Micro-bit 在 18 世纪的应用之旋转隔栅加密

李润家 陈誉霄

摘要：

旋转隔栅加密法是一种在 18 世纪颇为流行的短信息加密方法，曾被广泛应用于多次战争中，知道一战前才慢慢淡出了历史舞台，本项目使用 microbit 来实现此种加密方法，让加密与解密过程变得更加方便准确，同时展示此中加密方法相对于传统移位密码的关于短信息加密的优越性。

一、选题及创意介绍：

旋转隔栅加密最早出现在美国开国元勋亚历山大-汉密尔顿的一份文件中，1796 年兴登堡在《荷兰共和国的密码学和治国方略》首次对这一文献作了全面的描述，旋转隔栅密码逐渐被世人熟知和使用。由于其对于短信息操作的简便性和难以破解性，让其非常适用于战争前线的加密，在克里米亚战争中被广泛使用。

相比于换位密码的加密方式每加密一个字符都要查表（假设加密条件只有纸币和人脑），旋转隔栅加密只用在卡纸空出来的格子里顺序填数即可，对于短信息准确性高，速度快。同时旋转隔栅加密作为一种独特的加密方法很好的体现了简单的操作也可以带来巨大的效果这一观点，与 microbit 的存在不谋而合。

然而不幸的是，和凯撒密码等换位加密法相反，随着历史的发展，旋转隔栅已经越来越不为人所知。但是本小组成员认为其有趣的加密方式和对短信息所表现出来的超高效率值得用 microbit 进行再实现和改进。所以决定进行这样一个项目。

项目中将利用 microbit 实现旋转隔栅加密的过程的可视化，并利用无线电功能进行收发，从而实现完整的通讯过程，得到一个可用有效的加密通讯方式。

二、设计方案和硬件连接：

旋转隔栅加密的实现方案：

4*4 方格表，以及一个 4*4 方格表挖掉 4 个格形成的 “卡纸”，挖掉的四个方格处于不同的旋转轨道上（无法通过绕中心旋转得到另外一个）。明文是一个 16 位字符串，先将卡纸和方格表重合摆放，依次将明文前四位填入空中（空之间的先后次序提前约定好，这里采用先填左上的约定），旋转卡纸（这里约定顺时针）而方格表不动，再重复填入和旋转知道方格表被填满（明文 16 个数全部填入）。这时按照一定次序读出（约定先从上向下读第一列，在从上向下读第二列…）就得到了加密之后的密文。

演示如图：

如果要加密 “ABCDEFGHIJKLMNOP “

选取的卡纸，第一次写入，第二次写入，第三次写入，最后一一次

硬件连接：

2 块 microbit 主板，无需任何扩展

利用 microbit 前 4*4 表示卡纸，通过灯的亮灭来表示旋转，可视化加密过程。

三、 实现方案及代码分析：

两块 microbit，存储同样的程序。按 A 进入发送状态，按 B 进入接收状态。

按 A 后：

进入输入明文环节，明文以十进制方式输入，最多可以输入 8 位。按 B 调整数字，按 A 确认

全部输入完后不断按 B 直到显示 X，按 A 确认，输入完成

进入选择位置环节，按 A 调整位置，按 B 确认。选择四次

显示明文转化为四进制以后的 16 位（不足向后补零），增加校验位以显示补了多少个 0.

可视化加密过程，包括卡纸的旋转和数字的一次填入

进行配对：显示✅表示配对成功

逐位发送密文

按 B 后：

进入选择位置环节，同上面的选择位置

等待配对

配对成功，逐位接受明文

解密（不显示解密过程）

显示明文

代码：

```
from microbit import *
import radio
N0=Image("06660:06060:06060:06060:06660")
N1=Image("00600:06600:00600:00600:06660")
N2=Image("06660:00060:06660:06000:06660")
N3=Image("06660:00060:06660:00060:06660")
N4=Image("06060:06060:06660:00060:00060")
N5=Image("06660:06000:06660:00060:06660")
N6=Image("06000:06000:06660:06060:06660")
N7=Image("06660:00060:00060:00060:00060")
N8=Image("06660:06060:06660:06060:06660")
N9=Image("06660:06060:06660:00060:00060")
Stop=Image("60006:06060:00600:06060:60006")
Nlst=[N0,N1,N2,N3,N4,N5,N6,N7,N8,N9,Stop]

```

初始化，数字图片的输入

```
lst1=[(0,0),(0,3),(3,3),(3,0)]
lst2=[(1,0),(0,2),(2,3),(3,1)]
lst3=[(2,0),(0,1),(1,3),(3,2)]
lst4=[(1,1),(1,2),(2,2),(2,1)]

def nextt(x):
    for lst in [lst1,lst2,lst3,lst4]:
        if x in lst:
            return lst[lst.index(x)-1]

def turn(lst):
    return [nextt(x) for x in lst]

