**Project 2 Verman密码的破解**

# 1 实验内容

Vernam加密法也称一次一密(One-Time-Pad)，用随机的非重复的字符集合作为输出密文。这里最重要的是，一旦使用了变换的输入密文，就不再在任何其他消息中使用这个输入密文（因此是一次性的）。输入密文的长度等于原消息明文的长度。Verman密码的优点是，由于使用与消息等长的随机密钥，产生与原文没有任何统计关系的随机输出，因此一次一密方案不可破解，但也存在缺陷，密钥在传递和分发上存在很大困难；对于很长的字符串，加密很困难。

课程文件中提供了6段加密过的字符串，6段字符串总共使用了3个随机密钥。其中每个密钥都恰好使用了两次。6段明文分别是一段代码、一段莎士比亚的引言、一段歌词、三段其他的英语名言。

# 2 实验过程

考虑到Vernam密码的特点，当两个等长的明文字符串A和B使用了同一个密钥K时，生成相对应的密文分别为OUTA=A⊕K，OUTB=B⊕K。根据异或运算的交换律，OUTA⊕OUTB = (A⊕K)⊕(B⊕K)，即两个密文异或之后的结果中不再包含密钥K的信息，这时候问题就简化为已知A⊕B分别求消息明文A和B。这个问题从数学上来说给定的条件不足的，但考虑到明文为英语句子（和一段代码），所以利用英语语言（和程序代码）上的先验信息便有可能把明文破解出来。

## 2.1 密文分组

这一节首先根据每段密文所使用的密钥是否一样进行分组。

根据实验给出的条件，6段密文共使用了3个随机密钥，其中每个密钥都恰好使用了两次。给定两段密文，如果使用了同一个密钥，则它们之间按位异或的时候，大部分字符都是在英文字母之间进行异或，而如果使用了不同的密钥，则相当于在上面的结果基础上再跟两密钥异或的结果再进行异或，所以英文字母之间异或的概率就大大降低了。因此有必要研究一下英文字母之间进行异或的特点，作为密文分组的依据。

图1是7位的ACSII码表。不难发现，相同字母大小写形式的后4位是相同的，实际上前2位也是相同的，只有第3位是0或者1来区分大写和小写。因此英语句子之间进行按位异或得到的结果大概率还是字母（例外情况如，英文句子里存在的标点符号跟字母进行异或就可能不是字母，相同的字母做异或就得到空字符或者空格）。利用这个特点，就可以把6段密文两两做异或，然后统计异或结果中字母占字符个数的比例，其中比例最高的一对就可以认为是使用了同一个密钥。



**图1 7位ACSII码表**

为了简单起见，实际情况只统计了小写字母的比例（毕竟英文句子里小写字母是最多的）。异或结果的统计情况如表1所示。由于其中有一段明文是代码，不完全符合上述的统计规律，所以结果中某一对密文异或的结果中小写字母相对较少是正常的（即密文4和6）。根据统计结果可以初步推断，密文1和密文3使用了用一个密钥为一组，密文2和密文5为另一组，而密文4和密文6为代码所在的一组。

表1 密文之间两两异或得到的结果中小写字母所占的比例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 密文1 | 密文2 | 密文3 | 密文4 | 密文5 | 密文6 |
| 密文1 | 1 | 0.2031 | 0.6406 | 0.2734 | 0.2031 | 0.3125 |
| 密文2 | 0.2031 | 1 | 0.2891 | 0.2656 | 0.5703 | 0.2813 |
| 密文3 | 0.6406 | 0.2891 | 1 | 0.2656 | 0.1797 | 0.2734 |
| 密文4 | 0.2734 | 0.2656 | 0.2656 | 1 | 0.3203 | 0.4922 |
| 密文5 | 0.2031 | 0.5703 | 0.1797 | 0.3203 | 1 | 0.2734 |
| 密文6 | 0.3125 | 0.2813 | 0.2734 | 0.4922 | 0.2734 | 1 |

