学号：16430121

常州大学

实 验 报 告

课 程 名 称：信息安全

实 验 名 称：基于 DES 并结合 ECB、CBC 工作模式的加解密算法

指 导 教 师：孙霓刚

学 生 姓 名：林锦雄

学 院（系）：信息数理学院 专 业 班 级：计算机 162

实验起止时间：2019 年 05 月 08 日 至 2019 年 06 月 05 日

基于 DES 并结合 ECB、CBC 工作模式的加解密算法

一、实验目的：

使学生能够利用常用的加密算法、杂凑函数等，设计并实现有效、实用的加解密程序或杂凑程序。

二、实验内容：

1）结合 2 种及以上的分组密码的工作模式，设计并实现一个基于常用加密算法（如 DES、AES、RC5、RC6、RSA 等）的实用加解密程序；或者

2）基于常用的杂凑函数（如 MD5、SHA-1 等），设计并实现一个实用的杂凑程序。

三、实验要求：

1）编程语言不限；

2）所实现的程序应能够对任意文件进行正确的加解密；

3）实验报告内容完整，对设计思路和实现过程有详细的说明。

四、设计思路：

1）电子密码本（ECB）

电子密码本（Electronic codebook，ECB）模式。需要加密的消息按照块密码的块大小被分为数个块，并对每个块进行独立加密。ECB 的工作模式，如图 1 所示。

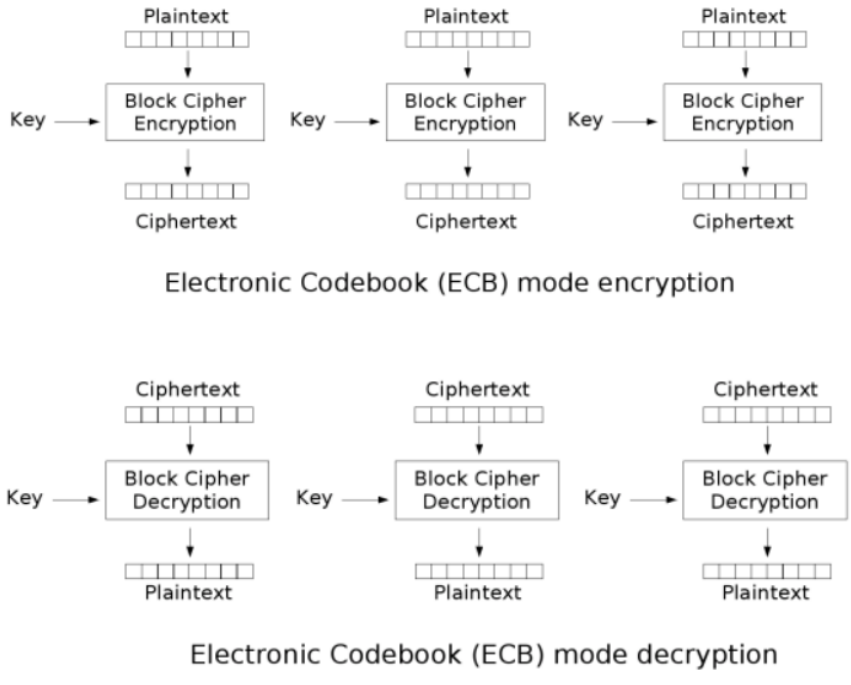


图 1. 电子密码本(ECB)模式

2）密码块链接（CBC）

1976 年，IBM 发明了密码分组链接（CBC，Cipher-block chaining）模式。在 CBC 模式中，每个明文块先与前一个密文块进行异或后，再进行加密。在这种方法中，每个密文块都依赖于它前面的所有明文块。同时，为了保证每条消息的唯一性，在第一个块中需要使用初始化向量。CBC 的工作模式，如图 2 所示。

