现代密码学 第二次作业

2017011441 计76 吕传承

一、作业概况

本次作业主要包括以下内容；

1. 编程实现DES、AES-128、SM4三个算法的加解密运算。
2. 编程实现RC4算法，生成16K的密钥流。
3. 使用B-M算法，生成一段序列最短LFSR。
4. 编码实现SHA-2、SM3、SHA-3算法，压缩16K左右的随机数据。

本次作业任务量较重，涉及到的细节非常多，因此需要参考大量文献（参见最后一节）并花费较多时间。

工程情况

本次实现的工程文件树如下：

```
-BM --------------BM.cpp                  B-M 算法的相关函数实现
    --------------BM.h                    B-M 算法的头文件
-Compression  ----compression_utils.h      对于 Hash 函数实现需要的头文件
              ----SHA2.cpp/SHA2.h          SHA2 算法的相关函数实现与头文件
              ----SHA3.cpp/SHA3.h          SHA3 算法的相关函数实现与头文件
              ----SM3.cpp/SM3.h            SM3 算法的相关函数实现与头文件
-Encryption ------AES128.cpp/AES128.h      AES 算法的相关函数实现与头文件
            ------DES.cpp/DES.h            DES 算法的相关函数实现与头文件
            ------SM4.cpp/SM4.h            SM4 算法的相关函数实现与头文件
            ------utils.h/utils.cpp        分组密码辅助工具包
-RC4  ------------RC4.cpp                  RC4 算法的辅助函数
-Makefile                                  Makefile 文件
-main.cpp                                  主函数入口，调用并测试相关算法实现
```

执行方法

1. 在文件根目录执行 `make` 可以自动进行编译链接等工作
2. 在文件根目录执行 `./main` 可以执行一梭子的算法测试工作，产生**较为详细的标准输出**并在根目录下生成对应的中间文件。产生的中间文件说明如下；

```
*.o                    : 编译生成的 OBJ 文件
main                   : 链接生成的主函数文件
plain_text.txt         : 随机生成的 16K bit 原文
*_cipher_text.txt      : 使用相对应的加密算法对原文进行加密得到的二进制密文
*_translate_text.txt   : 使用相对应的解密算法对密文进行解密得到的文件，期望与原文一致
*_hash.txt             : 使用对应的 hash 算法对原文进行 hash 得到的 hash 值
RC4_key_stream.txt     : 使用 RC4 算法得到的密钥流
```

二、分组密码

主要要求实现 DES、AES-128 以及 SM4 三个算法，这里对三个算法分别实现了三个类并对算法中的操作实现相对应的成员函数。

实现框架与实现细节

三个算法主要实现框架比较一致，包括以下部分：

1. **初始化**：产生随机的指定长度的密钥，并根据密钥产生对应的子密钥。
2. **加密轮函数**：对指定的字符串进行对应的加密操作，同时包含辅助操作。这里需要注意加解密的顺序问题。
3. **加密函数**：对原文进行分组并按组进行加密，同时不足一组的需要进行补齐操作。

主要实现细节如下：

1. 使用 string 进行相关原文与密文的读取和传递并使用原地存储的方式进行相关修改。
2. 加密算法在初始化时随机字符串得到指定长度的密钥并存储，使用此密钥进行密钥扩展得到子密钥。
3. 对于相关常数直接作为类的静态成员变量进行存储。但 AES 算法在浏览文献时发现了 S 表的生成方法因此实现了 S 表和逆 S 表的生成过程。
4. 因为需要对 String 使用异或以及移位等操作，因此在 `utils.h` 中进行了相关实现。
5. 经过测试加解密的实现**非常慢**，与作业期望的速度相差甚远（具体可查看标准输出），经过总结主要有以下原因：
    1. 加解密的各种操作的每一步全部使用函数进行封装操作，需要进行频繁的函数传参与调用。
    2. 测试的数据量（16K bit）比较小，使得加解密的权重占比比较小，导致平均速度较慢。
    3. 没有实现并行操作以及对已知结果进行打表等操作，这里可以优化的空间非常大，但由于时间原因没有进行相关优化的实现。

三、序列密码

主要是编程实现RC4算法，生成16K的密钥流以及使用B-M算法，生成一段序列最短LFSR。这里的两个算法并没有特别需要说明的地方，只是按照规定流程按部就班进行实现，并没有特别可以进行优化的地方，具体可见 `BM.cpp` 以及 `RC4.cpp` 的相关实现。

