International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 48 No. 2 January 2025, pp. 147-157

The Use Of Functions $f: D_1 \times D_2 \rightarrow \mathbb{R}^2$ For Encrypting And Decrypting Data In A Plane

Azir Jusufi[1] Blinera Zekaj[2] Gazmend Xhaferi[3] Altin Jusufi[4]

[1,3] University of Tetova, Tetovo, North Macedonia

[2]Nettxio, Prishtina, Kosovo

[4]MIG-Skopje, North Macedonia

[1]azir.jusufi@unite.edu.mk; [2]blinera.zekaaj@gmail.com;

[3]gazmend.xhaferi@unite.edu.mk; [4]altinjusufi2@gmail.com

Abstract – The study of cryptosystems is significant, and valuable scientific works are dedicated to them. Cryptosystems are mathematical structures that deal with data encryption and decryption, focusing on confidentiality and authenticity. Cryptography plays a crucial role in meeting these requirements, with numerous researchers proposing various solutions and developing algorithms that have contributed to the security of data confidentiality, integrity, and authentication. However, the issue of securing the internet through modern cryptography has also become increasingly complex. Modern encryption systems rely on complex mathematical algorithms and employ a combination of symmetric and asymmetric key encryption schemes to ensure secure communication. The use of $\mathbf{f: D_1 \times D_2 \rightarrow \mathbb{R}^2}$ for image encryption provides an effective and secure method for protecting the privacy, security, and confidentiality of sensitive data, such as artwork or sketches.

The use of functions$\mathbf{f: D_1 \times D_2 \rightarrow \mathbb{R}^2}$ operating in two-dimensional space encrypts images in a way that makes it difficult for unauthorized individuals to decipher their shape and structure.

The use of encryption functions for images has the potential to be applied in industries such as product design, computer graphics, virtual reality, engineering, and many other fields.

Keywords – Encryption, decryption, functions, commutative composite functions, image, artwork.

1. INTRODUCTION

Cryptography (from the Greek *kriptos* - secret, and *graphos* - writing) is the science that aims to conceal a message and transmit the hidden message to an authorized recipient, who can then revert it to its original state.[1] The transformation of the message, known as plaintext, into a concealed format referred to as ciphertext (or cryptogram) is achieved through the so-called encryption function, which enables the authorized recipient to convert the ciphertext back into plaintext. [1],[14],[15]. The entirety of these elements forms what is known as a cryptographic structure (abbreviated as cryptostructure or cryptosystem)[1], It is understood that the encryption process is carried out using an encryption algorithm.

The key is known to the sender and the authorized recipient, but not to those from whom the plaintext is concealed. The key and the ciphertext must uniquely determine the plaintext. The reconstruction of the plaintext from a given ciphertext is referred to as decryption and is accomplished through the so-called decryption function, which is defined by a decryption algorithm and the key k. For a given key, the decryption function is the inverse of the encryption function. The concepts described above can be summarized as follows:

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 48 No. 2 January 2025, pp. 147-157

Definition 1.1 A cryptosystem is called a quintuplet (*P, C, K, E, D*) , where

- *P*- is a finite family of all plaintexts;
- *C*- is a finite family of all ciphertexts;
- *K*- is a finite family of all possible keys;
- *E* and D are sets of mappings from *P* to *C* and from *C* to *P*, respectively, such that for every çdo $k \in K$, there exists an encryption function $e_k \in E$ and a decryption function $d_k \in D$
- that satisfies $\forall x \in P, d_k(e_k(x)) = x$.

The use of encryption and decryption is as old as communication itself. Before computers, the security of encryption was ensured by exchanging encryption keys between the sender and the recipient, a process that often posed risks.

The first encryption was achieved using the substitution method based on a set of substitution rules. More complex ciphers operate with the help of powerful computer algorithms that reconstruct the bits into digital signals. In order to retrieve the content of the encrypted signal, we need the correct decryption key.[3][4][5]

In cryptography, a key is a variable value or a string of characters applied to an algorithm on a block or string of plaintext or ciphertext.

Cryptosystems are classified into two types: secret key cryptosystems and public key cryptosystems. The secret key cryptosystem is the oldest type of secret notation, where both the sender and the recipient share the same secret key.

