应用密码学 RSA 实验报告

陈荣钊 2022213861

2022 年 11 月 15 日

1 作业目录

依文档要求，提交的作业分为./SRC，./BIN，./DOC 三个目录。
./SRC 目录存放源代码，包含：

- googletest: 开源的 googletest 组件，用于搭建项目测试；
- src: 由我自己实现的 RSA 源代码；
- test: 由我自己实现的功能测试和性能测试源代码。

./BIN 目录存放可执行文件，包含：

- RSA: Unix 可执行文件；
- sample: 示例数据。

./DOC 目录包含实验报告和参考文献。

2 使用说明

本次实验实现了一个命令行程序，打开终端，执行./BIN/RSA，进入程序初始界面如图1所示：

图 1 初始界面

初始界面展示了程序拥有的六个功能，与以下序号一一对应：

1. 生成 RSA 公钥/私钥；
2. 明文加密；
3. 密文解密；

4. 生成数字签名；

5. 验证数字签名；

6. 退出。

下面对功能 1 至功能 5 进行展示。

2.1 生成 RSA 公钥/私钥

在初始界面输入 1，按回车以进入生成 RSA 公钥/私钥功能，如图2所示：

```
Please select a function:
        1. Generate RSA public/private keys
        2. Encrypt(ascii plaintext -> hexadecimal ciphertext)
        3. Decrypt(hexadecimal ciphertext -> ascii plaintext)
        4. Sign(generate an ascii plaintext's digital signature)
        5. Verify(verify a digital signature with a plaintext)
        6. Quit
Please input the function index[1, 6]: 1
Please select RSA number:
        1. RSA-576(576-bits)
        2. RSA-640(640-bits)
        3. RSA-768(768-bits)
        4. RSA-896(896-bits)
        5. RSA-1024(1024-bits)
        6. RSA-1536(1536-bits)
        7. RSA-2048(2048-bits)
Enter RSA number[1, 7]: 3
Use default exponent(65537)?
        1. Yes
        2. No(generate random exponent)
Enter [1, 2]: 2
Successfully generate RSA public key on: ./public_key.txt
Successfully generate RSA private key on: ./private_key.txt
Successfully generate RSA public/private keys for module bit-length of 768, time-consuming: 0.0883s.
```

图 2 生成 RSA 公钥/私钥

依次选择：

1. RSA-number：包括 RSA-576，RSA-640，RSA-768，RSA-896，RSA-1024，RSA-1536，RSA-2048；

2. 是否使用默认指数：若是，则选用默认指数 65537；否则使用随机生成的大素数。

程序会生成公钥文件./public_key.txt 与私钥文件./private_key.txt，并储存于同目录下。

2.2 明文加密

在初始界面输入 2，按回车以进入明文加密功能，如图3所示：

```
Please select a function:
        1. Generate RSA public/private keys
        2. Encrypt(ascii plaintext -> hexadecimal ciphertext)
        3. Decrypt(hexadecimal ciphertext -> ascii plaintext)
        4. Sign(generate an ascii plaintext's digital signature)
        5. Verify(verify a digital signature with a plaintext)
        6. Quit
Please input the function index[1, 6]: 2
Please input your public_key file: ./public_key.txt
Successfully open file: ./public_key.txt
Successfully read RSA public key.
Please input your plaintext file: ./sample.txt
Successfully open file: ./sample.txt
Successfully read plaintext.
Successfully encrypt the plaintext, and the ciphertext is wrote on: ./ciphertext.txt
```

图 3 明文加密

依次输入：

1. 公钥文件；

2. 明文文件。

程序会使用公钥将明文文件加密，生成密文文件./ciphertext.txt，并储存于同目录下。可复制./sample 目录中的./sample.txt 文件至上级目录，以复现图3所示流程。

2.3 密文解密

在初始界面输入 3，按回车以进入密文解密功能，如图4所示：

```
Please select a function:
        1. Generate RSA public/private keys
        2. Encrypt(ascii plaintext -> hexadecimal ciphertext)
        3. Decrypt(hexadecimal ciphertext -> ascii plaintext)
        4. Sign(generate an ascii plaintext's digital signature)
        5. Verify(verify a digital signature with a plaintext)
        6. Quit
Please input the function index[1, 6]: 3
Please input your private_key file: ./private_key.txt
Successfully open file: ./private_key.txt
Successfully read RSA private key.
Please input your ciphertext file: ./ciphertext.txt
Successfully open file: ./ciphertext.txt
Successfully decrypt the ciphertext, and the plaintext is wrote on: ./plaintext.txt
```

