

**课 程 设 计 报 告**

**题目： 密码学课程设计**

**课程名称： 密码学课程设计**

**专业班级： 信息安全1901班**

**学 号： U201911657**

**姓 名： 李文重**

**指导教师： 汤学明**

**报告日期： 2021年10月12日**

**教师评语：**

**分数：**

**网络空间安全学院**

目录

[一、设计过程 1](#_Toc2411)

[1.1 SPN实现 1](#_Toc16601)

[1.2线性分析 1](#_Toc23490)

[1.3差分分析 2](#_Toc5980)

[1.4 SPN增强 3](#_Toc4029)

[1.5 RSA参数计算 4](#_Toc29826)

[1.6模重复平方 5](#_Toc11816)

[1.7中国剩余定理 6](#_Toc5187)

[1.8 PKCS7 7](#_Toc2041)

[1.9 彩虹表 8](#_Toc23862)

[二、 实验心得 9](#_Toc9492)

[三、对课程设计内容和过程的建议 10](#_Toc11436)

## 一、设计过程

### 1.1 SPN实现

（1）设计内容

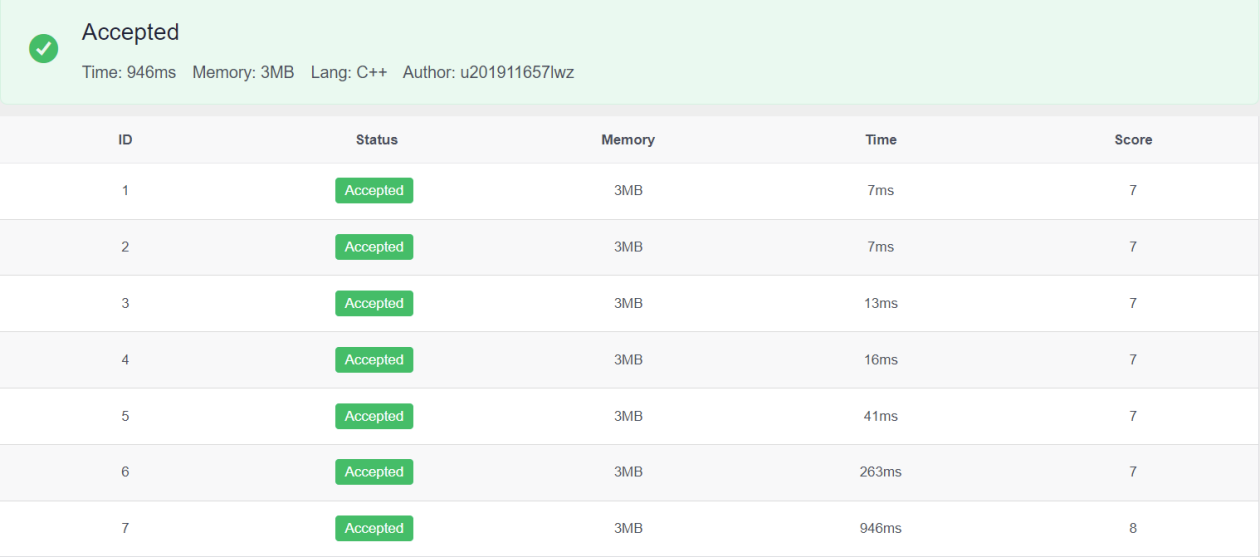
这个题目的要求是使用课本上给出的S盒和P盒，完成一个SPN，并且优化它的效率。OJ的评测方法则是给出一个明文和一个密钥，然后进行加密，把最后一bit取反之后再进行解密。输出两个操作之后的结果。最多可能会需要进行百万级别的加、解密操作。

（2）设计过程

最开始我的想法是没有考虑太多算法效率的问题，以为直接实现了网络就足够通过。结果就是运行效率惨不忍睹，根本没法通过。所以我就开始了优化，容易想到的优化首先是可以预处理小的S或者P盒为大的S盒，但是这样我原本设计的通过四个int型来存储四个半字节的方法就显得有些麻烦，而且每次都要进行合成再分解就会导致一笔不小的时间开销。因此我修改成直接使用一个int来存储中间步骤中的4个半字节，然后直接查表进行加解密。

这种方法比起之前速度已经快了不少，但是仍然不足以通过，时间差一点点。考虑到S、P盒在运用上实际上是很有规律的：在加密过程中，S盒经常和P盒连在一起；而在解密过程中反之。因此把他们的关系矩阵进行一个复合就能省下一半的时间。因此我就在原来的基础上做出了打的SP盒表，终于把时间卡到了1000ms以内。

（3）小结



我觉得这道题目我最大的不足之处就在于，为了尽可能地优化时间效能，而失去了一个原本很大的可扩展性。现在我的函数如果要处理更大的初始明文，或者更大的S，P盒就可能会有问题。因为更大的明文可能会挑战int型的存储上限，而更大的S、P盒则可能要求更大的空间去存储表格，因此只能进行代码的重构，而这样工作量就大得多了。而这个问题很快就会在第四题中出现。