In a public key cryptosystem, two keys are used: a public key known to everyone and a private or secret key known only to the recipient of the message. The public key and the secret key are related in such a way that the public key is used solely for encrypting messages, while the secret key is used exclusively for decrypting them. [1][6]

The purpose of this paper is to explore and develop effective methods for using functions $f: D_1 \times D_2 \to \mathbb{R}^2$ for image encryption to ensure the privacy, security, and confidentiality of sensitive data. The aim is to create secure and efficient algorithms for practical applications, enhancing the performance and outcomes of image encryption.

2.The use of functions $f: D_1 \times D_2 \to \mathbb{R}^2$ for data encryption and decryption in a plane

Definition 2.1. Let *L* and *L'* be vector spaces over the same field F with dimensions n and m, respectively. The function $f: L \to L'$ defined by $f(a_1, a_2, .., a_n)=(b_1, b_2, .., b_m)$, for any $(a_1, a_2, .., a_n) \in L$, is called a transformation of the space *L* into the space *L'*

Definition 2.2. Let *L* and *L'* be vector spaces over the same field *F*. For the function

$f: L \to L'$ we say it is linear or a linear transformation if:

a) $f(a + b) = f(a) + f(b)$, for $\forall\ a,b \in L$, and

b) $f(\lambda a) = \lambda f(a)$, $\forall\ a \in L$ and $\forall \lambda \in F$.

Or, if we combine the conditions mentioned above, we obtain:

Definition 2.3. . Let *L* and *L'* be vector spaces over the same field *F*. For the function

$f: L \to L'$ we say it is linear or a linear transformation if

$$f(\alpha a + \beta b) = \alpha f(a) + \beta f(b),\ \forall\ a,b \in L \text{ and } \forall\ \alpha,\beta \in F$$

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 48 No. 2 January 2025, pp. 147-157

Example 2.1. Show that the function $f: \mathbb{R}^2 \to \mathbb{R}^2$ given by $f(x,y)=(x,2x-y)$ for all $\forall x,y \in \mathbb{R}$ is a linear transformation in $\mathbb{R}^2$.

Solution. For all $\forall(x,y),(x_1,y_1) \in \mathbb{R}^2$ and for all $\forall a,b \in \mathbb{R}$, we have:

$$f[a(x,y)+b(x_1,y_1)]= f[ax+bx_1, ay+by_1]=[ax+bx_1, 2(ax+bx_1)-(ay+by_1)]=$$

$$=[ax+bx_1, a(2x-y)+b(2x_1-y_1)]=[ax,\ a(2x-y)+(bx_1, b(2x_1-y_1)]=$$

$$= a(x,2x-y)+b(x_1,2x_1-y_1) = af(x,y)+bf((x_1,y_1).$$

As a result f qualifies as a linear transformation in $\mathbb{R}^2$.

Next, we will examine a special type of transition functions $F: D_f \times D_g \to \mathbb{R}^2$ given by $F(x,y)=(f(x),g(y))$, for all $\forall(x,y) \in D_f \times D_g$, where $D_f, D_g \subseteq \mathbb{R}$. The functions $f: D_f \to \mathbb{R}$ and $g: D_g \to \mathbb{R}$ are bijective functions, thus there exist inverse functions $f^{-1}(x)$ and $g^{-1}(y)$. Therefore, we can form the inverse of F(x.y), which is $F^{-1}(x,y)=(f^{-1}(x),g^{-1}(y))$.

We will use the transition function $F(x,y)=(f(x),g(y))$, as the encryption key for figures in the plane, while the inverse function $F^{-1}(x,y)=(f^{-1}(x),g^{-1}(y))$, of the function $F(x,y)=(f(x),g(y))$ will serve as the decryption key.

Encryption Procedure:

1. We place the figure we wish to encrypt in a coordinate plane xOy.
2. We take several characteristic points $P_i(x_i,y_i)$, i=1,2,…,n, on the figure.
3. As the encryption key, we use a transition function $F(x,y)=(f(x),g(y))$, where

   (x_i,y_i), - represent the points of the unencrypted figure A, while the values $(f(x_i),g(y_i)$represent the points of the encrypted figure A_k.

Example 2.2 Encrypt the quadrilateral with vertices P(2,3), Q(3,5), R(1,4), S(0,2) with the help of the key function $F(x,y)=(\frac{x+3}{4},2^y)$.

