密码学第七次实验报告

# RSA公钥密码体制

## 原理

RSA公钥密码体制是现行的一种公钥密码体制，主要基于求离散对数和大整数因子分解的困难性。它原理比较简单，实现相对方便，是基于模幂的原理。

## 伪代码

### 素数生成算法

static BigNum genPrime (int bitlength) {

do {

生成0到(1 << bitlength)之间的随机数;

} while (生成的随机数经过素性测试算法检验不通过);

return 生成的随机数;

}

### 生成公私钥对算法

static BigNum[][] genKeyPair(BigNum p, BigNum q) {

计算出n, phi(n), d

令 e = 65537

以 {d, n, p, q} 作为私钥，{e, n} 作为公钥

return 生成的公私钥对;

}

### 加密算法

static BigNum encrypt (BigNum plainText, BigNum[] publicKey) {

return (mod);

}

### 解密算法

static BigNum decrypt (BigNum cipherText, BigNum[] privateKey) {

return 方程组的解;

}

### OEAP加解密算法

按照文档中7.1.1和7.1.2节进行。

### 文件加密算法

static void encryptFile (Path plainTextPath, Path cipherTextPath, Path publicKeyPath) {

从publicKeyPath对应的文件中获得公钥;

令每个密文分组的长度为n所占的字节长度;

令每个明文分组的长度为n所占的字节长度-1;

读plainTextPath对应的文件，对每个分组加密，并写入cipherTextPath对应的文件中去,若最后一组长度不够，则后面用0字节填充;

}

### 文件解密算法

static void decryptFile (Path plainTextPath, Path cipherTextPath, Path privateKeyPath) {

从privateKeyPath对应的文件中获得私钥;

令每个密文分组的长度为n所占的字节长度;

令每个明文分组的长度为n所占的字节长度-1;

读cipherTextPath对应的文件，对每个分组解密，并写入cipherTextPath对应的文件中去,若最后一组长度不够，则后面用0字节填充;

}

### OAEP文件加解密算法

与普通文件加解密算法一样，不同的是每个明文分组长度固定，都是64个字节，而且用相应的OAEP算法加解密。

## 分析

### 素数生成算法和生成公私钥对算法

素数分布本身是有规律的，但是因为本次实验生成素数的范围确定，而且素数生成带有概率性，因此需要的时空复杂度都是。

### 加解密类算法

由于加解密算法数据规模确定，因此需要的时空复杂度都是。

### 文件加解密类算法

由于文件加解密类算法需要分块，所以时间复杂度与文件长度有关。每次加解密之间的数据中，有关联的都是常数级别大小。因此，时间复杂度是，空间复杂度是。

# 总结

## 大整数操作

由于这次不让用Python语言，我用Java重写了一遍大整数操作，包含一些数论操作。

## RSA相关算法

这次RSA相关算法数学上是比较简单的，但是关于大整数的细节确实有些卡人。我花了两个小时调一个bug，结果发现是大整数算法中的一个函数写错了。

## OAEP相关算法

OAEP相关算法其实是比较严格的，但是比较难以调试。我花了不少时间才把算法调试到正常水平。

## 算法评估与优化

这次算法评估与优化主要是在大整数算法上。我在大整数算法中使用了一定的优化技巧，比如乘法首先判断最高位，这使得乘法的复杂度大大降低。

## 系统设计与维护

这次良好的系统设计与维护起到了很好的作用。由于我实现大整数相关算法的时候考虑了模块化和系统设计，这使得我到后面执行复杂操作的时候越来越不需要考虑下层的需要，在最后实现OAEP的时候，也能比较快地发现bug。这也能体现良好系统设计的加速型回报。

## 对课程的建议

感觉以后可以把细节抽象出来，不要让我们写大整数运算了。我感觉大整数运算相对密码学可能有一点点细节。所以，我们经过一番细节的洗礼，不仅没法独立思考，更没法对这个课程有更深的兴趣。感觉密码学实验本来就很累，占用了我几乎所有的休息时间，这样一来更累了。这个实验大概花了我四五十个小时，一大半时间都耗在大整数类上了。更何况我们还有其它科目的实验，尤其是我们致真的同学，还多了一个星期十几个小时的数据结构，时间和精力真的耗不起了。感觉这样真的有点像做苦力，不仅东西没怎么学到（更不用说体会到），人还受累，也不知道对未来有什么好结果，更不知道密码学实验这样做对以后有什么意义。

虽然我知道大整数运算在算法中比较关键，但是算法本身其实并没有动态规划之类那么难，似乎有点麻烦了。而且，感觉以后真的不敢用Python了，因为这个星期好不容易写出了模块，下下个星期就连着语言被ban了。如果老师或者学长真的想半路改掉开学设下的规定，放弃Python，那还不如尽早声明的好，免得我们这些用Python的同学们到最后还是得学其它语言。而且，我们本来Java就不精，感觉对学习其它语言的锻炼也比较有限。更何况频繁换语言，应该也对更深刻地掌握密码学不利。所以我感觉对Python能不能用，今早确定比较好，别让我们的劲头被挫伤，到头来失去兴趣，还得重新干活。

## 总结

这次实验算一次从头重写大整数库的实验，RSA本身倒不难了。感觉这次把我两个星期的所有空余时间都吃了，因此我无法写出更多的东西了。