## 2.2 密文2和密文5的破解

在上一节做密文之间的按位异或时有一个意外的发现，即密文2和密文5的异或结果不但出现了较高比例的小写字母，而且在结尾处出现了EAR tHE bEATLES的字符串。随机字符串组成刚好是有意义的英文字母组合的概率是非常小的，而且这个字母组合的大小写反了，不难推测这样的字符串是字母跟空格符按位异或得到的结果（空格符的2进制ACSII值为0010000与字母异或刚好改变了大小写）。显然密文2和密文5中有一个对应的明文的结尾是ear The Beatles，而另一个对应的明文的结尾填充了相同长度的空格符。

由于已知6段密文中有1段对应的明文是歌词，利用google搜索The Beatles（英国摇滚乐队）的歌中以ear结尾的歌词，便发现了明文出自于歌曲Do you want to know a secret中的一句歌词，再把它与密文2和5异或的结果进行异或，得到另一密文对应的明文是Albert Einstein的一句名言，结果如下（无法区分这两段明文分别对应密文2还是密文5，下同）。

Listen - Do you want to know a secret? Do you promise not to tell? whoa oh, oh Closer - Let me whisper in your ear -The Beatles

I never teach my pupils. . .I only attempt to provide the conditions in which they can learn. -Albert Einstein

**图2 密文2和密文5的破解结果**

## 2.3 密文1和密文3的破解

根据给定的条件，剩余的4段密文其中有一段对应莎士比亚的引言。受到上面已破解的密文2和密文5的形式启发，利用莎士比亚的姓和名“ William Shakespeare ”分别对剩余两组密文异或的结果进行滑窗按位异或。其中跟密文1和密文3的异或结果进行异或时，发现结果的末尾出出现了有意义的字母组合“k rules Neil Gaiman”，可以推测Neil Gaiman是该组其中一段明文的作者。同样利用google搜索Neil Gaiman（英国作家）以“k rules ”结尾的名言，找到了对应的明文，同理异或得到另一段明文的莎士比亚引言，结果如下。

And now go, and make interesting mistakes, make amazing mistakes, make glorious and fantastic mistakes. Break rules Neil Gaiman

Therefore, all hearts in love use their own tongues;

Let every eye negotiate for itself And trust no agent. William Shakespeare

**图2 密文1和密文3的破解结果**

## 2.3 密文4和密文6的破解

经过前面两组密文的破解，现在可以肯定代码存在于密文4和密文6之中了。代码中大量存在英语中不常见的字符，大大增加了破解的难度。

首先还是从这一组密文中对应英语句子明文的那一段入手。这里利用了wikipedia上的常见英语单词库（https://en.wikipedia.org/wiki/Most\_common\_words\_in\_English），用其中的高频单词对密文4和密文6的异或结果进行滑窗按位异或。这是一个穷举搜索的过程，会耗费大量的时间精力，目的是希望匹配到明文中出现的相同单词从而显示出代码段。

功夫不负有心人。利用单词穷举的方法在代码中发现了cookie这一单词，并且出现在明文的前几个字符处，可以推断cookie是代码开始时声明的变量。既然是变量，就会在后面的代码段中反复出现，于是利用类似上面的方法拿cookie做滑窗进行按位异或，得到了英语句子明文中的若干个单词，利用google搜索得到完整明文，结果如下。

C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do, it blows away your whole leg. -Stroustrup

{int cookie;char buf[80];printf("buf: %08x cookie: %08x\n", &buf, &cookie);gets(buf);if (cookie == 0x41424344) printf("win!");}

**图4 密文4和密文6的破解结果**

至此6段密文全部破解完成！

# 3 实验结论

这个实验最大的启示是，相同的密钥绝不能在同样的场合下反复使用，因为在第三者掌握一定先验知识的情况下是很有可能把密文破解的。