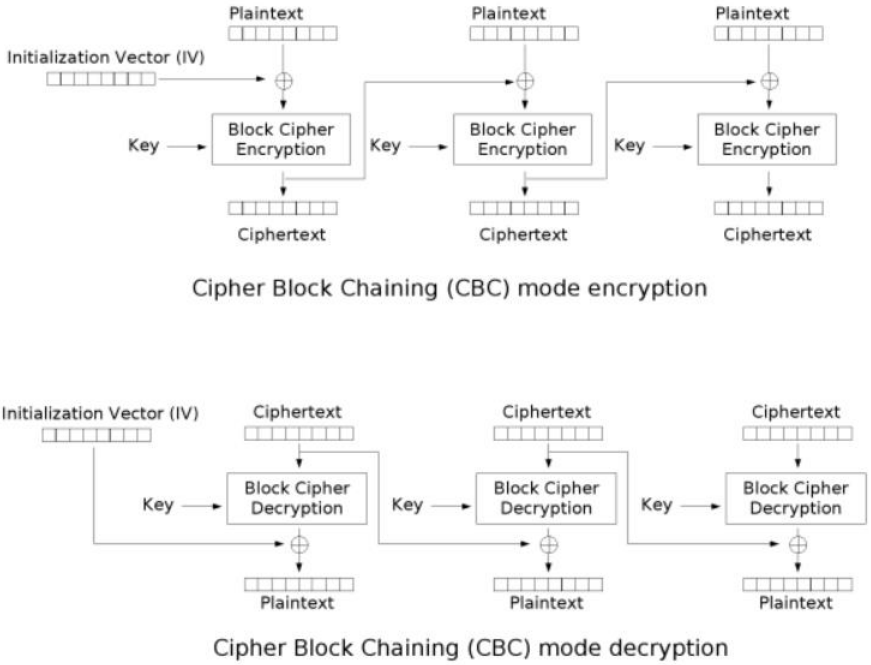


图 2. 密码块链接(CBC)模式

在加密时，明文中的微小改变会导致其后的全部密文块发生改变，而在解密时，从两个邻接的密文块中即可得到一个明文块。因此，解密过程可以被并行化，而解密时，密文中一位的改变只会导致其对应的明文块完全改变和下一个明文块中对应位发生改变，不会影响到其它明文的内容。

CBC 模式加解密过程，如图 3 所示。

若第一个块的下标为1，则CBC模式的加密过程为

$$C_i = E_K(P_i \oplus C_{i-1}),$$
$$C_0 = IV$$

而其解密过程则为

$$P_i = D_K(C_i) \oplus C_{i-1},$$
$$C_0 = IV$$

图 3. CBC 模式加密解密过程

3）DES 加解密算法原理

DES 是 Data Encryption Standard（数据加密标准）的缩写。它是由 IBM 公司研制的一种对称密码算法，美国国家标准局于 1977 年公布把它作为非机要部门使用的数据加密标准，三十年来，它一直活跃在国际保密通信的舞台上，扮演了十分重要的角色。

DES 是一个分组加密算法，典型的 DES 以 64 位为分组对数据加密，加密和解密用的是同一个算法。它的密钥长度是 56 位（因为每个第 8 位都用作奇偶校验），密钥可以是任意的 56 位的数，而且可以任意时候改变。其中有极少数被认为是易破解的弱密钥，但是很容易避开它们不用。所以保密性依赖于密钥。

DES 对称加密算法的工作流程图，如图 4 所示。

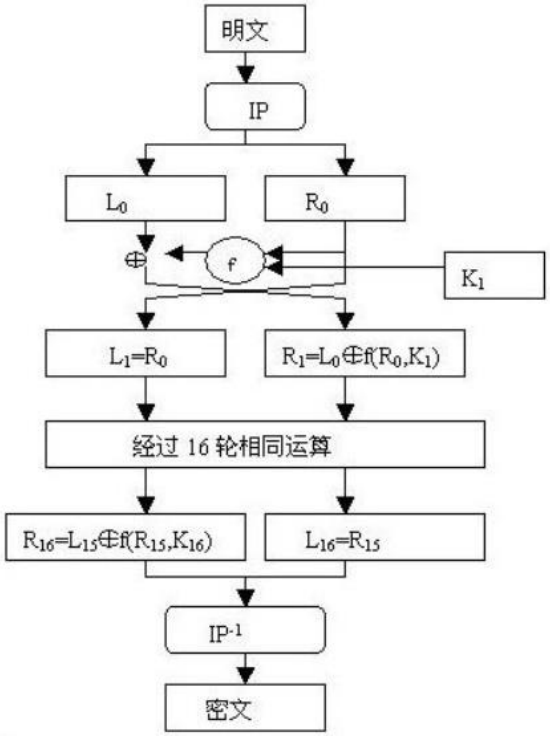


图 4. DES 算法工作流程图

4）结合电子密码本(ECB)和密码块链接(CBC)