图 4 密文解密

依次输入：

1. 私钥文件；

2. 密文文件。

程序会使用私钥将密文文件解密，生成明文文件./plaintext.txt，并储存于同目录下。可验证，./sample.txt 与./plaintext.txt 文件完全一致。

2.4 生成数字签名

在初始界面输入 4，按回车以进入生成数字签名功能，如图5所示：

```
Please select a function:
        1. Generate RSA public/private keys
        2. Encrypt(ascii plaintext -> hexadecimal ciphertext)
        3. Decrypt(hexadecimal ciphertext -> ascii plaintext)
        4. Sign(generate an ascii plaintext's digital signature)
        5. Verify(verify a digital signature with a plaintext)
        6. Quit
Please input the function index[1, 6]: 4
Please input your private_key file: ./private_key.txt
Successfully open file: ./private_key.txt
Successfully read RSA private key.
Please input your plaintext file: ./plaintext.txt
Successfully open file: ./plaintext.txt
Successfully read plaintext.
Successfully generate digital signature on: ./signature.txt
```

图 5 生成数字签名

依次输入：

1. 私钥文件；

2. 明文文件。

程序会先计算明文文件的哈希值，然后将哈希值使用私钥加密，以生成数字签名文件./signature.txt，并储存于同目录下。

2.5 验证数字签名

在初始界面输入 5，按回车以进入验证数字签名功能，如图6所示：

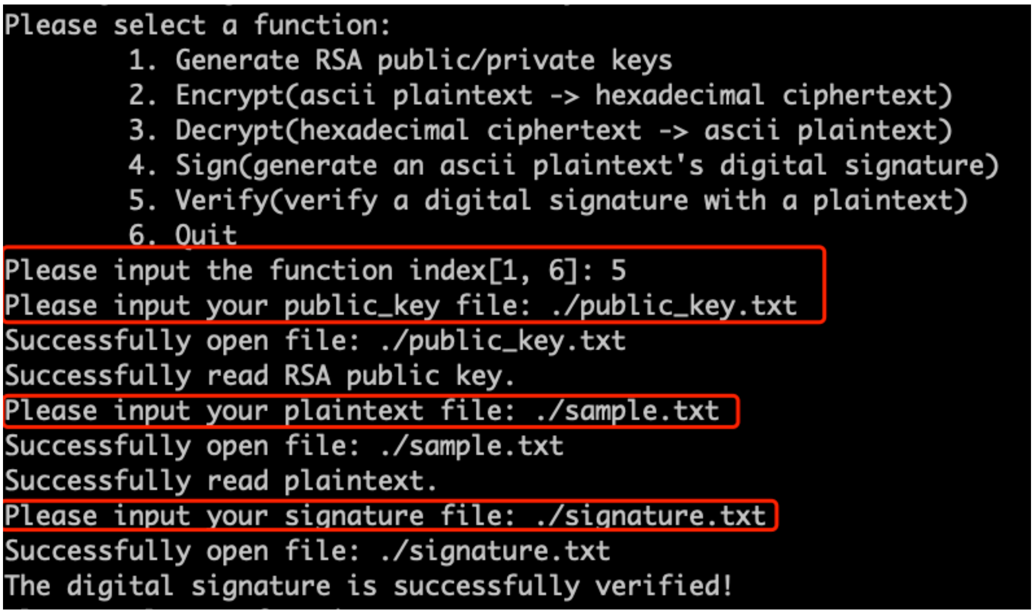

图 6 验证数字签名

依次输入：

1. 公钥文件；

2. 明文文件；

3. 数字签名文件。

程序会先计算明文文件的哈希值，然后将哈希值使用公钥加密，将加密结果与数字签名比对，一致时输出验证成功，否则验证失败。

3 代码实现

本节从以下两个方面介绍本次实验中的代码实现亮点：

3.1 BigInteger

本实验的基础是实现效率高的大整数，因此，我定义大整数类 BigInteger 如下：

```
class BigInteger {

private:
    // The sign of this
    // this is positive if sign == 1,
    // this is zero if sign == 0,
```

```
    // this is negative if sign == -1.
    int sign;
    // The length of number array
    int length;
    // The bitLength of number array
    int bitLength;
    unsigned int *number;
}
```