四、Hash函数

主要需要编码实现SHA-2、SM3、SHA-3算法，压缩16K左右的随机数据，这里由于处理方式和加密不同，各个块之间存在数据相关，因此使用不同的数据结构来存储。

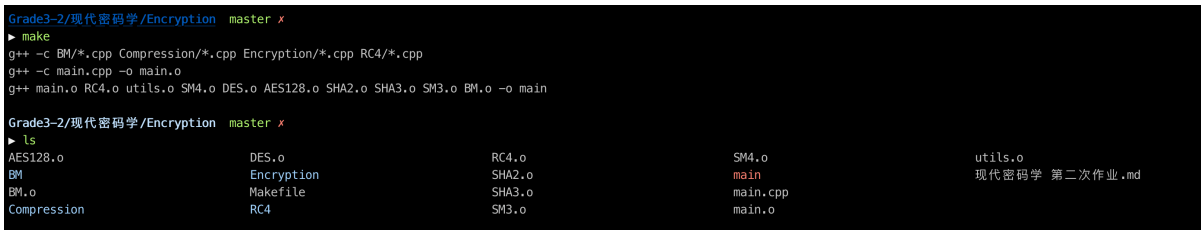
实现框架与实现细节

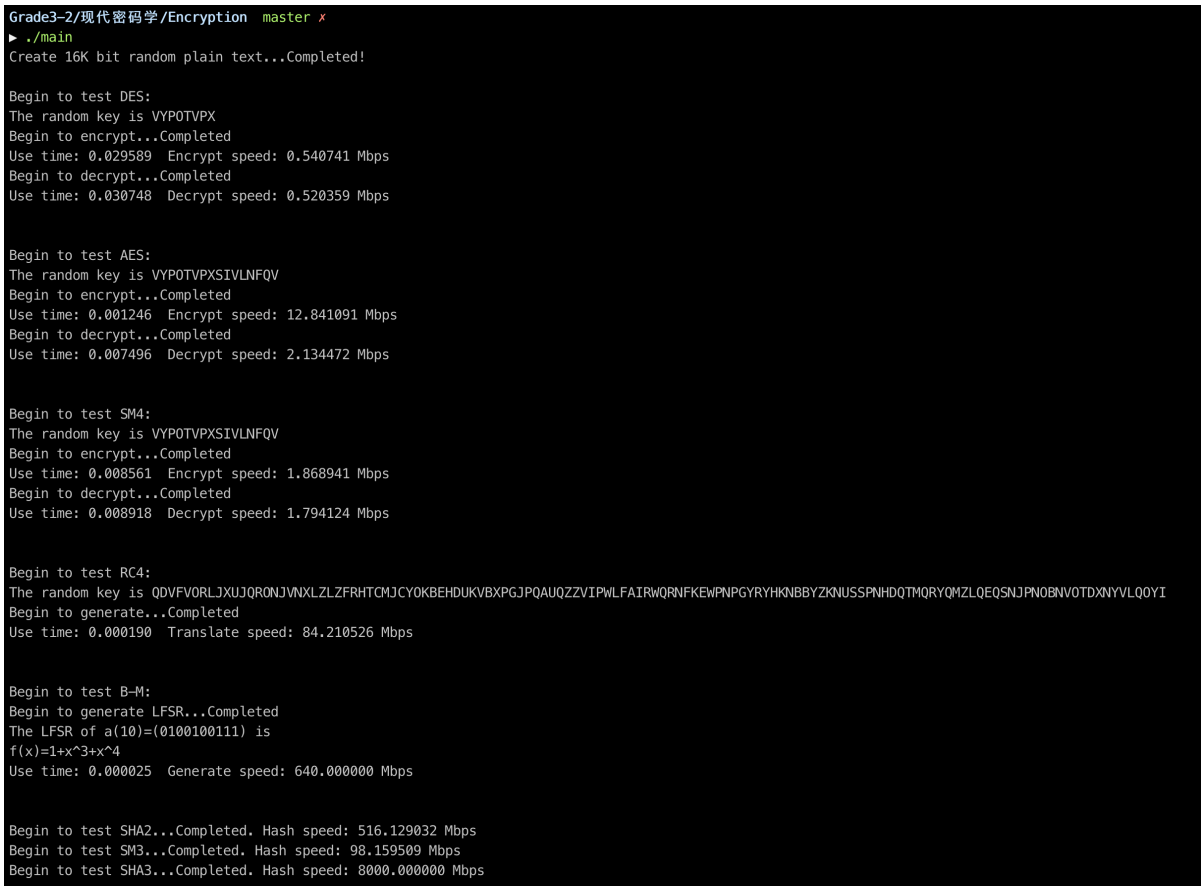
三者主要实现框架大体一致，主要包括：

1. 使用 struct 存储当前 hash 状态，包括需要哈希的数据以及当前块的哈希状态等。
2. 实现初始化函数与更新函数，对 hash struct 进行初始化以及相关的更新操作。
3. 这里进行相关操作的粒度较小，因此使用 unsigned int 以及 unsigned char 进行相关存储，其实分组密码的相关操作和 hash 函数可以进行数据格式的统一，但因为时间原因并没有进行严格的统一。

五、实验结果

1. 参数设置：实验运行时将产生 16K bit 的随机数据，其他参数设置与算法需要的参数相同。
2. 实验数据：实验将随机产生实验数据，请运行后查看标准输出以及产生的中间文件。
3. 实验截图：如下所示标准输出与产生的中间文件（文件对应的含义见上文）





六、实验感想

这是目前本学期的量最重的一次实验，因为需要自己动手写非常多的已有的 “轮子”。同时还有的问题是对部分需要实现的内容模糊不清，没有官方的文档可以查询（或者官方的文档难以寻找）导致大部分时间需要去学习前人已经造好的相关代码进行学习然后自己再实现相关的代码。因此我本次作业主要时间在于学习其他人的实现细节上，花费了非常多的时间，因此我希望之后的作业对于较为模糊不清的算法给出更加详细的文档说明与测试用例。

感谢老师、助教的帮助和指导！

七、参考文献

本人在部分算法内容与实现细节上参考了以下文献。

1. https://blog.csdn.net/weixin_42700740/article/details/102667012
2. https://blog.csdn.net/cg129054036/article/details/83016958
3. https://blog.csdn.net/qq_28205153/article/details/55798628
4. https://www.cnblogs.com/elpsycongroo/p/7819814.html
5. https://github.com/B-Con/crypto-algorithms/blob/master/sha256.h
6. https://zhuanlan.zhihu.com/p/94619052
7. https://wenku.baidu.com/view/8d67d80178563c1ec5da50e2524de518964bd3b6.html?qq-pf-to=pcqq.c2c
8. https://blog.csdn.net/a344288106/article/details/80094878