### 1.2线性分析

（1）设计内容

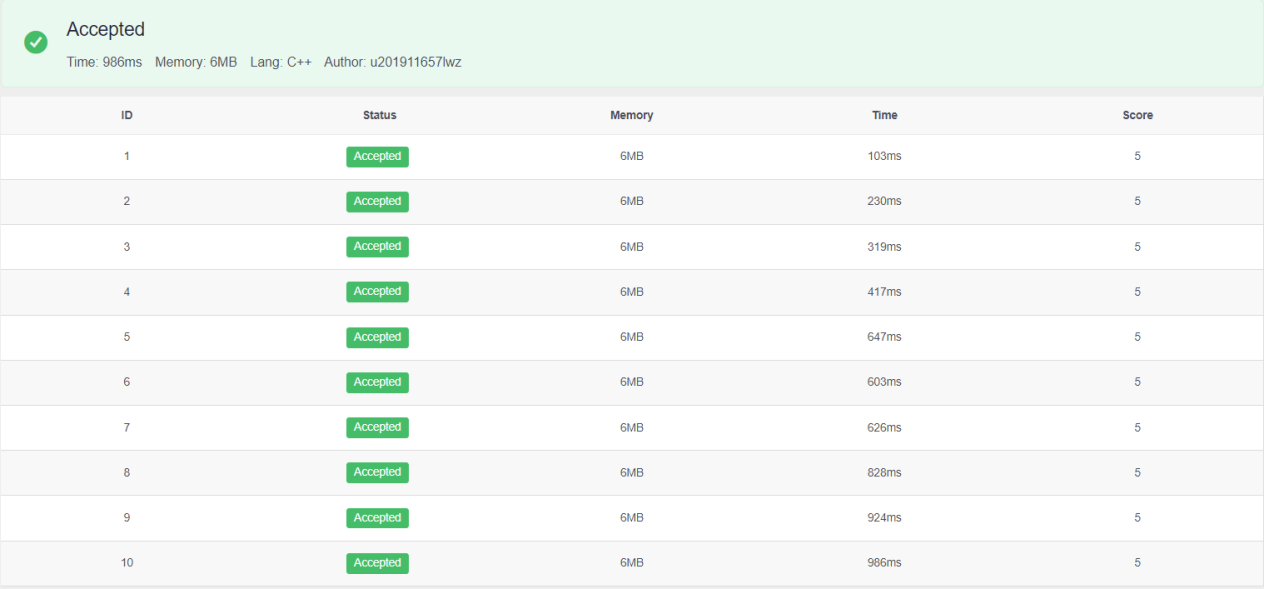
对于上文所示的SPN，要进行一次线性攻击，来进行密钥的破解。每一组需要破译的密钥都给出了8000组用此密钥进行加密的明密文对。而需要破译的组数可能会很大，达到1000组。

（2）设计过程

首先考虑到根据课本上的结论，8000组数据应该足以破解偏差达到的线性攻击，而如果偏差更小就很有可能破译不了。所以在选择线性攻击的时候偏不能太小，这就限制了很有可能并不能选到一组最理想的线性分析，即与第一组所分析的半字节正好互补，且偏差比较大。

因为自己最开始找线性分析并不是很顺利，往往都是最后有三个半字节，而且偏差小于的（虽然后来在实际测试中发现，如果三个半字节的密钥中有一个是固定且正确的的话，跑出来的结果甚至比预计的两倍还要高）。我想到利用计算机来完成这一部分的工作，基础思路也很简单，就以第一组的8000组明密文对，枚举线性攻击的掩码（或许可以这么说），然后判断实际中的偏差。如果对于明密文对这个掩码会导致比较大的偏差（比如其中偏差的最大值接近），那么就有理由相信现在找到的就是一组线性分析的结果。程序运行了几个小时之后，给出了几个还算不错的答案，于是我就选出了其中的一个作为第二组线性攻击。这个攻击依然包含了U4中的三个字节，但是已经足够了。在此基础之上，这道题还有一部分面向数据的调参，比如要枚举前多少大的偏差对应的密钥，都不是正确密钥才能认为当前是失败的；又比如每个密钥的验证需要进行几次加密都成功才能被判断为成功。最终我面向数据（？）的找到了一个能保证正确性前提下的最少的次数。

1. 小结



这个题目虽然险之又险的通过了，但是因为我是面向数据改的参数，所以这个程序很有可能并不具有通过随机测试的能力。而且同样的这道题目我的代码也存在难以适应其他规模SPN的问题。不过这道题目应该是极大的锻炼了我的卡常数意识和能力，在最初的代码里我有一些无用循环，还有多次使用不直接预处理等等的问题，好像只过了最多两个点。后来一步一步的思考判断优化，才成了现在代码的样子。