Solution. Let the quadrilateral PQRS, be given, with vertices P(2,3), Q(3,5), R(1,4), S(0,2)

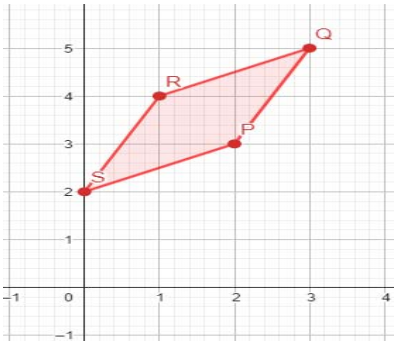

Figure 1. The unencrypted quadrilateral PQRS

With the help of the key function $F(x,y)=(\frac{x+3}{4},2^y)$ we encrypt the vertices and obtain:

- For P(2,3) we have $F(2,3)=(\frac{2+3}{4},2^3)=\left(\frac{5}{4},8\right)$, therefore P'$\left(\frac{5}{4},8\right)$;
- For Q(3,5) we have $F(3,5)=(\frac{3+3}{4},2^5)=\left(\frac{3}{2},32\right)$, therefore Q'$\left(\frac{3}{2},32\right)$;

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 48 No. 2 January 2025, pp. 147-157

- For R(1,4) we have $F(1,4) = (\frac{1+3}{4}, 2^4)$=(1,16), therefore R'(1,16);
- For S(0,2) we have $F(0,2) = (\frac{0+3}{4}, 2^2)$=$\left(\frac{3}{4}, 4\right)$, therefore S' $\left(\frac{3}{4}, 4\right)$,

As a result, we obtained the encrypted figure P' Q' R' S'.

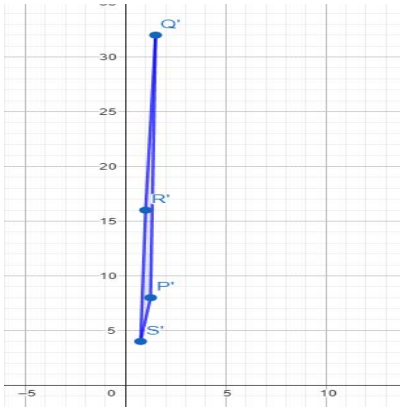

Figure 2. The encrypted quadrilateral P' Q' R' S'

Decryption Procedure:

1. We take the characteristic points$P_i(x_i, y_i)$, i=1,2,…,n, of the encrypted figure.
2. As the decryption key, we use the inverse function of the transition function $F(x, y) = (f(x), g(y))$, thus $F^{-1}(x, y) = (f^{-1}(x), g^{-1}(y))$, where (x_i, y_i), - represent the points of the encrypted figure A_k, while the pairs $(f^{-1}(x_i), g^{-1}(y_i))$, represent the points of the decrypted figure A.

Example 2.3. Decrypt the figure P' Q' R' S', given the encrypted vertices P'$\left(\frac{5}{4}, 8\right)$; Q'$\left(\frac{3}{2}, 32\right)$; R'(1,16); S' $\left(\frac{3}{4}, 4\right)$, as well as the key function $F(x, y) = \left(\frac{x+3}{4}, 2^y\right)$.

Solution. Let the encrypted figure P' Q' R' S' be given with the encrypted vertices P'$\left(\frac{5}{4}, 8\right)$; Q'$\left(\frac{3}{2}, 32\right)$; R'(1,16); S' $\left(\frac{3}{4}, 4\right)$, thus

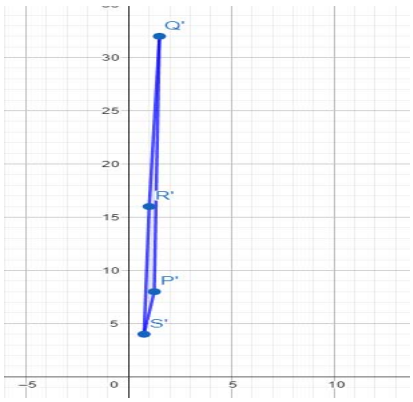

Figure.3 The encrypted figure P' Q' R' S'

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 48 No. 2 January 2025, pp. 147-157

- First, we find the inverse function $F^{-1}(x,y)$, which serves as the decryption key:

From $F(x,y) = \big(f(x), g(y)\big) = \left(\frac{x+3}{4}, 2^y\right)$, we obtain:

$$f(x) = \frac{x+3}{4} \Rightarrow f^{-1}(x) = 4x - 3$$

$$g(y) = 2^y \Rightarrow g^{-1}(y) = log_2 y$$

Thus, the inverse function $F^{-1}(x,y)$ is given by $F^{-1}(x,y) = (4x-3, log_2 y)$.

