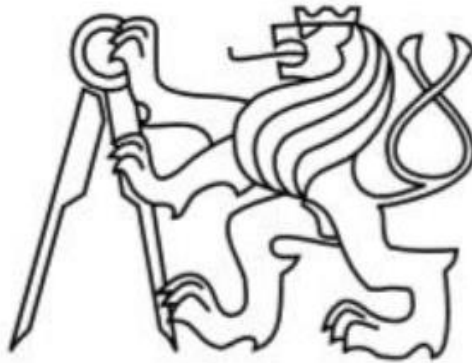


Czech Technical University in Prague  
Faculty of Electrical Engineering  
Department of Computer Science



Master's Thesis

**Data Security while Using Cloud Services**

Adelina Akhmedzianova

Supervisor: Ing. Václav Rechtberger

Study Program: Open Informatics  
Field of Study: Software Engineering

May 22, 2020





# MASTER'S THESIS ASSIGNMENT

## I. Personal and study details

Student's name: **Akhmedzianova Adelina** Personal ID number: **492134**  
Faculty / Institute: **Faculty of Electrical Engineering**  
Department / Institute: **Department of Computer Science**  
Study program: **Open Informatics**  
Specialisation: **Software Engineering**

## II. Master's thesis details

Master's thesis title in English:

**Data security while using cloud services**

Master's thesis title in Czech:

**Bezpečnost dat při využívání cloudových služeb**

Guidelines:

Design and implement a system to ensure the confidentiality of data in cloud storage. The system provides automatic integration with Yandex.Disk cloud using API interface, two-factor authentication based on one-time passwords, implementation of user-side encryption algorithms, and implementation of a method for generating encryption keys from media files.

Bibliography / sources:

[1] W.Stallings, "Cryptography and Network Security: Principles and Practice", 6th ed. Prentice Hall Press Upper Saddle River, New Jersey: 2013, pp. 68-89. [2] Omer A.Shqeer, "Judgment of Extracting Encryption Keys From Image Data", IJCSNS vol.14, no.2, pp.116-119. [3] S.Katzenbeisser, Fabien A.P.Petitcolas, "Information Hiding Techniques for Steganography"

Name and workplace of master's thesis supervisor:

**Ing. Václav Rechtberger, Department of Computer Science, FEE**

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **12.03.2020** Deadline for master's thesis submission: **22.05.2020**

Assignment valid until: **19.02.2022**

\_\_\_\_\_  
Ing. Václav Rechtberger  
Supervisor's signature

\_\_\_\_\_  
Head of department's signature

\_\_\_\_\_  
prof. Mgr. Petr Páta, Ph.D.  
Dean's signature

## III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

\_\_\_\_\_  
Date of assignment receipt

\_\_\_\_\_  
Student's signature



## **Acknowledgements**

I would like to express my sincere gratitude to all of my teachers, especially E.A. Turilova, and N.R. Boukharaev (Kazan Federal University, Kazan, Russia) for their patience and guidance in my study. I express my appreciation to Innocenter VAO LLC (Moscow, Russia) and especially to A.S. Budyakov for the idea and cooperation in the process of software development and testing. I especially thank my family for its support during the entire period of my study. Finally, many thanks to the Department of Computer Science at CVUT for creating such an enjoyable working environment.

## Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.

I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

Prague, May, 2020

---

## **Abstract**

AKHMEDZIANOVA, Adelina: Data security while using cloud services. [Master's Thesis] Czech Technical University in Prague. Faculty of Electrical Engineering, Department of Computer Science. Supervisor: Ing. Václav Rechtberger.

The main problem in the use of cloud storage services is data protection: cloud service providers do not provide encryption functions. The existing solutions, that realize a principle of client-side encryption, do not provide secure maintenance in encryption keys management – a user is responsible for a task of safe transfer of encryption keys by himself. This thesis aims to develop and implement software that provides data security while using cloud storage services with the use of methods of secure generation and transfer of encryption keys. The main emphasis in the study is on the development and implementation of methods to derivate encryption keys on both image and password: using fractal curves and using DCT coefficients. The analysis with NIST statistical tests demonstrates good statistical properties of the output sequences while using medium size contrast images. The result of the thesis is desktop software that provides data security while using a cloud storage service. Implemented software interacts with the Yandex Disk cloud storage via REST API and ensures data security by performing encryption functions (derivation of encryption keys using developed methods, encryption and decryption). The proposed system carries basic functionality and ready to further expansion. Currently, the implemented software was handed over to a fables-company for acceptance testing and to determine the directions of further improvements.

Keywords: cloud storage service, data security, encryption key derivation.

# Contents

Chapter 1: Introduction .....	1
1.1. Problem and tasks statement.....	1
1.2. Thesis organization .....	2
Chapter 2: Review of existing methods for securing data in cloud storage.....	3
2.1. Client-side encryption in cloud storage services.....	3
2.2. Description of the encryption scheme.....	4
2.3. Encryption key management and cryptanalysis.....	5
2.4. Existing methods for image and password based encryption key derivation overview...7	
2.5. Summary.....	10
Chapter 3: Methods of image and password based encryption key derivation .....	11
3.1. The main idea of developed methods.....	11
3.2. Algorithm of image and password based key derivation using fractal curves.....	12
3.3. Algorithm of image and password based key derivation using coefficients of DCT .....	14
3.4. Algorithm of derivation the output key from the vector of values .....	15
3.5. Summary.....	17
Chapter 4: Implementation and analysis of the methods of image and password based encryption key derivation .....	18
4.1. Description of security module of the developed system .....	18
4.2. Implementation of the algorithm of image and password based key derivation using fractal curves.....	20
4.3. Implementation of the algorithm of image and password based key derivation using coefficients of DCT .....	23
4.4. Comparative analysis of statistical properties of implemented algorithms for image and password based key derivation .....	24
4.5. Summary.....	28
Chapter 5: Implementation of a developed system for ensuring data security while using cloud storage services.....	29
5.1. Interaction with REST API of Yandex Disk .....	29
5.2. Description of system's modules .....	31
5.3. Description of system user interfaces.....	33
5.4. Summary.....	36
Chapter 6: Summary and Conclusions.....	38
6.1. Conclusion .....	38



6.2. Future directions .....	39
Bibliography.....	40

## List of figures

Figure 1 - Model of symmetric cryptosystem .....	5
Figure 2 - Scheme of possible ways to use PBKDF to generate an encryption key.....	7
Figure 3 – JPEG compression scheme .....	7
Figure 4 – Distribution of changes in F5 algorithm .....	8
Figure 5 – Image and password based key derivation algorithms requirements.....	11
Figure 6 - Image and password based key derivation algorithms scheme.....	12
Figure 7 - Stages of the algorithm for extraction of the value vector using fractal curves .....	13
Figure 8 – Construction of Koch's fractal curve .....	13
Figure 9 – Fractal curves example .....	14
Figure 10 - Stages of the algorithm for extraction of the value vector using coefficients of DCT	15
Figure 11 - Stages of the algorithm for derivation output encryption key from the value vector .	16
Figure 12 – Feistel network round scheme .....	17
Figure 13 – Component Diagram of security module of developed system .....	18
Figure 14 – Sequence Diagram of implemented image and password based key derivation algorithm using fractal curves .....	20
Figure 15 – Visualization of the work of the algorithm of extraction of the initial vector of values using fractal curves.....	22
Figure 16 - Sequence Diagram of implemented image and password based key derivation algorithm using DCT coefficients .....	23
Figure 17 – Examples of images from the test sample.....	24
Figure 18 – Uploading a file to a cloud storage .....	29
Figure 19 – Downloading a file from a cloud storage.....	29
Figure 20 – Application registration on OAuth server of Yandex Disk .....	30
Figure 21 – Authentication process.....	30
Figure 22 – Structure of the developed system .....	32
Figure 23 – Sequence Diagram for the "Uploading file to cloud storage with intermediate encryption" usage scenario .....	34
Figure 24 – “FileSystem” user interface designer.....	34
Figure 25 – Main window of the system .....	35
Figure 26 - System main window with established connections .....	35
Figure 27 - Uploading a file to cloud storage.....	36

## List of tables

Table 1 - Time of complete brute-force of different key spaces (at one million tests per second)	6
Table 2 – Average lengths of source value vectors .....	25
Table 3 – Test results of statistical tests of sequences generated by the algorithm of image and password based key derivation using fractal curves (A1) .....	26
Table 4 - Test results of statistical tests of sequences generated by the algorithm of image and password based key derivation using DCT coefficients (A2) .....	27
Table 5 – Percentage of sequences that passed all statistical tests .....	27

# Chapter 1: Introduction

## 1.1. Problem and tasks statement

Nowadays cloud data storages, such as Google.Drive or Yandex Disk [1], are becoming more and more popular, bringing innovations in terms of cost of resources, optimization of company processes and become widely used for both personal and corporate purposes for storage and transfer data, including proprietary information.

The high demand in the use of cloud storage can be demonstrated by the case of fabless-companies - small companies that manufacture microchips and other high-tech devices. Such sort of companies is unable to provide their own secure server storage. In the course of organizations' activities, employees develop intellectual property that have high economic value. This fact makes developed intellectual property limited access information [2]. Due to the specifics of the target companies, the most appropriate way to store and transfer files is to store them in cloud storages.

Storage of data in a cloud storage has various advantages. The main benefits are following: there is no need in corporate servers, there is an ability to provide access to files simultaneously to several users. However, cloud storage security policies do not provide the desired level of data protection, including protection from cloud service providers themselves [1].

The existing solutions in this area have various drawbacks, therefore they cannot be used in companies interested in ensuring the security of processed data, in particular, in fabless-companies. The most critical disadvantage of the existing solutions is the low level of security in the implementation of the encryption key management process, which is a major component of the entire cryptographic system [3]. Thereby there is a need to develop software that will ensure data security while using a cloud storage with the focus on the use of methods of secure generation and transfer of encryption keys.

In the present work, the task of ensuring data security while using cloud storage services is considered. The aim of the work is to develop a method to provide high level of security in encryption key derivation and management processes while using cloud storage services. To achieve this goal, this work solves the following tasks:

1. Development of methods of generating encryption keys takes into account the area of application of the system.
2. Implementation of the developed methods and analysis of statistical properties of generated sequences using NIST Statistical Test Suite [4].
3. Development and testing of the software that provides data security while using cloud storage services. System should meet the following requirements:
  - interaction with the Yandex.Disk cloud storage service by using REST API;
  - use of developed methods to generate encryption keys;
  - implementation of encryption and decryption methods for various types of files.

The object of the study is the system that provides data security while using cloud data storage by means of cryptographic functions. The subject of the study is the methods and the models to generate encryption keys. The following points determine the scientific novelty of this work:

1. Development of the methods to derivate encryption keys based on both image and password.
2. Implementation of the developed methods in Python 3.7.
3. Comparative analysis of the statistical properties of the generated sequences. Investigation of the dependence between statistical properties of the generated sequences and input image parameters.

## **1.2. Thesis organization**

The Master's Thesis consists of an introduction, theoretical, algorithmic, practical chapters and a conclusion. The source code of the main components of the developed software is presented in the application at the end of the work.

The first chapter gives an overview of existing methods to provide data security while using cloud storage. This chapter considers the main principles of implementation of the client-side data encryption scheme in cloud data services, methods of managing and transferring encryption keys and identifies the main vulnerabilities.

The second chapter describes the developed methods to derivate encryption keys based on both image and password: the first method uses fractal curves and the second method uses discrete cosine transform (DCT) coefficients.

The third chapter describes the details of implementation of the developed methods and results of the analysis of the statistical properties of generated sequences.

The fourth chapter contains the description of the developed software to ensure data security while using cloud storage services.

## Chapter 2: Review of existing methods for securing data in cloud storage

### 2.1. Client-side encryption in cloud storage services

Cloud storage is a system that uses cloud technology and provides a storage interface. These services allow users to access data regardless of their location and device used [5].

The most vulnerable point of cloud storage is data security: most cloud service providers do not provide encryption services. Thus, when data is uploaded to cloud storage, it becomes available to the cloud provider and the probability of the threat of unauthorized access and modification of data becomes critically high. Attacks such as password brute-force and cloud storage compromise are become highly possible [3]. Therefore, the use of cloud services without an additional level of security jeopardizes the confidentiality and integrity of the data.

While using cloud storage for corporate purposes, the risks from unauthorized access threats and modification of sensitive data in cloud storage are critical. There are the following ways to process the risks of this type:

1. **Building own secure storage** - deploying and installing a file server on the local network and creating secure connections to it (for example, using VPN technology). This method is efficient and provides a high level of protection against external intruders, but it is very expensive in development and maintenance, which makes it inapplicable in a small business environment.

2. **Client-side encryption** is the best way to secure data while using cloud storage. The method consists of encrypting the files on the user's local machine before uploading them to the storage and locally decrypting the files after downloading them to the local machine. In this case, the user doesn't have to worry about cloud storage security policies - data security is provided by encryption features. The most important task in client-side encryption is to ensure secure transmission of the encryption key [3].

At this moment there are various software solutions that partially solve the problem of protecting data stored in cloud data storages: BoxCryptor [6], Mega [7], TrueCrypt [8]. These software solutions are a powerful tool for ensuring data privacy: they use reliable algorithms of symmetric encryption (AES-256, RSA-4096), but most of them do not meet the requirements of easy access to the system, as well as the requirements of providing a high level of data availability and adaptability to the needs of a particular organization. The general disadvantages of these services are the following:

- the absence of integration with the Russian service Yandex.Disk [1], which is critical for domestic companies in accordance with the Decree of the Government of the Russian Federation from 16.11.2015 № 1236 "About establishing a prohibition on the

admission of software originating from foreign countries" [8], which prohibits Russian organizations from using software of foreign origin;

- inconvenient and insecure key management: keys in all services are generated in the form of text files - the problem of secure key transfer and storage falls entirely on users' shoulders, which threatens the confidentiality of protected data.