### 1.3差分分析

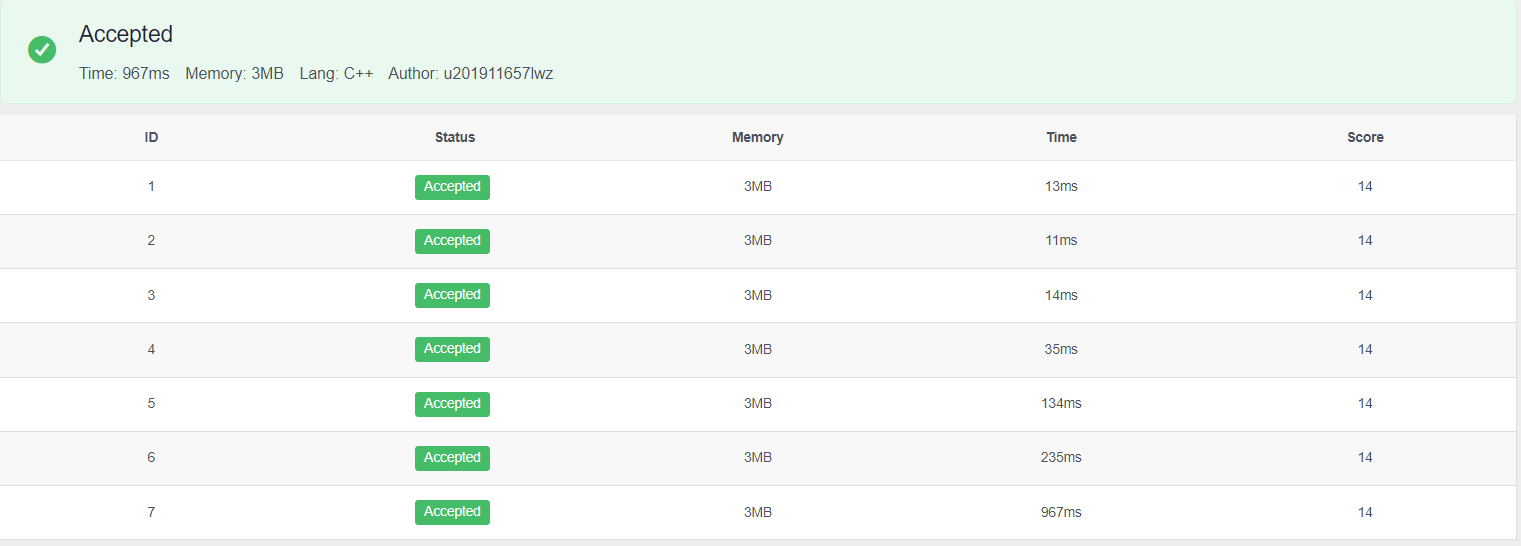
1. 设计内容

目的类似于上一题的线性分析，只不过这次采用差分分析的方法进行，差分分析所需要的数据量要大于线性分析，但是差分分析的效果显著的好于线性分析。每一组中会给出0x0000到0xffff的对应密文。最多会有500组密钥需要破解。

1. 设计过程

大概复习了一下差分分析的思路方法，发现盒线性分析西实际上差距不是很大，于是直接手动找了一组差分分析链，然后结合课本上的例子，再第二题基础上稍微改动一下。然后直接提交，直接RE了。然后也不知道RE是什么原因，就开始二分查错结果查了一遍，各个部分交上去都能RE，哪一部分都会挂，这就完全超出我认知范围了，连续好几天都没突破RE。后来在和别的同学讨论的时候，他告诉我，可能是头文件有问题，之后我又二分头文件，发现iostream.h这个头文件会出错，于是就重构了以下代码，去掉了必须要iostream的地方，又经历了一番艰难的改错，最后终于能过前几个点了。之后借用系统中的时间函数，进行了运行时间分析，发现实际上原本我预计时间会很长的地方反而时间很短，倒是读入时间很长，于是就优化了读入做了一个循环展开，程序运行速度显著的提高了，但是仍然有很多测试点通过不了。最后一个同学提醒我修改一下差分分析函数里面的循环枚举顺序，就可以稍作很多步重复操作，因为一个循环中，如果涉及到到的都是同一个明密文对，显然可以优化成只计算一次。修改之后，快了一点点，正好卡进了时限内。

（3）小结



前面iostream会出现未知错误，可能是OJ本身有些问题。因为完全没有预料到头文件可能会直接导致错误，所以花了很长时间也没有能够发现问题。不过在后面部分我在阅读了代码很多次之后仍然没有发现这个循环顺序调换的优化是有点不应该。归根到底还是自己对于能够优化的常数不够敏感，还是沿用了高中OI时期的主要分析时间复杂度的阶，忽略常数的想法，所以吃了不少的亏。

### 1.4 SPN增强

（1）设计内容

这个题目和前面的有很大的不同，是一个设计题目，对于最终的判定只有要求低信息熵之类的要求，没有正确性的要求（因为对于不同的设计，输出都可能不一样）。而且加密信息盒密钥也都比较长，加密信息长度达到比特，密钥长度达到128这就给了实现各种方法的空间。