```

允许对位置（二元数组）进行旋转

```
def shuru():
    cin=""
    curN=0
    stop=False
    while not stop:
        if button_b.was_pressed():
            display.clear()
            curN=(curN+1)%11
        display.show(Nlst[curN])
        if button_a.was_pressed():
            if curN==10:
                stop=True
            else:
                cin+=str(curN)
                curN=0
    return cin

```

输入明文，按 B 当前数字+1，按 A 选择，如果当前为 Stop 且按 A 则结束

```
46  def Ten_To_Four(n):
47      out=""
48      while n!=0:
49          out=str(n%4)+out
50          n=n//4
51      return out
52  def Four_To_Ten(s):
53      out=0
54      for i in range(len(list(s))):
55          out+=int(list(s)[-i-1])*(4**i)
56      return out
```

进制转换

```
57  def pulse():
58          for i in range(5):
59              display.set_pixel(i,4,8)
60              sleep(10)
61
62          for j in range(4):
63              display.set_pixel(4,3-j,8)
64              sleep(10)
65          for i in range(5):
66              display.set_pixel(i,4,0)
67              sleep(10)
68
69          for j in range(4):
70              display.set_pixel(4,3-j,0)
71              sleep(10)
72
```

装饰，在非 4*4 区域显示脉冲

```
def choose_pos():
    pos=[]
    cur=0
    for x in [lst1,lst2,lst3,lst4]:
        display.set_pixel(x[0][0],x[0][1],5)
        display.set_pixel(x[1][0],x[1][1],5)
        display.set_pixel(x[2][0],x[2][1],5)
        display.set_pixel(x[3][0],x[3][1],5)
        while not button_b.was_pressed():
            pulse()
            display.set_pixel(x[cur][0],x[cur][1],9)
            sleep(50)
            display.set_pixel(x[cur][0],x[cur][1],0)
            sleep(50)
            if button_a.was_pressed():
                display.set_pixel(x[cur][0],x[cur][1],5)
                cur=(cur+1)%4
        display.set_pixel(x[0][0],x[0][1],0)
        display.set_pixel(x[1][0],x[1][1],0)
        display.set_pixel(x[2][0],x[2][1],0)
        display.set_pixel(x[3][0],x[3][1],0)
        pos.append(x[cur])
    pos=sorted(pos)
    return pos
```

选择位置，四个列表就是四个旋转类，按 B 选择，按 A 对当前进行旋转

返回一个四个二元数组组成的列表，即为卡纸被挖去的格子

```
def showw(pos):
    for x in range(4):
        for y in range(4):
            if (x,y) not in pos:
                display.set_pixel(x,y,7)
```

显示卡纸

```
#s长为4
def tick(pos,s):
    for i in range(5):
        display.set_pixel(i,4,0)

    if len(s)==1:
        display.set_pixel(0,4,int(s[0]))
        sleep(1000)
        display.set_pixel(0,4,0)
        display.set_pixel(pos[0][0],pos[0][1],int(s[0]))
        sleep(1000)
    else:
        for i in range(len(s)):
            display.set_pixel(i,4,int(s[i]))
        sleep(1000)
        display.set_pixel(pos[0][0],pos[0][1],int(s[0]))
        tick(pos[1:],s[1:])
```

定义了一个辅助函数，可以将长为四的字符串逐位填入到卡纸空着的地方中，次序为左上优先

```
def Encrypt(cin4,pos):
    mingwen=cin4+"0"*(16-len(cin4))
    display.show(Image("00000:00000:00000:00000:00000"))

    for i in range(4):
        for j in range(4):
            display.set_pixel(i,j,3*int(mingwen[4*i+j]))
            sleep(400)
```

加密部分一：显示明文（按列从上向下）

```
    #装饰
    display.show(Image(
        "00000:"
        "00900:"
        "09990:"
        "00900:"
        "00000"))
    sleep(1000)
    display.show(Image(
        "00000:"
        "09990:"
        "09990:"
        "09990:"
        "00000"))
    sleep(1000)
    display.show(Image(
        "90909:"
        "09990:"
        "99999:"
        "09990:"
        "90909"))
    sleep(1000)
```

装饰

```
    display.show(Image("00000:00000:00000:00000:00000"))
    sleep(500)
    dic={}
    curpos=pos
    for i in range(4):
        showw(curpos)
        sleep(1000)
        s=[3*int(x) for x in mingwen[4*i:4*i+4]]
        tick(curpos,s)
        for x in curpos:
            dic[x]=display.get_pixel(x[0],x[1])
        curpos=turn(curpos)
        display.clear()
        sleep(200)

    sleep(1000)
```

可视化加密过程，具体就是使用 turn 函数旋转卡纸，再用 tick 函数逐位填入，并将格子中的值使用 get_pixel 写入到方格表中（这里用字典来实现）

```
    miwen=""
    for x in range(4):
        for y in range(4):
            miwen+=str(dic[(x,y)]//3)
            sleep(30)
    miwen=miwen+"_"+str(len(str(cin4)))
    return miwen
```