Thus, there is a need for a software that would meet the following requirements:

- providing convenient and efficient tools for fast downloading and downloading of files in safe mode;
- integration with the cloud storage service Yandex.Disk;
- ensuring data privacy (key generation and transfer, data encryption), taking into account the specifics of the system's application area.

The following paragraph provides an overview of data encryption algorithms used on the client side, their shortcomings and methods of their elimination.

## 2.2. Description of the encryption scheme

An **encryption algorithm** is the translation of plaintext into key-encrypted text. Encryption function  $E$  with key  $K$  acts on the plaintext  $M$ , creating ciphertext  $C$ , so that the decryption function  $D$  with the same key  $K$  applied to text  $C$  gives the original  $M$  message at the output:

$$C = E_K(M)$$

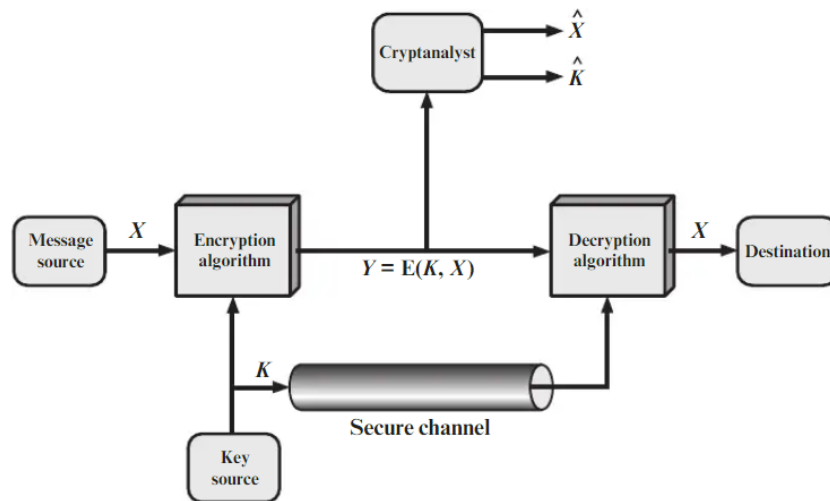
$$M = D_K(C)$$

It should be noted that the encryption and decryption algorithms  $E$  and  $D$  are open and public, and the secrecy of the original message  $M$  in ciphertext  $C$  completely depends on the secrecy of key  $K$  [10]. There are two main classes of key-based encryption algorithms: symmetric and asymmetric (public-key algorithms).

**Asymmetric encryption** algorithms use two independent keys: the public key used to encrypt the original message and the private key used to decrypt the cipher text. Although asymmetric encryption algorithms have high cryptographic endurance, their use is not optimal in terms of the scope of the system. While using cloud storage for corporate purposes, all team members should be able to quickly access the files stored on the disk.

In **symmetric algorithms**, the encryption and decryption keys are identical, or the decryption key can be calculated based on the encryption key. Thus, the security of algorithms in this class depends entirely on the security of the key. The most popular algorithm for symmetric encryption is AES (Advanced Encryption Standard). AES is a block cipher that operates with blocks of 128 bits in length. The length of the key for the algorithm can be 128, 192 or 256 bits. To ensure cryptographic stability, AES includes repeating rounds, each of which consists of substitutions, permutations, and key additions. The round function operates using four operations: SubBytes, ShiftRows, MixColumns, AddRoundKey [10].

The main problem of using symmetric cryptosystems is a key distribution (Figure 1). In order to be able to exchange information between two sides, the key must be generated by one side and safely passed to the other. At present the schemes of safe transfer of keys of encryption (for example, the Diffie-Hellman protocol) are developed. However, all of them demand additional actions from users and do not approach to the area of application of developed system where the resource added to cloud storage should become accessible to command of users for a short period of time.



**Figure 1 - Model of symmetric cryptosystem**

### 2.3. Encryption key management and cryptanalysis

The essence of cryptography is to keep the plaintext and encryption key secret from attackers, assuming that they have full control over the communication lines between the sender and the recipient. The process of obtaining plaintext without the key is called cryptanalysis. The basic assumption of cryptanalysis is that security is fully defined by the key (Dutchman A. Kerckhoffs) [10].

At present, the most difficult part of cryptography is key management: the security of cryptographic algorithms is proved by many theoretical studies, and the key secret is much harder to keep because of the human factor. For example, DiskLock for Macintosh used the DES algorithm for encryption, the security of which has been proven. However, when encrypting files, the encryption keys were saved along with the encrypted algorithm: to get the original file, an attacker only had to read the key from the encrypted file at a certain position. Another example is a vulnerability in the code of the BitCoin wallet application, where the encryption key generator was not initialized with salt so that the source keys of all users had the same value. [11]

All cryptosystems except the one-time pad are subject to brute-force opening. An algorithm is considered computationally secure if it cannot be cracked using available computing resources in the foreseeable future. Table 1 shows the time required to completely brute-force all possible keys that meet the specified conditions at a million attempts per second [12]. For a



complete search, any hardware and parallel implementations can be used. Thus, the length of the encryption key and the characters included in it critically affect the cryptographic stability of the whole cryptosystem.

	<b>4 bytes</b>	<b>6 bytes</b>	<b>8 bytes</b>
Highercase letters and numbers (36)	1.7 seconds	36 minutes	33 days
Alphabetical and digital symbols (62)	15 seconds	16 hours	6.9 years
Printable symbols (95)	1.4 minutes	8.5 days	210 years
ASCII Symbols (128)	4.5 minutes	51 days	2300 years
8-bit ASCII characters (256)	1.2 hours	8.9 years	580000 years

**Table 1** - Time of complete brute-force of different key spaces (at one million tests per second)

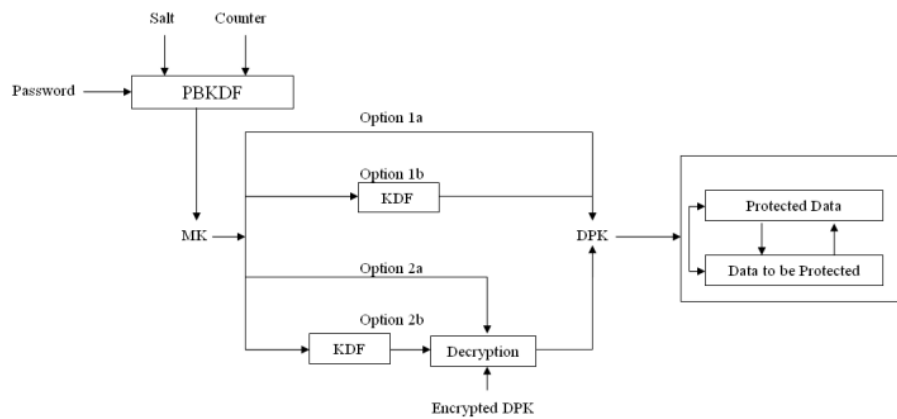
The main method of generating encryption keys for symmetric encryption algorithms is using pseudo-random number generators (PRNG). Pseudo-random sequence generation is based on a physical source of random, hard to predict information called sand. PRNGs are well researched at the moment and meet the requirements for statistical properties of the generated sequences.

The key generation standard ANSI X9.17 defines two types of keys: data encryption keys, which encrypt data keys, and data keys, which encrypt the messages themselves [12]. The following requirements must be met while using this approach:

- both sides should be able to generate data keys based on the key-encryption keys;
- both sides must have a channel of communication to transfer the key-encryption key;
- the transfer of key encryption keys must be implemented in a secure manner.

This approach finds application in various methods, each of which has its advantages and disadvantages. **Password-Based Key Derivation Function (PBKDF)** is the most popular and uses the principles of pseudo-random number generators [13]. When generating a password-based encryption key, the function's input consists from password, sand and the required length of the encryption key. The scheme of possible ways to apply the PBKDF algorithm to generate the encryption key is shown in Figure 2.

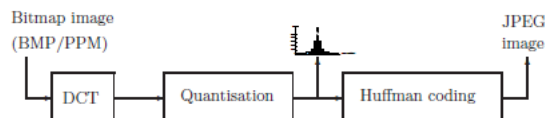
Nowadays there are standardized algorithms for generating keys based on a password with proven persistence: for example, the standard PBKDF2 [14]. However, the disadvantage of this type of algorithms is that they are vulnerable to Man in the Middle attacks since the password is transmitted through open communication channels. To solve this problem, it is necessary to use complex secure methods of generating and transmitting encryption keys.



**Figure 2** - Scheme of possible ways to use PBKDF to generate an encryption key

## 2.4. Existing methods for image and password based encryption key derivation overview

Considering the current trends, one of the most popular ways to transfer encryption keys is to use media files. The most convenient format for processing is JPEG. Files in JPEG format store image data in a compressed form with losses in the form of quantized frequency factors. The Figure 3 shows a scheme for compressing images of JPEG.



**Figure 3** – JPEG compression scheme

The original raster image is divided into blocks of 8x8 pixels, to which Discrete Cosine Transform (DCT) is applied. DCT converts pixel values into coefficients, forming an 8x8 matrix at the output. In output matrix coefficients in the upper left corner correspond to the low-frequency component of the image, and in the lower right corner - to the high-frequency component. Then the quantization of the coefficients is applied: DCT coefficients are converted into integers in the range from -2048 to 2047 using the quantization matrix. The final part of the process is encoding: the coefficients in matrix are bypassed in zigzag order and the first (most significant coefficients) are encoded using the repeats encoding algorithm, after which the final result is encoded by the Huffman algorithm [15].

Two main approaches can be implemented while using JPEG media files to generate and transfer encryption keys are following:

1. Using media files as containers for embedding and hidden transfer of information.
2. Extraction of information from media files.

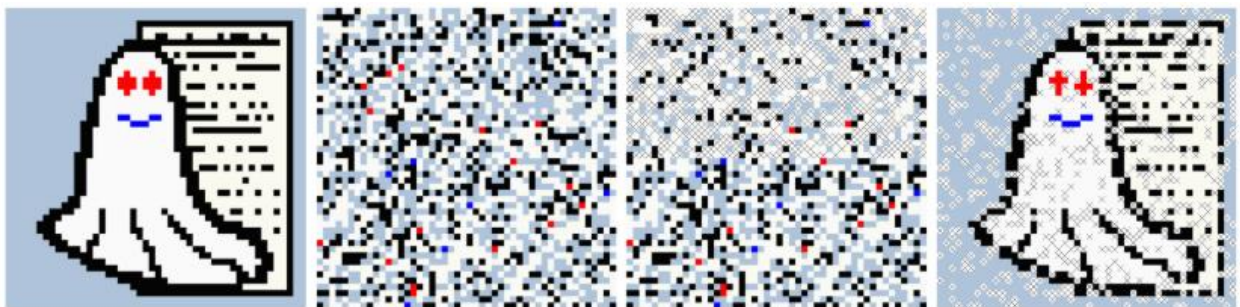
### 2.4.1. Steganography algorithms

Methods of the first class are implemented in steganography algorithms. The main task of steganography is to hide the fact of message transmission. This task is solved by embedding the message in the container, the transmission of which is performed on a regular basis and does not cause suspicion. The most commonly used type of container used in steganography algorithms is graphic containers of JPEG format. Two main approaches are used in steganography algorithms:

1. Embedding the message by overwriting the least significant bits of DCT coefficients with the message bits. The main property of the algorithms of this type is that their bandwidth can be increased by increasing the number of overwritten bits.
2. Embedding the message by changing certain DCT values provided that the given ratio is satisfied, thus implementing the principle of relative replacement of DCT coefficients. Algorithms of this type have low bandwidth [16].

Various steganography algorithms for embedding data into JPEG images are discussed in [15]. In particular, this paper describes the F5 algorithm, which has the highest secrecy and resistance to statistical attacks. A specific feature of the algorithm is the uniform distribution of changes throughout the image (Figure 4). In general, the algorithm F5 contains the following stages:

1. Execution of JPEG compression stages: obtaining DCT coefficients and quantization.
2. Generation of pseudo-random number permutation from 1 to N (number of DCT coefficients) by initializing the random number generator with the password entered by the user.
3. Embedding message bits in the modified sequence of DCT coefficients using the matrix encoding algorithm.
4. Performing the remaining stages of JPEG compression (Huffman encoding).



**Figure 4** – Distribution of changes in F5 algorithm

The F5 algorithm can be used to transfer encryption keys by embedding them in media files and transferring them over open communication channels. The advantages of F5 algorithm are high resistance to statistical attacks and high bandwidth [15]. The disadvantages of the algorithm are high sensitivity to image changes (if the brightness changes, it will be impossible to extract the embedded message). In addition, the algorithm has a relatively high complexity,

respectively - the program running time, which is important while using the system under development. All users from the command should receive new files in a short period of time and without extra effort.

#### **2.4.2. Image based key derivation algorithms**

The methods of second class, in which the encryption key is extracted from the image itself are not widely understood. The main problem in solving the problem of generating keys from images is the problem of achieving high entropy of the obtained keys at any parameters of the original image (e.g., size and contrast). With regard to images, entropy is determined by the contrast of the image and the smooth transition of colors. Smooth images with large areas of smooth color transitions have low entropy. Contrast mottled images, in contrast, have high entropy [17]. Therefore, in order to develop sequences meeting the requirements to the statistical properties of the produced encryption keys, it is necessary to provide the additional processing to data obtained from the image (e.g., permutation or encryption).

At present, the majority of works in this area are devoted to biometric data based key derivation algorithms. Standard algorithms of this type, which operate only with biometric data by users, have such disadvantages as small key lengths and high probability of second-generation errors (more than 20%) due to inaccurate reproducibility of data. The method of "fuzzy extractors" suggested in [18] requires storage of open user data for exact reproducibility of the key. The method generates an encryption key from the original input data and then restores this sequence correctly from any data similar to the original data. The disadvantages of key generation algorithms based on biometric data include the following:

- key change problem - to ensure a high level of security, encryption keys must change over time, which is not possible with the same biometric data;
- impossibility to hide biometric data, as a consequence - probability of key compromising by an intruder.