各字段定义如下:

- sign: 大整数符号位，sign == 1 时表示正整数，sign == 0 时表示 0，sign == -1 时表示负整数;
- length: 大整数长度，即 number 数组的长度;
- bitLength: 大整数去除前导 0 后剩余的有效位数。
- number: 大整数具体数值，采用小端模式保存。

此设计中，符号位与具体数值分离，因此 number 数组几乎每一个 bit 都得到有效利用，且运算时不必考虑补码；大整数采用 2^{32} 进制，便于在具体实现中使用位运算替代四则运算。

对于大整数 u, v 的四则运算，不妨设 $u.bitLength = n + m > n = v.bitLength$，最终实现复杂度如下:

- 加法：$\mathcal{O}(\frac{n+m}{32})$
- 减法：$\mathcal{O}(\frac{n+m}{32})$
- 乘法：$\mathcal{O}(\frac{n+m}{32} \times \frac{m}{32})$
- 除法与模运算：$\mathcal{O}(\frac{m}{32} \times \frac{n}{32})$

大数加法、减法和乘法的朴素实现即可达到上述复杂度，对于大数除法与模运算的复杂度将在下一章详细分析。

3.2 测试框架

为确保本次实验完成的程序具有较高准确性，我在工程中引入 googletest 框架，使用 Java8 标准库的 BigInteger 类生成测试数据，并在./SRC/test/FunctionalTests.cpp 源文件中实现了以下功能测试:

- addTest: 验证大数加法;
- subtractTest: 验证大数减法;
- multiplyTest: 验证大数乘法;
- divideTest: 验证大数除法;
- smallModTest: 验证大数对单个 unsigned int 取模;
- modTest: 验证大数取模;
- powModTest: 验证大数模快速幂;
- inverseTest: 验证大数乘法逆元;
- isPrimeTest: 验证素数检测（Miller-Rabin 算法）;
- cipherTest: 验证 RSA 加密与解密;
- signatureTest: 验证 RSA 数字签名。

4 算法实现

本节从以下两个方面介绍本次实验中的算法实现亮点：

4.1 大数除法与模运算

大数除法与模运算是本次实验中的主要复杂度瓶颈，本小节讨论具体实现的优化。

令大整数进制为 $b=2^{32}$，设被除数为 U，设除数为 V。对 $V<b$ 的情形，使用秦九韶算法即可；当 $v>=b$ 时，采用 Donald E. Knuth 在《The Art of Computer Programming》Vol 2. section 4.3.1 介绍的长除算法。

4.1.1 长除算法简介

本小节简介长除算法。

长除算法的输入为 b 进制下的 $N+M$ 位被除数 $U=(U_{N+M-1}\cdots U_1U_0)_b$ 和 N 位除数 $V=(V_{N-1}\cdots V_1V_0)_b$；输出为 $M+1$ 位商 $Q=(Q_M\cdots Q_1Q_0)_b$ 和 N 位余数 $R=(R_{N-1}\cdots R_1R_0)_b$。

首先需要构造规范化的输入 u,v：

$$v=(v_{n-1}\cdots v_1v_0)_b=V<<T,$$

$$T=\min\{t:t\geq 0,\hat{V}=(\hat{V}_{\hat{N}-1}\cdots\hat{V}_1\hat{V}_0)_b=V<<t,\hat{V}_{\hat{N}-1}\geq b/2\}$$

