.utcluj.ro, Raul.Malutan@com.utcluj.ro
http://ares.utcluj.ro

Abstract: The first part of the paper is dedicated to a new method for random sequence generation, based on DNA structures. The length of the sequence is as long wanted, intended to be used for one time pad (OTP) cryptosystems. Four different ways are presented to ensure the desired lengths. The second and third parts are dealing with OTP cryptosystems for duplex transmissions, respectively storage, along with the corresponding protocols. The advantages of the proposed method and OTP systems are ending the paper.

Key-Words: random sequence generation; genomic databases; one-time-pad cryptosystems.

1 Introduction

Pure random sequences are widely used in cryptographic applications for cryptographic keys [1], [2]. The difficulty is to generate such pure random sequences and also a great problem is the management of these keys (to be transmitted and stored securely). Of great interest in cryptography is to use OTP, meaning to use only once a key in a confidential communication and the length of the key to be at least as long as the message in clear. Such a system was proved to be unbreakable [1], [3]. The main problems which occur in OTP implementation are: the great number of required very long keys and their management (transmission on trusty channels).
The randomness of DNA sequences, proved by the fact that they practically cannot be compressed [4], [5], [6], [7], [8], can be used to generate as long desired random binary sequences. Such a random sequence generator is presented in Section 2; four different ways of obtaining the desired length are also proposed and also the format of the secret key which need to be shared between the users.
Section 3 is presenting the block-scheme and the communication protocol of an OTP duplex cryptosystem based on DNA random sequences.
In section 4 is presented the block-scheme and the corresponding protocol of an OTP cryptosystem used for storage purposes.
The last, 5th section, is dedicated to illustrate the advantages of the proposed DNA based method for random sequence generation, respectively OTP cryptosystems for transmission and storage.

2 Method for DNA Based Random Sequence Generation

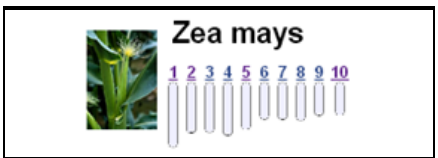
Random sequences based on DNA sequences (chromosomes, genes, etc.) can be provided by biological databases, the

greatest being [9], [10], [11]. The structure of a DNA sequence belonging to a living organism (human, animal, plant) is random (cannot be compressed, or the compression ratio is extremely small) [4], [5], [6], [7], [8].
The genetic code has four bases: A – adenine, C – cytosine, G – guanine, T – thymine [12] which can be converted into binary, using a uniform encoding:

A	00
C	01
G	10
T	11

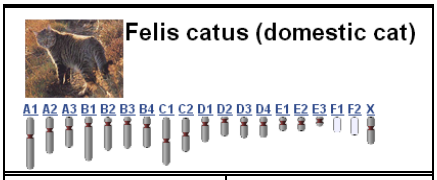
Table 1. Conversion table from DNA to binary

These substitutions can be easily realized using switch-case selection of a programming language. The lengths of DNA sequences in genetic databases are variable: from tens (a gene) to hundreds of millions (a chromosome) bases. Tables 2, 3, and 4 are giving lengths of DNA sequences (in base pairs - bp) for some living organisms.



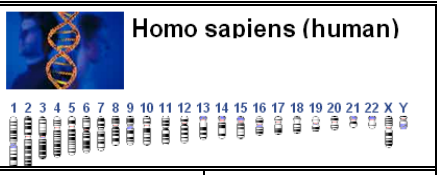
Chromosome	Length in bp
1	301,354,135
2	237,068,873
3	232,140,174
4	241,473,504
5	217,872,852
6	169,174,353
7	176,764,762
8	175,793,759
9	156,750,706
10	150,189,435

Table 2. DNA sequences lengths corresponding to Zea mays (corn) chromosomes



Chromosome	Length in bp
A1	239,302,903
A2	169,043,629
A3	142,459,683
B1	205,241,052
B2	154,261,789
B3	148,491,654
B4	144,259,557
C1	221,441,202
C2	157,659,299
D1	116,869,131
D2	89,822,065
D3	95,741,729
D4	96,020,406
E1	63,002,102
E2	64,039,838
E3	43,024,555
F1	68,669,167
F2	82,763,536
X	126,427,096

Table 3. DNA sequences lengths corresponding to cat chromosomes



Chromosome	Length in bp
1	249,250,621
2	243,199,373
3	198,022,430
4	191,154,276
5	180,915,260
6	171,115,067
7	159,138,663
8	146,364,022
9	141,213,431
10	135,534,747

11	135,006,516
12	133,851,895
13	115,169,878
14	107,349,540
15	102,531,392
16	90,354,753
17	81,195,210
18	78,077,248
19	59,128,983
20	63,025,520
21	48,129,895
22	51,304,566
X	155,270,560
Y	59,373,566

Table 4. DNA sequences lengths corresponding to human chromosomes

In order to obtain random sequences (RS) for cryptographic OTP applications, the sequence need to have the length at least equal with that of the cleartext message and to be used only once. For this reason, the length of the generated sequence needs to match the message length, which is defined by:

- Message type: - text
  - image
  - sound (bitrate)
  - video (bitrate)
- Transmission time for sound and video

These input data will determine the required length of the random sequence used as the key. Table 5 is illustrating some examples of these input data.