Because different people should have access to data in the system under design, existing methods of key generation based on biometric data are not suitable. The suggested methods of generation of encryption keys based on images take the image and key length for input and calculate the hash based on bit representation of the image or apply fixed pseudo-random permutation [17]. The analysis of statistical properties of the proposed methods has been made based on the following statistical tests of NIST pseudo-random sequences: Frequency Test, Serial Test, Poker Test at the selected significance level  $\alpha=0.05$ . The results of testing the generated sequences have shown the following results:

1. Contrast of the original images directly influences the entropy of the obtained keys (sequences obtained from low entropy images have not passed any tests).

2. The smaller the length of the generated sequences, the more successfully they pass the tests (all tests passed 52.2% of the generated sequences of 256 bits length, and only 42.1% of the generated sequences of 512 bits length).

The disadvantage of the proposed methods is that the algorithms are not resistant to man-in-the-middle attacks - obtaining the image will be equivalent to obtaining the encryption key.

Thus, the existing methods of generating encryption keys based on images have a number of critical shortcomings that do not allow using them in the system to ensure the confidentiality of corporate data. The task of developing an algorithm for generating encryption keys based on images is relevant.

## **2.5. Summary**

The security of a cryptosystem depends primarily on the security of the procedure to generate and manage encryption keys. One of the most secure methods of key management is to apply the principle of using a key pair: the key-encryption key and the data encryption key. At the moment, a various key generation algorithms have been developed to use this principle, e.g. steganography algorithms, password-based, biometric data-based or image-based key derivation algorithms. The main advantages of the developed methods include high speed and good statistical properties. Nevertheless, almost all algorithms have disadvantages that do not allow to utilize them while using cloud data storages. The algorithms of steganography require additional computing resources on both sides. All developed algorithms are based on the input data, and take the key length and only an input parameter (image or password). This fact makes them vulnerable to man-in-the-middle attacks.

Thus, the development of new algorithms to derivate encryption keys based on user-defined information, that combine high performance and good statistical properties of output sequences as well as a high degree of protection from man-in-the-middle attacks, is an actual scientific and practical task.

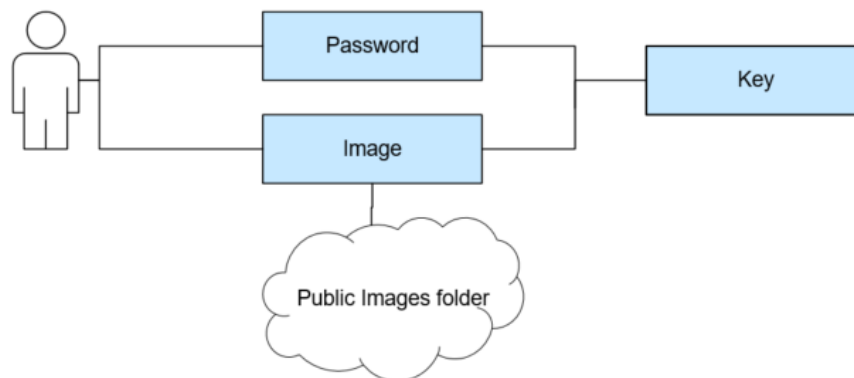
## Chapter 3: Methods of image and password based encryption key derivation

This chapter reviews the developed methods of derivation of encryption keys based on images and password.

### 3.1. The main idea of developed methods

As it was described earlier, existing methods of generating encryption keys based on the source data (password or image) take only one input value and unambiguously generate an output sequence from it. This approach makes the algorithms vulnerable to man-in-the-middle attacks.

The developing data security system implies the need for quick access to newly downloaded data to the cloud storage, respectively - fast transfer of encryption keys between several users. The best approach is to use images as the source data for generating encryption keys: the cloud storage can host a public folder with images, some of which can be the basis for derivation of encryption keys. However, in order to provide different levels of access to the stored files, another level of security is required. Thus, an algorithm of derivation encryption keys based on the images and the password entered by the user is required (Figure 5).



**Figure 5** – Image and password based key derivation algorithms requirements

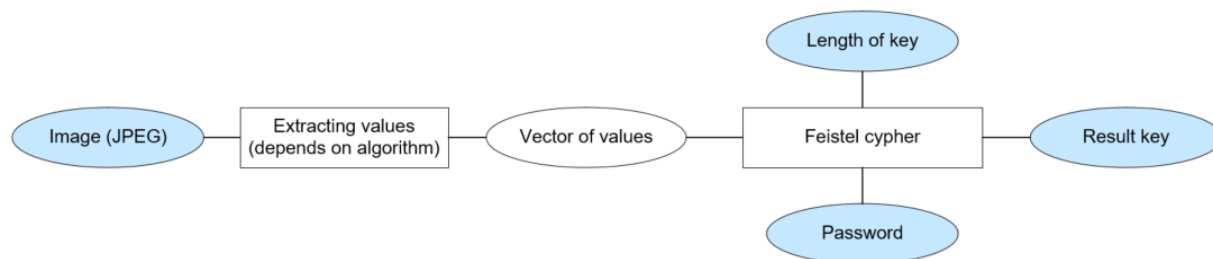
In general, data from the image in JPEG format can be presented in two types: as bit representation of pixel values and as spectral interpretation generated using DCT. Therefore, methods of derivation an encryption key from the image can use two following approaches:

1. Use of pixel values of the source image.
2. Use of DCT coefficients of the source image.

In case of using pixel values of the source image, the problem of low entropy of a number of standing pixels arises - to solve this problem, an algorithm of nontrivial generation of pixel position sequence based on fractal curves is proposed.

To ensure good statistical properties of output sequences, the developed methods use the Feistel cipher (block encryption algorithm) [10]. Reproducibility of the obtained sequences is

provided by initialization of the original round key by the hash function from the password provided by the user. The general scheme of the developed methods is presented in Figure 6.



**Figure 6** - Image and password based key derivation algorithms scheme

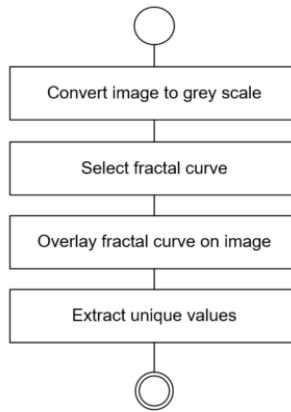
Next paragraphs describe in detail the developed algorithms for derivation the encryption key from the image and password. Each algorithm consists of two parts: extraction of the value vector from the specified image and its encryption via the Feistel cipher using the specified password. The developed algorithms differ in the first part: the first algorithm uses pixel values generated using fractal curves, and the second algorithm uses DCT coefficients. The second parts of the algorithms are identical, so it will be described separately.

### **3.2. Algorithm of image and password based key derivation using fractal curves**

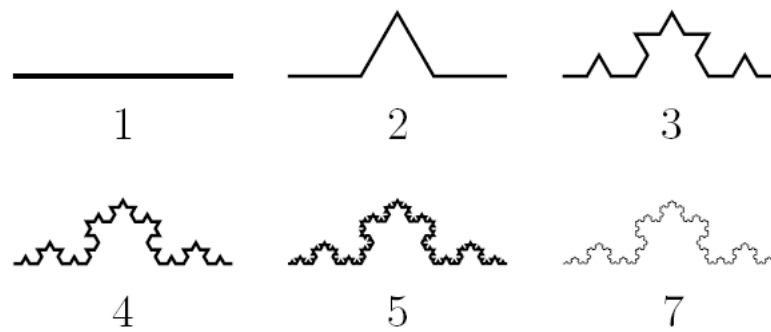
In the algorithm of the value vector derivation from the image using fractal curves, the approach of using pixel values is implemented. The main problem of this approach is the fact that the neighbouring pixels in the image are equivalent or close to each other. As a result, using a sequence of neighbouring pixels is not an optimal solution - values should be extracted according to a given algorithm. On the other hand, the output sequences should be reproducible.

In the considered approach, fractal curves [19] are used to solve the problem mentioned above. An input of the algorithm is image. Then an intermediate vector of values is generated by overlaying the fractal curve, selected depending on metadata, on the image, converted into grayscale. The output sequence is obtained by extraction of unique values from the intermediate vector. The steps of the algorithm are shown in Figure 7.

In the first step, the image is converted to the grayscale. Each pixel of the original color image is a set of three values of additive RGB color model - one value from the interval [0, 255] for each color. These values are often redundant, so further work is done on the grayscale image, where each pixel corresponds to one value from the interval [0, 255].



**Figure 7** - Stages of the algorithm for extraction of the value vector using fractal curves



**Figure 8** – Construction of Koch's fractal curve

At the second stage of the algorithm, a fractal curve is defined from a predefined set that will be used to extract the output value vector. The initial set of fractal curves is predefined and can be extended while using the program. The number of the fractal curve is uniquely defined for each image by the following formula:

$$i_{fractal} = \frac{\sum pixel[x,y]}{Count\ of\ pixels} \% (Count\ of\ fractals)$$

The set of fractal curves contains 15 curves, including the following ones:

1. **Koch snowflake** is a continuous fractal curve. At zero iteration, the curve is a line segment. At each subsequent iteration the segment is divided into 3 equal parts, and the central part is replaced by two segments obtained by finishing to the correct triangle. The process is repeated to the specified number of iterations (Figure 9, a).

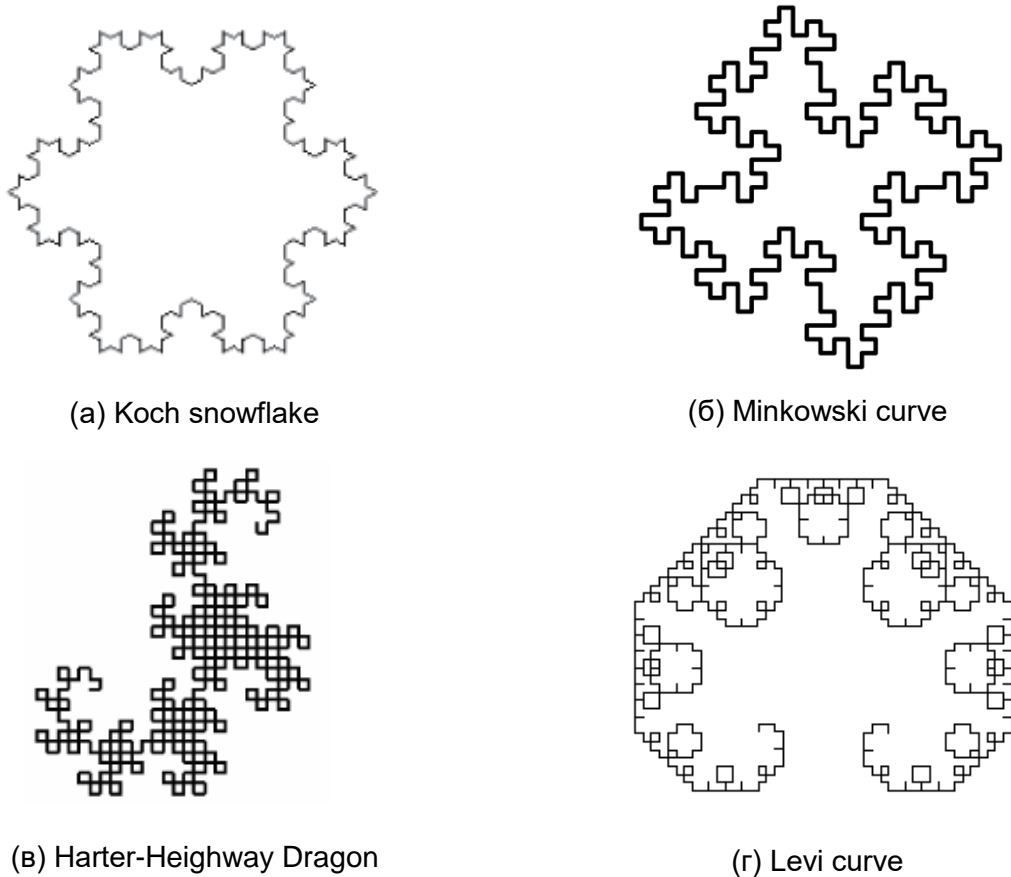
2. **Minkowski curve**. At zero iteration, the curve is a horizontal line segment. At each subsequent iteration, each of the segments is replaced by a broken segment of 8. The process is repeated until the specified number of iterations (Figure 9, b).

3. **Harter-Heighway Dragon**. At zero iteration, the curve is a horizontal line. At the first iteration, the segment is split in half and joined at a right angle. At each subsequent iteration, the



right angles are built again on each side of the right corner, with alternating convexity directions (Figure 9, c).

4. **Levi curve.** At zero iteration, the curve represents two line segments connected at an angle of 90. At each subsequent iteration, each side is replaced by the same fragment. The process is repeated until the specified number of iterations (Figure 9, d).



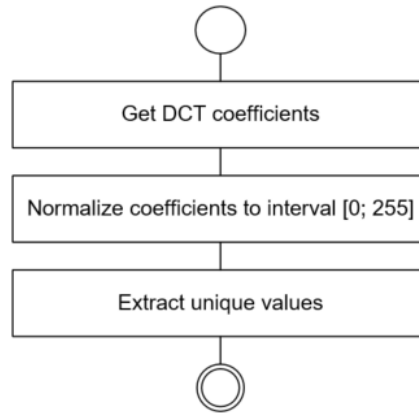
**Figure 9** – Fractal curves example

At the final stage of the algorithm, the elected fractal curve is overlaid on the original image in grayscale. The output vector is formed from unique values of pixels, located at positions in the of fractal curve construction points. As can be seen from the figures, the fractal curves have a nontrivial form, which allows obtaining values from pixels, located in different areas of the image. For obtaining the output encryption key, the generated vector of values is provided to the algorithm implementing the Feistel cipher for obtaining good statistical properties.

### 3.3. Algorithm of image and password based key derivation using coefficients of DCT

The second algorithm for generating the encryption key based on the image and password implements the approach, which is based on the DCT coefficients of the original image in JPEG format. The main problem with this approach is the fact that the coefficients may be negative.