在上述条件下，令 $\hat{U}=(\hat{U}_{n+m-1}\cdots\hat{U}_1\hat{U}_0)_b=U<<T$，为 $\hat{U}$ 补上前导 0，得到：

$$u=(u_{n+m}u_{n+m-1}\cdots u_1u_0)_b,u_{n+m}=0$$

于是可得具有以下性质的规范化输入 u,v：

$$u_{n+m}=0 \tag{1}$$

$$v_{n-1}\geq b/2 \tag{2}$$

设规范化输入对应的商 q 和余数 r 如下：

$$q=(q_m\cdots q_1q_0)_b$$

$$r=(r_{n-1}\cdots r_1r_0)_b$$

初始化 $j=m$，考虑 q 的最高位 q_j，有 $q_j=\lfloor\frac{(u_{j+n}u_{j+n-1}\cdots u_{j+1}u_j)_b}{(v_{n-1}\cdots v_1v_0)_b}\rfloor$。由式(1)可得：

$$(u_{j+n}u_{j+n-1}\cdots u_{j+1})_b<(v_{n-1}\cdots v_1v_0)_b \tag{3}$$

可以用近似算法快速确定 q_j 的准确值，定义近似值 $\hat{q}_j$ 如下：

$$\hat{q}_j=\min\{\lfloor\frac{u_{j+n}b+u_{j+n-1}}{v_{n-1}}\rfloor,b-1\}$$

可证明在式2和式3的条件下，式4成立（详见附录）：

$$\hat{q}_j-2\leq q_j\leq\hat{q}_j \tag{4}$$

进一步检查近似值 $\hat{q}_j$，若式5成立：

$$\hat{q}_j(v_{n-1}b+v_{n-2})>u_{j+n}b^2+u_{j+n-1}b+u_{j+n-2} \tag{5}$$

因为对准确值 q_j，下式总是成立的：

$$u_{j+n}b^2 + u_{j+n-1}b + u_{j+n-2} \geq q_j(v_{n-1}b + v_{n-2})$$

所以式6成立：

$$q_j < \hat{q}_j \tag{6}$$

在式4和6的条件下，令 $\hat{q}_j \leftarrow \hat{q}_j - 1$，可得到更精确的近似值估计，如式7所示：

$$\hat{q}_j - 1 \leq q_j \leq \hat{q}_j \tag{7}$$

在式7的约束下，很容易得出准确值 q_j，令：

$$(u_{j+n}u_{j+n-1}\cdots u_{j+1}u_j)_b \leftarrow (u_{j+n}u_{j+n-1}\cdots u_{j+1}u_j)_b - q_j(v_{n-1}\cdots v_1v_0)_b \tag{8}$$

式8的含义为，求解 $\frac{(u_{j+n}u_{j+n-1}\cdots u_{j+1}u_j)_b}{(v_{n-1}\cdots v_1v_0)_b}$，得到商 q_j，并将余数赋值回自身，于是根据除法的性质可得：

$$(u_{j+n-1}u_{j+n-2}\cdots u_j)_b < (v_{n-1}\cdots v_1v_0)_b \tag{9}$$

式9即为式3的下一步循环，而式2恒成立，因此在循环 $j \leftarrow m, m-1, \cdots, 1, 0$ 的过程中，总能通过上述过程快速确认 q_j。

最后，回顾开头对输入 U, V 执行的规范化过程，规范化并不影响商，所以 $Q = q = (q_m \cdots q_1q_0)_b$。

对于余数，经过如下非规范化过程即可：

1. 去除前导 0：$r = (r_{n-1}r_1r_0)_b = u = (u_{n+m}\cdots u_1u_0)$
2. 右移：$R = r >> T$

4.1.2 复杂度分析

由上文式7的约束，可供选择的 $\hat{q}_j$ 是 $\mathcal{O}(1)$ 的，不妨枚举每个可能的 $\hat{q}_j$，利用式8进行验证：

$$q_j = \max\{\hat{q}_j : (u_{j+n}u_{j+n-1}\cdots u_{j+1}u_j)_b - \hat{q}_j(v_{n-1}\cdots v_1v_0)_b > 0\}$$

计算上式的复杂度为 $\mathcal{O}(1) \times \mathcal{O}(\frac{n}{32}) = \mathcal{O}(\frac{n}{32})$，需要对 $\mathcal{O}(m)$ 位依次求商，因此长除算法的时间复杂度为：

$$\mathcal{O}(\frac{n}{32}) \times \mathcal{O}(\frac{m}{32}) = \mathcal{O}(\frac{n}{32} \times \frac{m}{32})$$