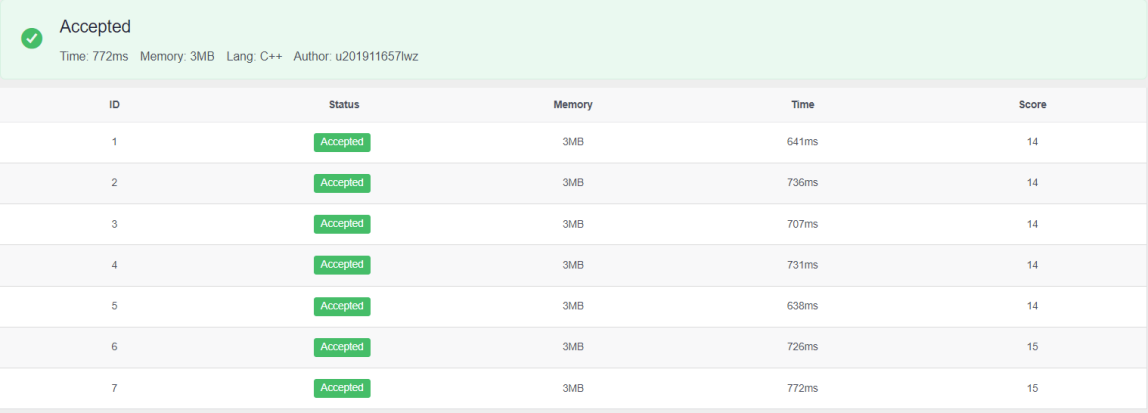
（2）设计过程

这题目一看就是个乱搞题，只要能够解密出来就可以，甚至可以说，只要加密过程是可逆的，不需要管解密过程有多复杂。首先考虑因为并不能评测是否加密可逆，所以我直接交了一个随机数上去，过了，因此就有了一个很朴素的想法，只要我的加密结果越像随机数，就越有可能通过，因此往加密过程里面掺伪随机数就成了我的核心思路，问题就是怎么掺随机数进去。

显然的是，明密文长度相同，因此如果必须做成一一映射。而且需要一个高效的做法，所以我先重构了一部分代码，把它变成依次处理4个字节，进行12轮（实际上轮数完全没必要这么高，因为评测标准只涉及信息熵）的函数。然后尝试掺入随机化因子。我第一个成功通过的程序使用的是这样的一个方法，每一次进行加密的时候随机生成S盒，但是使用的种子已知，比如使用密钥作为随机数种子，这样显然只要知道密钥就能还原出整个S盒然后进行解密操作。从这个思路出发，容易得到很容易就可以得到许多的加密方法，这是因为利用一个已知的文字总可以使其作为种子，rand出一个近乎随机的信息加入到里面。这就无形中增大了信息熵。但是这个增大与我自己所写的部分无关，直接使用这种方法，用key生成随机数，然后直接异或也可以通过。显然这种加密方法是可以解密的。

不过最终让我放弃这种做法的是另一位同学的反驳，即我的做法违背了Kerckoff原则。我所使用的密码体制并不是一种被大家所知的体制，因为他的S盒和密钥绑在了一起，当然也可以说S盒是密钥使用的一部分，不过这就违背了SPN的定义。所以最终我使用了CBC模式，但考虑到因为名密文等长，初始向量不能加入到密文里面，所以我还是采用了key通过伪随机数生成初始向量的方法，这样解密的时候要先通过相同的密钥进行rand获得初始向量。

（3）小结



总的来说这道题非常的有意思，他让我思考了很多，包括可行性，安全性甚至是否简洁。最终得到的这个方法我自己还是比较满意的，不仅仅是最终结果更是我的思考过程，我觉得这使得我得到了很大的提升。而且在过程中同学们对我的帮助，我们之间的讨论都让我受益匪浅。