In this algorithm, normalization is applied to solve the problem mentioned above. An image is given to the input of the algorithm. The output sequence is the result of extraction of unique values from the coefficient vector normalized in the interval [0, 255]. The algorithm stages are shown in Figure 10.



**Figure 10** - Stages of the algorithm for extraction of the value vector using coefficients of DCT

The input to the algorithm is the original image. The first step is to calculate the DCT coefficients and gather them into 1-D array  $\{k_1, \dots, k_N\}$ . Then the generated values are normalized to the values  $\{\bar{k}_1, \dots, \bar{k}_N\}$  from the interval  $[a, b]$  as follows:

$$\bar{k}_i = a + \frac{(b - a) * (k_i - \min_j k_j)}{(\max_j k_j - \min_j k_j)}$$

Due to the fact that 8-bit ASCII characters should be used for the output sequence, the initial DCT coefficients should be normalized into integer values from the interval [0, 255]. In this case, the formula is converted to a following:

$$\bar{k}_i = \frac{255 * (k_i - \min_j k_j)}{(\max_j k_j - \min_j k_j)}$$

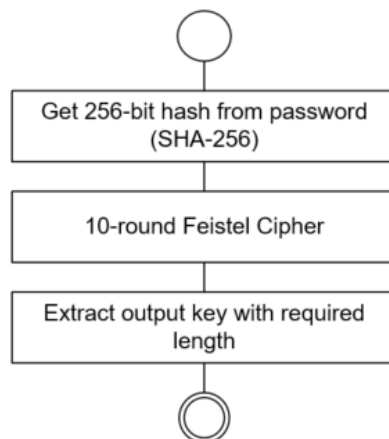
At the last stage, unique values are extracted from the generated sequence – this procedure significantly reduces the vector size. To obtain the encryption output key, the obtained value vector is then sent to the input of the algorithm that performs the Feistel cipher described in the next paragraph.

### 3.4. Algorithm of derivation the output key from the vector of values

To provide good statistical properties of the output sequences and to increase the cryptographic stability of the whole value vector system, additional processing of the obtained value vectors is applied. The developed method suggests using such a basic cryptographic algorithm as the Feistel network (Feistel cipher). The Feistel network is a block encryption method developed by Horst Feistel in IBM laboratory in 1971. The algorithm converts plain text into similar-sized, keyed, encrypted text. The choice of the algorithm is explained, on the one hand,

by proven cryptographic stability and good statistical properties, and on the other hand, by simplicity and compactness of calculations [10].

The algorithm receives the input vector values  $\{k_1, \dots, k_n\}$  from the interval  $[0, 255]$ , generated from the input image by one of the algorithms described above, the password entered by the user, and the required key length. The general scheme of the algorithm operation is shown in Figure 11.



**Figure 11** - Stages of the algorithm for derivation output encryption key from the value vector

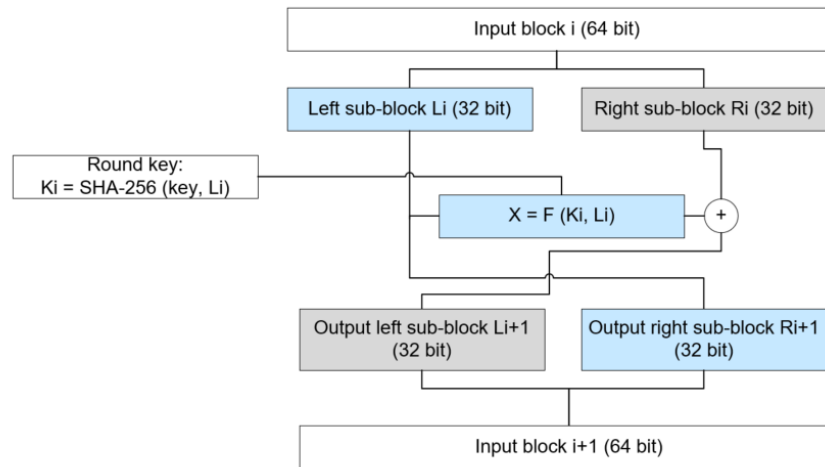
On the first stage of the algorithm, the first round key  $key_0$  (256-bit) of the Feistel network is generated based on the entered password - SHA-256 hash function. For reproducibility of the output sequence, the hash function is initialized with sand - user-defined password.

At the second stage of the algorithm, five rounds of the Feistel network are implemented. The choice of the number of rounds is explained by the fact that in [21] it is proved that the output value after applying five rounds is a function of all bits of public text and all bits of the key, and after eight rounds - a random function of all bits of public text and the key. Due to the fact that the source public text (source value vector) is generated using a non-trivial algorithm, the number of rounds equal to five has been selected to reduce the running time. The network operates with blocks of 64 bits of open text. On each of the rounds, the input block is divided into two equal sub-blocks  $L_i$  and  $R_i$ . After the division, function  $f$  is applied to the left subblock  $L_i$  based on the round key  $key_i$ , and it is added to the right subblock  $R_i$  by module 2. Function  $f$  is bitwise multiplication of block  $L_i$  and round key  $key_i$ , raised in the power of the current round number  $i$ :

$$f(L_i) = pow(L_i * key_i, i)$$

Then the left and right sub-blocks are swapped and joined together to form a 64-bit output block for the current round. For each following round, a new round key  $key_{i+1}$  is formed by taking a hash function from the string which is the result of joining the current round key and the resulting right subblock  $R_i$ . The algorithm of one round of the Feistel network is shown in Figure 12.

After receiving the encrypted sequence, the last step is to extract the encryption output key of the specified length. The length of the key is set depending on the encryption algorithm used.



**Figure 12 – Feistel network round scheme**

### 3.5. Summary

This chapter describes the methods developed to derive encryption keys based on both images and passwords. The methods consist of two stages: extraction of the value vector from the input image and encryption of the value vector with the password in order to increase its cryptographic stability. The developed methods use two principles to extract the value vector from the input image:

1. Extracting of pixel values from the input image converted to grayscale by overlaying a fractal curve selected based on a parameters of that image.
2. Extracting of DCT values normalized to the interval  $[0, 255]$ .

The output sequences generated by these algorithms contain unique values from the interval  $[0, 255]$  in a random order. The maximum sequence length is 256 bytes and depends on such parameters of the source image as contrast and size.

It is proposed to use a block encryption algorithm based on the Feistel network as a method to source vector processing. The initial round key is a hash function that is generated using the SHA-256 algorithm based on the password in order to ensure reproducibility of the output sequence

Encryption of vector allows to increase pseudo-randomness of the generated sequences. Analysis of statistical properties of the implemented algorithms will be described in the next chapter.

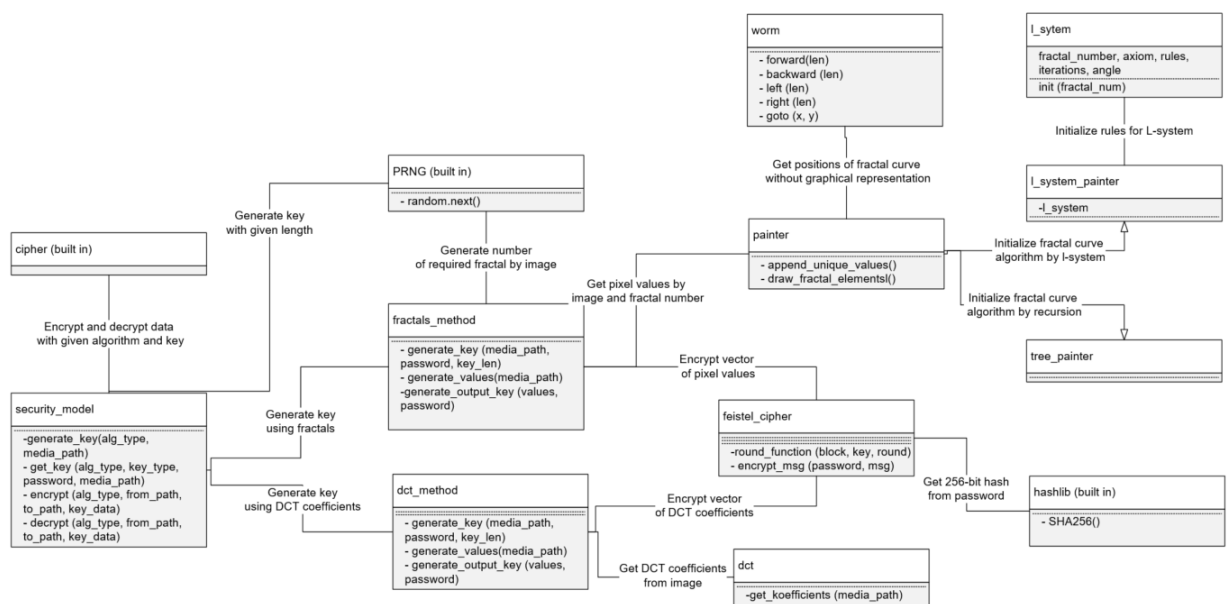
## Chapter 4: Implementation and analysis of the methods of image and password based encryption key derivation

This chapter describes the implementation of developed methods for generating encryption keys based on images and passwords. It describes the process of analyzing the crypt stability of the developed algorithms based on statistical tests of the generated sequences, the results obtained and conclusions.

The developed methods are a part of the developed system for ensuring data confidentiality while using cloud storages. The methods are implemented in Python 3.7 programming language in JetBrains PyCharm integrated development environment and used in the system as a part of security\_model component. The convenience and flexibility of the language, as well as the presence of a large number of ready-to-use libraries explain the choice of a programming language. Next chapter gives the full description of the developed system.

### 4.1. Description of security module of the developed system

The Component Diagram of the module of the developed system named "security\_model" is shown in Figure 13.



**Figure 13** – Component Diagram of security module of developed system

The main control class of the module is “security\_model”. Public functions of this class has the following arguments: the path to the open text file “from\_path”, the required path to the output encrypted file “to\_path”, and the set of values “encryption\_data”. Set “**encryption\_data**” has the following fields:

- encryption algorithm (“algorithm”) - selected from a pre-defined set of symmetric encryption algorithms (3DES, AES);

- type of key (“key\_type”) - the following key types are available: symbol string, binary file, image and password-based key using fractal curves, image and password-based key using DCT coefficients;
- key (“key”) - for the "Symbolic string" key type contains a symbolic string, which is an encryption key, for the image and password-based key types contains a symbolic string - a password;
- file path (“media\_path”) - for the "Binary file" key type contains a path to the file with the encryption key, for the image and password-based key types contains the path to the image.

The “security\_model” class provides functions such as following:

1. **Key generation** (“generate\_key” function) - Generate\_key - depending on the key type, a new encryption key is generated with the length according to the selected encryption algorithm:

- for the "Symbol string", "Binary file" key type, a pseudo-random sequence of a specified length is generated and this sequence is written to the binary file (optional);
- for the "Image and password-based key using fractals\_method" key type, the "fractals\_method" function is called. Function receives the path to the image and the required key length as arguments. The method returns a character string, which will later be the password. The implementation of this class is described below;
- for the "Image and password-based key using DCP coefficients" key type, the "dct\_method" function is called. Function receives the image path and the necessary key length as arguments. The method returns a character string, which will later be the password. The implementation of this class is described below.

2. **Data encryption** (“encrypt” function) - performs encryption of the source file “from\_path” according to the “encryption\_data” parameters and writes the encrypted data to the “out\_path” file. During the execution of the function, the method of built-in “cipher” class from “cryptography” library is called. In order to provide flexibility of the system and allow to edit set of used cryptographic algorithms Factory Method design pattern is used (Algorithm 1).

```

1. def get_algorithm(self, algorithm, key):
2.     if algorithm == SecurityAlgorithm.none:
3.         return None
4.     elif algorithm == SecurityAlgorithm.aes:
5.         return algorithms.AES(key)
6.     elif algorithm == SecurityAlgorithm.camellia:
7.         return algorithms.Camellia(key)
8.     elif algorithm == SecurityAlgorithm.triple_des:
9.         return algorithms.TripleDES(key)
10.    elif algorithm == SecurityAlgorithm.seed:
11.        return algorithms.SEED(key)
12.    else:
13.        logger.error("Not supportable algorithm")
14. algorithm = self.get_algorithm(algorithm_type, key)
15. cipher=Cipher(algorithm, mode=modes.ECB(), backend=default_backend())

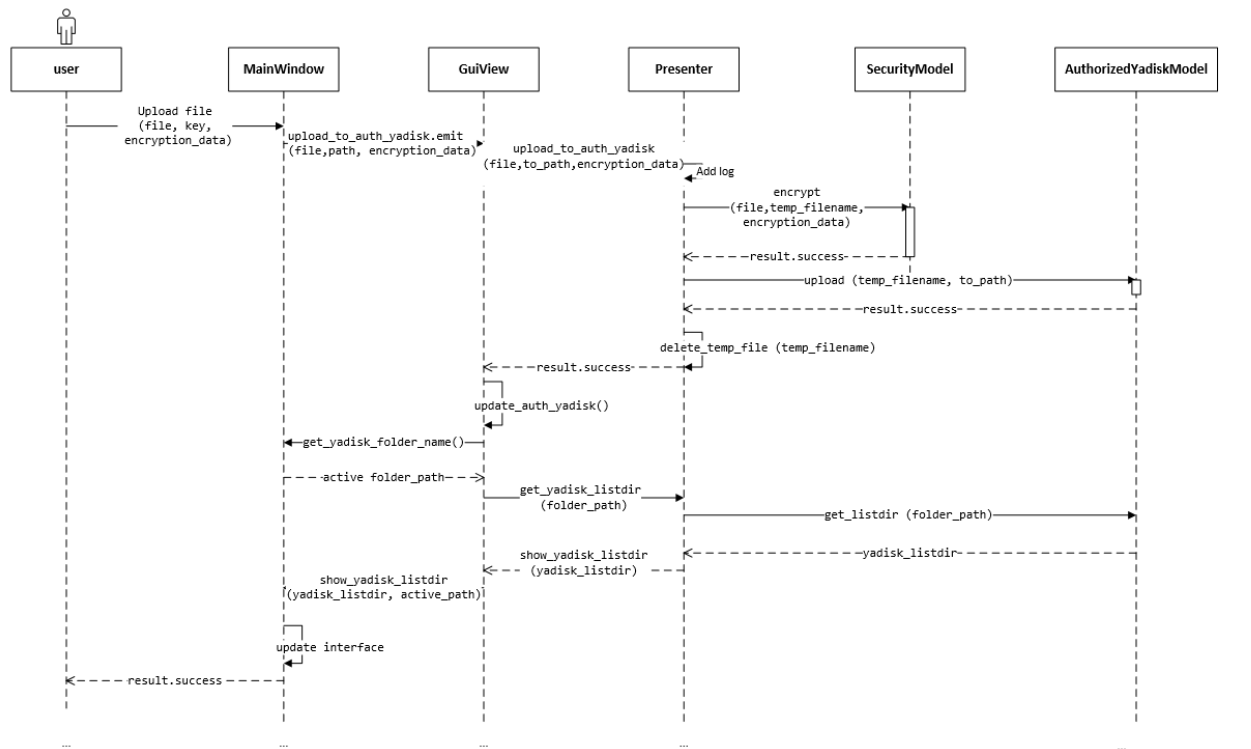
```

**Algorithm 1** - Using Factory Method design pattern in “security\_model” class

3. **Data decryption** (“decrypt” function) - performs decryption of the source file “from\_path” according to the “encryption\_data” decryption parameters and writes the decrypted data to an “out\_path” file. During the execution of the function, the method of built-in “cipher” class from “cryptography” library is called.

