تشفير النصوص بالاعتماد على كثيرات حدود تشيبشيف–تطبيقات عملية بلغة C++

مروه ملا خلف*، نصر الدين عيد، محمد هشام بوادقجي***،**

عبد القادر جوخدار****

* طالبة دراسات عليا (ماجستير) قسم الرياضيات، كلية العلوم، جامعة حلب

**أستاذ، قسم الرياضيات، كلية العلوم، جامعة حلب

***مدرس، قسم الرياضيات، كلية العلوم، جامعة حلب

****أستاذ، قسم هندسة الميكاترونيكس، كلية الهندسة الكهربائية والالكترونية، جامعة حلب

الملخص

في هذا البحث تم العمل على تقديم تطبيقات عملية لتشفير النصوص باستخدام خوارزمية تشفير غير متماثلة مستندة لكثيرات حدود تشيبشيف Chebyshev من النوع الأول.

تم تقديم برنامج حاسوبي بلغة البرمجة C++ يقوم بتنفيذ عملية التشفير وفك التشفير على رسالة نصية مدخلة من قبل المستخدم.

تم الاعتماد في كتابة البرنامج على ترميز ASCII (ASCII encoding) لعلامات وأحرف النص المدخل (باللغة الإنكليزية)، واستخدام العلاقة التدريجية ذاتية الاستدعاء لكثيرات حدود تشيبشيف لحساب تطبيق تشيبشيف القياسي في كل من علاقات التشفير وفك التشفير وانشاء المفاتيح العامة وكذلك تم استخدام بعض المكتبات والدوال الموجودة في برنامج Dev-C++ لإنجاز عمليتي التشفير وفك التشفير.

الكلمات المفتاحية: تشفير غير متماثل، مفتاح عام، مفتاح خاص، ملف نصي، ترميز آسكي، كثيرات حدود تشيبشيف.

ورد البحث للنشر بتاريخ 3 /10 / 2022

قبل البحث للنشر بتاريخ 11/7 / 2022

Text Encoding Based on Chebyshev Polynomials - Practical Application in C++

Marwa Malla Khalaf*, Nasr Al Din Ide,**
Mohammed Hisham Bawadekji*, Abdulkader Joukhadar******
* Post Graduate student (Msc), Dept of Mathematics, Faculty of Science, University of Aleppo
**Prof, Dept of Mathematics, Faculty of Science, University of Aleppo
***Teacher, Dept of Mathematics, Faculty of Science, University of Aleppo
**** Prof, Dept of Mechatronics Engineering, Faculty of Electrical & Electronic Engineering, University of Aleppo

Abstract

This paper presents practical implementations of texts encryption using an asymmetric encryption algorithm utilizing Chebyshev polynomials of the first kind.

A computer program has been introduced in the C++ programming language, where it implements the encryption and decryption process on a text message entered by the user.

The program is written on the ASCII coding of the characters and signs of the entered text (in English) and used the incremental self-referencing relationship of Chebyshev polynomials to calculate modified Chebyshev map in both encryption and decryption relations and the creation of public keys, as well as some libraries and functions in Dev-C++ compiler to accomplish encryption and decryption processes.

Keywords: Asymmetric encryption, Public key, Private key, Text file, ASCII encoding, Chebyshev polynomials.

Received 3 /10 / 2022
Accepted 7 / 11 / 2022

1. مقدمة:

يعد التشفير أداة من أدوات أمن المعلومات وحمايتها وأحد أهم وسائل أمن المعلومات وأكثرها فاعلية، وهو يضمن عدم تسرب البيانات، سواءً المخزنة منها في الحاسوب أو المنقولة عبر الشبكات الحاسوبية إلى الجهات غير المخولة.

هناك العديد من خوارزميات التشفير الكلاسيكية والمعاصرة والمتماثلة وغير المتماثلة، ولكل منها استخداماتها وتطبيقاتها وتضمن خوارزميات التشفير تشفيراً سهلاً للمعلومات كما تضمن استحالة اختراقها أو كسرها بدون معرفة المفتاح المستخدم في عملية التشفير [1].