In genetic databases there are DNA sequences of very great length (chromosomes), acceptable for many applications. Table 6 exemplifies different cleartext messages, the length of the corresponding required DNA key (in bp), and the dimension of the corresponding secret key information.

Text	Image	Sound	Video
15KB	402KB	5.34MB/278s (160kbps)	113 MB/1140s (audio 111 kbps, video 704 kbps)
639KB	573KB	4.74MB/248s (160kbps)	329MB/3359s (audio 112 kbps, video 695 kbps)
1.14MB	1.31MB	8.48MB/222s (320kbps)	349MB/2640s (audio 153 kbps, video 934 kbps)
207MB	3.44MB	3.16MB/195s (128kbps)	699MB/5708s (audio 99 kbps, video 909 kbps)

Table 5. Examples of different input data (obtained by measuring real files)

Cleartext message	DNA sequence (key) length	Secret key information
Text - 15KB	61,440 bp	1 ID of DNA sequence (8 bytes)
Image - 402KB	1,646,592 bp	1 ID of DNA sequence (8 bytes)
Sound (160kbps) 5.34MB/278s	22,397,583 bp	1 ID of DNA sequence (8 bytes)
Video - 329MB/3359s (audio 112 kbps, video 695 kbps)	1,384,120,320 bp	~ 6 IDs of DNA sequences (48 bytes)

Table 6. Example of different cleartext messages, the corresponding DNA key length (bp), and respectively the secret key information

In order to obtain a high number of DNA sequences of great lengths, there are more possibilities:

a) one chromosome taken from a given database (see Tables 2 - 4)
b) multiplexing, cycling (shifting) and concatenation of more sequences obtained from the same chromosome (Fig. 1)
c) multiplexing sequences from different chromosomes belonging to the same species (Fig. 2)
d) multiplexing sequences from different species (Fig. 3)

The header (secret key) used for transmitting the data necessary to generate the key (need to be transmitted confidentially) is given in Fig. 4.

Example:

For a cleartext message corresponding to a video file of 329 MB (Table 6), a 1,384,120,320 bp DNA sequence is required; it can be obtained multiplexing 5 – 6 different DNA sequences. Consequently, the secret key will transmit the corresponding IDs. Fig. 5 is illustrating the sequences in GenBank format and their IDs. Fig. 6 is illustrating the secret key format.

The last step in the generation of the binary RS is to convert the DNA-RS in binary coding (Table 1).

Original sequence: AATAGCACAATAA TCACATTCTTG GCTTCTACTCATCT

Modified sequence: GCTTCTACTCATCT AATAGCACAATAA TCACATTCTTG

Figure 1. Modification of the original DNA sequence

Zea mays Cr. 4: AAGCTTCTACTCATCTCCCGGCAAACAGATAT...
Zea mays Cr. 7: GGAATAGCACAATAAGTGCGCAAAATCGAAG...
Zea mays Cr. 9: GATCACATTCTTGGATTTTTGGTGGAGACCAT...
MUX(Zea mays{Cr. 4, Cr. 7, Cr.9}) = AGGAGAGATCACTTATAC...

Figure 2. Multiplexing DNA sequences from different chromosomes of the same species

Homo Sapiens Cr. 5 → l_1 = 180,915,260 bp = 361,830,520 bits

Zea mays Cr. 8 → l_2 = 175,793,759 bp = 351,587,518 bits

Felis catus Cr. C1 → l_3 = 221,441,202 bp = 442,882,404 bits

→ $l_{mux} = 3*l_{min}(l_1, l_2, l_3)$ = 1,054,762,554 bits

Figure 3. Multiplexing DNA sequences of different species, obtained key can be used to encrypt 125.7 MB of the data