#### 4.2. Implementation of the algorithm of image and password based key derivation using fractal curves

The method of generating the encryption key based on the image and password using fractal curves is implemented in “fractal\_method” class. The Sequence Diagram for derivation an encryption key using this method is shown in Figure 14.



**Figure 14** – Sequence Diagram of implemented image and password based key derivation algorithm using fractal curves

While implementing this method, a special difficulty was to implement a program code for building fractal curves. Construction of fractal curves is traditionally realized with the help of “turtle” graphic module. This graphical module is built on the metaphor of the device, which moves on the screen in discrete steps in given directions and leaves a trace of a given colour and width. The basic methods of the module are the following commands: forward (len), backward (len), left (angle), right (angle), goto (x, y). Calling the functions of an instance of the turtle class opens a graphical window, which is not suitable for the developed system. Therefore, own similar “worm” class was implemented. “Worm” class has x, y (current coordinates) and a (direction) fields, and basic functions similar to functions of the turtle module.

Fractal curves can be set in two ways:

1. **Recursive function** - a polyline with a finite number of segments is set as an initial value. Then each segment in initial polyline is replaced by a similar polyline, and so on. Example of the recursive function that sets a fractal curve “Koch snowflake” is shown in Algorithm 1.

```

1. def snowflake(self, length, depth):
2.     if depth == 0:
3.         self.turtle.forward(length)
4.         self.append_unique_values()
5.         return
6.     length /= 3.0
7.     self.snowflake(length, depth-1)
8.     self.worm.left(60)
9.     self.snowflake(length, depth-1)
10.    self.worm.right(60)
11.    self.snowflake(length, depth-1)
12.    self.worm.left(60)
13.    self.snowflake(length, depth-1)
14. def fractal_elements(self, image):
15.     Painter.fractal_elements(self, image)
16.     for i in range(100):
17.         self.snowflake(300, 4)
18.         self.worm.right(60)

```

**Algorithm 2 – Recursive function for fractal curve representation**

2. **The Lindenmayer system (L-system)** is a way to represent recursive structures as a string of characters and multiple rewrites of a given string. The alphabet in L-system is a set of symbols, which the system operates with [20]. According to the available methods of “worm” class, the alphabet consists of characters {F, +, -} where "F" indicates moving forward, "+" - rotation to the right, "-" - rotation to the left (left). The L-system consists of the following elements:

- axiom - source string that defines the initial state of the system;
- set of rules - instructions describing how to change each character describing the system state to another iteration;
- angle - angle of rotation of the direction at the next iteration;
- number of iterations.

For example, Algorithm 3 shows L-system of “Koch snowflake” fractal curve.

```

1. axiom = "F--F--F"
2. rules = {"F+F--F+F"}
3. iterations = 300
4. self.angle = 60

```

**Algorithm 3 – L-system for fractal curve representation**

However, not all fractal curves can be defined as an L-system (e.g., crop-shaped structures) [20], so hierarchal structured classes were developed in order to work with fractal curves. The superclass “Painter” has methods of sequential filling of the resulting vector with



unique values of pixels at the positions of intersection given image and fractal curve. The following sub-classes were developed:

1. “LSystemPainter” class, which processing fractal curves defined in L-system. The class provides “draw\_curve\_l\_system” function that constructs the fractal curves that defined in the L-system. The function consistently reads the rules line, executes the corresponding commands of the “worm” class and supplements the resulting vector of values (Algorithm 4).

```

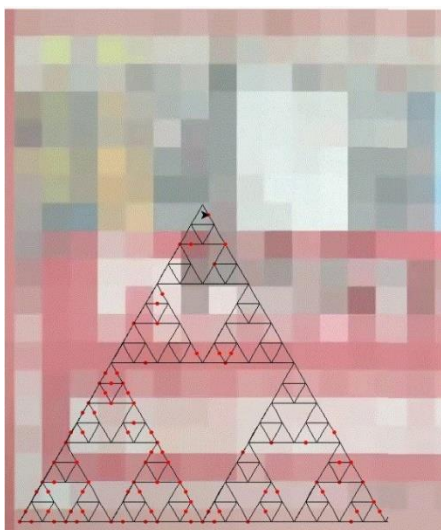
1. def draw_l_system(self,instructions, angle, distance):
2.     for cmd in instructions:
3.         if cmd == 'F':
4.             self.worm.forward(distance)
5.             self.append_unique_values()
6.         elif cmd == '+':
7.             self.worm.right(angle)
8.             self.append_unique_values()
9.         elif cmd == '-':
10.            self.worm.left(angle)
11.            self.append_unique_values()

```

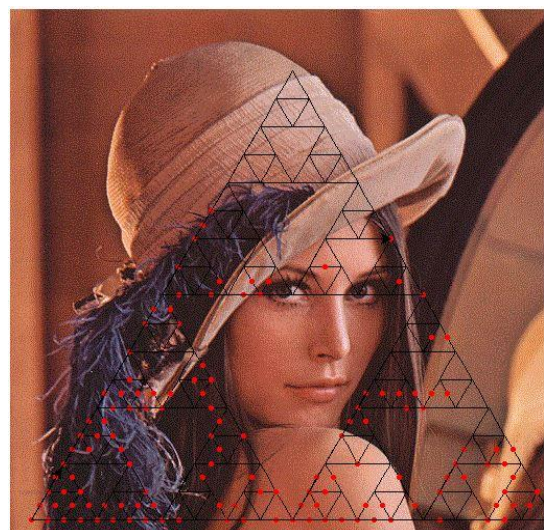
**Algorithm 4** – Function for construction fractal curve represented in L-system

2. Separate classes for each fractal curve that defined as a recursive function (for example, TreePainter). Each class contains its own function for constructing a fractal curve by "moving" an instance of worm class. After each change of "worm" position, the resulting vector of values is supplemented if necessary.

Visualization of the algorithm that extracts source vector of values with the "Sierpinski Triangle" fractal curve is shown in Figure 15. Positions, where pixel values have been appended into the resulting vector, are marked with red color. The numbers in brackets indicate size of extracted source vector.



a) Visualization on smooth image (80)



b) Visualization on contrasting image with monotonous areas (134)

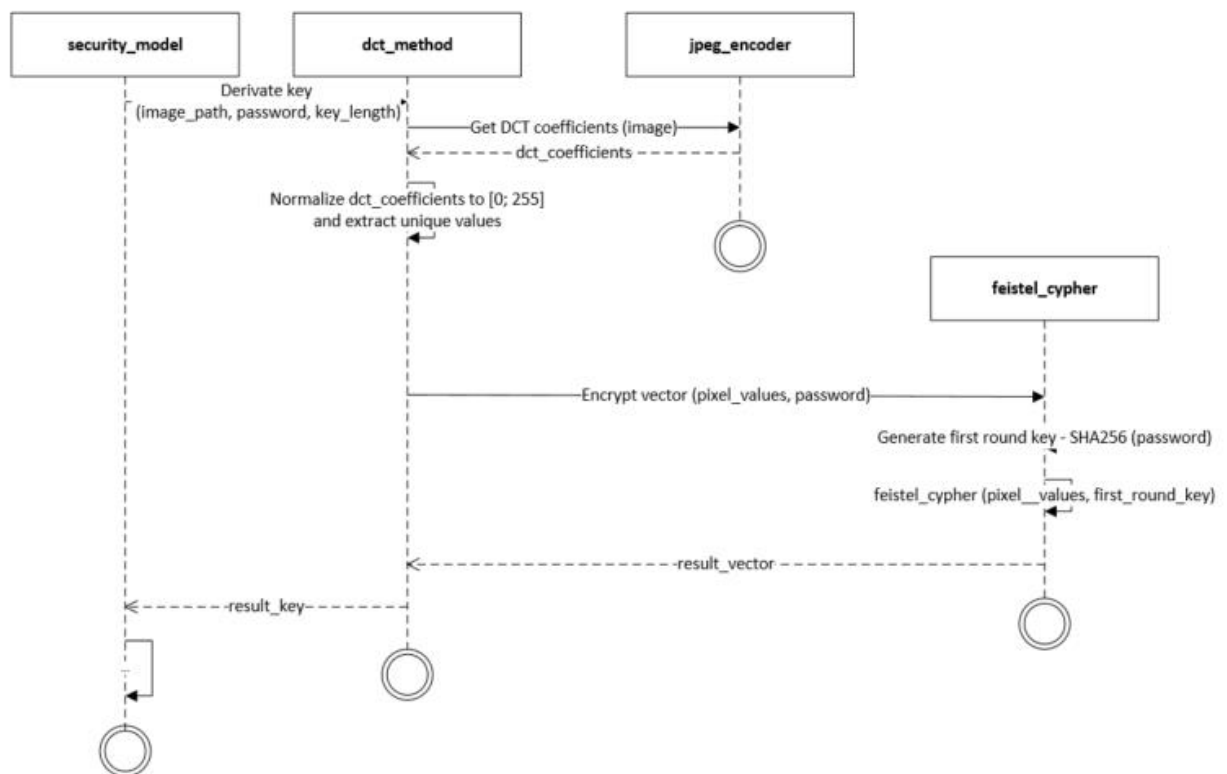
**Figure 15** – Visualization of the work of the algorithm of extraction of the initial vector of values using fractal curves

As can be seen from the figure, the length of the output sequence directly depends on the contrast of the image. The results of deeper examination of the dependence of the length of the obtained value vector on the image type and size will be presented in the next chapter.

After receiving the initial vector of values, the method of the “feistel\_cypher” class is called, which implements the sequential execution of Feistel network rounds according to the description from the third chapter. The source round key is generated from the password using the “hashlib” library method.

### 4.3. Implementation of the algorithm of image and password based key derivation using coefficients of DCT

The implementation of the method to derivate encryption key are based both on image and password using DCT coefficients is provided by “dct\_method” class. The sequence diagram for generation of the encryption key using this method is shown in Figure 16.



**Figure 16** - Sequence Diagram of implemented image and password based key derivation algorithm using DCT coefficients

The most significant part of the implementation of this method is the calculation of DCT coefficients in the “get\_coeff” function of the “jpeg\_encoder” class. The algorithm consistently converts blocks of 8\*8 pixels into a matrix of frequency coefficients 8\*8 using the function “forward\_dct”. Normalization of coefficients allows reducing the length of the source vector significantly. The result of the intermediate stage is a vector of values from the interval [0; 255]. The results of the examination of the dependence of the length of the obtained value vector on the type and size of the image will be presented in the next chapter.

After receiving the initial vector of values, the method of the “feistel\_cypher” class is called, which implements the sequential execution of Feistel network rounds.

#### 4.4. Comparative analysis of statistical properties of implemented algorithms for image and password based key derivation

As noted earlier, the properties of the output sequences generated in the developed algorithms depend on the properties of the original images. It was assumed that properties such as image size and contrast have the greatest influence. With this, the examination of the dependence between image size and contrast on statistical properties of generated sequences of both developed algorithms has been performed.

A sample of 184 images in JPEG format was prepared to examine statistical properties of generated sequences. All images were divided into 9 classes depending on their contrast and size. In terms of contrast, the following groups were selected:

- $Image_1$  – smooth images with smooth color transitions;
- $Image_2$  – contrasting images with monotonous areas;
- $Image_3$  – contrasting mottled images.

The following classes were selected in terms of size:

- Small size (from 225\*225 to 512\*512);
- Medium size (from 512\*512 to 1100\*1100);
- Large size (from 1100\*1100).

Examples of images of each contrast group are shown in Figure 17.



**Figure 17** – Examples of images from the test sample

The first stage of comparison is the comparison of lengths of source vectors of values. Table 2 shows the average lengths of source value vectors for each group of images for each method. Value "A1" corresponds to the algorithm that uses fractal curves, "A2" - to the algorithm that uses DCT coefficients.

Group \ Size and algorithm	Small size		Medium size		Large size	
	A1	A2	A1	A2	A1	A2
I <sub>1</sub>	83	223	127	237	152	245
I <sub>2</sub>	104	236	152	239	172	247
I <sub>3</sub>	122	237	151	231	225	252

**Table 2** – Average lengths of source value vectors

As can be seen from the table, the greatest influence of parameters of the initial image on the length of produced sequences is observed in the algorithm A1 (algorithm that uses fractal curves). The size and contrast of the image directly affect the length. This dependence can be critical while using the system - for example, the average sequence length for small smooth images with smooth color transitions (I<sub>1</sub>) is 83 bytes (249 bits) - with such a sequence length generation of encryption key for AES algorithm is impossible (key length in AES is 256 bits). In the A2 algorithm (the algorithm using DCT coefficients) influence is also present, but it is not critical. Thus, the developed algorithm that uses fractal curves has a critical usage limitation: strong influence of the size and contrast type of source images on the length of the output sequences.