### 1.5 RSA参数计算

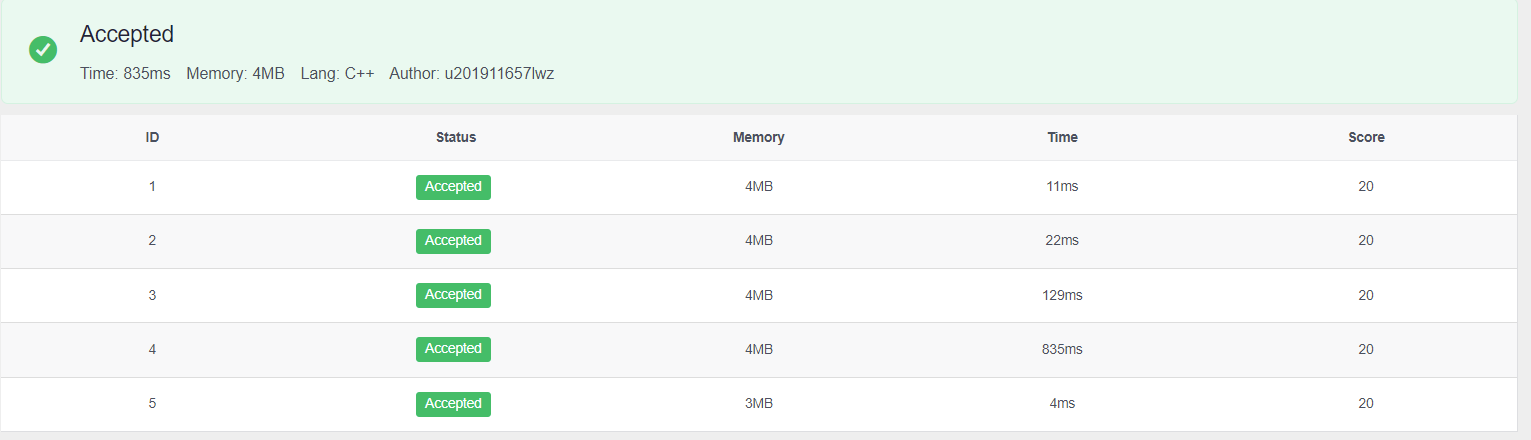
（1）设计内容

已知RSA的陷门，和公钥，去求一个私钥。但是数据可能会很大，所以需要使用高精度算法，然后还要判断当前的o，q和公私钥对是否可能会被常用攻击所攻击。

（2）设计过程

这道题目我花在配环境的时间远大于写代码的时间。为了安这个gmp，我把dev C++的整个编译器都换掉了，原本是TDM，都换成了MINGW，然后利用mingw-get下载了gmp模块。相当的麻烦，但总算实现了gmp库，之后为了方便我整个都是使用的mpz\_class，然后定义了一个BigInt 其他部分都是把他当成int来写的，但是这样的坏处就是没法用已有的一些函数，比如gcd。不过也不麻烦，重写一个也不算很难。判断指数选择的是带二次判断的Miller\_Rabin算法。有一些常见的不优秀参数选择是容易想到的，比如p、q过大过小，或者非素数，包括Wiener低解密指数攻击。但是用上这些之后，仍然有一个点过不去。我尝试做调整参数，发现卡过他去比较麻烦，而且不够优雅。后来乱用factor做分解的时候发现，似乎p-1和q-1有个共同特征，最大质因子比较小。但是虽说比较小，也有100bit以上。因此这个东西我没有想到非常好的方法，我不知道哪个算法能够在1s时间要求内，分解一个600+bit的合数。但是在http://www.factordb.com/index.php? 这个网站上，他能够用很短的时间分解一个巨大的合数，不清楚它采用的是什么算法？它的算力又有多少？不过后来汤老师在群里说了一种神秘的判别方法，就是对p-1和q-1做gcd，我至今没能理解这个方法为什么能起到检测最大因子的作用，不过恰当调整参数之后似乎就可以通过了。

（3）小结



这个题目的完成给后面两个题提供了环境的基础，后面两道题在不存在环境问题之后，很快就通过了。

我觉得这个题最大的难点还是在于常见的rsa攻击上。我一直不太喜欢面向数据来编程，但是所谓的过大过小的参数还是需要根据数据来调整，这就有点不太友好，因为一组rsa公钥是否合法，要取决于别人的算法，更要取决于算力。抛开计算能力说是否会被攻击成功，那么标准就应该非常的高，以至于对于较强的算力也无法破解，但是从测试数据来看，并没有设置非常高的门槛，这就有点让人费解了。尤其是我上文所说的p-1，q-1分解的地方，不知道有没有什么更合理的分析方法可以通过测试数据。

### 1.6模重复平方

1. 设计内容

模重复平方指快速计算模意义下的一个数的次方。时间非常的宽松，以至于只要实现了log级别的算法就几乎不可能超时。但是实际上这个题目完全可以作为快速幂优化算法的试炼场，在这个的基础之上去完成下一题。

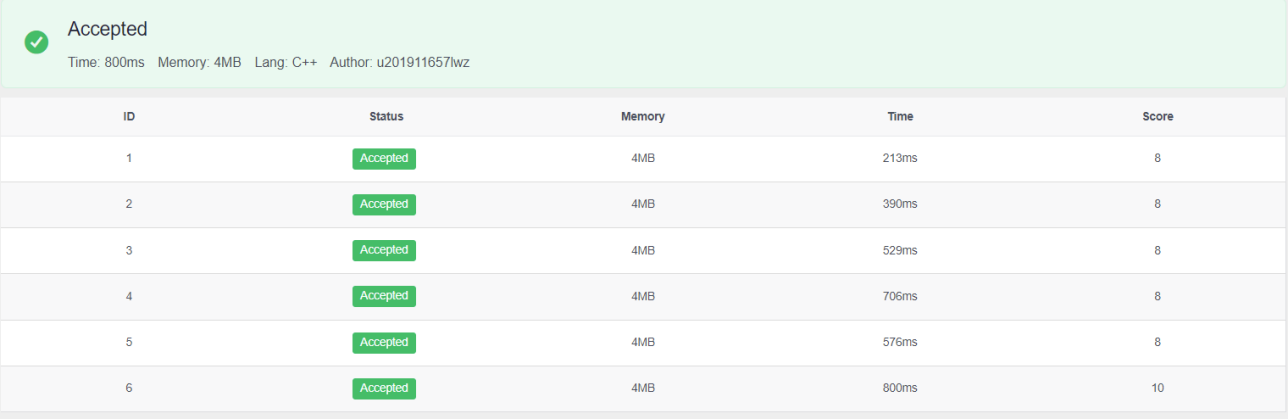
（2）设计过程

一般在原来我们称呼他为快速模幂，所以乍一看这个名字还显得有点高大上。我立刻就实现了一个最慢的log级别算法，直接通过了。我以为这道题目就结束了，但后来在做下一题的时候发现，下一题是在这道题的基础上进行的，而一般的快速模幂通过不了下一题，因此这道题就成了优化模幂运算和中国剩余定理的试炼场。

考虑到对于大整数来说，加减运算都是位数线性的，而乘法运算是位数平方级别的。所以首要目的是减少乘法的使用。我首先使用的是通过引入逆也就是-1次方的方法来完成，这种方法最主要的作用是把一连串的1编程一个更高位上的1和最低为上的一个-1。通过这种方法，大概是把整体的1的个数变成原来的，但是实际运行中，这个方法的优化可以忽略的小，也通过不了这个题目。

