CSE 4803: Graph Theory

Application of Euler Digraph in Cryptography

IUT CSE 18, Section 1

Longest Circular Sequence

(without repeating words)

Stream Cipher

Encryption

Key Stream is generated from a pseudorandom generator

Plain Text :	10011001
Key Stream :	11000011
Cipher Text: (XOR)	01011010

Stream Cipher

Key Generation Condition

- Pseudo-random
- Non-repeating
- Long

de Bruijn Sequence

- Sequence of length **k^n** from **A**
- Every string of length **n** occurs only once

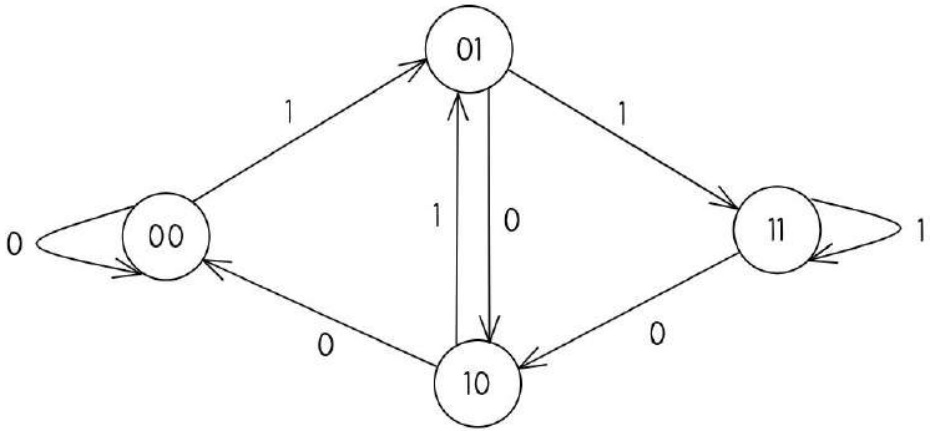
- B (2, 3) = **00010111**
  = **11101000**

- **n-ordered de Bruijn Graph** with (n-1) string size and alphabet size **k** generates **B(k, n)**

Alphabet,	A	=	{0, 1}
Size,	k	=	2
Order,	n	=	3
Sequence		=	B(k, n)

de Bruijn Graph (n ordered)

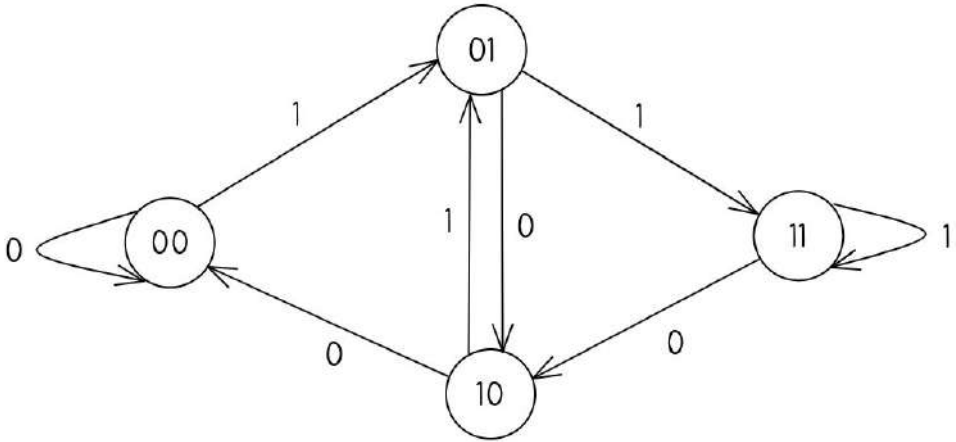
- Vertices = (n-1)-length unique strings from alphabet A
- Edges = Alphabets from A (Append)
- **Eulerian Directed Graphs**



