MỘT SỐ ỨNG DỤNG CỦA CÁC HÀM LOGIC TRONG MÃ HÓA THÔNG TIN

TS. Nguyễn Đăng Minh[1], ThS. Vũ Anh Tú[2]
[1]Khoa Công nghệ thông tin & Điện tử viễn thông, Trường Đại học Hòa Bình
[2] Trung tâm Công nghệ thông tin, Trường Đại học Công nghiệp Hà Nội
[*]Tác giả liên hệ: ndminh@daihochoabinh.edu.vn

Ngày nhận: 13/02/2022
Ngày nhận bản sửa: 02/3/2022
Ngày duyệt đăng: 18/3/2022

Tóm tắt

Bài báo phân tích các nguyên tắc xây dựng thuật toán mã hóa thông tin dạng văn bản sử dụng các hàm logic phổ biến. Phân tích cụ thể và xây dựng phần mềm để mã hóa thông tin bằng cách sử dụng hàm logic Not và hàm Xor1.

Khi mã hóa thông tin bằng hai hàm Not và Xor1[1], mức độ bảo mật thông tin cao hơn hẳn so với nhiều phương pháp khác nhờ áp dụng các yếu tố cho phép sử dụng hàm Xor1 trong mọi trường hợp.

Phương pháp này cũng cho phép sử dụng các cổng máy tính giao tiếp để truyền văn bản được mã hóa qua mạng.

Có phần mềm mã hóa và giải mã với các ví dụ cụ thể và phân tích độ bảo mật của phương pháp mã hóa dùng các hàm logic.

Từ khóa: *Xor, mã hóa đối xứng, mã hóa, giải mã.*

Some applications of logic functions in cryptography

Abstract

The article analyzes the principles to follow while building rules for encoding textual information using common logic functions. It provides specific analysis and software built to encode information using the Logic NOT Function and the Xor1[2] Function. When information is encrypted using the above two functions, the security level attained is much higher level than that of many other methods, as a result of applying the factors which allow the use of the Xor1 function in all cases. This method also allows the use of communication computer ports to transmit encrypted text over the network. Encryption and decryption software are demonstrated with specific examples and security analysis of encryption methods using logical functions.

Keywords: *Xor, symmetric encryption, encryption, decryption.*

1. Đặt vấn đề

Thông tin ngày càng đóng vai trò quan trọng trong kinh tế và quốc phòng đối với bất kỳ quốc gia nào. Giữ được bí mật thông tin của nhà nước, cơ quan đóng vai trò quan trọng trong việc đảm bảo an ninh, an toàn cho các hoạt động của nhà nước, toàn quân, toàn dân, góp phần giữ vững an ninh quốc gia, trật tự an toàn xã hội.

Bảo vệ thông tin bằng cách biến đổi nó để loại trừ khả năng người ngoài cố ý hoặc tình cờ có thể đọc được đã được nhân loại chú ý từ rất lâu và đã hình thành ngành Bảo mật thông tin (Cryptographia). Lịch sử của ngành Bảo mật thông tin gần như đồng thời với lịch sử loài người. Hơn thế, ban đầu, ngay chữ viết tự thân nó cũng đã là một hệ thống bảo mật thông tin vì rằng ở thời xa xưa, chỉ những người đọc và hiểu được chữ viết mới có thể biết được thông tin. Những cuốn sách cổ Hi Lạp, Ấn Độ là các ví dụ.

[1] Đây là hàm mới được đề xuất cho mục đích mã hóa thông tin.
[2] It is a new function.

Với sự phổ cập của chữ viết, Bảo mật thông tin dần trở thành một ngành khoa học độc lập. Các hệ thống bảo mật thông tin đầu tiên ra đời từ đầu kỷ nguyên của chúng ta. Sezar trong các thư của mình thường sử dụng một cách có hệ thống các mã, mà sau này, người ta gọi là mã Sezar.

Cuộc cách mạng công nghiệp 4.0 với sự số hóa toàn diện của cuộc sống đang ngày càng đòi hỏi phải bảo mật thông tin tốt hơn nữa.

Thông tin tồn tại ở rất nhiều dạng khác nhau. Âm thanh, chữ viết, dấu hiệu, hình ảnh đều có thể được biến đổi, mà ta gọi là mã hóa (cryptographia) để người khác không hiểu được. Trong phạm vi của bài viết này, chúng tôi chỉ xét sự biến đổi của bản chữ viết (text) thành một bản chữ viết khác (text đã được mã hóa) trong phạm vi áp dụng trên máy tính, chứ không bàn đến bảo mật thông tin tồn tại dưới các dạng khác hoặc bằng thiết bị phần cứng.