之后我有尝试了一些常数优化，结果不是很理想。后来在群里同学们的讨论中，听到了一种叫滑动窗口的做法，我就去学习了一下。看完之后，感觉智力又被碾压了，我自己在思考的时候一直局限在指数的位按照从小到大来完成整个的过程，没有想到从大到小枚举指数的位。我最开始实现的滑动窗口似乎优化有点问题，做的仍然很慢。后来我转换成了一种直接的压位的模重复平方，也就是一次直接处理掉最低的好几个二进制位的指数，加上一些预处理的操作，就能够实现比原来快了不少的优化。但实际上因为每次仍然需要进行自平方取模，而这个操作仍然是log(指数)级别的，所以实际上优化也不算很大，仍然是常数级别的。这样就过掉了这个题目。

1. 小结



这个题目是一个典型的常数优化题目，而这一部分恰恰是我原来经常忽略的。也没有很思考这方面的优化方法，以至于需要阅读有关资料才能获得优化的思路。然后实现的时候也因为自己代码习惯的问题，凭空就把常数写的大了。这种思路和代码习惯都是以后的我需要注意改善的。

### 1.7中国剩余定理

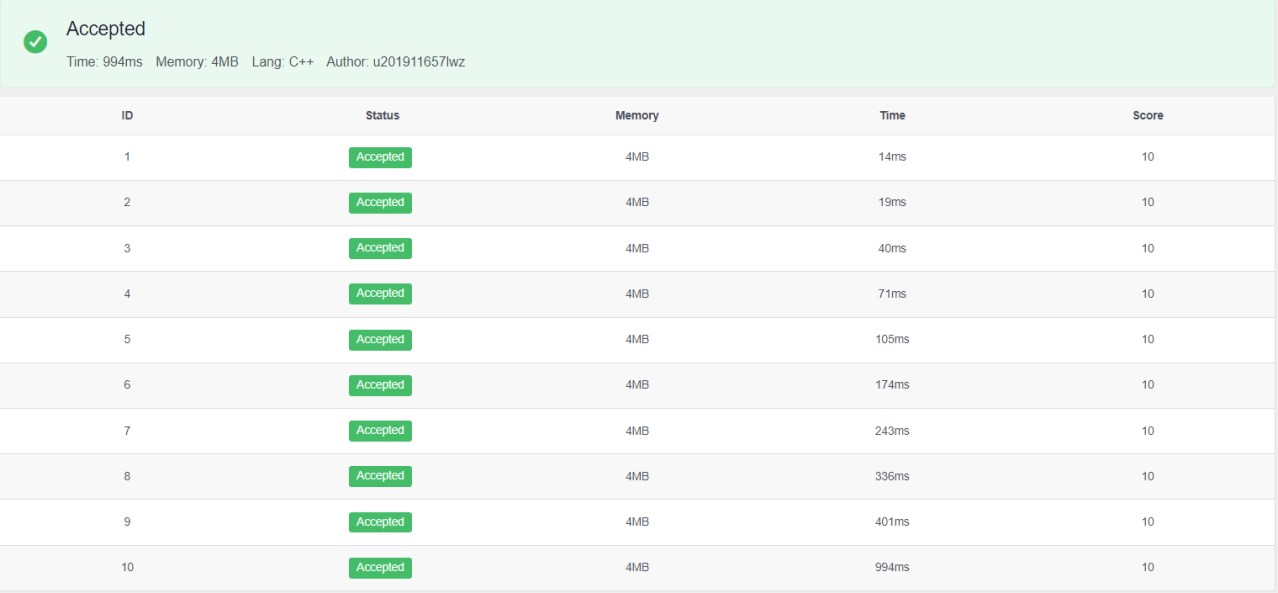
1. 设计内容

本题目的目的是已知rsa的陷门，和加密密钥，来做到给一个密文解密。简单来说就是第五题+第六题。

（2）设计过程

因为第六题我直接用最简单的快速幂就通过了，而一看这个题目几乎就是前两题的复合，所以直接就把两个代码连在一起，又写了一个控制的主函数就交了。当时我还有些疑惑，为什么这个题目叫中国剩余定理呢，我好像也没用到中国剩余定理了就完成了任务。结果自然是毫无悬念的Tle掉了。后来我冷静了一下，发现似乎分开模数，然后CRT组合确实能给运行时间几乎去掉一半（因为指数的位数减少了一半），但是仅仅使用CRT而没有优化快速幂的方法仍然不能通过，所以我就用了第六题作为实验，写出了一个效率比较优秀的模重复平方（具体过程可以见第六题的设计过程部分），然后通过了本个题目。

（3）小结



这个题目算是对公钥密码学体制题目的一个最后总结，基本就是把前面的结合起来。而且要求前面的实现都要做的比较优秀才可以，不过总的来说，这一部分并没有花掉了很多的时间，甚至不如安装gmp用的时间长。