To analyze the statistical properties of output sequences, NIST statistical tests were used [4]. At present, these tests best meet the needs of all interested parties and are the most reliable methods for testing the statistical properties of pseudo-random sequences. The set of statistical tests offers 15 different statistical tests. The following tests have been selected for this work:

1. **The Frequency Mono-bit Test** - this test determines if the frequency of occurrence of 0 and 1 in the whole sequence is approximately equal in the whole sequence. If the probability for the sequence is  $p < 0.01$ , then the sequence is considered not to be truly random and the rest of the tests are not performed in this case.

2. **The Approximate Entropy Test** is based on calculating the frequencies of all possible overlapping patterns in the sequence under study. In an absolutely random sequence, each of  $2^n$  patterns of length  $n$  should appear with approximately the same probability. To pass the test, the probability must be  $p \geq 0.01$ .

3. **Non-overlapping Template Matching Test** - this test calculates the number of finds of templates previously found in the sequence.

4. **The Random Excursions Test** is based on calculating the number of cycles with a given number of hits during an arbitrary traversal of the cumulative sum (the sum starts with partial sums after the sequence (0; 1)).

To perform statistical tests, program generated sequences for each of the images in the test sample by both of developed methods. The length of the sequence submitted to the generator input should be at least  $10^6$  bits. Therefore, 1000 sequences were generated for each image and

the result of concatenation of these sequences was submitted to the input of the test. Algorithm 5 shows pseudo code for generating test sequences for a particular image.

```

Input: image (img)
Output: generated sequences of bits (sequence_A1, sequence_A2)
1. Initialize sequence sequence_A1 = [], sequence_A2 = []
2. for i in (1, 1000) do
3.   Initialize pseudo-random string (psswd)
4.   Generate 1024-bit key from image img and password psswd by algorithm
   A1 (key_A1)
5.   Append key_A1 to sequence_A1
6.   Generate 1024-bit key from image img and password psswd by algorithm
   A2 (key_A2)
7.   Append key_A2 to sequence_A2
8. end for
9. return generated sequences sequence_A1, sequence_A2

```

**Algorithm 5 – Pseudo code for generating test sequences for image**

The tests were performed by executing program code in Python language using the ready-to-use library “sp800\_22\_tests” [22]. The results of testing the sequences produced by executing the algorithm that uses fractal curves (A1) are given in Table 3. Table 4 shows the results of testing the sequences produced by executing the algorithm that uses DCT coefficients (A2). The values in the cells of table indicate the percentage of sequences that passed the respective NIST test for corresponding image group and algorithm.

Group	Size	The Frequency Mono-bit Test	Approximate Entropy Test	Non-overlapping Template Matching Test	Random Excursions Test
I <sub>1</sub>	Small	83.3 %	66.6 %	83.3 %	66.6 %
	Medium	91.3 %	91.3 %	82.6 %	78.3 %
	Large	95 %	95 %	85 %	90 %
I <sub>2</sub>	Small	84.2 %	84.2 %	89.4 %	68 %
	Medium	90.4 %	95.2 %	85.7 %	85.7 %
	Large	95.7 %	91.3 %	86.9 %	91.3 %
I <sub>3</sub>	Small	95 %	75 %	75 %	70 %
	Medium	73.9 %	95.6 %	86.9 %	73.9 %
	Large	94.1 %	94.1 %	82.4 %	82.4 %
All:		88.8 %	87.3 %	83.7 %	86.7 %

**Table 3 – Test results of statistical tests of sequences generated by the algorithm of image and password based key derivation using fractal curves (A1)**

Group	Size	The Frequency Mono-bit Test	Approximate Entropy Test	Non-overlapping Template Matching Test	Random Excursions Test
I <sub>1</sub>	Small	88.9 %	94.4 %	61.1%	66.6 %
	Medium	91.3 %	86.9 %	65.2 %	82.6 %
	Large	95 %	95 %	85 %	85 %
I <sub>2</sub>	Small	89.4 %	84.2 %	89.4 %	68 %
	Medium	95.2 %	95.2 %	90.4 %	61.9 %
	Large	95.7 %	91.3 %	86.9 %	82.6 %

Group	Size	The Frequency Mono-bit Test	Approximate Entropy Test	Non-overlapping Template Matching Test	Random Excursions Test
I <sub>3</sub>	Small	95 %	85 %	85 %	65 %
	Medium	73.9 %	86.9 %	86.9 %	86.9 %
	Large	88.2 %	94.1 %	94.1 %	88.2 %
All:		90.1 %	88.9 %	81.1 %	74.4 %

**Table 4** - Test results of statistical tests of sequences generated by the algorithm of image and password based key derivation using DCT coefficients (A2)

As can be seen from the tables, the best results were obtained by The Frequency Mono-bit Test, the worst results were obtained by the Random Excursions Test. The average values for each of the tests are approximately the same. The sequences obtained as a result of the algorithm using fractal curves (A2) on contrast images (groups I<sub>1</sub> and I<sub>2</sub>) with a larger size managed the tests more successfully than the algorithm using DCT coefficients (A1). On images of small size, in opposite, the algorithm A2 showed better results. Table 5 shows the percentage of sequences that passed all tests by image groups.

Size and algorithm Group	Small size		Medium size		Large size	
	A1	A2	A1	A2	A1	A2
I <sub>1</sub>	38.9 %	55.5 %	73.9 %	69.5 %	73.9 %	78.2 %
I <sub>2</sub>	78.9 %	78.9 %	76.2 %	61.9 %	86.9 %	78.2 %
I <sub>3</sub>	70 %	65 %	73.9 %	73.9 %	82.3 %	88.2 %

**Table 5** – Percentage of sequences that passed all statistical tests

According to the data in the table, as expected, there is a clear correlation between image parameters and statistical properties of the output sequences. Critically low results (38.9% and 55.5%) showed smooth monotonous images of small size. In the case of the algorithm with the use of fractal curves (A1), this is explained by the low variability of pixel values in images of this type translated into grayscale. As it was noticed in the present chapter earlier, in this case the length of the source vector is also not large. In case of the use of DCT coefficients (A2), the low percentage of sequences that passed the tests is explained by the low entropy of such images, respectively - most of the DCT coefficients will be negligibly small, which will lead to reduction when the values are normalized.

Contrast images of large size showed the best results. For the algorithm using fractal curves, the highest percentage of sequences that passed statistical tests belongs to the sequences generated from images of the second group - contrast images with monotonous areas. Presumably, this result was obtained due to the fact that in such images there are areas of smooth and sharp color transition, which affects the spread of values of the original value vector. In colorful images, on the contrary, often the tone of the image is the same in the majority of the image, which leads to close values of pixels. Increase of statistical properties of output sequences

of the given algorithm can be achieved by dynamic calculation of length of a step at construction of a fractal curve in the algorithm of calculation of an initial vector of values, and also by the increasing number in rounds of a Feistel network at the second stage of algorithm.

For the algorithm using DCT coefficients, the highest percentage of sequences that passed statistical tests belongs to the sequences generated on the basis of images of the third group - contrast blurred images. This fact can be explained by the definition of DCT - in colorful images the entropy is high enough. Improvement of statistical properties of the output sequences of this algorithm can be achieved by increasing the Feistel network rounds in the second stage of the algorithm.

The results of statistical tests have shown that the developed methods of generating encryption keys based on image and password have relatively good statistical properties and can be used to generate encryption keys for further data encryption. In combination with a secure approach to key transmission, the method is capable of ensuring a high level of data confidentiality.

#### **4.5. Summary**

This chapter describes the implementation of the methods to derivate an encryption key based on both image and password: method that uses fractal curves and method that uses DCT coefficients. NIST statistical tests were used to test statistical properties of the output sequences. The test sample consists of sequences generated from a predefined set of images. This set consists of images grouped into 9 classes by contrast type and size. Test results enable to draw the following conclusions:

1. Statistical properties of the output sequences critically depend on the size of the input image for both algorithms: the output sequences are considered random in less than half of cases for small-size images. The sequences derived from large-size images show the best results.

2. Statistical properties of output sequences depend on image types for both algorithms. For the algorithm that uses fractal curves, contrast images with areas of smooth color transition have the best statistical properties (86.9% of sequences that passed all statistical tests). For the algorithm that uses DCT coefficients the best statistical properties are obtained for contrast blur images (88.9 % of sequences that passed all statistical tests).

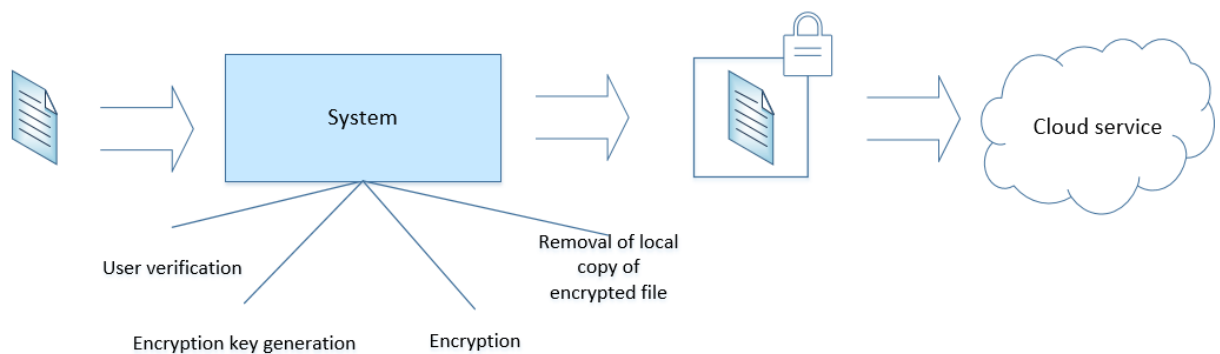
3. Limited length of the produced key (1024 bits) is considered as a weakness of the algorithms.

The advantage of the algorithms is the high resistance to man-in-the-middle attacks: two input values must be used to generate an encryption key. The algorithm's crypto stability is determined by the crypto stability of the Feistel cipher. Thus, the developed algorithms can be used as methods for generating encryption keys in a complex cryptographic system using an encryption algorithm that requires an encryption key with a length up to 1024 bits.

## Chapter 5: Implementation of a developed system for ensuring data security while using cloud storage services

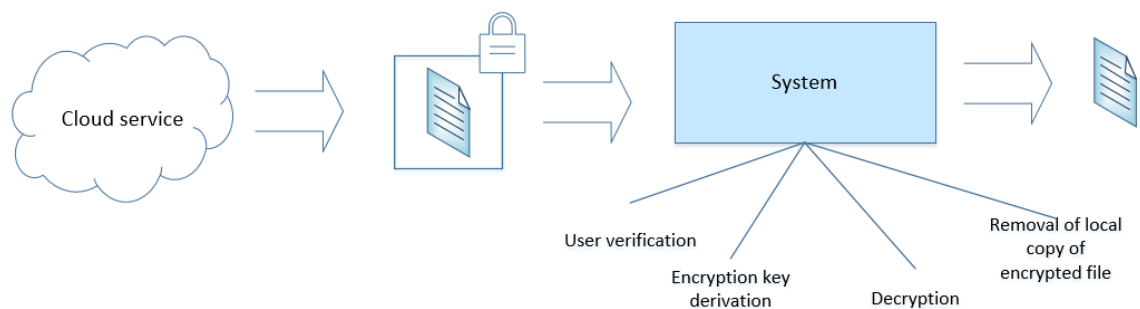
The system for providing data security while using cloud storage service Yandex Disk [1] was implemented in the Python 3.7 programming language using the PyQt5 library to implement a graphical interface of the system. In general, the developed system can be used to perform the following use cases:

1. Uploading a file to cloud storage with intermediate encryption. The user selects the file on the local machine, the encryption algorithm, and enters the encryption key. The system encrypts the specified file and downloads an encrypted copy of the file to the user's cloud storage (Figure 18).



**Figure 18** – Uploading a file to a cloud storage

2. Downloading a file from cloud storage with intermediate decryption. The user selects the file in the cloud storage, an encryption algorithm and an encryption key. The system downloads the encrypted copy of the file to the local machine, decrypts the specified file and saves the decrypted copy of the file on the user's local machine (Figure 19).



**Figure 19** – Downloading a file from a cloud storage


### 5.1. Interaction with REST API of Yandex Disk

As a cloud storage with which the system is synchronized, the cloud storage Yandex Disk of Russian origin was selected [1]. Interaction with cloud service is performed via REST API with the help of “YaDisk” library [23].



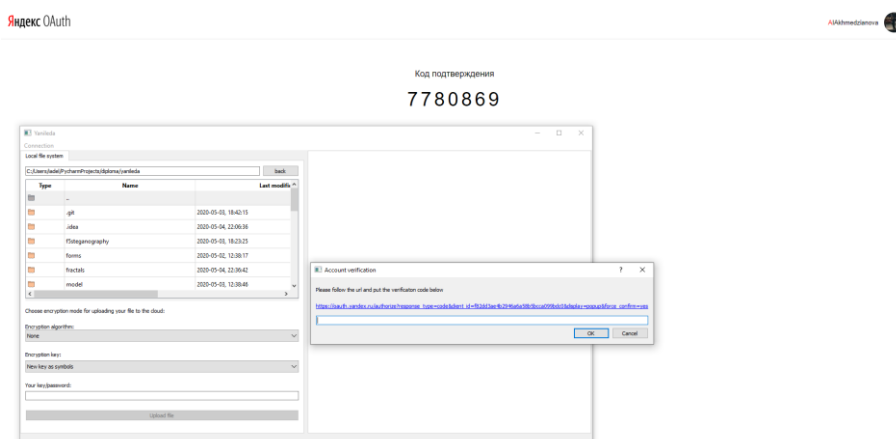
Before the beginning of works the application has been registered on an OAuth-server - this procedure is necessary in order to get access to the data of the user Yandex.Disk by using the authorization protocol OAuth 2.0 (Figure 20).