Có một văn bản (text_source), muốn biến nó thành văn bản khác để người khác không đọc được - gọi là văn bản đã mã hóa (text_cryptographic).

Nguyên tắc chuyển đổi có thể thấy trong rất nhiều tài liệu kinh điển [1,2,3].

Gọi các phần tử của văn bản ban đầu (text_source) là x_i với i=1,K,

Gọi các phần tử của văn bản đã mã hóa (text_cryptographic) là y_j với j=1,N,

Thông thường [2] K=N. Từ đây, ta xem N=K và chỉ dùng i cho cả x và y. Như vậy, quá trình mã hóa biến $x_i \rightarrow y_i$,

Quá trình mã hóa yêu cầu, từ y_i ta phải khôi phục được x_i- quá trình này gọi là giải mã (decryption). Như vậy, cần hai quá trình biến đổi **mã hóa** và **giải mã** đồng thời tồn tại.

Mã hóa dùng hàm f_{crpt} $y_i=fcrpt(x_i)$;

Còn giải mã dùng hàm $x_i=fdecrpt(y_i)$;

Khi xây dựng các hàm mã hóa, có 2 phương án được phát triển. Mã đối xứng và mã không đối xứng [6]. Trong phạm vi bài viết này, chỉ bàn đến mã đối xứng, chúng bị ràng buộc bởi các yêu cầu sau đây:

$y_i=fcrpt(x_i)$ và $x_i=fdecrpt(y_i)$ cần thỏa mãn một số yêu cầu [2].

- Tồn tại hàm $x_i=fdecrpt(y_i)$ được suy ra từ hàm $y_i=fcrpt(x_i)$ - fdecrypt gọi là hàm thuận nghịch;

-Các hàm xi=fdecrpt(yi) và yi=fcrpt(xi) phải dễ thực hiện trên các phương tiện thông dụng và phải luôn đơn trị (trên máy tính và các phương tiện truyền tin thông thường).

- Khi làm việc trên các máy tính thông dụng, các chữ dùng trong text_source và text_cryptographic thì các biến phải thay đổi trong phạm vi của mã ascii (là các số từ 1 đến 127) hoặc các ký tự (character) tương ứng.

- Bảng mã ascii phải là bảng mã mở rộng để có thể dùng được tiếng Việt (bảng mã ascii mở rộng gồm 255 ký tự).

Ví dụ về bảng mã ascii:

DEC	OCT	HEX	BIN	SYMBOL
0	000	00	00000000	NUL
1	001	01	00000001	SOH
2	002	02	00000010	STX
3	003	03	00000011	ETX
4	004	04	00000100	EOT
5	005	05	00000101	ENQ
6	006	06	00000110	ACK
7	007	07	00000111	BEL
8	010	08	00001000	BS
9	011	09	00001001	HT
10	012	0A	00001010	LF
11	013	0B	00001011	VT
12	014	0C	00001100	FF
13	015	0D	00001101	CR
14	016	0E	00001110	SO
15	017	0F	00001111	SI
16	020	10	00010000	DLE
17	021	11	00010001	DC1
18	022	12	00010010	DC2
19	023	13	00010011	DC3
20	024	14	00010100	DC4
21	025	15	00010101	NAK
22	026	16	00010110	SYN
23	027	17	00010111	ETB
24	030	18	00011000	CAN
25	031	19	00011001	EM
26	032	1A	00011010	SUB
27	033	1B	00011011	ESC
28	034	1C	00011100	FS
29	035	1D	00011101	GS
30	036	1E	00011110	RS
31	037	1F	00011111	US

Hình 1. Ví dụ về bảng mã ascii

2. Lựa chọn hàm

a) Hàm đại số

y_i=fcrpt(xi) có thể chọn là một hàm đại số, ví dụ fy(x)=$3x^2$

Khi đó, hàm ngược $fx(y) = \sqrt{\frac{y}{3}}$, muốn 2 hàm này có thể dùng làm bộ đôi hàm thuận nghịch cho mã hóa, cần các yếu tố như trong yêu cầu trên đây: $0 \le x, y \le M$. Thông thường, người ta dùng hàm mod() để đảm bảo điều kiện này. Cần rất nhiều những yếu tố đảm bảo để có thể làm việc với các quá trình trên trên máy tính. Có nhiều cách áp dụng các hàm đại số [2,3,4,5].