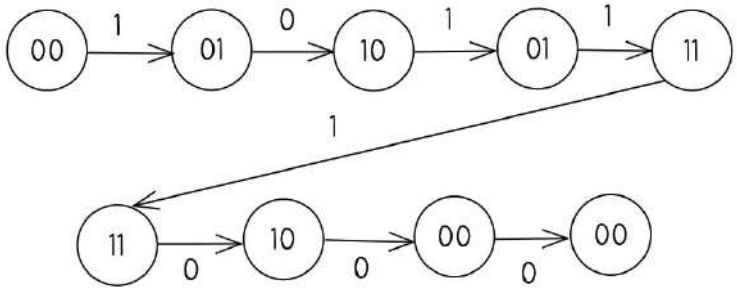
3-ordered de Bruijn graph

de Bruijn Graph (Sequence Generation)

- Find a Euler Circuit
- Edge values of the Euler Circuit is a de Bruijn sequence



3-ordered de Bruijn graph



Euler Path = **10111000**

Encryption & Decryption

Encryption

- Predefined coordination for same de Bruijn sequence [B(k,n) & Euler Circuit]

- Plain Text: 11111111
  Key Stream: **10111000**

  Cipher Text: 01000111

- Key = (2, 3) => B(2, 3)
- Sequence starting with 00

Decryption

- Predefined coordination for same de Bruijn sequence [B(k,n) & Euler Circuit]

- Cipher Text: 01000111
  Key Stream: **10111000**

  Plain Text: 11111111

- Key = (2, 3) => B(2, 3)
- Sequence starting with 00

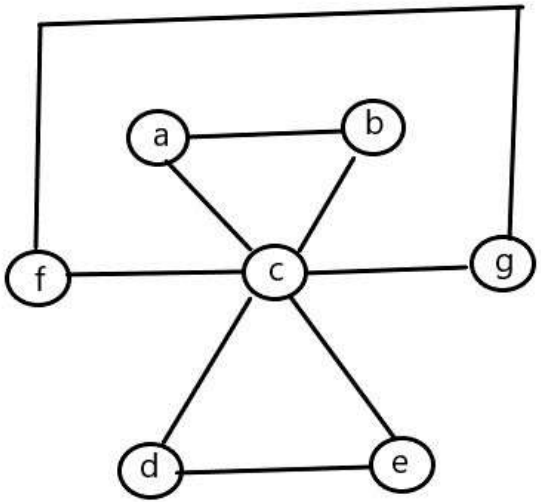
Data Encryption Basing on the Existence of Eulerian Circuit in a Group of Random Graphs

(ICTCS,2021)

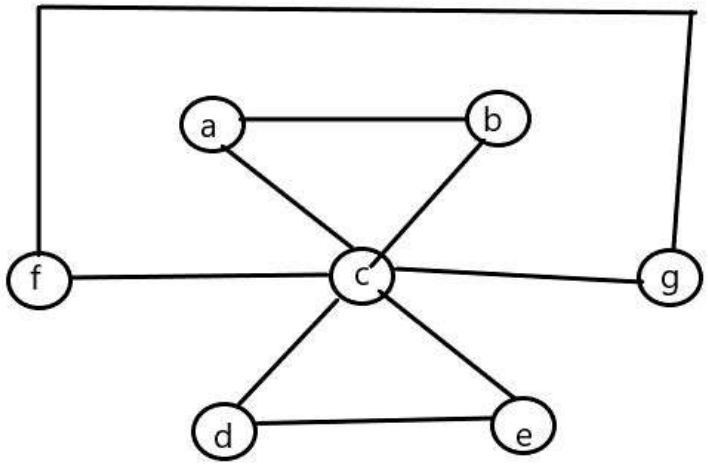
Introduction

- An Euler circuit can be drawn in many ways using different vertices of the euler graph in different directions.
- A weighted euler graph with all of it’s euler circuits representation is our KEY.
- KEY can be produced by user or any central authority, authors did not specify.
- Proposed Encryption is Symmetric.