With the help of the decryption key function $F^{-1}(x,y) = (4x-3, log_2 y)$ we decrypt the vertices P', Q', R', S' and obtain:"

- For P'$\left(\frac{5}{4}, 8\right)$ we have $F^{-1}\left(\frac{5}{4}, 8\right) = \left(4 \cdot \frac{5}{4} - 3, log_2 8\right) = (5-3, log_2 2^3) = (2,3)$, therefore P(2,3);
- for Q'$\left(\frac{3}{2}, 32\right)$ we have $F^{-1}\left(\frac{3}{2}, 32\right) = \left(4 \cdot \frac{3}{2} - 3, log_2 32\right) = (6-3, log_2 2^5) = (3,5)$, therefore Q(3,5);
- For R'$(1,16)$ we have $F^{-1}(1,16) = (4 \cdot 1 - 3, log_2 16) = (4-3, log_2 2^4) = (1,4)$ therefore R(1,4);
- For S'$\left(\frac{3}{4}, 4\right)$ we have $F^{-1}\left(\frac{3}{4}, 4\right) = \left(4 \cdot \frac{3}{4} - 3, log_2 4\right) = (3-3, log_2 2^2) = (0,2)$, therefore S(0,2).

This is how we obtain the decrypted figure PQRS

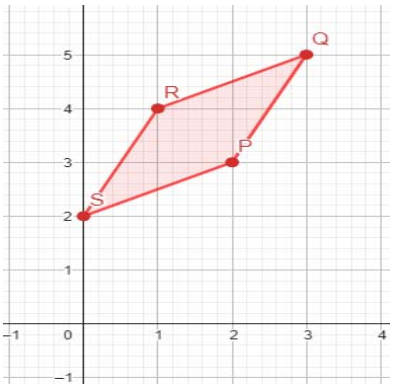

Figure.4 The decrypted figure PQRS

3.Program Implementation And Results

3.1. Implementation

This is a program created in Python that utilizes various libraries to develop an interface for the encryption and decryption of 2D coordinates. The program is structured into several functions that allow the user to input options and view different visualizations of the encrypted and decrypted shapes.

To use this program, you can follow these steps:

- After executing the program, you will see the option for 2D. Press 1 for 2D.
- You can also take advantage of the option **"?"** to read how to use the program and how to input the necessary data for the coordinates and functions."

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 48 No. 2 January 2025, pp. 147-157

Encryption and Decryption

After selecting the dimension, the program will prompt you to choose between encryption and decryption.

If you select encryption, it will ask you to provide the coordinates and functions for each dimension.

If you choose decryption, it will request the encrypted coordinates and functions for each dimension

```
def Load_Encrypt_2D():
  os.system('cls')
  n = int(input("Write number of n - vertixes: "))
  print("")
  startChar = 'A';
  cordinates = []
  print(f"* Write the cordinates for the following {n} vertixes *")
  print("")
  i = 0
  cpyN = n
  while cpyN != 0:
    cordsString = input(f"{chr(ord(startChar) + i)}: ")
    try:
      if(len(cordsString.split()) == 2):
        cordinates.append([eval(cordinate) for cordinate in cordsString.split()])
        i += 1
        cpyN -= 1
      else:
        print(f"{Fore.YELLOW} oops.. try again{Fore.RESET}")
    except:
      print(f"{Fore.YELLOW} oops.. try again{Fore.RESET}")
  print("")
  print("")
  print(f"* Write the functions for the following 2 lines *")
  print("")
  cpyN = 2
  functionsList = []
  funLetters = 'fgh'
```