b) Hàm logic

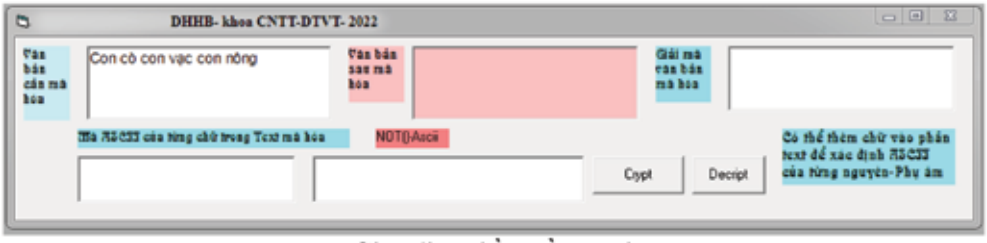
Có thể kiểm tra khả năng dùng các hàm logic sau đây trong mã hóa thông tin: hàm NOT, AND, OR và XOR. Các hàm này được sử dụng rộng rãi trong các ngôn ngữ lập trình hiện nay. Để nghiên cứu khả năng sử dụng các hàm logic cho mã hóa, VisualBasic 98 được sử dụng.

c) Hàm NOT

Nếu y=Not(x), có thể thấy các yêu cầu cho cặp đôi hàm mã hóa được thỏa mãn.

Thực hiện mã hóa bằng hàm Not() cho các kết quả như dưới đây:

Phần mềm tạo Form:



Hình 2. Giao diện phần mềm mã hóa

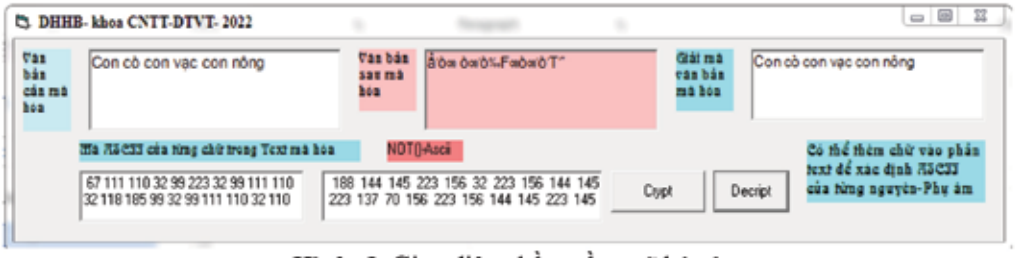
```
Dim a As Byte
Dim b As Byte
Dim c As Byte
Dim Dau_de As String

Private Sub Command1_Click()
Text2.Text = Chr(Val(Text1.Text))
Text3.Text=(Not(Val(Text1.Text)))
End Sub

Private Sub Command2_Click()
Text5.Text = ""
Text4.Text = ""
Text2.Text = ""
L = Len(Text1.Text)
 For i = 1 To L
   a = (Asc(Mid(Text1.Text, i, 1)))
Text4.Text = Text4.Text + Str(a)
   a = Not (a)
Text5.Text = Text5.Text & " " & Str(a)
Text2.Text = Text2.Text & Chr(a)
 Next
End Sub

Private Sub Command3_Click()
Text3.Text = ""

L = Len(Text2.Text)
 For i = 1 To L
   a = (Asc(Mid(Text2.Text, i, 1)))
   a = Not (a)
   Text3.Text = Text3.Text + Chr(a)
 Next
End Sub

Private Sub Form_Load()
Dau_de="DHHB-khoaCNTT-DTVT-2022"
Me.Caption = Dau_de
End Sub
Private Sub Timer1_Timer()
Me.Caption = " " + Me.Caption
If Len(Me.Caption) > 150 Then Me.Caption =
Dau_de
End Sub
```

Nhấn vào Encrypt, ta được:

Dòng chữ *Con cò con vạc con nông* đã được mã hóa bằng hàm Not thành:

¼□‘ßœ ßœ□‘ß‰Fœßœ□‘ß‘T‘˜

Có thể giải mã bằng việc nhấn vào Decrypt, ta lại được:



Hình 3. Giao diện phần mềm mã hóa 1

Vậy dòng chữ thông tin mã hóa ban đầu lại xuất hiện trên textBox thứ 3.

d) Hàm AND

Hàm AND cần 2 tham số trở lên. Với 2 tham số x1, x2, ta có:

y=x1 and x2 như sau:

Bảng 1.

x1	x2	y(x1 and x2)	Số dòng
0	0	0	1
0	1	0	2
1	0	0	3
1	1	1	4

Không khó để xác định hàm y không đơn trị với x1 và x2 (dòng 1 và dòng 2, dòng 1 và dòng 3). Theo [3], hàm AND vi phạm nguyên tắc đơn trị (hàm thuận nghịch).

