DES 算法加解密实验

张海斌*

2019 年 11 月

目 录

1 交互界面介绍

我使用 Python 完成了 DES 加密与解密的过程，并且用了 Python 的 Flask 框架制作了一个简单的网页前端界面，如图 1。所以这个项目的依赖只有 Python3 和 Flask 框架。

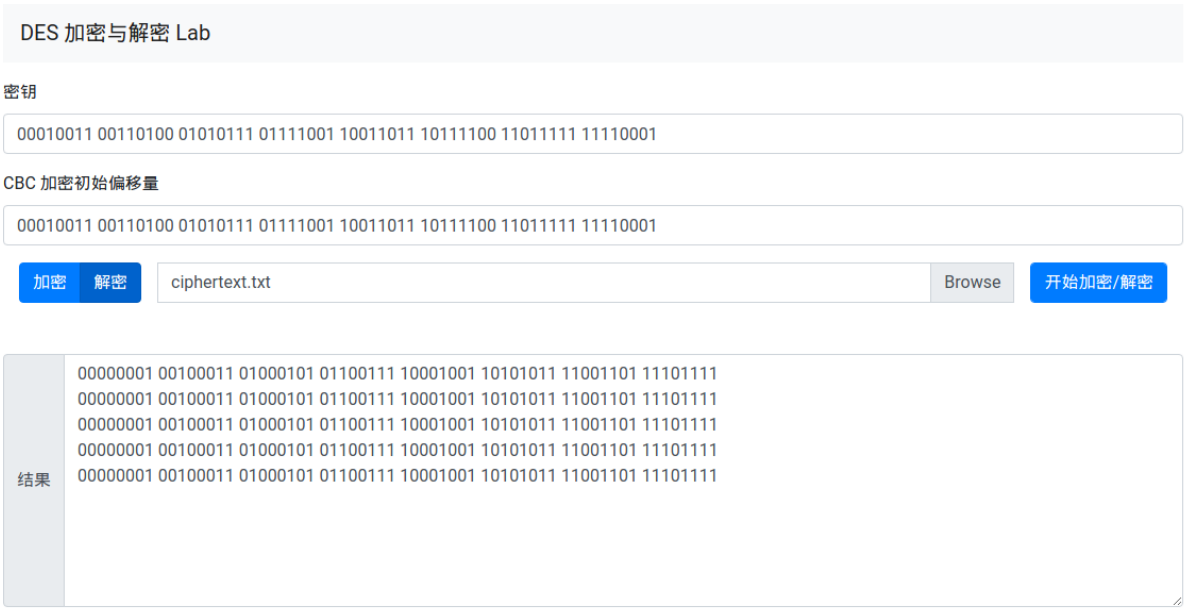

图 1: 前端网页界面

*学号 17307130118

所有数据都是通过 0 和 1 组成的字符串来表示的。第一栏密钥填写使用 DES 算法加密/解密所需的密钥。第二栏 CBC 加密初始偏移量表示使用 CBC 分组加密模式时，明文组前 64 位会与它进行异或运算后再进行加密。第三栏可以选择进行加密还是解密，同时可以选择需要加密/解密的数据文件。最后可以点击” 开始加密/解密” 的按钮开始加密和解密运算，如果输入的数据没有问题，加密的结果就会显示在下面的结果文本框中。

为正确进行 DES 加密与解密，我对输入的数据有一定的限制与判断：密钥的有效位长度为 64 位，只有 0 和 1 为有效字符，在输入框中可以夹杂空格与其它字符，但它们不会被计入有效长度中。同样，CBC 初始偏移量也要求 64 位的有效长度。对于加密时的上传文件来说，文件中的有效长度可以是任意长度。如果有效长度没有与 64 位对齐，在 DES 加密时会在明文后补充 0 直至与 64 位对齐。而对于需要被解密的文本来说，文件中的有效长度必须是 64 的倍数。

对于各种异常情况，前端界面都能正确提示用户，当前两栏内容不符合条件时，如图 2；当没有选择文件时，如图 3；当解密密文文件内容不符合条件时，如图 4。

DES 加密或解密成功的时候，会有提示框进行提示，如图 5。

图 2: 输入框中有错误

2 源代码目录结构

列表 1 展示了实验源代码的目录结构，其中，`实验报告.pdf` 即本文档。`example` 目录中是一些可供参考的样例输入。以 `key.txt` 为密钥，`iv.txt` 为初始偏移量，`text.txt` 为明文经过 CBC 模式的 DES 加密可以得到 `ciphertext.txt` 中的内容。反过来，将输入文件改为 `ciphertext.txt` 密文进行解密也可以得到原来的明文 `text.txt` 中的内容。解密结果如图 5，每个 64 位的明文组都是一样的内容，但是密文中每 64 个密文组都是不同的内容。

`static` 和 `templates` 目录包括了前端网页的 Javascript 代码和 HTML 代码。`des.py` 包括了 Python 实现的 DES 加解密算法和 CBC 分组加密模式。`web.py` 包括了 Flask 接收和处理 HTTP 请求的代码。最后，`run.sh` 和 `run.bat` 分别为在 Linux 系统和 Windows 系统上的运行脚本（前提是安装了 Python 和 Flask）。Flask 可以通过 pip 方便地安装：`pip install flask`。脚本开始运

图 3: 未选择上传文件

图 4: 解密密文长度错误

图 5: 成功解密

```
.
|-- example
|   |-- ciphertext.txt
|   |-- iv.txt
|   |-- key.txt
|   |-- text.txt
|-- static
|   |-- main.js
|-- templates
|   |-- index.html
|-- des.py
|-- run.bat
|-- run.sh
|-- web.py
|-- 实验报告.pdf
```

Listing 1: 代码文件结构树状图

行后会显示类似`Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)`的内容，此时可以打开浏览器，输入网址`http://localhost:5000`可访问前端页面。

3 源代码说明

`des.py`中，首先集中定义了各种矩阵常量。之后是各种功能函数的实现。我根据 DES 算法内容将整个加解密过程拆分为许多部分，每部分都对应于 DES 算法中的某一步。如`F(x, k)`函数就是实现了 DES 算法中的轮函数 F。之后的一些函数如`runDES(text, cipher, mode)`，`runDESwithCBC(text, cipher, mode, IV)`是对整体功能的包装。

`getDESResult(text, cipher, mode, IV)`是`web.py`中直接调用的函数,相比与`runDESwithCBC`函数，它增加了一些对错误输入数据的判断与处理。

对于 8 个 S 盒，我做了一个小的改动。原本每个 S 盒是二维的矩阵，但是我们可以把它看作一维的数组，在该数组中查找时使用二进制下标 $x_0x_5x_1x_2x_3x_4$，其结果与二维 S 盒结果完全相同。所以每个 S 盒可以简化为一维存储。