本实验结合分组密码的电子密码本(ECB)工作模式和密码块链接(CBC)工作模式，基于 DES 对称加密算法。

加密算法：

A. 获取指定文件中的明文；

B. 使用电子密码本(ECB)工作模式，利用密钥，按照 7 位一组，进行加密；

C. 使用密码块链接(CBC)工作模式，初始化一个二进制 7 位全 1 的数，明文按照 7 位一组分组，初始化数首先与第一组进行亦或运算，得到的数据再使用密钥加密，得到的第一组密文作为第二组的初始化数，依次加密；

D. 得到密文写回文件。

解密算法：

A. 获取指定文件中的密文；

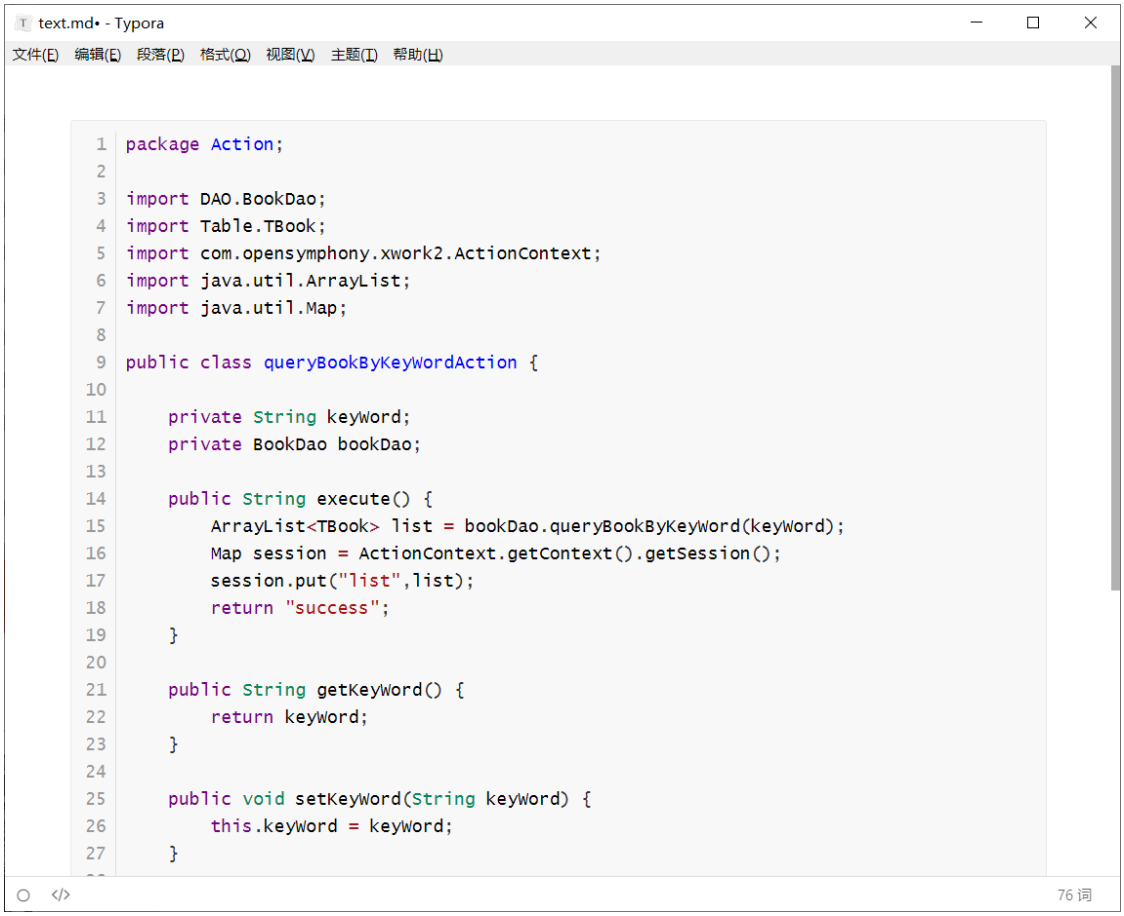
B. 使用密码块链接(CBC)工作模式，密文按照 7 位一组分组，利用密钥进行解密，然后初始化一个二进制 7 位全 1 的数，与解密后的数据进行亦或运算，得到一次解密的数据；

C. 使用电子密码本(ECB)工作模式，利用密钥，按照 7 位一组，进行解密；

D. 得到明文写回文件。

五、实现过程：

创建一个测试文件，并写入测试明文。如图 5 所示。




```
package Action;

import DAO.BookDao;
import Table.TBook;
import com.opensymphony.xwork2.ActionContext;
import java.util.ArrayList;
import java.util.Map;

public class queryBookByKeyWordAction {

    private String keyWord;
    private BookDao bookDao;

    public String execute() {
        ArrayList<TBook> list = bookDao.queryBookByKeyWord(keyWord);
        Map session = ActionContext.getContext().getSession();
        session.put("list",list);
        return "success";
    }

    public String getKeyWord() {
        return keyWord;
    }

    public void setKeyWord(String keyWord) {
        this.keyWord = keyWord;
    }
```