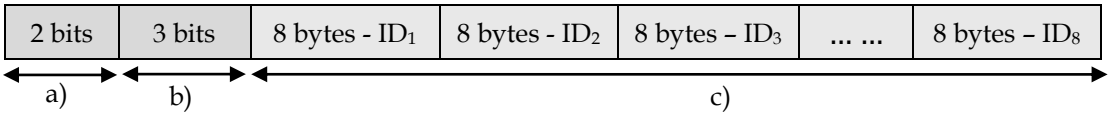
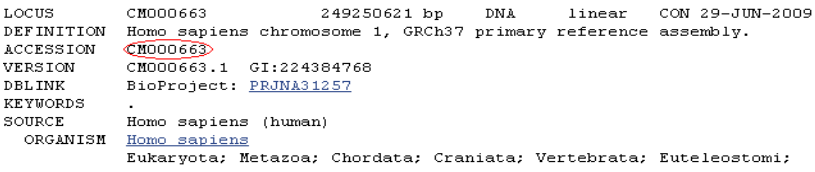
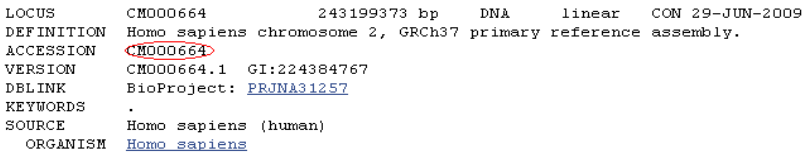


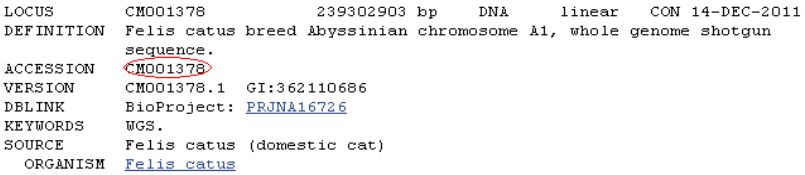
Figure 4. Format of the secret key (header which contains the initial data for DNA - RS) a) 2 bits specifing the method a – 00, b – 01, c – 10, d – 11, b) 3 bits specifing the number of transmitted IDs, c) the actual secret key formed of 1 – 8 IDs, each of 8 bytes



a) ID_1 – CM000663



b) ID_2 – CM000664



c) ID_3 – CM001378

Figure 5. DNA sequences in GenBank format and their IDs

10	101	ID_1 CM000663	ID_2 CM000664	ID_3 CM001378	... ...

Figure 6. Example of secret key format

The presented method used for DNA-RS generation has the following advantages:

- generation of binary random sequences of any length using DNA structures from public or private databases
- the number of distinct random sequences is practically unlimited, taken into consideration the versatility of the possible ways (a ÷ d)
- the key is not necessary to be transmitted entirely, only the secret key (Fig. 4) of 9 - 65 bytes length requiring confidentiality in transmission and storage