返回密文，由于显示实现的原因将数值除 3，添加了校验位

```
def Decrypt(miwen,pos):
    l=int(miwen[17])
    dic={(x,y):0 for x in range(4)
                 for y in range(4)}
    for x in range(4):
        for y in range(4):
            dic[(x,y)]=int(miwen[4*x+y])
    mingwen=""
    curpos=pos
    for i in range(4):
        for x in curpos:
            mingwen+=str(dic[x])
        curpos=turn(curpos)
    return mingwen[0:l]
```

解密：逆转加密过程，卡纸第一个位置时的第一二三四个空就是明文的第一二三四位，第二个位置时的第一二三四个空就是明文的第五六七八位，以此类推，其中使用校验位来判断哪些 0 是明文自带的哪些是为了凑 16 位而补上的

```
def A():#加密
    cin=int(shuru())
    cin4=Ten_To_Four(cin)
    display.clear()
    pos=choose_pos()
    miwen=Encrypt(cin4,pos)
    display.scroll(miwen)
    sleep(3000)
    display.scroll("sending")
    radio.on()
    radio.send(miwen[miwen.index("_")+1:])
    recive=None
    while not recive:
        display.show(Stop)
        recive=radio.receive()
        sleep(500)
        display.clear()
    display.show(Image("00000:00009:00090:90900:09000"))
    sleep(1000)
    for x in miwen[:miwen.index("_")]:
        display.show(x)
        radio.send(x)
        sleep(1000)
        display.clear()
        sleep(100)
```

最终加密分支程序，在最开始按 A 进入，具体函数功能见前面叙述

```
def B():
    radio.on()
    pos=choose_pos()
    while True:
        countt=radio.receive()
        if countt:
            display.show(Image("00000:00009:00090:90900:09000"))
            sleep(500)
            radio.send("hello")

            miwen=""
            c=1
            while c<=16:
                x=radio.receive()
                if x:
                    display.show(x)
                    sleep(1000)
                    display.clear()
                    sleep(100)
                    miwen+=x
                    c+=1
            mingwen=Four_To_Ten(Decrypt(miwen+"_"+countt,pos))
            display.scroll(mingwen)
            sleep(10000)
            break
        else:
            showw(pos)
```

最终解密分支程序，最开始按 B 进入

```
    display.scroll("sending")
    radio.on()
    radio.send(miwen[miwen.index("_")+1:])
    recive=None
    while not recive:
        display.show(Stop)
        recive=radio.receive()
        sleep(500)
        display.clear()
    display.show(Image("00000:00009:00090:90900:09000"))
    sleep(1000)
    for x in miwen[:miwen.index("_")]:
        display.show(x)
        radio.send(x)
        sleep(1000)
        display.clear()
        sleep(100)
```

发送设备的校验及逐位发送

```
while True:
    countt=radio.receive()
    if countt:
        display.show(Image("00000:00009:00090:90900:09000"))
        sleep(500)
        radio.send("hello")

        miwen=""
        c=1
        while c<=16:
            x=radio.receive()
            if x:
                display.show(x)
                sleep(1000)
                display.clear()
                sleep(100)
                miwen+=x
                c+=1
```

接收设备的校验及逐位接收

```
254
255 while True:
256     if button_a.was_pressed():
257         A()
258     if button_b.was_pressed():
259         B()
260
```

最终程序

四、 后续工作展望：

1、 每次都要重新手动选取位置显得有些麻烦，而且发送者和接受者都需要输入增加了出错的可能性，可以使用 RSA 加密的方式，发送者在发送密文的同时发送用 RSA 加密后的卡纸，这样解密者拥有正确的素因子分解就可以准确无误的得到卡纸。但是这个方案的问题在于还不如用 RSA 直接加密密文，不过这个设想还是挺有趣的。

2、 在输入过程中增加后退见，以防止输入错误需要重新输入的情况

3、 为了防止有学生利用这个考试作弊，结合 microbit 无限电发送人人均可接受的特性，可以为老师配备一款破解机型，问题在于破解的计算量会比较大，还需要结合自然语言处理的知识，超出了我们的任务范围。

总结：

通过这次项目，首先我们对在硬件上编程有了新的认识，发现在硬件上编程也不想之前想象的那么可怕，甚至可以说是很有趣。很遗憾由于程序过于简单以及二代 microbit 的加强并没有出现 memory error 的情况，不过我们也同时感受到了好的思想想法，就算很简单，也能带来巨大威力的这个道理。旋转隔栅加密法说起来也就寥寥数行字，但是它的准确易行高效性让他在历史上被广泛应用。生活中很多事情也是这样的，就像 RSA，就像课上我们学习的各种排序算法，好的思想，合适的实现方式，将会带来超越其载体的巨大威力，这也许就是计算机科学（密码学）吸引人的地方！

五、 小组分工合作：

李润家：组长，方案设计，初步编程，部分报告撰写

陈誉霄：方案设计，程序调试，部分报告撰写，制作视频