يمكن اللجوء إلى التشفير غير المتماثل لإنجاز التوثيق والسرية على التوالي بآن واحد حيث إن مفاتيح المستقبل تستخدم في مرحلة تأمين السرية بينما مفاتيح المرسل تستخدم في مرحلة إنجاز التوثيق [2]، لقد ظهرت طرائق عدة للتشفير غير المتماثل، مثل التشفير بطريقة RSA والتشفير بطريقة Diffie-Hellman والتشفير بطريقة الجمل ELGamal وغيرها...

في هذا البحث عرضنا خوارزمية تشفير غير متماثلة اقترحها الباحث كوكاريف (Ljupco Kocarev) [3,4]، حيث لها القدرة أيضاً على توفير التوقيع الرقمي (digital signature)، وتعتمد هذه الخوارزمية بشكل أساسي على كثيرات حدود تشيبشيف (Chebyshev polynomials) من النوع الأول والتي تحقق خاصية شبه الزمرة، وهي كثيرات حدود متعامدة يعود اسمها إلى عالم الرياضيات الروسي بافنوتي تشيبشيف (Pafnuty Chebyshev) حيث يمكن أن تُعرَّف بعلاقة ذاتية الاستدعاء كما سنبين ذلك لاحقاً. كما عرضنا أمثلة عملية لكيفية التشفير وفك التشفير لرسائل نصية وفقاً للخوارزمية المستندة لكثيرات حدود تشيبشيف Chebyshev من النوع الأول باستخدام لغة البرمجة C++.

2. مفاهيم ومبرهنات ذات صلة:

2.1. تطبيق كثيرات حدود تشيبشيف[5,6] (Chebyshev polynomial map):

كثيرات حدود تشيبشيف هي عبارة عن متتالية من كثيرات حدود متعامدة، تعرف كثيرات حدود تشيبشيف من النوع الأول بالشكل:

$$T_k(x) = \cos(k \cos^{-1} x) \quad \dots\dots (1)$$

حيث: $T_k(x): [-1, +1] \to [-1, +1]$

و $k \in \mathbb{N}^*$ تمثل درجة كثير الحدود، ومن أجل $x \in \mathbb{Z}$ يكون $T_k(x) \in \mathbb{Z}$.

وتعطى العلاقة التدريجية (ذاتية الاستدعاء) لأي تابع من توابع تشيبشيف بدلالة تابعين سابقين بالصيغة:

$$T_k(x) = 2x\, T_{k-1}(x) - T_{k-2}(x) \quad ; k \geq 2 \quad \dots\dots (2)$$

ونظراً لأن $T_0(x) = 1$ و $T_1(x) = x$ فإن بعض كثيرات حدود تشيبشيف الأولى هي:

$$T_2(x) = 2x^2 - 1\,, T_3(x) = 4x^3 - 3x\,, T_4(x) = 8x^4 - 8x^3 + 1\,,$$
$$T_5(x) = 16x^5 - 20x^2 + 5x\,, T_6(x) = 32x^6 - 48x^4 + 18x^2 - 1, \dots$$

وهكذا يمكن إيجاد بقية الحدود.

2.2. مبرهنة [7]: إن كثيرات حدود تشيبشيف تحقق خاصة شبه الزمرة(Semi-group):

$$T_p\left(T_q(x)\right) = T_{pq}(x) \qquad x \in \mathbb{Z};\ و\ p, q \in \mathbb{N}^*$$

وكذلك تحقق خاصة تبديل تحت التركيب (Commute under compo-sition):

$$T_p\left(T_q(x)\right) = T_q\left(T_p(x)\right)$$

يمكن بسهولة اثبات المبرهنة **(2-2)** باستخدام العلاقة الجبرية (1) [8].