### 1.8 PKCS7

（1）设计内容

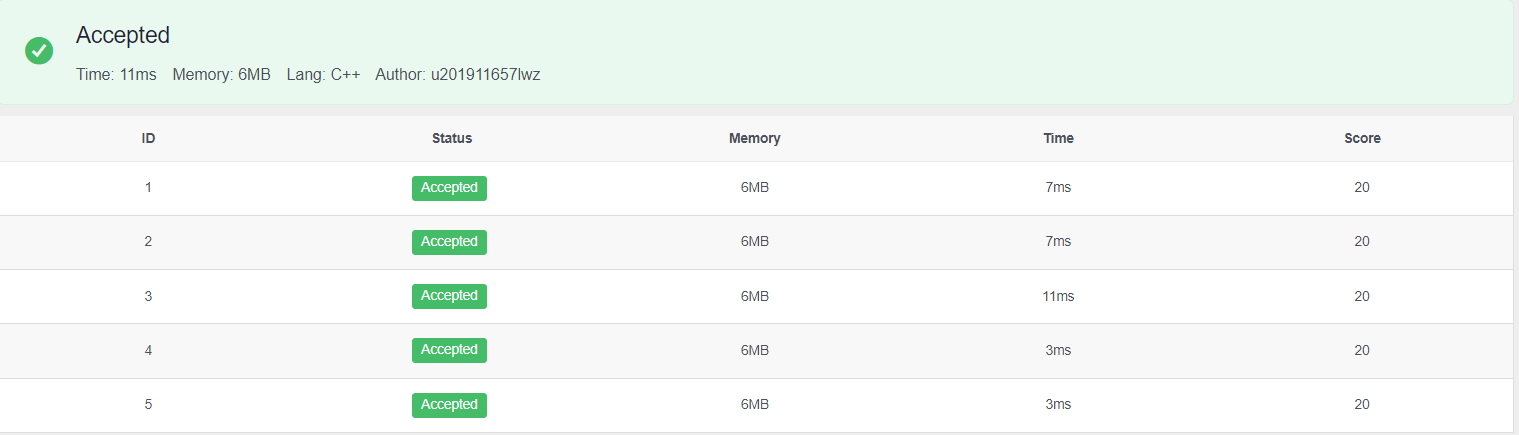
本题是对PKCS#7内容的考察。PKCS#7是PKI中用于消息加密的语法标准。可以用于给拥有公钥的用户发加密邮件、传送加密文件等。首先给出一个可信的CA的公钥。用户B是PKCS#7格式加密消息的接收者，给出用户B的私钥（包括公开参数和密钥两部分）。目标是解开PKCS#7的包装，获取明文信息。

（2）设计过程

这个题目一开始我是毫无思路的。刚开始的时候以为需要自己进行加解密，信息摘要，验证签名之类的，去网上学了一些杂七杂八的东西。后来同学通过了告诉我只需要调用一堆的openssl里面的函数就好了。后面就成了变成了根据名字猜函数用途和理解别人的代码。在经历了各种神奇的理解之后，才大概能搞懂具体的实现过程了。

首先初始化，引入各种函数。之后创建一个内存型BIO，读入并解析出原始数据，从BIO中解析出原始数据。解密出明文，之后获得签名者信息栈，获取签名者信息之后利用ca公钥逐一验证签名。这个过程中使用了各种各样的函数来完成，还涉及到了各种的数据类型。

（3）小结



这个题目是我做的最不明白的题目了，虽然通过了题目，但是仅仅局限在理解了系统函数的程度，也没有去仔细理解函数的具体实现。而且因为并没有使用openssl进行本机测试，而是直接用OJ来进行测试，所以也限制了我的调试，好在都是比较固定的函数调用而且基本上各个模块都能找到别人的成品代码，所以还是很快的通过了。

### 1.9 彩虹表

（1）设计内容

这个题目的题面总体来说就很谜语，几个同学第一次理解的题目或多或少都有不同，不少同学都是因为理解错了题目的意思导致通过不聊这个题目。

题目的题意大概是给出了若干条彩虹链，其生成过程如下：首先获得一个链首（为字符串），然后进行一百轮的迭代。其中每一轮的迭代是这样一个过程：计数器cnt从1到100，每次首先对字符串进行SHA1摘要，随后使用R（cnt）使其从一个摘要回到字符串。