3 OTP Cryptosystem for Duplex Transmission Based on DNA-RS

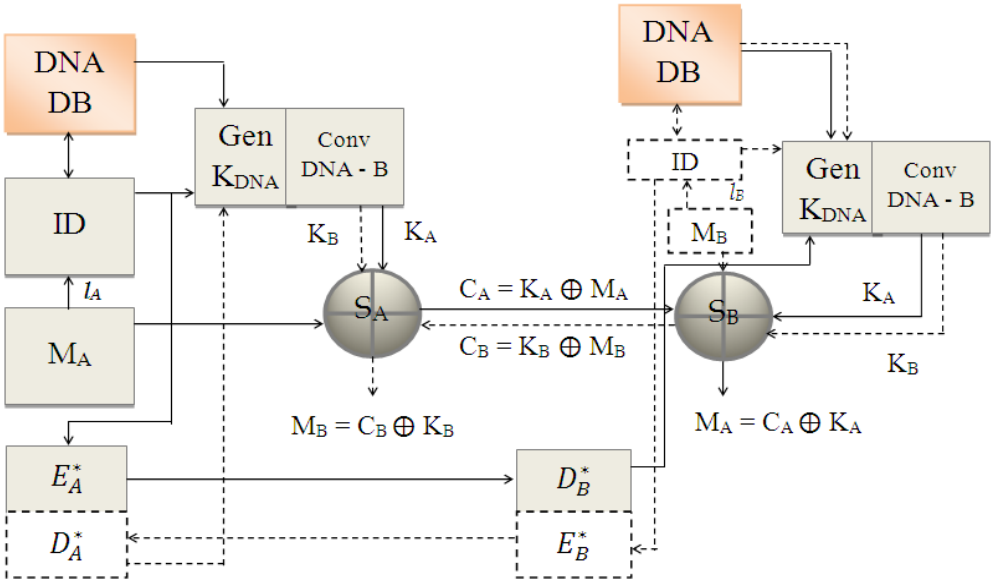


Figure 7. Scheme of OTP cryptosystem for duplex transmission based on DNA-RS key

Legend of Figure 7:
DNA DB – a database of DNA sequences (identical for 2 parts: A and B), which can be public or private
ID – block of input data containing: required DNA sequence length according to the size of the input data; IDs of the sequences to be used at the key generation; method of key computation
M_A – cleartext message of user A
M_B – cleartext message of user B
Gen K_{DNA} – block of DNA sequence generation, used as the key (K_{DNA})
Conv DNA-B – transformation of DNA sequence in binary (K_A, K_B)
$E^*_{A,B}$ – block of input data (ID) encryption, using a symmetric or public algorithm for users A and B
$D^*_{A,B}$ – block of input data (ID) decryption, using the algorithm chosen at $E^*_{A,B}$
S_A – modulo-2 adder used by A for OTP encryption and decryption
S_B – modulo-2 adder used by B for OTP encryption and decryption

Protocol of using duplex OTP Cryptosystem:

I. Transmission A → B
a. A side encryption
(1) Providing the input data: length of the cleartext message, choosing DNA sequences and key generation method.
(2) DNA key generation using input data and genetic database: K_{DNA} of user A.
(3) Generation of OTP key (K_A) using conversion DNA → binary
(4) Encryption of input data using a symmetric or public algorithm (block E^*_A) and its transmission to part B
(5) OTP encryption of cleartext message M_A: $C_A = K_A \oplus M_A$ and transmission of the obtained cryptogram (C_A) to part B

b. B side decryption
(6) Decryption of input data (ID) realized in block D^*_B
(7) Generation of K_A key using input data obtained at (6), DNA database identical with

part A, and the same key generator (*Gen* K_{DNA} and *Conv DNA-B*)
(8) OTP decryption of C_A cryptogram using K_A key obtained at (7): $C_A \oplus K_A = M_A$

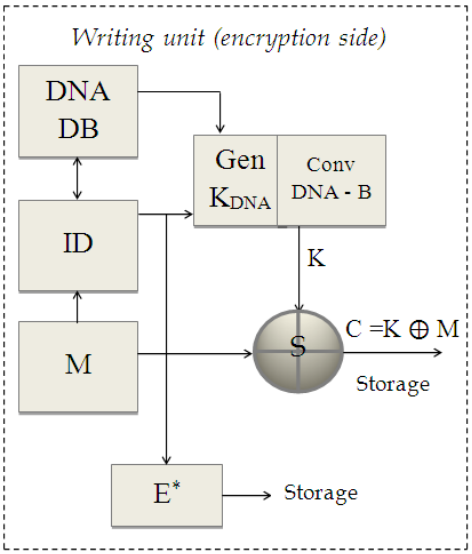
II. Transmission B → A
a. B side encryption
(1) Providing the input data: length of the cleartext message, choosing DNA sequences and key generation method.
(2) DNA key generation using input data and genetic database: K_{DNA} of user B.
(3) Generation of OTP key (K_B) using conversion DNA → binary
(4) Encryption of input data using a symmetric or public algorithm (block E^*_B) and its transmission to part A
(5) OTP encryption of cleartext message M_B: $C_B = K_B \oplus M_B$ and transmission of the obtained cryptogram (C_B) to part A

b. A side decryption
(6) Decryption of ID realized in block D^*_A
(7) Generation of K_B key using input data obtained at (6), DNA database identical with part B, and the same key generator (*Gen* K_{DNA} and *Conv DNA-B*)
(8) OTP decryption of C_B cryptogram using K_B key obtained at (7): $C_B \oplus K_B = M_B$

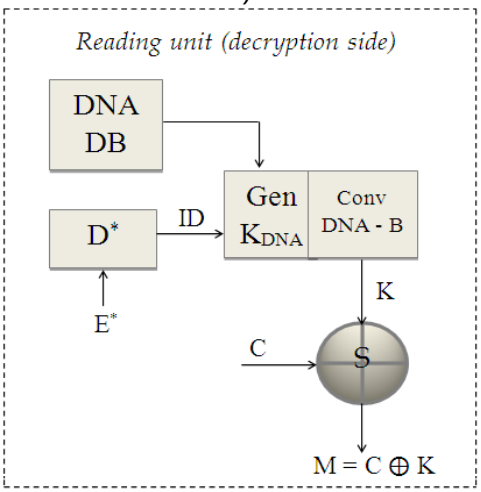
The OTP cryptosystem for duplex transmission based on DNA-RS key has the following advantages:

- The OTP system (a secret key used only once and having the length at least equal to that of the cleartext message) was shown to be unbreakable [3], [13]
- The OTP key doesn't have to be transmitted entirely, the recipient can easily generate the key using confidential transmission (E^*_A, D^*_B) of ID and by using the same DNA database, which provides an easy key management, thus removing the main drawback of symmetric cryptography: difficult key management, especially when the number of users grows.
- Security of the cryptosystem is given by the security of the algorithm used at input data encryption (E*, D*)

4 OTP Cryptosystem for storage, based on DNA-RS key



a)



b)

Figure 8. OTP cryptosystem for storage, based on DNA-RS key a) encryption side of writing unit b) decryption side of reading unit

Obs.: block notations used in Fig. 8 are identical to the notations used in Fig. 7.

Protocol of writing – reading using OTP cryptosystem based on DNA-RS key:
a. Encryption (writing unit)
(1) Providing the input data: length of the cleartext message, choosing DNA sequences and key generation method.
(2) DNA key generation using input data and genetic database: K_{DNA}.
(3) Generation of OTP key (K) using conversion DNA → binary
(4) Encryption of input data using a symmetric or public algorithm (block E^*) and depositing on the storage media
(5) OTP encryption of data M: $C = K \oplus M$ and depositing of the cryptogram (C) on the storage media (CD, DVD, etc.)

b. Decryption (reading unit)
(6) Decryption of input data (ID) realized in block D^*
(7) Generation of key (K) using input data obtained at (6), DNA database identical with the one used at writing unit, and the same key generator (*Gen* K_{DNA} and *Conv DNA-B*)
(8) OTP decryption of cryptogram (C) using key (K) obtained at (7): $C \oplus K = M$

OTP cryptosystem for storage, based on DNA-RS key has the following advantages:

- Storage of encryption keys is easier than in the classical system because it consists only of input data (ID), so it has a much shorter length than the encryption key (K)

5 Conclusion

A new method for random sequences generation based on DNA structures and two OTP systems for transmission, respectively for storage, along with their protocols were presented [14]. The main advantage of the method for random sequence generation is the possibility to generate practically an infinite number of such sequences starting from DNA structures and as long desired. If these sequences are used in cryptographic applications as keys, they require only a header of 9 – 65 bytes long which need to be confidentially transmitted and not the entire key, meaning that the key management becomes very easy.
The two OTP systems for transmission and storage have the advantage of the OTP: greatest security based on using a key once and the lengths of the key being at least equal with that of the cleartext message. The major drawback of OTP systems is the key management, but this is overcome by the advantage of the key generation described previously.

References:

[1] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, Inc, 1996.
[2] W. Stallings, *Cryptography and Network Security: Principles and Practice*, (5th Ed.), Prentice Hall, 2011.
[3] C.E. Shannon, Communication Theory of Secrecy Systems, *Bell System Technical Journal*, Vol. 28, No. 4, 1949, pp. 656-715.
[4] C.G. Nevill-Manning, I.H. Witten, Protein is incompressible, *In Proceedings of the Conference on Data Compression (DCC '99)*, 1999, pp. 257.
[5] S. Christley, Y. Lu, C. Li, X. Xie, Human genomes as email attachments, *Bioinformatics*, Vol. 25, 2009, pp. 274–275.
[6] M.H.Y. Fritz, R. Leinonen, G. Cochrane and E. Birney, Efficient storage of high throughput DNA sequencing data using reference-based compression, *Genome Res*, Vol. 21, 2011, pp. 734-740.
[7] K. Daily, P. Rigor, S. Christley, X. Xie, P. Baldi, Data structures and compression algorithms for high-throughput sequencing technologies, *BMC Bioinformatics*, Vol. 11, No. 514, 2010.
[8] P. Rajarajeswari, A. Apparao, DNABIT Compress – Genome compression algorithm, *Bioinformation*, Vol. 5, No. 8, 2011, pp. 350–360.
[9]http://www.ddbj.nig.ac.jp/Welcome-e.html
[10] http://www.ebi.ac.uk/ena/
[11]http://www.ncbi.nlm.nih.gov/pubmed/21071399
[12] C.R. Calladine, H.R. Drew, B.F. Luisi, A.A. Travers, *Understanding DNA The Molecule & How It Works*, Academic Press, 2004.
[13] G.S. Vernam, Cipher Printing Telegraph Systems, *Journal of the American Institute of Electrical Engineers*, Vol. 11, No. 5, 1926, pp. 109-115.
[14] M. Borda, O. Tornea, R. Terebes, R. Malutan, "Method and cryptographic OTP system based on DNA random sequences", best Patent request at ProInvent 2013, No. of patent application: A10003/14.02.2013.