2.3. تطبيق كثيرات حدود تشيبشيف القياسي (Modified Chebyshev polynomials map):

يتم إجراء تعديل بسيط للتطبيق الأساسي لكثيرات حدود تشيبشيف من أجل التعامل مع حقل معين من الأعداد الصحيحة [9,10] ، وذلك بأخذ تعديل(mod) لـ $T_k(x)$ على عدد أولي N، وبذلك يكون التطبيق القياسي معطى بالشكل:

$$Y = T_k(x) \bmod \mathrm{N} = \left(2x\, T_{k-1}(x) - T_{k-2}(x)\right) mod\ N \quad \dots\dots (3)$$

حيث x و Y أقل من N و $k \geq 2$.

2.4. مبرهنة [11]: يحقق تطبيق كثيرات حدود تشيبشيف القياسي الخاصة الهامة الآتية:

$$T_p\big(T_q(x) mod\ N\big) mod\ N = T_{pq}(x) mod\ N = T_q\big(T_p(x) mod\ N\big) mod\ N$$

3.خوارزمية المفتاح العام المستندة لتطبيق كثيرات حدود تشيبشيف من النوع الأول (L. Kocarev 2003):

تتكون هذه الخوارزمية من ثلاثة مراحل، من أجل ارسال رسالة من Bob (مرسل) إلىAlice (مستقبل) [3, 10]:

1. انشاء المفتاح العام، سيقوم المستقبل بتوليد المفتاح العام كما يلي:

1) يتم اختيار عدداً أولياً كبيراً N، وعدداً عشوائياً $x \in \mathbb{Z}$ أصغر من N.
2) يتم حساب $K_1 = T_p(x) \bmod N$، حيث $p \in \mathbb{Z}$ عدد عشوائي ويمثل مفتاحاً خاصاً لـ Alice.
3) أخيراً المفتاح العام لـ Alice هو (x, N, K_1).

2. تنفيذ عملية التشفير:

بعد توليد المفتاح العام والخاص لـ Alice، وتعميم المفتاح العام على جميع أطراف شبكة الاتصال، لإرسال العدد المشفر $m \in [0, N-1]$ سيقوم Bob – مستعملاً المفتاح العام– بما يلي:

1) يتم حساب $K_2 = T_q(x) \bmod N$، حيث $q \in \mathbb{Z}$ أصغر من N، ويمثل K_2 مفتاحاً عاماً لفك التشفير و q المفتاح الخاص بـ Bob.
2) يتم تشفير العدد m وفق العلاقة: $C = m\, T_q(K_1) \bmod N$.
3) يتم ارسال الرسالة المشفرة (K_2, C) إلى Alice .

3. تنفيذ عملية فك التشفير (استعادة الرسالة الأصلية):

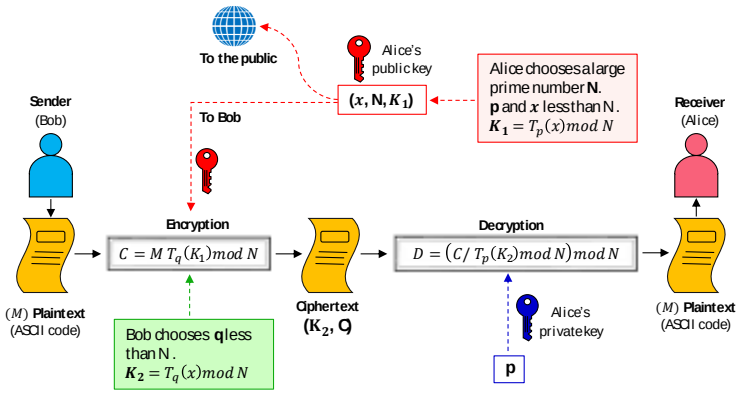
باستخدام K_2 المفتاح العام لـ Bob و p المفتاح الخاص بـ Alice يتم فك التشفير من خلال تطبيق العلاقة: $D = (C / T_p(K_2) \bmod N) \bmod N$.