Список зарегистрированных приложений

Приложение	Права
 YaNLД	<b>API Яндекс.Паспорта</b> Доступ к логину, имени и фамилии, полу Доступ к адресу электронной почты Доступ к портрету пользователя  <b>Яндекс.Диск REST API</b> Запись в любом месте на Диске Чтение всего Диска Доступ к информации о Диске Доступ к папке приложения на Диске  <b>Яндекс.Диск WebDAV API</b> Доступ к Яндекс.Диску для приложений

**Figure 20** – Application registration on OAuth server of Yandex Disk

Each time the system is started, system generates a link using the “AuthorizedYadiskModel.get\_code\_url()” method. When a user clicks on this link, he gets to the window that permits system access to user data. After that a one-time password is generated, and using the “AuthorizedYadiskModel.check\_token()” function the user is authorized in Yandex Disk (Figure 21).



**Figure 21** – Authentication process

The REST API of Yandex Disk allows viewing information about authorized user or public folder files, downloading and uploading files by using HTTP requests, for example:

1. “AuthorizedYadiskModel.listdir (path)” method allows to get the list of folders and files by the specified path;
2. “AuthorizedYadiskModel.upload (from\_path, to\_path, overwrite=True)” method uploads a file located on the local machine along the path “from\_path” to the cloud storage folder “to\_path”;
3. “AuthorizedYadiskModel.download (from\_path, to\_path)” method performs download of a file located on the path “from\_path” in cloud storage to the folder “to\_path” on user's local machine.

In order to handle with errors during interaction with Yandex Disk API, Decorator structural pattern is used. Function “yadisk\_error\_handle (fn)” decorates all functions that requires connection to Yandex Disk via HTTP requests by adding detection of errors. Algorithm 6 presents decorator and example of it’s use.

```
1. def yadisk_error_handle(fn):
2.     def wrapped(*args):
3.         try:
4.             result = fn(*args)
5.         except YaDiskError:
6.             logger.info("Connection problems")
7.             result = Result.failed("Connection problems to Yadisk")
8.         return result
9.     return wrapped
10.
11. @utils.yadisk_error_handle
12. def download(self, from_path, to_path):
13.     logger.info("download {} to {}".format(from_path, to_path))
14.     self.disk.download(from_path, to_path)
15.     return Result.success()
```

**Algorithm 6** – Using Decorator structural pattern for handling with connection errors

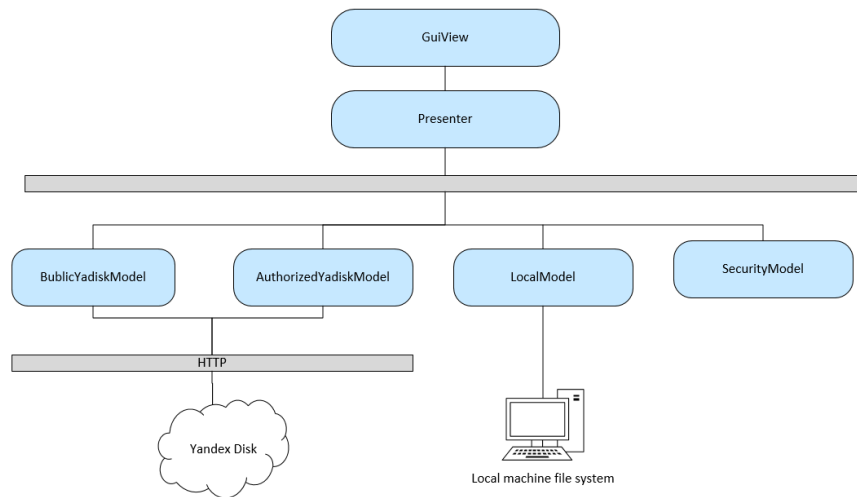
## 5.2. Description of system’s modules

The developed system has a modular structure and has the following modules (Figure 22):

- user interface management module – *GuiView*;
- control module - *Presenter*;
- module of work with authorized users - *AuthorizedYadiskModel*;
- module of work with public folders - *PublicYadiskModel*;
- module of work with file system on the local machine - *LocalModel*;
- data protection module - *SecurityModel* (this module was described earlier).

The **GuiView** user interface control module provides a graphical interface display and user actions processing. A description of the graphical interfaces of the system will be provided later. The module handles the event initiated by a user action (click a button, select an item, change the field value) and calls the Presenter module methods.

The **Presenter** module is the main control module of the system. Class methods sequentially perform a set of actions required to process a user request, calling methods of other classes. The main methods of the Presenter class are as follows:



**Figure 22** – Structure of the developed system

1. The method of getting information about a folder in the cloud storage *get\_yadisk\_listdir*, which gets the path to the folder in the cloud storage and performs the following steps:

- calls the *get\_listdir* method of an instance of the *AuthorizedYadiskModel* class, which in its turn sends an HTTP request to the REST API Yandex Disk and receives a list of files and folders on the specified path. A similar method is also implemented for handling public folders - in this case the call is made to the *PublicYadiskModel* class;
- calls the *show\_yadisk\_listdir* method of *GuiView* class, which updates the graphical window in *MainWindow* user interface according to the obtained list.

2. The method of getting information about a folder on the user's local machine *get\_local\_listdir* gets the path to a folder on the user's local machine and performs the following steps:

- the *get\_listdir* method calls the *LocalModel* class, which addresses the operating system through the “os” built-in module and gets the list of files and folders at the specified path;
- calls the *show\_local\_listdir* method of the *GuiView* class, which updates the graphical window in the *MainWindow* user interface according to the received list.

3. The method of encrypting and uploading a file to the cloud storage *upload\_to\_auth\_yadisk* gets the input path to the file on the local machine *from\_path*, the desired path for uploading the file in cloud storage *to\_path*, and a set of encryption parameters *crypto\_data*. The method performs the following actions:

- creates a new thread and calls the *show\_progress\_dialog* method of the *GuiView* class to display the waiting window;
- calls the *encrypt* method of *SecurityModel* class, which generates the encryption key, implements encryption algorithms and returns the path to the encrypted file on the user's local machine;

- calls the upload method of the `AuthorizedYadiskModel` class, which in turn sends an HTTP request to the REST API Yandex Disk uploads the specified file to the user's cloud storage;
- calls the `delete_file` method of the `LocalModel` class, which deletes the created local copy of the encrypted file on the user's machine;
- closes the stream with the waiting window created in the first step, referring to the `GuiView` class.

A similar method is implemented for working with public links - in this case, the **PublicYadiskModel** class is used. In addition, the class implements methods for downloading and decrypting files using a similar scheme.

The module of work with authorized users **AuthorizedYadiskModel** is implemented to interact with the REST API of Yandex Disk by sending HTTP requests. Methods of the class process requests from the Presenter class, add parameters such as `APP_ID` and `APP_SECRET` password to authorize the application on the OAuth server, and process the responses received. A similar **PublicYadiskModel** class is implemented for working with public links.

The module of work with the file system on the local machine of the *LocalModel* user uses functions of the built-in os library for work with the operating system. Methods of the class allow to get information about the file system of the local machine. For example, the `get_listdir` method gets the input path to a file or folder, and outputs a list of the contents of the file or folder with the corresponding metadata (type, size, date of last modification). All modules also log events using the built-in logging library - all logs are saved to a text file in the root folder of the system.

The sequence diagram for the "Uploading file to cloud storage with intermediate encryption" usage scenario is shown in Figure 23.

### 5.3. Description of system user interfaces

System user interfaces are created using the Qt Creator environment for the appearance constructor. Each user interface has its own class that describes event handler functions in the corresponding user interface. The following classes of user interfaces are implemented in the system:

1. **MainWindow** class describes the main working window and handles requests to establish a new connection to the Yandex Disk cloud storage.
2. **FileSystem** class describes a window that is displayed for each explorer (local machine or cloud storage) with a set of fields and buttons to provide encryption functions (Figure 24).

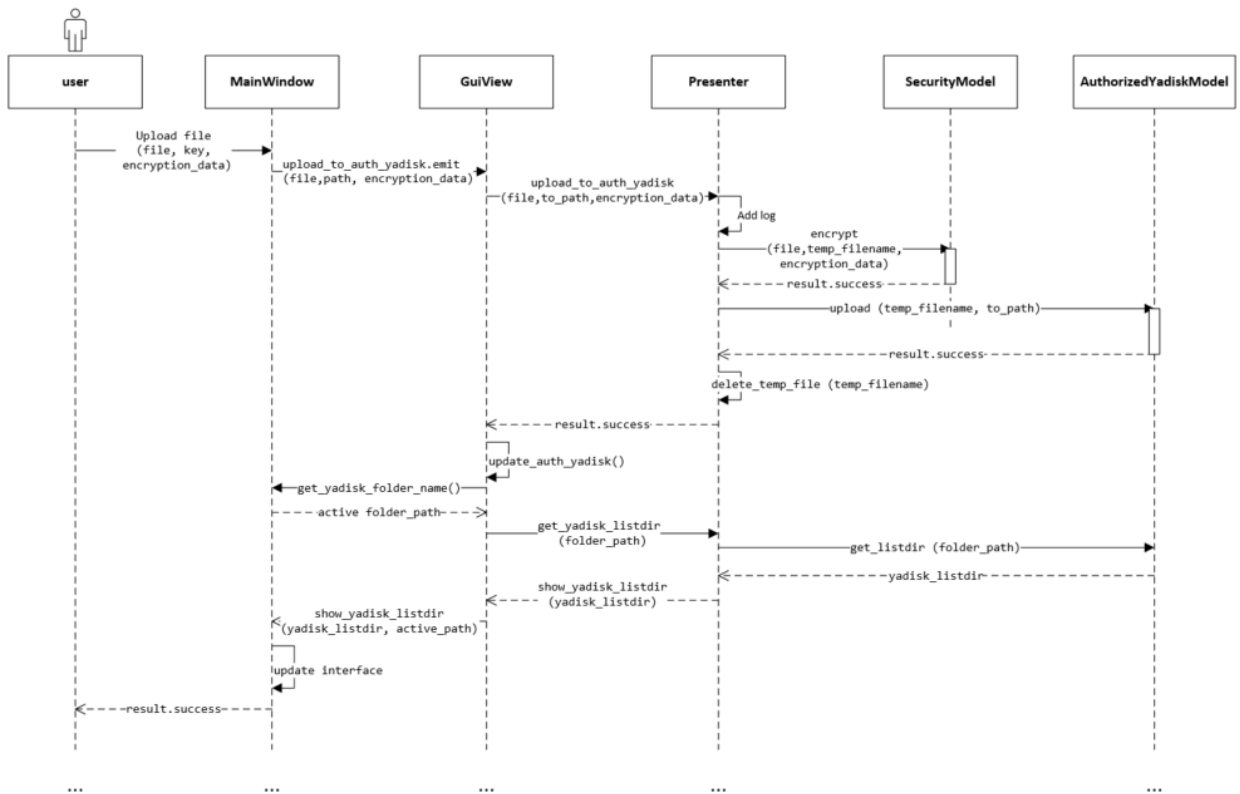


Figure 23 – Sequence Diagram for the "Uploading file to cloud storage with intermediate encryption" usage scenario

3. **ConnectionDialog** class describes the window that opens when an authorized user requests a new connection to the cloud storage.
4. **OpenLinkDialog** class describes the window that opens when requesting a new connection to a public folder in the cloud storage.

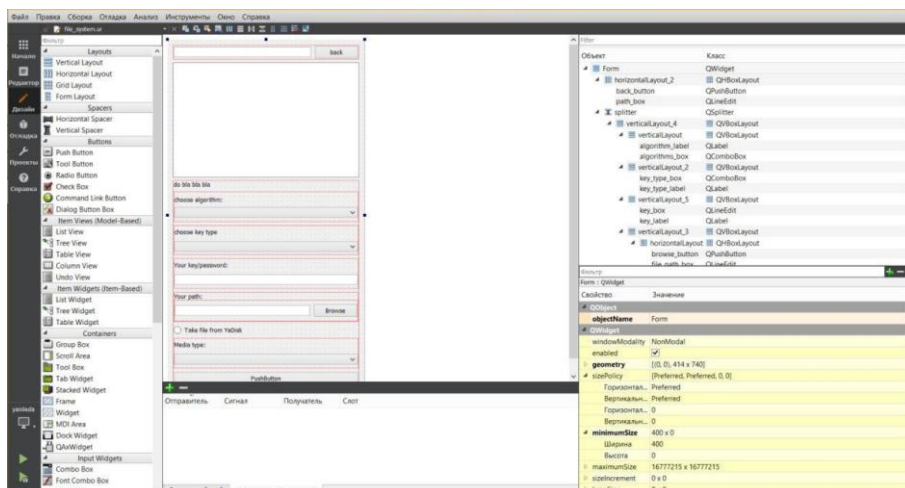
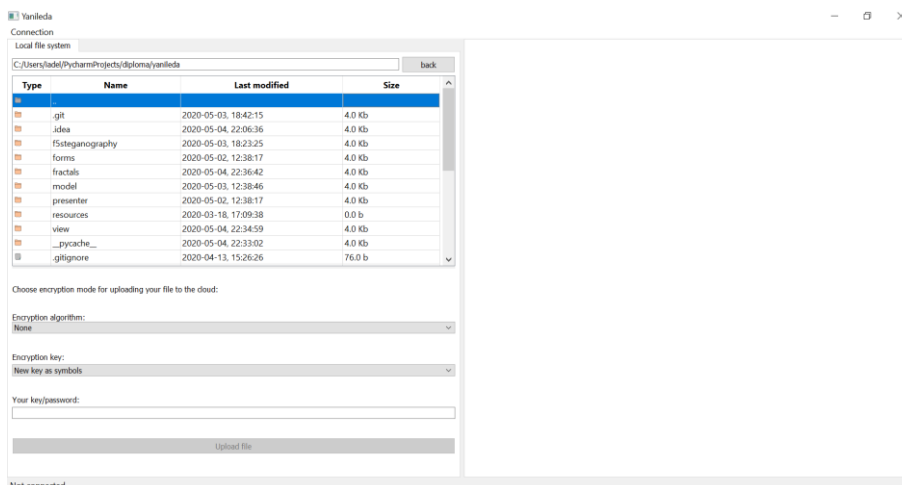