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 48 No. 2 January 2025, pp. 147-157

```
paramLetters = 'xyz'
startChar = 'X'
i = 0
while cpyN != 0:
    funcString = input(f"{funLetters[i]}({paramLetters[i]}) for cordinate {chr(ord(startChar) + i)}: ")
    if('^' in funcString):
        funcString = funcString.replace('^', '**')
    if('y' in funcString):
        funcString = funcString.replace('y', 'x')
    if('z' in funcString):
        funcString = funcString.replace('z', 'x')
    functionsList.append(funcString)
    i += 1
    cpyN -= 1
x, y = symbols('x y')
try:
    fun1String = sp.sympify(functionsList[0])
    fun2String = sp.sympify(functionsList[1])
    f1 = sp.lambdify(x, fun1String, 'numpy')
    f2 = sp.lambdify(x, fun2String, 'numpy')
    f1_ = sp.Eq(fun1String, y)
    f2_ = sp.Eq(fun2String, y)

    f1_inverse = sp.solve(f1_, x)
    f2_inverse = sp.solve(f2_, x)
    if(len(f1_inverse) > 1 or len(f2_inverse) > 1):
        print(f"{Fore.YELLOW} oops.. looks like one of the functions doesn't have an inverse!{Fore.RESET}")
        print("")
        time.sleep(3)
        Load_Decrypt_2D()
    print("")
    points = [(f1(cordinate[0]), f2(cordinate[1])) for cordinate in cordinates]
```

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 48 No. 2 January 2025, pp. 147-157

```
        points.append((f1(cordinates[0][0]), f2(cordinates[0][1])))
        print("")
        print("\n1) View Again")
        print("0) Go Back\n")
        show2D(points, "Encrypted")
    except:
        print(f"{Fore.YELLOW} oops.. looks like we ran into an issue :({Fore.RESET}")
        print("")
        time.sleep(3)
        Load_Encrypt_2D()
```

Figure 5 . Portion of the code implemented for encrypting 2D images

```
def Load_Decrypt_2D():
    os.system('cls')
    n = int(input("Write number of n - vertixes: "))
    print("")
    startChar = 'A';
    cordinates = []
    print(f"* Write the encrypted cordinates for the following {n} vertixes *\n")
    i = 0
    cpyN = n
    while cpyN != 0:
        cordsString = input(f"{chr(ord(startChar) + i)}': ")
        try:
            if(len(cordsString.split()) == 2):
                cordinates.append([eval(cordinate) for cordinate in cordsString.split()])
                i += 1
                cpyN -= 1
            else:
                print(f"{Fore.YELLOW} oops.. try again{Fore.RESET}")
        except:
            print(f"{Fore.YELLOW} oops.. try again{Fore.RESET}")
```

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 48 No. 2 January 2025, pp. 147-157

```
print(f"\n\n* Write the functions that match the vertexes for the following {n} lines *\n")
cpyN = 2
functionsList = []
funLetters = 'fgh'
paramLetters = 'xyz'
startChar = 'X'
i = 0
while cpyN != 0:
    funcString = input(f"{funLetters[i]}({paramLetters[i]}) for cordinate {chr(ord(startChar) + i)}: ")
    if('^' in funcString):
        funcString = funcString.replace('^', '**')
    if('y' in funcString):
        funcString = funcString.replace('y', 'x')
    if('z' in funcString):
        funcString = funcString.replace('z', 'x')
    functionsList.append(funcString)
    i += 1
    cpyN -= 1
x, y = symbols('x y')
try:
    fun1String = sp.sympify(functionsList[0])
    fun2String = sp.sympify(functionsList[1])
    f1 = sp.Eq(fun1String, y)
    f2 = sp.Eq(fun2String, y)
    f1_inverse = sp.solve(f1, x)
    f2_inverse = sp.solve(f2, x)
    if(len(f1_inverse) > 1 or len(f2_inverse) > 1):
        print(f"{Fore.YELLOW} oops.. looks like one of the functions doesn't have an inverse!{Fore.RESET}")
        print("")
        time.sleep(3)
        Load_Decrypt_2D()
    points = [(f1_inverse[0].subs(y, c[0]), f2_inverse[0].subs(y, c[1])) for c in cordinates]
```