4. آلية تشفير وفك تشفير رسالة نصية وفقاً للخوارزمية المستندة لكثيرات حدود تشيبشيف من النوع الأول:

يتم تحويل الرسالة النصية إلى أعداد بحيث تتوافق مع العمليات الحسابية في عمليتي التشفير وفك التشفير.

وقد تم الاعتماد على ترميز ASCII (ASCII encoding) للتعامل مع حروف وعلامات النص، حيث يتم بدايةً مقابلة محارف النص الأصلي بأعداد وفق نظام الترميز ASCII ، وأخيراً في عملية فك التشفير يتم أخذ المقابل المحرفي لأعداد النص المشفر للحصول على النص الأصلي (Reverse ASCII).

تم تمثيل مخطط لخوارزمية المفتاح العام المستندة لتطبيق كثيرات حدود تشيبشيف من أجل رسالة نصية متبادلة بين طرفي اتصال، الشكل (1).



الشكل (1)، آلية التشفير وفك التشفير لرسالة نصية وفقاً للخوارزمية المستندة لتطبيق تشيبشيف القياسي

يتضح من الشكل (1) و(2) أن هناك مفتاحان لكل من طرفي جهة الاتصال، مفتاح عام (معلن) وآخر خاص (سري).

	Sender	Receiver
Private key	q	p
Public key	k_2	k_1

الشكل (2)، المفاتيح العامة والخاصة لطرفي جهة الاتصال

وتعمل هذه المفاتيح على النحو الآتي:

تقوم الجهة المستقبلة بتعميم المفتاح العام k_1 مع التوصيفات N وx على جميع أطراف شبكة الاتصال، وهو المفتاح الذي يستخدمه المرسل في عملية التشفير فقط.

وتبقي على نفسها $\boldsymbol{p}$ كمفتاح خاص والذي يجب أن يبقى سراً ولا تتبادله مع أحد، حيث تمثل $\boldsymbol{p}$ درجة تطبيق تشيبشيف عند x في حساب $\boldsymbol{K_1}$.

سيقوم الطرف الثاني(المرسل) بتشفير رسالة M تحت المفتاح العام $(x, N, \boldsymbol{K_1})$ مع استخدام المفتاح الخاص به $\boldsymbol{q}$، بعد ذلك سيرسل الرسالة المشفرة مرفقةً بمفتاحه العام $\boldsymbol{k_2}$ والذي يتعلق بمفتاحه الخاص $\boldsymbol{q}$. تمثل $\boldsymbol{q}$ درجة تطبيق تشيبشيف عند x في حساب $\boldsymbol{k_2}$ ،حيث يستخدم $\boldsymbol{k_2}$ في عملية فك التشفير.

إذن يمكن أن يرسل كل من الطرفين مفتاحه العام كقيمة متعلقة بالمفتاح الخاص دون أن يرسل كل منهما المفتاح الخاص به نفسهُ، وهذا ما يميز التشفير غير المتماثل عن التشفير المتماثل. إذ أن مشكلة طرق التشفير المتماثل هي تناقل المفتاح نفسه بين المرسل والمستقبل.

5. أمثلة عملية:

مثال (1):

لنوضح خطوات الخوارزمية السابقة من خلال تشفير الرسالة:

"Hello Alice, I'm in Aleppo. "

باستخدام المفاتيح العددية الآتية:

$N = 131, x = 14, p = 12, q = 15$

$K_1 = T_p(x) \bmod \mathrm{N} = T_{12}(14) \bmod 131 = 41$

$K_2 = T_q(x) \bmod \mathrm{N} = T_{15}(14) \bmod 131 = 113$

1. مرحلة التشفير:

H	e	l	l	o	sp	A	l	i	c	e	,	sp	I
'	m	sp	i	n	sp	A	l	e	p	p	o	.	sp

- لنكتب النص على شكل مصفوفة:
- نضع المقابلات العددية لمحارف النص الأصلي للرسالة (ASCII encoding):