4.2 素数生成算法

参考论文《快速素数生成方法综述》，我设计了一个简单有效的素数生成算法。

首先需要准备一个小素数筛，我在实现时参考 Java8 标准库的取值，筛选了不超过 10000 的全部素数。

接着随机生成一个大奇数 n，执行以下步骤：

1. 若 n 不是所有小素数的倍数，且通过 10 次 Miller-Rabin 算法，返回 n；
2. $n \leftarrow n + 2$。

5 性能评估

我在./SRC/test/PerformanceTests.cpp 文件中完成了对生成以下标准 RSA-number 公私钥的性能测试：RSA-576, RSA-640, RSA-768, RSA-896, RSA-1024, RSA-1536, RSA-2048。

5.1 测试环境

- 操作系统：macOS 12.6
- 芯片：Apple M1 Pro
- 内存：16 GB

5.2 测试结果

为方便测试，固定公钥中的 exponent=65537，随后执行以下流程：对每个 RSA-number，随机生成 100 次公私钥，记录每次的生成时间，最后统计对每个 RSA-number 生成时间的最大值、最小值、平均值和方差，测试结果如表1所示（结果保留 3 位有效数字）：

表 1 性能测试

RSA-number	Maximum(ms)	Minimum(ms)	Average(ms)	Sigama
RSA-576	53.6	9.88	20.8	55.6
RSA-640	78.2	14	29	155
RSA-768	107	18.7	47.6	381
RSA-896	210	32.2	76.7	1.15×10^3
RSA-1024	426	47.5	134	4.51×10^3
RSA-1536	3.74×10^3	204	945	4.67×10^5
RSA-2048	9.1×10^3	804	3.16×10^3	3.52×10^6

由于素数的分布是不规则的，所以随着 RSA-number 量级的增加，极差和方差也显著增加。主要得益于本次实验中实现的效率较高的大数除法与模运算，从性能测试的平均值来看，生成 RSA-576 到 RSA-2048 的开销都是可以接受到。

6 感想和收获

本次实验基于 C++，从零开始实现了 RSA 算法。由于实现的大部分代码均为底层计算相关，思考如何实现高效算法，对我的 C++ 编码能力起到很大帮助。在本次实验中，我认真学习并理解了 Knuth 教授介绍的长除算法，还依据对算法所用各个不等式的总结，完成了一个比较精简的算法实现，这对我的数学能力也起到了很大锻炼。

7 课程建议

1. 交互界面不是本次实验的重点，交互界面设计与本课程无关，将交互界面设为附加项并不合理；
2. 若能提供一个基于 C++ 实现的交互界面模板，能较好地规范各功能定义，也可以让我们有更多时间放在算法研究和实现上，因为本实验对算法效率有较高要求，所以大部分同学应该还是会选用 C++ 完成；
3. 适当提供一些参考文献，指引一些有益的研究思路，这对本课程来说应该是有意义的。

8 附录

此处补上对上文所提，若式2和式3成立，那么式4成立的证明。

此处为了表述方便，不失一般性，仍取 $b = 2^{32}$，重新定义被除数 u 与除数 v 如下：

$$u = (u_n \cdots u_1 u_0)_b$$

$$v = (v_{n-1} \cdots v_1 v_0)_b$$

重新定义准确值 q 的近似值 $\hat{q}$ 的表达式如下：

$$\hat{q} = \min\{\lfloor \frac{u_n b + u_{n-1}}{v_{n-1}} \rfloor, b-1\}$$

那么所证的等价表达即为，若式10和式11成立，那么式12成立。

$$v_{n-1} \geq b/2 \tag{10}$$

$$(u_n u_{n-1} \cdots u_2 u_1)_b < (v_{n-1} \cdots v_1 v_0)_b \tag{11}$$

$$\hat{q} - 2 \leq q \leq \hat{q} \tag{12}$$

实际上在长除算法运行过程中，也总是在循环取出 $n+1$ 位的被除数和 n 位的除数，以此得到 1 位的商 q 和 n 位的余数 $r = (r_{n-1} \cdots r_1 r_0)$，故上述简化完全遵循长除算法，且不影响结果。

下证 $q \leq \hat{q}$，首先由式11可得：

$$\lfloor \frac{u}{b} \rfloor = (u_n u_{n-1} \cdots u_2 u_1)_b < (v_{n-1} \cdots v_1 v_0)_b$$