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 48 No. 2 January 2025, pp. 147-157

```
        points.append((f1_inverse[0].subs(y, cordinates[0][0]), f2_inverse[0].subs(y, cordinates[0][1])))
        numericPoints = [(round(float(p[0].evalf()), 2), round(float(p[1].evalf()), 2)) for p in points]
        print("\n\n")
        print("1) View Again")
        print("0) Go Back\n")
        show2D(numericPoints, "Decrypted")
    except:
        print(f"{Fore.YELLOW} oops.. looks like we ran into an issue :({Fore.RESET}")
        print("")
        time.sleep(3)
        Load_Decrypt_2D()
```

Figure. 6. Portion of the code implemented for decrypting 2D images

- **Display of Results:**

After entering the data, the program will display the visualizations of the encrypted or decrypted shapes using the Matplotlib library.

```
def show2D(points, caller = ""):
    x, y = zip(*points)
    plt.plot(x, y)
    plt.scatter(x, y, color='blue')
    offset = 0.3
    for i, (xi, yi) in enumerate(zip(x, y)):
        plt.text(xi + offset, yi - offset, f'({xi}, {yi})', fontsize = 8, ha='left')
    plt.title(f"{caller} Shape")
    plt.xlabel("X")
    plt.ylabel("Y")
    padding = 4
    plt.xlim(min(x) - padding, max(x) + padding)
    plt.ylim(min(y) - padding, max(y) + padding)
    plt.show()
    if(caller == "Decrypted"):
        choice = input(": ")
        if(choice == "0"):
```

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 48 No. 2 January 2025, pp. 147-157

```
        Load_2D()
    elif(choice == "1"):
        show2D(points, "Decrypted")
elif(caller == "Encrypted"):
    choice = input(": ")
    if(choice == "0"):
        Load_2D()
    elif(choice == "1"):
        show2D(points, "En crypted")
```

Figure 7. Function for visualizing 2D images.

CONCLUSIONS

In this paper, we aimed to present a new and interesting application of functions $F: D_f \times D_g \to \mathbb{R}^2$. This cryptosystem is user-friendly. After the theoretical treatment of the problem, we attempted to concretize it with examples. We also implemented a programmatic version of this cryptosystem. In the aforementioned examples, we used simple key functions to ensure that the work is easily understandable. By complicating the key functions, the difficulty of breaking this cryptosystem increases.

REFERENCES

[1] Azir Jusufi Kristaq Filipi, Matematikë Diskrete dhe Aplikime, Prishtine, 2022

[2] https://tresorit.com/blog/the-history-of-encryption-the-roots-of-modern-day-cyber-security/

[3] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In23rd Annual ACM Symposium on Theory of Computing,1991.

[4] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography, 1998.

[5] C. Dwork and M. Naor. Method for message authentication from non-malleable cryptosystems, 1996.

[6] I. Damgard. Towards practical public key cryptosystems secure againstchosen ciphertext attacks. In Advances in Cryptology.

[7] K. Filipi, A. Jusufi, Xh. Beqiri, Matematika për ekonomistë, Tetovë, 2012

[8] Discrete Math, Mohamed Jamaloodeen, Kathy Pinzon, Daniel Pragel, Joshua Roberts, Sebastien Siva

[9] Mathematical Foundations and Aspects of Discrete Mathematics, Jean Gallier and Jocelyn Quaintance, March 2022

[10] Discrete Mathematics an Open Introduction, Oscar Levin, July 2021

[11] Discrete Maths in Computer Science, Gary Haggard, John Schlipf, Sue Whitesides

[12] A. Jusufi, D, Berisha, M, Reqica, Permbledhje detyrash nga matematika diskrete, Prishtine, 2022

[13] F. Kabashi, A. Jusufi, USE OF COMPOSITE COMMUTATION FUNCTIONS IN DETERMINING THE DIFFIE-HELLMAN KEYS, PROCEEDINGS OF THE 7TH ANNUAL INTERNATIONAL CONFERENCE-UBT, 2018

[14] B. Zekaj, A. Jusufi, B. Imeri-Jusufi, Using incomplete polynomial functions of the odd degree n and their inverses for data encryption and decryption, IFAC-PapersOnLine, Volume 55, Issue 39, 2022, Pages 241-246

[15] Mirlinda Reqica, Diellza Berisha, Azir Jusufi, Meriton Reqica, Exploitation of exponential and logarithmic functions for data encryption and decryption, IFAC-PapersOnLine, Volume 55, Issue 39, 2022, Pages 286-291