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}
72	**101**	**108**	**108**	**111**	**32**	**65**	**108**	**105**	**99**	**101**	**44**	**32**	**73**
M_{15}	M_{16}	M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}	M_{25}	M_{26}	M_{27}	M_{28}
39	**109**	**32**	**105**	**110**	**32**	**65**	**108**	**101**	**112**	**112**	**111**	**46**	**32**

- نستخدم العلاقة $C_i = M_{\ i} \times T_q(K_1) \bmod \mathrm{N}$ لتشفير النص، حيث i عدد محارف النص، فمثلاً:

$C_1 = M_{\ 1} \times T_{15}(41) \bmod 131 = 72 \times 18 = 1296$
$C_2 = M_{\ 2} \times T_{15}(41) \bmod 131 = 101 \times 18 = 1818$
$C_3 = M_{\ 3} \times T_{15}(41) \bmod 131 = 108 \times 18 = 1944$

C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}
1296	**1818**	**1944**	**1944**	**1998**	**576**	**1170**	**1944**	**1890**	**1782**	**1818**	**792**	**576**	**1314**
C_{15}	C_{16}	C_{17}	C_{18}	C_{19}	C_{20}	C_{21}	C_{22}	C_{23}	C_{24}	C_{25}	C_{26}	C_{27}	C_{28}
702	**1962**	**576**	**1890**	**1980**	**576**	**1170**	**1944**	**1818**	**2016**	**2016**	**1988**	**828**	**576**

- وهكذا يتم تشفير بقية محارف النص كما يلي:
- إذن النص المشفر الرقمي هو:

"1296181819441944199857611701944189017821818792576131470219625761890198057611701944181820162016199882857 6 "

2. مرحلة فك التشفير:

- لفك التشفير عن النص السابق نستخدم العلاقة الآتية:

$D_i = (C_i / \ T_p(K_2) \bmod \mathrm{N}\ ) \bmod \mathrm{N}$

حيث i عدد المحارف العددية للنص المشفر.

لدينا: $T_p(K_2) \bmod \mathrm{N} = \ T_{12}(113) \bmod 131 = 18$

وبالتالي يتم فك التشفير بالشكل:

$D_1 = (C_1 / \ T_{12}(113) \bmod 131) \bmod 131 = (1296/18) \bmod 113 = 72$
$D_2 = (C_2 / \ T_{12}(113) \bmod 131) \bmod 131 = (1818/18) \bmod 113 = 101$

وهكذا...

D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}
72	**101**	**108**	**108**	**111**	**32**	**65**	**108**	**105**	**99**	**101**	**44**	**32**	**73**
D_{15}	D_{16}	D_{17}	D_{18}	D_{19}	D_{20}	D_{21}	D_{22}	D_{23}	D_{24}	D_{25}	D_{26}	D_{27}	D_{28}
39	**109**	**32**	**105**	**110**	**32**	**65**	**108**	**101**	**112**	**112**	**111**	**46**	**32**

- وأخيراً نعوض بالمقابل المحرفي (Reverse ASCII) نحصل على المصفوفة:

H	**e**	**l**	**l**	**o**	**sp**	**A**	**l**	**i**	**c**	**e**	**,**	**sp**	**I**
'	**m**	**sp**	**i**	**n**	**sp**	**A**	**l**	**e**	**p**	**p**	**o**	**.**	**sp**

وبالتالي حصلنا مجدداً على النص الأصلي:" Hello Alice, I'm in Aleppo." .

3. التطبيق البرمجي:

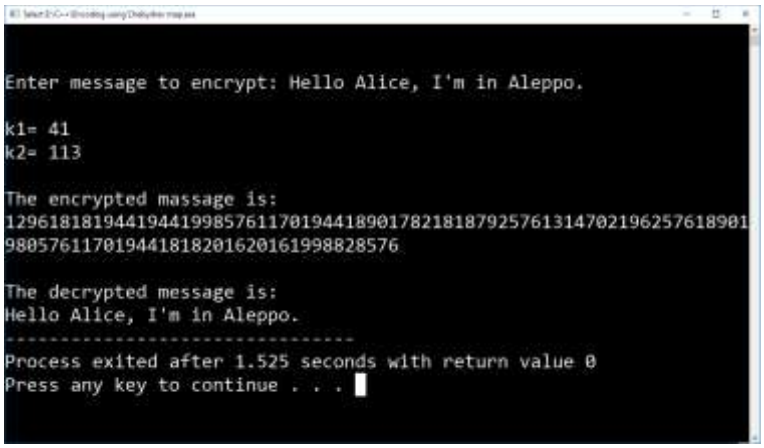
فيما يلي برنامج بلغة C++ حيث تم فيه تنفيذ عمليتي التشفير وفك التشفير:

```
// Encryption & Decryption Using Chebyshev polynomials map
#include<iostream>
#include<fstream>
#include<stdlib.h>
#include<cstring>
#include<math.h>
using namespace std;
void Plain_Text (string& user);
int Public_key1(int p, int x, int N);
int Public_key2(int q, int x, int N);
int Cheb_Mod (int n, int x, int N);
void File_CipherdTxt (ofstream& CipheredText);
void Encryption (string& st, int p, int q, int x, int N, ofstream& CipheredText);
void Decryption (string& st, int p, int q, int x, int N);
int main ()
{
string user; int N=131; int x=14; int p=12; int q=15; ofstream CipheredText;
Plain_Text(user); //Original message
cout<< "k1= "<<Public_key1(p, x, N) <<endl;
cout<< "k2= "<<Public_key2(q, x, N) <<endl;
File_CipherdTxt (CipheredText); // Create a file(txt)
Encryption (user, p, q, x, N, CipheredText); //Encode message
Decryption (user, p, q, x, N); //Decode message
return 0;
}
void Plain_Text (string& user)
{
   cout<< "\n\nEnter message to encrypt: ";
   getline (cin, user);
   cout<<endl;
}
int Public_key1(int p, int x, int N)
{
    int key1=Cheb_Mod (p, x, N);
    return(key1);
}
int Public_key2(int q, int x, int N)
{
    int key2=Cheb_Mod (q, x, N);
    return(key2);
}
void File_CipherdTxt (ofstream& CipheredText)
```

```
{
    CipheredText.open("encrypted message file.txt");
}
void Encryption (string& st, int p, int q, int x, int N, ofstream& CipheredText)
{
    int K1=Public_key1(p, x, N);
    cout<<endl<< "The encrypted massage is: " <<endl;
    for (int i=0; i<st. length (); i++)
    {
          int E=st[i]*Cheb_Mod (q, K1, N);
          cout<<E;
          CipheredText<<E<<" ";
    }
    cout<<endl;
    CipheredText.close();
}
void Decryption (string& st, int p, int q, int x, int N)
{
    ifstream EnterText;
    EnterText.open("encrypted message file.txt");
    if (EnterText.fail())
   {
          cerr<< "fail could not be opened" <<endl;
          exit (1);
    }
    cout<< "\nThe decrypted message is: " <<endl;
    int m [st. length ()];
    for (int i=0; i<st. length (); i++)
    {
          EnterText>>m[i];
    }
    int K2=Public_key2(q, x, N);
    for (int i=0; i<st. length (); i++)
    {
         int D=(m[i]/Cheb_Mod (p, K2, N)) %N;
         char Dec=(char)D;
         cout<<Dec;
    }
}
int Cheb_Mod (int n, int x, int N) // Modified Chebyshev polynomials map
{
     int T1, T2;
     if (n==0)
          return 1;
     else
           if (n==1)
                return x;
```

```
        else
        {
            T1=Cheb_Mod (n-1, x, N) %N;
            T2=Cheb_Mod (n-2, x, N) %N;
            if ((2*x*T1-T2) %N >0)
                return (2*x*T1-T2) %N;
            else
                return (2*x*T1-T2) %N +N;
        }
}
```