Figure 24 – "FileSystem" user interface designer

After opening the system the main window ("MainWindow") is displayed. The working window is divided into two parts: on the left, an explorer of the user's local machine is displayed, and on the right, an explorer of the cloud storage is displayed. The full path of the current folder

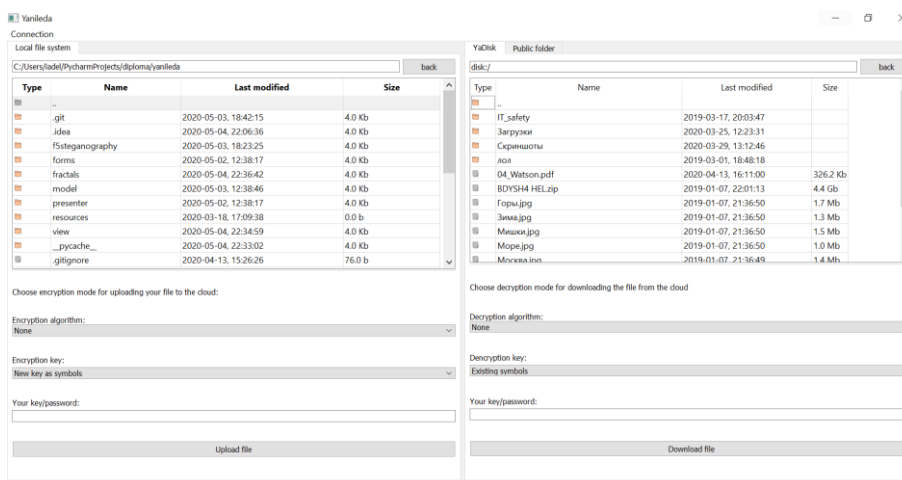
is displayed in the upper left part of the window. Then an explorer corresponding to the specified path is displayed. The symbols indicate the type of the item: folder or file (the icon is set depending on the item type - "dir" or "file") (Figure 25).



**Figure 25** – Main window of the system

In order to create a new connection to the cloud storage, a user should click on the "Connect" button and choose the type of connection - an authorized user or a public folder. If a user connects to a public folder, he/she should insert a link to the public folder. When an authorized user connects to a cloud storage, a link will be generated in the system. When passing to it, the user should give the application access to his or her data, after which a one-time password will be generated. After entering the one-time password in the system and successful authorization, a list of user folders and files will appear on the right side of the main window (Figure 26). The user has the option to set the folder as the default image folder (to store images that are the basis for generating encryption keys) - to do this, right-click on the folder and select "Set as media folder".

There is a possibility to create several connections at once (up to 5) - for each connection a new tab will be opened. The open tab is active - all files are uploaded to it.



**Figure 26** - System main window with established connections

To upload a file to a disk, the user must:

1. Select an element in the local explorer (in the left part of the window).
2. Select an encryption algorithm or "None" to upload the file in its original form (algorithms such as 3DES, AES are available).
3. Select the key type: character string, binary file, image and password based key derivation using fractal curves, image and password based key derivation using DCT coefficients.
4. Specify the key (when generating the key, fields are filled automatically):
  - for "Symbolic string" type, a string is entered into the field, which will be the encryption key;
  - for "Binary file" type, pressing the "Browse" button opens an explorer where user can select the binary file containing the encryption key;
  - for the types " image and password based key derivation using fractal curves " and " image and password based key derivation using DCT coefficients" it is necessary to enter the password string and the path to the image (absolute path on the local machine or the name of the image while using images from the folder selected in the cloud storage);
5. Click on the "Upload" button - after the encryption process is complete and you upload to the cloud storage, the explorer on the right side of the screen will be updated (Figure 27).

To download a file from the cloud storage you need to do the same on the right side of the screen.

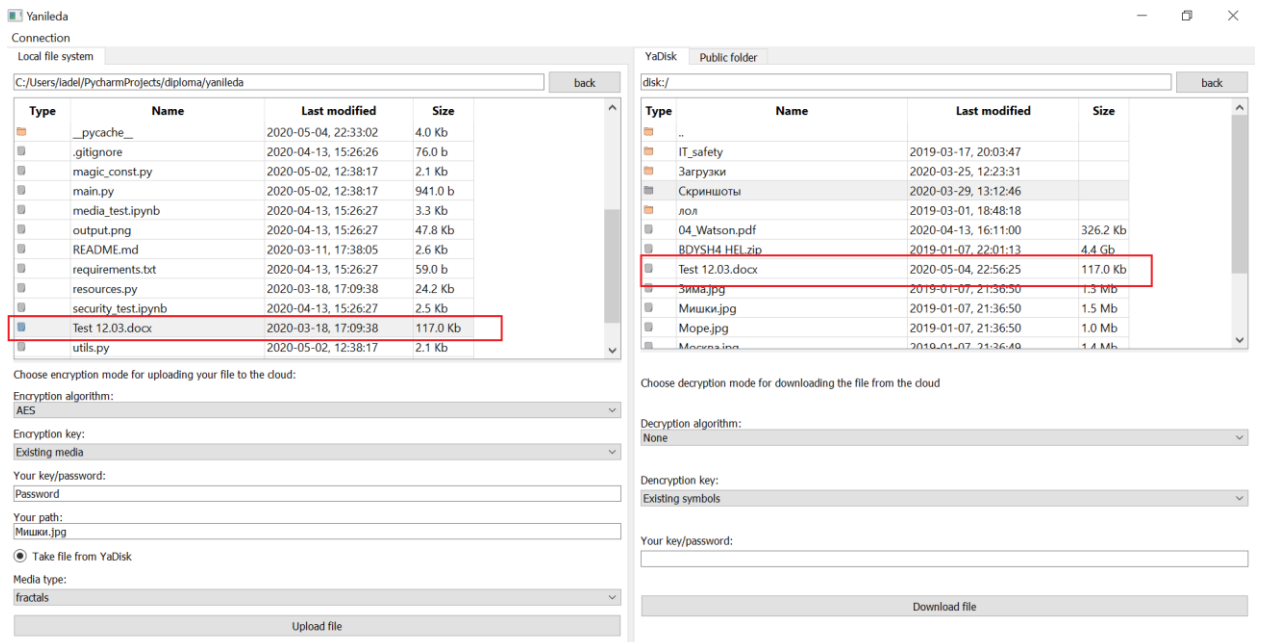


Figure 27 - Uploading a file to cloud storage

## 5.4. Summary

This chapter describes the implementation of the system to provide data security while using cloud storage. The system provides synchronization with cloud storage Yandex Disk via

REST API. The system provides such cryptographic functions as encryption key generation, encryption and decryption. Generation of encryption keys is performed by using the following algorithms:

- generation of the encryption key based on password;
- generation of the encryption key based on both password and image using fractal curves;
- generation of the encryption key based on both password and image using DCT coefficients.

Symmetric encryption algorithms such as AES, 3DES, Feistel cypher are available in the system. The system meets all the requirements proposed at the early stage of the development process and can be used to ensure data security while using cloud storage for corporate purposes, e.g. in fabless-companies. Now the system is under testing in a fabless-company.

There is a potential to improve the functionality of the system: adding the ability to display a preview of files from cloud storage, implementing of functions to ensure the availability and integrity of files and developing cross-platform application.



## Chapter 6: Summary and Conclusions

### 6.1. Conclusion

The work presented in this thesis aims to increase the level of data security in client-side encryption process while using cloud storage services in theoretical and practical aspects. In order to address this aim, the work proposes the methodology for client-side encryption systematically builds upon the methods to derivate an encryption key on both image and password. The following results were achieved in this thesis:

1. Methods to derivate encryption key based on both image and password were developed, in particular: method using fractal curves and method using DCT coefficients. The developed methods consist of two stages: extraction of the value vector from the image and encryption of the extracted vector with the password using the Feistel cipher. The algorithm's crypto stability is determined by the crypto stability of the second stage - Feistel cipher. The main advantage of the proposed algorithms is that two parameters are employed to generate an encryption key. Therefore, proposed methods have high level of protection from Man-in-the-middle attacks. Limited possible length of the generated sequences (1024 bits) is considered as a main drawback of the developed methods.

2. Developed methods are implemented in Python 3.7. The comparative analysis of statistical properties of the sequences generated from different types of images using NIST statistical tests is conducted. The test results allow to make the following conclusions:

- statistical properties of the generated sequences depend on contrast and size of an input image;
- size of an input image drastically influences on statistical properties of sequences generated by the method using fractal curves: only 38.9 % of sequences passed all tests for small images (up to 512x512 pixels);
- sequences that generated from large and contrast images show the best statistical properties for both algorithms. Therefore, the methods can be used as algorithms to generate encryption keys in crypto systems while using these images.

3. The software that provides data security while using cloud storage services was developed and implemented. This software integrated with Yandex Disk cloud storage service via REST API. System provides such functions as uploading files of any type to an authorized user's cloud storage or a public folder with intermediate encryption or downloading files with intermediate decryption. System provides the following types of the cryptographic functions:

- generating an encryption key by password-based key derivation algorithm (PBKDF-2), image and password based key derivation algorithms (using fractal curves and using DCT coefficients);
- symmetric encryption and decryption algorithms (3-DES, AES).

The developed software was handed over to a fables-company for testing and to determine the areas of further system improvement. Test results confirmed the practical importance of the system. Thus, the developed software is ready be used for corporate purposes to ensure data security while using cloud storages.

## **6.2. Future directions**

Test results enable to draw the following directions for further work:

1. Improvement of the developed methods of encryption key generation in order to decrease the dependency between the statistical properties of output sequences and parameters of input images. For example, the method that uses fractal curves can be improved by dynamic calculation of the step length while generating the initial vector of values by drawing fractal curve. Both methods can be improved by increasing the number of rounds in the Feistel network.

2. Improvement of the developed software in a manner that it is capable to:

- check the integrity of files stored on the cloud storage in order to detect unauthorized data modification;
- log actions for files stored on the cloud in order to detect the facts of attempts to unauthorized access to the protected data;
- control the versions of the files to be downloaded and notifying the user that the current version of the file may not be the last.

## Bibliography

1. Yandex Disk. URL: <https://disk.yandex.ru/>. Accessed: 2020-05-05.
2. What is a Fabless Company (2019). URL: <https://anysilicon.com/what-is-a-fabless-company/> . Accessed: 2020-05-05.
3. C.Dotson. Practical Cloud Security: A Guide for Secure Design and Deployment. O'Reilly (2019), 196 p.
4. NIST SP 800-22rev1a, A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications (2010). URL: <https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software> . Accessed: 2020-05-05.
5. M.Borgmann, M.Waidner. On the Security of Cloud Storage Services. Fraunhofer Verlag (2012), pp.21-53.
6. BoxCryptor – Encryption Software to Secure Cloud Files. URL: <https://www.boxcryptor.com/en/> . Accessed: 2020-05-05.
7. MEGA - User-encrypted cloud services. URL: <https://mega.nz/>. Accessed: 2020-05-05.
8. TrueCrypt – utility for on-the-fly encryption. URL: <http://truecrypt.sourceforge.net/>. Accessed: 2020-05-05.
9. The Decree of the Government of the Russian Federation from 16.11.2015 № 1236 "On the establishment of a prohibition on the admission of software originating from foreign countries, for the purposes of procurement to ensure state and municipal needs. (in Russian) URL: [http://www.consultant.ru/document/cons\\_doc\\_LAW\\_189116/](http://www.consultant.ru/document/cons_doc_LAW_189116/) . Accessed: 2020-05-05.
10. W.Stallings. Cryptography and Network Security: Principles and Practice, 7<sup>th</sup> Edition. Pearson (2017), pp.85-202.
11. The Android BitCoin vulnerability explained. URL: <https://dataprotectioncenter.com/general/the-android-bitcoin-vulnerability-explained/>. Accessed: 2020-05-05.
12. S. Zapechniko. Encyclopedia of Theoretical and Applied Cryptography. URL: [http://cryptowiki.net/index.php?title=Main\\_Page](http://cryptowiki.net/index.php?title=Main_Page). Accessed: 2020-05-05.
13. M.Turan, E.Barker, W.Burr. Computer Security. Resomendation for Password-Based Key Derivation. NIST Special Publication (2010)
14. P 50.1.111–2016 “Information technology. Cryptographic protection of information. Password protection of key information” (in Russian)
15. A.Westfeld. F5 – A Steganographic Algorithm. High Capacity Despite Better Steganalysis. S. Moskowitz (Ed.): IH 2001, LNCS 2137, pp. 289–302, 2001.
16. M.A.Druchenko. Algorithm of steganographic data hiding in JPEG image based on convolution function (in Russian). Proceedings of VSU №3 (2018).

17. M.B.Younes, A.Jantan. An image encryption approach using a combination of permutation technology (2008).
18. Y.Dodis, L.Reyzin, A.Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. Advances in Cryptology - EUROCRYPT (2004), pp.523-540
19. J.Feder. Fractals. Springer (1988), pp.6-36.
20. R.M.Kronover. Fractals and Chaos in Dynamic Systems. Fundamentals of Theory (in Russian). Moscow (2006), 352 c.
21. B.Schneier. Applied Cryptography. Protocols, Algorithms and Source Code in C. Willey, 20<sup>th</sup> Ed. (2015), pp.331-354.
22. A python implementation of the SP800-22 Rev 1a PRNG test suite. URL: [https://github.com/dj-on-github/sp800\\_22\\_tests](https://github.com/dj-on-github/sp800_22_tests). Accessed: 2020-05-05.
23. YaDisk - Yandex.Disk REST API client library. URL: <https://yadisk.readthedocs.io/en/latest/index.html>. Accessed: 2020-05-05.