然后现在给出m条彩虹链中，问现在的摘要是否出现在这m条链中，如果出现，他的前一步字符串是什么。这当然也就是彩虹链加速信息摘要碰撞的思路和做法。

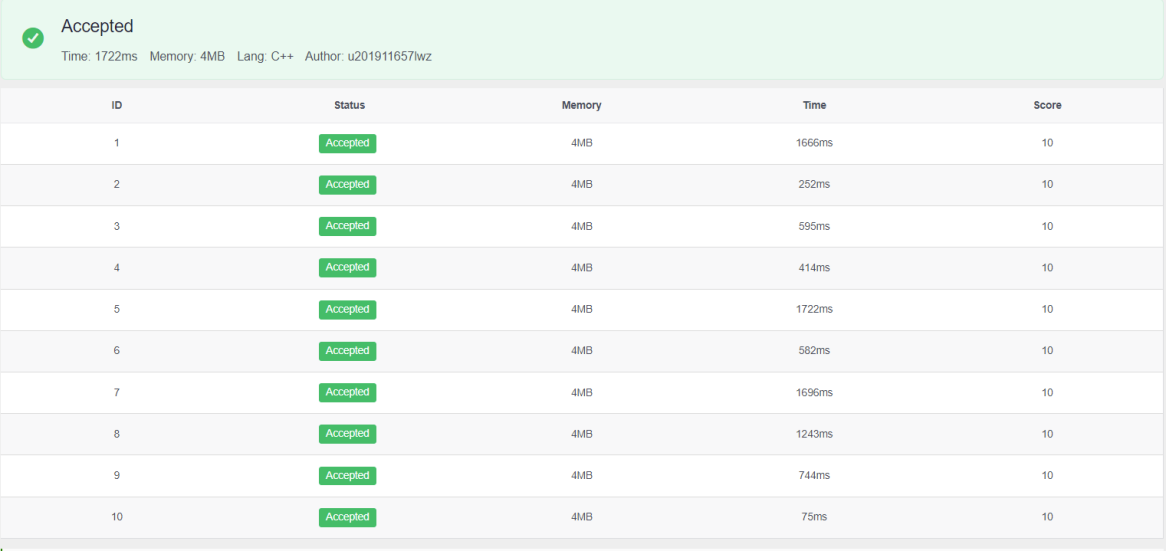
（2）设计过程

首先必须吐槽一句，OJ上给出的代码是有问题的，在getstr函数里面，只生成了字符串的前面非零部分，但是根据C的规则，一个字符串应该包含其合法结尾，即’\0’。而这个函数根本没有在后面加上一个字符零，所以实际上返回的并不是一个完整的C意义上的字符串。因为想少开变量，我在迭代得过程中复用了一个变量（字符串指针），但是直接结果就是没有0隔离导致前面的字符串加入到了下一轮的字符串中，导致了一些很难察觉的错误。而我在很长时间内都没有怀疑到这个地方，浪费了很多时间。

下面具体说我的解题过程：

首先为了验证我读的题意是不是正确的，我首先写了一个test函数，实现了每条链从链头走到链尾，目的就是为了验证每一条链是不是按照我的理解生成的，结果确实如此，之后我才开始实现要求的功能。简单来说就是枚举初始值在弈论迭代中的位置，然后往后进行最多100轮去撞链尾。而每一个彩虹链使用一个map存储，key为链尾，value为链首。如果撞到了链尾，那么当前这个值就可能在这个链里面。这种情况下就应当去跑整条链，去撞这个字符串，如果能撞得到就输出，如果全都不能。那么就是error。

（3）小结



这个题目是所有题目的最后一个，这个题目没有在实现的常数上卡的很死，没有很进行常数优化就通过了。而且因为我读题目的意思第一次就对了，所以也没有很卡题。而且首先写了一个测试程序，确认了我的理解是正确的，而且这一部分完全可以复用在后面跑链的步骤里。但我仍然觉得，这个题目中给出的实现代码中有问题确实不太应该，这个地方是不太容易察觉到的地方，我也是在经历了单步调试之后才机缘巧合下发现了问题。

## 实验心得

这次密码学实验也算是一个比较新颖的体验，也算是巩固了上学期的密码学知识。从SPN到公钥加密（RSA）还有已经成熟了的加密消息的语法标准以及hash的攻击方法。覆盖的方面比较全，题量也比较适中。在完成这些题目中，附带这也让我复习了很多相关的内容（比如第五题中对RSA的常见攻击），也算是对密码学进行了一个比较全面的复习吧。不过这次题目并没有对那些比较成熟的算法设计题目，比如，sha，AES，DES，利用CDH问题的各种密码体制以及椭圆曲线出题。（当然如果都出了可能题量就太大了）

而且这次试验和之前算法相关任务有很大不同，比如这次尤其的重视对于常数的优化，往往通过和不通过之间只差了几倍的时间，很多的题目从几乎没有分数到通过都是常数级别的优化。而这部分在以往往往是被我忽略的，借着这次试验，让我重视到了这部分的重要性。同时也修正了一些不太好的代码习惯，使得自己能够注意代码是否存在压低了效率的部分，并且去改正它。

从事后的角度来看，除了第四题，其他题目基本上都有比较固定的优化方式（?）。而频繁的卡题也导致了大家几乎对每个题都有讨论，所以我感觉大家基本上交流着交流着思路就到一起去了，所以最终优化的结果也比较相似。

## 三、对课程设计内容和过程的建议

首先是有的题目的题面描述确实不太容易让人理解或者说有歧义，不同同学的理解可能都会有些不同。

除此之外很多题目的时限卡的还是比较的紧，需要优化的点比较多。有的时候长时间得到调试不通过还是体验不太好，虽然老师说这次实验可能有些同学得不了满分，但是我看最后大概是147个同学都得了满分，看得出同学们还是都很认真的。

除此之外希望像第八题这种的题目老师可以准备一些必要的资料，如果全靠自己入门的话还是比较麻烦的。而且大家也往往选择复制、粘贴网上的实现代码中的模块而不是自己实现，因为自己实现需要学习的东西还是比较多而且容易做的复杂了。

除此之外，感觉实验课的OJ似乎比起网络上的其他OJ来说似乎有一些奇奇怪怪的问题，比如我至今也不知道为什么使用iostream会导致RE。