يتم ادخال نص الرسالة المراد تشفيرها من المستخدم، ثم يتم عرض النص المشفر بصيغته العددية وتخزين نتيجة التشفير في ملف(مستند) نصي txt مساراً من نفس مسار ملف المشروع الذي نعمل فيه، وكذلك يتم عرض النص الأصلي للرسالة بعد عملية فك التشفير عن النص المشفر حيث يتم إحضار دخل فك التشفير (البيانات العددية المشفرة) من الاتصال بالملف **encrypted message file.txt** في القرص الصلب من جهاز الكومبيوتر. والشكل (3) يبين نتيجة تنفيذ البرنامج أعلاه:



الشكل (3)، ناتج عملية التشفير وفك التشفير على الرسالة النصية في المثال (1)

مثال (2):

شفر الرسالة النصية:

"The Euphrates River is one of the oldest rivers in the Middle East, with a length of 2,800 km. Many ancient countries and cities have flourished on the banks of this river since ancient times, including the Mesopotamian Empire. The waters of the Euphrates River are mostly rain and melting snow. Giant dams were built on the Euphrates River,

the most famous of which is the Euphrates Dam, with a height of 200 feet. This river may overlap at the city of Jarablus, as it enters many cities, including Deir ez-Zor and Raqqa. And this river expands greatly in many cities to form the marshes. It also unites with the Tigris River in Iraq to form the Shatt al-Arab and then flows into the Arabian Gulf."

وفقاً لخوارزمية التشفير المستندة لكثيرات حدود تشيبشيف من النوع الأول باستخدام المفاتيح العددية الآتية:

$N = 157, x = 17, p = 20, q = 14.$

الحل: بتنفيذ البرنامج التطبيقي في المثال السابق على النص المراد تشفيره مع الأخذ بالمفاتيح المعطاة، وحيث لدينا:

$$K_1 = T_p(x) \bmod \mathrm{N} = T_{20}(17) \bmod 157 = 28$$
$$K_2 = T_q(x) \bmod \mathrm{N} = T_{14}(17) \bmod 157 = 94$$

نحصل على النص العددي المشفر:

"44525512535316963657620159365512604251416148535360951696434655656254535360421696560951696588358305355169643465565625453536042169655656095169658835830535316965883540616966148551253531696588357245300535360956148169660425565625453536042609516965565583016966145512535316965883572453005353609561481696604255656254535360426095169655655830169661455125353169658835724530053536095614823321696630755656148551216965141169657245353583054596148551216965883540616962650233229682544254416965671577724381696408151415830641316965141583052475565535358306148169652475883620158306148604255655353609516965141583053001696524755656148556553536095169655125141625453531696540657245883620160425565609555125353530016965883583016966148551253531696519451415830567160951696588354061696614855125565609516966042556562545353604216966095556558305247535316965141583052475565535358306148169661485565577753536095233216965565583052475724620153005565583054591696614855125353169640815353609558835936588361485141577755655141583016963657577759365565604253532438169644525512535316966307514161485353604260951696588354061696614855125353169636576201593655126042514161485353609516964346556562545353604216965141604253531696577758836095614857246413169660425141556558301696514158305300169657775353572461436042514155655830169651415830530016965777535357246143604251415565583016965141583053001696577753535724614

85565583054591696609558305883630724381696376355655141583061481696530051415777609516966307535360425353169651946201556557246148169658835830169661485512535316963630576201593655126042514161485353609516963604514157772332169663075512556524755121696556560951696614855125353169636576201593655126042514161485353609516963604514157772332169663075512556755656148551216965141169655125353556554595512614816965880354062616962650254425441696540653535353614824381696445255125565609516966042556562545353604216965777514164131696588362545353604257245141593616965141614816965883625453536042572451415936169651416148169661485512535316965247556561486413169658835406169639225141604251415194572462016095233216965141609516965565614816965353583061485353604260951696577751415830641316965247556561485565535360952332169655655830524757246201530055655830545916963604535355656042169653536462385477058836042169651415830530016964346514159895989514124381696344558305300169661485512556560951696604255656254535360421696535363605936514158305300609516965459604253535141614857246413169655655830169657775141583064131696524755656148556553536095169661485883169654065883604257771696614855125353169657775141604260955512535360952438169638696148169651415724609558831696620158305565614853536095169663075565614855121696614855125353169644525565545960425565609516964346556562545336042169655655830169638696042514159891696614858831696540658836042577716966148551253531696439955125141614861481696514157242385344560425141519416965141583053001696614855125353583016965406572458836307609516965565583061485883169661485512535316963445604251415194169651415830169637636201572454062438".