Cũng làm tương tự như trên đối với hàm OR và cũng rút ra kết luận không dùng được hàm này cho mục đích mã hóa hoặc muốn dùng phải thêm các điều kiện bổ sung.

e. Hàm Xor

Hàm Xor cần 2 tham số trở lên. Với 2 tham số x1, x2, ta có:

y=x1 Xor x2 như sau:

Bảng 2.

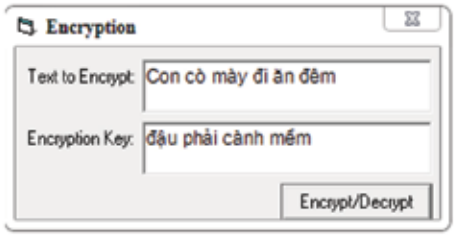
x1	x2	y(x1 Xor x2)	Số dòng
0	0	0	1
0	1	1	2
1	0	1	3
1	1	0	4

Bảng trên cho ta thấy y là hàm đơn trị đối với x1, x2.

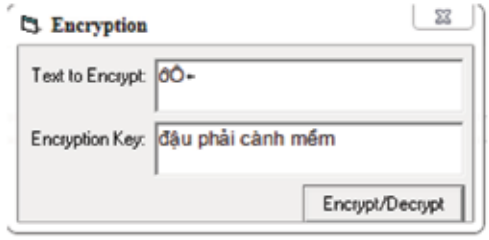
Một đặc điểm cần thấy: y=x1 Xor x2 thì y= x2 Xor x1 và x1=y Xor x2 và x2= y Xor x1.

Đặc điểm này của hàm cho phép dùng chung một hàm cho cả mã hóa và giải mã.

Ví dụ:



Thực hiện mã hóa bằng hàm Xor cho các kết quả như dưới dây. Phần mềm tạo Form [ ]:



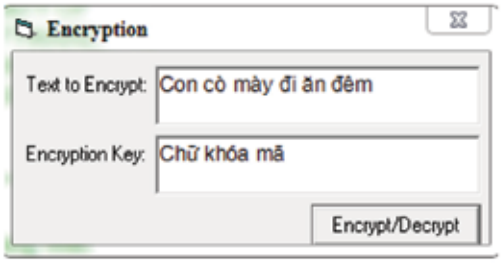
Khi nhấn vào Encrypt, ta mã hóa và được:

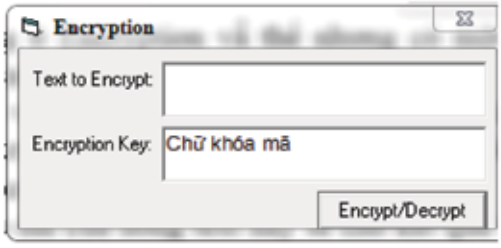
Số ký tự trong ô Encryption vẫn thế, nhưng có một số không hiện hình. Nếu nhấn vào decryption, ta lại được text ban đầu.

3. Xử lý lỗi

Khi Encryption Key hơi khác đi một chút, phần mềm này cho kết quả sai. Ví dụ: khi cho Encryption Key là "Chữ khóa ma", sẽ cho

kết quả như bên phải. Ở ô Text to Encrypt text, mã hóa đã biến mất.





Nếu tính chiều dài của dòng text này, sẽ cho kết quả 0.

Điều này chứng tỏ thuật toán đã không làm việc bình thường.

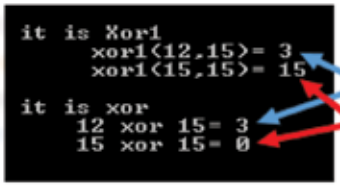
Để xử lý lỗi, ta cần phải thay đổi thuật toán cho phù hợp.