Two Different Euler Circuit Representation



ab-bc-cd-de-ec-cf-fg-gc-ca
2 - 2 - 6 - 2 - 2 - 6 - 2- 2- 6



ec-ca-ab-bc-cf-fg-gc-cd-dc
2 - 6- 2 - 2 - 6 - 2- 2- 6 - 2

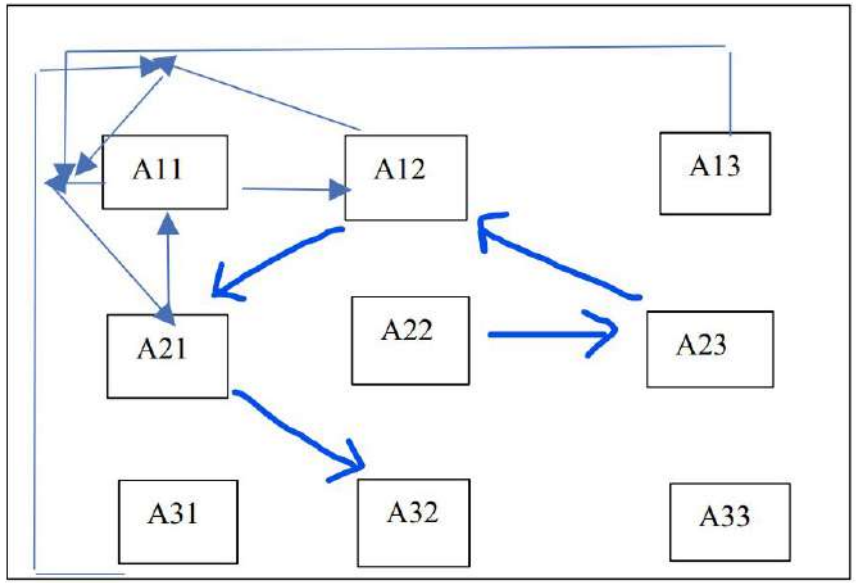
Figure 1:Different Euler Circuit Representation

Encryption

1. Split plain text into blocks of length equal to #edges specified.
2. Convert each character to ASCII of 8-bit, reduced to mod 256 if necessary.
3. Each binary number undergo right circular shift operation, and the number of times equal to degree of the corresponding vertex of the key.
4. Let's say first block first character is E = 69 = 0100 0101, then according to figure 1, we need to do 2 circular right shift. So 0100 0101 becomes 0101 0001 = 81.
5. Make a nxn matrix where n^2 = #edges.
6. Do this aij XOR ai(j + 1) XOR a(i−1) j XOR ai(j−1) XOR a(i + 1)j and replace a[i][j] with the calculated value.
7. Convert the binary to ASCII characters, which is our cipher text for this block.
8. Do 2 to 7 for each block and send those cipher text serially.

Right Shift & XOR

1	0	1	1	0	0	1	0
0	1	0	1	1	0	0	1
1	0	1	0	1	1	0	0



Example

- We have plain text “EULER XOSS” and “GRAPHS FUN” for two circuits given in figure 1 respectively.
- EULERXOSS = 69 | 85 | 76 | 69 | 82 | 88 | 79 | 83 | 83
- E = 69 = 0100 0101 >2= 0101 0001 = 81
- U = 85 = 0101 0101 >2= 0101 0101 = 85
- L = 76 = 0100 1100 >6= 0011 0001 = 49
- Doing other similarly and convert it to matrix 3x3 we get

- Then applying XOR operation we get

XOR C++ Code

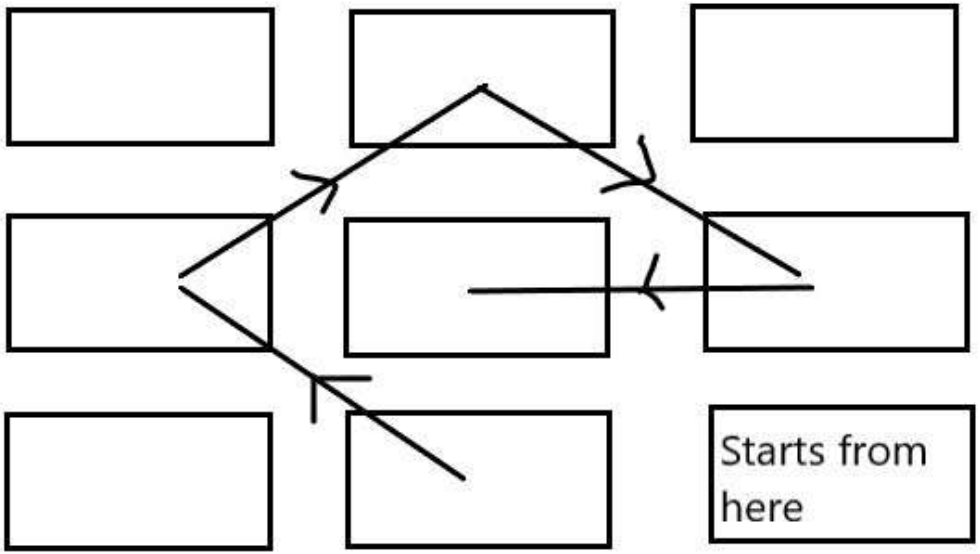
```
void encrypt()
{
    cout<<"encrypt"<<'\n';

    for(int i=0;i<3;i++)
    {
        for(int j=0;j<3;j++)
        {
            int vv = ar[i][j];

            vv ^= ar[i][(j+1)%3];

            if(i-1 < 0) vv ^= ar[i-1+3][j];
            else vv ^= ar[i-1][j];

            if(j-1 < 0) vv ^= ar[i][j-1+3];
            else vv ^= ar[i][j-1];

            vv ^= ar[(i+1)%3][j];

            ar[i][j] = vv;
        }
    }

    for(int i=0;i<3;i++)
    {
        for(int j=0;j<3;j++)
        {
            cout<<ar[i][j]<<'|';
        }
        cout<<'\n';
    }
    cout<<'\n';
    cout<<"decrypt"<<'\n';
}
```