والشكل (4) يبين ناتج التنفيذ البرمجي لكل من عمليتي التشفير وفك التشفير على نص الرسالة السابق:

الشكل (4)، ناتج عملية التشفير وفك التشفير على الرسالة النصية في المثال (2)

6.نتيجة البحث:

تم في هذا البحث تطوير خوارزمية مفتاح عام مستندة لكثيرات حدود تشيبشيف وذلك باستخدامها في تشفير النصوص، تم اختبار العمل بإجراء تطبيقات عملية بلغة البرمجة C++ يتم فيها تنفيذ التشفير وفك التشفير على بعض النصوص (باللغة الإنكليزية).

7. التوصيات:

1. لزيادة أمان الخوارزمية المذكورة نقترح تجزئة النص وتشفير أجزاء منها بواسطة خوارزمية RSA أو الجمل ELGamal.
2. تطبيق الخوارزمية ضمن لغة برمجية أخرى مثل C# أو Python.

المراجع:

1. **النائب** إبراهيم، **دبش** محمد، 2008 – **أمن المعلومات**. كلية الاقتصاد، منشورات جامعة حلب، سوريا، 364 صفحة.
2. **فريد** يحيى، 2012 – **أمن المعلومات في نظم الاتصالات**. كلية الهندسة الكهربائية والإلكترونية، منشورات جامعة حلب، سوريا، 256 صفحة.
3. KOCAREV L.; TASEV Z., 2003 – **PUBLIC-KEY ENCRYPTION BASED ON CHEBYSHEV MAPS**. *ISCAS, IEEE international symposium on circuits systems*, 28-31.

4.KOCAREV L.; MAKRADULI J.; AMATO P., 2005 – **PUBLIC-KEY ENCRYPTION BASED ON CHEBYSHEV POLYNOMIALS**. *Circuits System and Signal Processing*, **24(5),** 497-517.

5.MASON J.C.; HANDSCOMB D.C., 2003 – **CHEBYSHEV POLYNOMIALS**. Chapman and Hall/CRC, USA, 335.

6. **عيد** نصر الدين، 2011 – **التحليل العددي**. كلية العلوم، منشورات جامعة حلب، سوريا، 421 صفحة.

7. YAN S.; ZHEN P.; MIN L., 2015 – **Provably Secure Public Key Cryptosystem Based on Chebyshev Polynomials**. *Journal of Communication*, **10(6)**, 380-384.

8.TAHAT N.; ABDALLAH E., 2018 – **Hybrid Public Verifiable Authenticated Encryption Scheme Based on Chaotic Maps and Factoring Problems**. *Journal of Applied Security Research*, **13(3)**, 304-314.

9.SUN J.; ZHAO G.; LI X., 2013 – **An Improved Key Encryption Algorithm Based on Chebyshev Polynomials**. *TELKOMNIKEA*, **11(2)**, 864-870.

10. NAJAF Y.; MAHMOOD M., 2020 – **AN IMPROVED PUBLIC KEY CRYPTOSYSTEM BASED ON CHEBYSHEV CHAOTIC MAP OVER FINITE FIELD**. *Journal of Engineering and Sustainable Development*, 304-314.

11. TAHAT N.; TAHAT A.; ABD-DALU M.; ALBADARNEH R.; ABDALLAH A.; HAZAIMEH O., 2020 – **A new RSA Public Key encryption scheme with chaotic maps**. *International Journal of Electrical and Computer Engineering*, **10(2)**, 1430-1437.