Cần đưa thêm một phép toán Xor1 khác biệt so với Xor như hàm trong C++ dưới đây:

Phần mềm sử dụng thuật toán Xor1 như sau:

```
int xor1(int a,int b){ if (a==b) { return a;}
else {return a xor b;}}
```

Thực nghiệm trong C++ như sau:



Khi chạy phần mềm với Xor và Xor1, ta được như hình bên.
Với a<>b ta có a xor b= xor1(a,b)
Với a=b thì a xor b <> xor1(a,b).

Hình 4. Sự khác biệt giữa Xor và Xor1

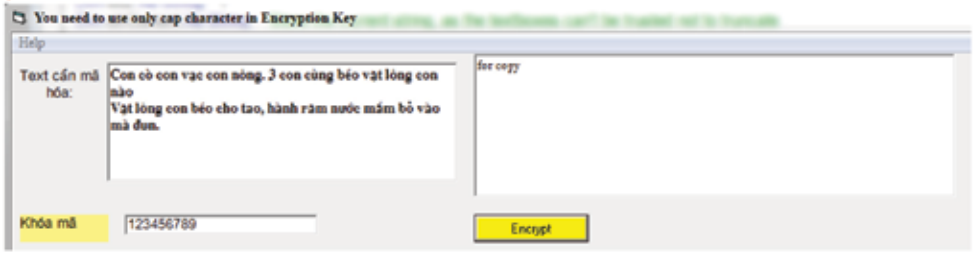
Phần mềm trong C++ là:

```
#include <iostream>
using namespace std;
int a,b,c;

int xor1(int a,int b){
   if (a==b) {
    return a;
        }
    else {
    return a xor b;
    }
   }
int main()
{
   cout<<endl;
   cout<<" it is Xor1"<<endl;
   a=12;
   b=15;
   c=xor1(a,b);
   cout<<"   "<<" xor1("<<a<<","<<b<<")=
              "<<c<<endl;
   a=15;
   b=15;
   c=xor1(a,b);
   cout<<"   "<<" xor1("<<a<<","<<b<<")=
              "<<c<<endl;
   cout<<endl;
   cout<<" it is xor"<<endl;
   a=12;
   b=15;
   c=a xor b;
   cout<<"   "<<a <<" xor " <<b<<"=
              "<<c<<endl;
   a=15;
   b=15;
   c=a xor b;
   cout<<"   "<<a <<" xor " <<b<<"=
              "<<c<<endl;
   cin.get();
```

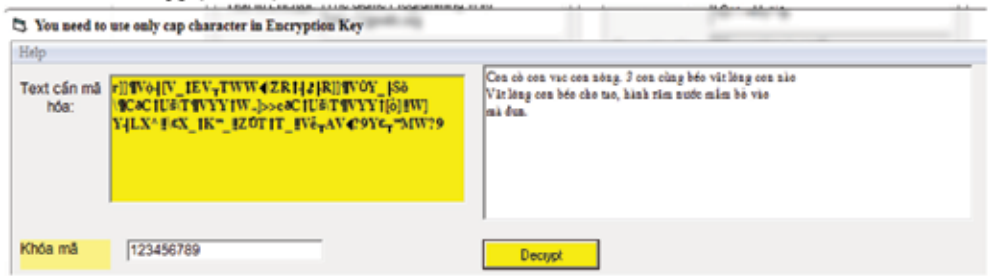
```
    cin.get();
    return 0;
}
```

4. Xây dựng thuật toán

Tạo Form trong VB6, ta có:



Nhấn vào Encrypt, ta được



Hình 5. Giao diện mã hóa và giải mã

Khóa mã ở đây là 123456789…

Kết luận: Khảo sát 2 hàm logic và có các thực nghiệm để áp dụng chúng cho mã hóa.

Tiềm năng của 2 hàm này rất lớn thông qua xác định số khóa mã vô cùng lớn.

Độ bảo mật thông tin:

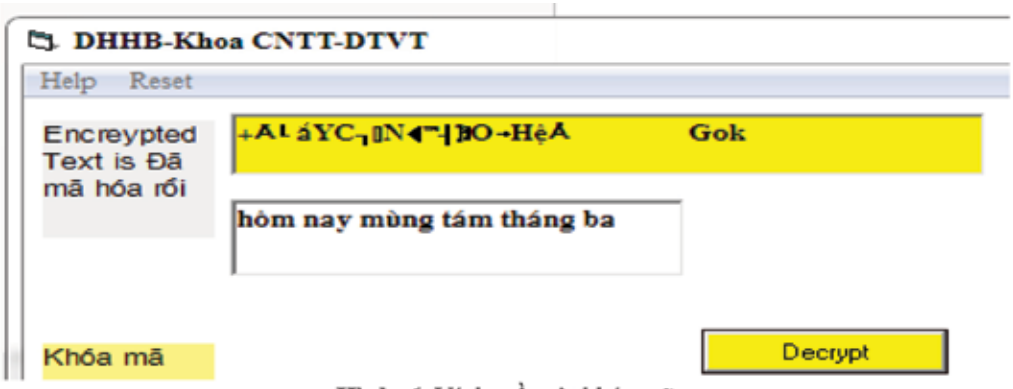
Áp dụng thuật toán Xor và Xor1.