Decrypt

- We convert the ASCII cipher text to binary values
- Then convert those value to a nxn matrix.
- Apply XOR operation in opposite order. Do this a(i+1)j XOR ai (j-1) XOR a(i-1)j XOR ai(j+1) XOR aij and replace a[i][j] with the calculated value. While encryption we go 0,0 to 2,2 but in decryption we go 2,2 to 0,0.
- Then Apply LEFT circular shift operation, and the number of times equal to degree of the corresponding vertex of the key.
- Convert each element of the matrix to ASCII, and we get our plain text back.

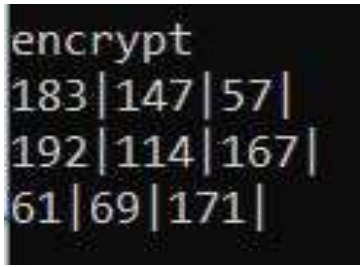
Left Shift & Opposite XOR

1	0	1	1	0	0	1	0
0	1	0	1	1	0	0	1
1	0	1	0	1	1	0	0

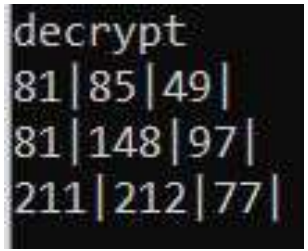


Example

- So after converting each character to ASCII and forming a 3x3 matrix, we get



- Using reverse XOR operation we get
  It's the same as input matrix while en



- 81 = 0101 0001 <2= 0100 0101 = 69 = E
- 85 = 0101 0101 <2= 0101 0101 = 85 = U
- 49 = 0011 0001 <6= 0100 1100 = 76 = L
- If we do for other values we will get EULERXOSS.

Opposite XOR C++ Code

```
void decrypt()
{
    for(int i=2;i>=0;i--)
    {
        for(int j=2;j>=0;j--)
        {
            int vv;

            vv = ar[(i+1)%3][j];

            if(j-1 < 0) vv ^= ar[i][j-1+3];
            else vv ^= ar[i][j-1];

            if(i-1 < 0) vv ^= ar[i-1+3][j];
            else vv ^= ar[i-1][j];

            vv ^= ar[i][(j+1)%3];

            vv ^= ar[i][j];

            ar[i][j] = vv;
        }
    }

    for(int i=0;i<3;i++)
    {
        for(int j=0;j<3;j++)
        {
            cout<<ar[i][j]<<'|';
        }
        cout<<'\n';
    }
}
```

Conclusion

- We do GRAPHS FUN using the second euler circuit of figure 1.
- We can convert same block of plain text to a different cipher text according to a different representation of euler circuit.
- So we can not predict key using cipher also can not predict plain using cipher, so this scheme satisfies both confusion and diffusion property of encryption.
- The above proposed scheme is very fast than well known Etaiwai scheme.
- Etaiwai scheme use concept of cyclic graphs, complete graphs and minimum spanning tree.