因此在式11条件下，有 $\lfloor \frac{u}{b} \rfloor < v$，因此 $\frac{u}{v} < b$，即 $q < b$。

回顾 $\hat{q}$ 的定义，可知若取 $\hat{q} = b-1$，那么必有 $q \leq \hat{q}$。

还需分析 $\hat{q} = \lfloor \frac{u_n b + u_{n-1}}{v_{n-1}} \rfloor$ 的情形，根据下取整的定义和条件式10可得：

$$\begin{aligned}
\hat{q} v_{n-1} &\geq u_n b + u_{n-1} - 1 \\
&\geq u_n b + u_{n-1} - (b/2 - 1) \\
&\geq u_n b + u_{n-1} - (v_{n-1} - 1)
\end{aligned}$$

即 $\hat{q} v_{n-1} \geq u_n b + u_{n-1} - v_{n-1} + 1$，进而下述关系成立：

$$\begin{aligned}
u - \hat{q} v &\leq u - \hat{q} v_{n-1} b^{n-1} \\
&\leq (\sum_{i=0}^{n} u_i b^i) - (u_n b + u_{n-1} - v_{n-1} + 1) b^{n-1} \\
&= (\sum_{i=0}^{n} u_i b^i) - (u_n b^n + u_{n-1} b^{n-1} - v_{n-1} b^{n-1} + b^{n-1}) \\
&= (\sum_{i=0}^{n-2} u_i b^i) - (b^{n-1} - v_{n-1} b^{n-1}) \\
&= v_{n-1} b^{n-1} - (b^{n-1} - \sum_{i=0}^{n-2} u_i b^i) \\
&< v_{n-1} b^{n-1} \\
&\leq v
\end{aligned}$$

由该关系可知，$u - \hat{q} v < v$，根据除法的定义：

$$q = \min\{k : u - vk < v\}$$

因此当 $\hat{q} = \lfloor \frac{u_n b + u_{n-1}}{v_{n-1}} \rfloor$ 时，仍有 $q \leq \hat{q}$。

下面再用反证法证明 $\hat{q} - 2 \leq q$，假设：

$$\hat{q} \geq q + 3 \tag{13}$$

根据 $\hat{q}$ 的定义，可得：

$$\hat{q} \leq \frac{u_n b + u_{n-1}}{v_{n-1}} = \frac{u_n b^n + u_{n-1} b^{n-1}}{v_{n-1} b^{n-1}} \leq \frac{u}{v_{n-1} b^{n-1}}$$

由于 $v = (100 \cdots 0)_b$ 时，总有 $\hat{q} = q$，所以只需考虑 $v > (100 \cdots 0)_b$ 的情形，此时有：

$$0 < (v_{n-1} - 1) b^{n-1} + \sum_{i=0}^{n-2} v_i b^i = v - b^{n-1} < v_{n-1} b^{n-1}$$

即：

$$(v_{n-1} - 1) b^{n-1} \leq v - b^{n-1} < v_{n-1} b^{n-1}$$

所以有：

$$\hat{q} \leq \frac{u}{v_{n-1} b^{n-1}} < \frac{u}{v - b^{n-1}} \tag{14}$$

由准确值 q 的定义可得：

$$q > \frac{u}{v} - 1 \tag{15}$$

由式13，式14和式15，可得：

$$3 \leq \hat{q} - q < \frac{u}{v - b^{n-1}} - (\frac{u}{v} - 1) = \frac{u}{v}(\frac{b^{n-1}}{v - b^{n-1}}) + 1$$

所以：

$$3 < \frac{u}{v}(\frac{b^{n-1}}{v - b^{n-1}}) + 1$$

进而：

$$\frac{u}{v} > 2(\frac{v - b^{n-1}}{b^{n-1}}) \geq 2(v_{n-1} - 1)$$

因为：

$$b - 4 \geq \hat{q} - 3 \geq q = \lfloor \frac{u}{v} \rfloor \geq 2(v_{n-1} - 1)$$

所以 $v_{n-1} < b/2$，与式10矛盾，故 $\hat{q} - 2 \leq q$。

综上所述，若式2和式3成立，那么式4成立。