Có dòng text, đánh dấu từng ký tự của dòng text này thông qua biến x, x(i), trong đó i là thứ tự của ký tự trong dòng text ban đầu.

Để mã hóa, dùng hàm Xor1, ta dùng dòng thứ 2, tùy chọn và nó đóng vai trò chìa khóa mã, ký hiệu là k, cũng đánh thứ tự k(i). Giả sử số ký tự của 2 dòng như nhau.

Khi đó: y(i)=x(i) Xor k(i) sẽ là dòng chữ đã được mã hóa. Rõ ràng, từ y(i) muốn tìm x(i), cần có k(i).

Sau mã hóa, ta được:



Hình 6. Ví dụ về một khóa mã

Nếu giải mã, ta sẽ được giá trị ban đầu là "Con cò con vạc con nông".

Nếu thay đổi bất kỳ ký tự nào trong khóa mã "Hôm nay mùng tám tháng ba", giải mã sẽ không cho dòng trên.

5. Độ bảo mật

"Hôm nay mùng tám tháng ba" là khóa mã. Không biết khóa mã, gần như không thể giải mã được. Khả năng giải mã được khi không có khóa mã của một thuật toán mã hóa gọi là độ bảo mật của phương pháp mã hóa [3,6]. Người ta cho rằng, độ bảo mật này tỷ lệ thuận với số lượng khóa mã có thể có đối với thuật toán mã hóa bất kỳ.

Hiện tại, máy tính có khoảng 70 ký tự có thể gõ từ bàn phím. Mỗi ký tự trở thành 1 khóa mã cho một ký tự của text. Như vậy, với mỗi ký tự, ta có 70 khóa. Dòng chữ với 20 ký tự sẽ có 70^20=7.9792266297612E+36 khóa mã khác nhau.

Nếu khóa mã không dùng trực tiếp là ký tự, mà dùng các con số sao cho cũng vẫn biến chúng thành ký tự thì độ bảo mật có thể tăng lên đáng kể. Chúng ta có thể dùng là các con số từ 1 đến 255, khi đó, phần mềm mã hóa thay đổi đôi chút:

Khi dùng các con số là mã ascii của các ký tự, ví dụ 1,2,3,4,11,34,56,32,21,99,125,130,205,255 để thay cho việc gõ từ bàn phím, ta được dòng key như sau:

Hình 7. Sử dụng mã khóa dạng số

Nhìn key bây giờ đã có những ký tự không thể gõ được bằng bàn phím. Tổng số key như thế sẽ là 255. Cũng với dòng chữ 20 ký tự như trên, bây giờ, số khóa mã sẽ là 255^20 = 1.35146128375559E+48 khóa mã. Nó vượt xa con số trước đây, độ bảo mật đã tăng lên đáng kể!

Với khả năng của máy tính hiện nay, cho dù độ bảo mật được xác định như trên có vẻ như chưa phải là trở ngại chính, nhưng xét đến các yếu tố khác nữa, có thể kết luận, nếu chọn khóa mã hợp lý, việc bẻ khóa khi không có khóa mã khi áp dụng thuật toán Xor1 hợp lý là điều không thể.

Tài liệu tham khảo:

[1]. http://gpwiki.org RYAN CLARK.

[2]. *Баричев Сергей,* КРИПТОГРАФИЯ БЕЗ СЕКРЕТОВ.

[3]. В. В. Ященко ,Введение в криптографию, Москва Издательство МЦНМО-2012.

[4]. Баричев С. Криптография без секретов, IMHO.

[5]. https://ozlib.com/1004780/tehnika/diskretnye_logarifmy_konechnom_pole.

[6]. Nguyễn Xuân Dũng (2007), *Bảo mật thông tin - Mô hình & ứng dụng*, Nxb. Thống kê, Hà Nội.