图 5. text.md 中的明文

1）加密方法：

- 首先通过 OpenFile 方法打开一个待加密的文件；
- 调用 ReadFile 方法从待加密文件中获取全部明文，记录；
- 使用 ECB 工作模式，利用密钥对明文进行一轮加密；
- 使用 CBC 工作模式，将一轮加密的数据 7 位一组与初始化数进行亦或，并把亦或结果通过密钥进行二轮加密；
- 调用 WriteFile 方法将加密后的密文写回文件。

```
1.  //加密方法
2.  public void Encrypt() {
3.      chooseFile = OpenFile("请选择待加密文件");
4.      ReadFile(); //从文件中读取明文
5.      //ECB
6.      for(int i = 0;i < sb.length();i++) {
7.          sb.setCharAt(i, (char) ((sb.charAt(i) + key) % 128));
8.      }
```

```
9.      //CBC
10.     int init = initialization;
11.     for(int i = 0;i < sb.length();i++) {
12.         sb.setCharAt(i, (char) (init ^ sb.charAt(i)));
13.         sb.setCharAt(i, (char) ((sb.charAt(i) + key) % 128));
14.         init = sb.charAt(i);
15.     }
16.     WriteFile();     //将密文写回文件
17. }
```

调用加密方法对测试文件进行加密后的密文，如图 6 所示。

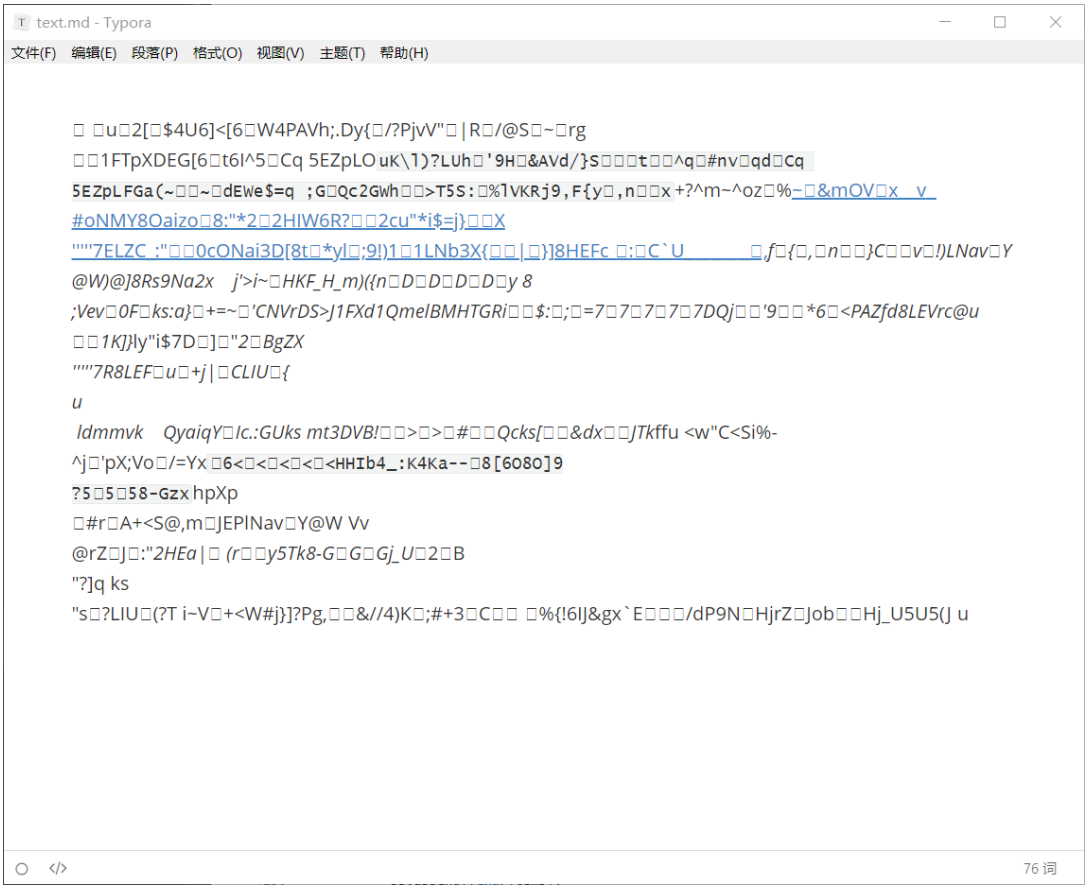


图 6. text.md 加密后的密文

OpenFile 方法：传入标题，利用 Swing 中的 JfileChooser 来打开一个可视化的仅选择文件的文件选择器，最后返回使用者选择的文件。

ReadFile 方法：利用 FileInputStream 文件输入流打开待操作文件，并逐字读取文件中的数据，存储在 StringBuffer 对象中。

WriteFile 方法：利用 FileWriter 对象打开待操作文件，将处理好的保存在 StringBuffer 对象中的数据覆盖式写入文件。

```
1.  //使用文件选择器打开待操作文件
2.  public File OpenFile(String Title) {
3.      jfc = new JFileChooser();
4.      jfc.setFileSelectionMode(JFileChooser.FILES_ONLY);
```

```
5.      jfc.showDialog(new Label(),Title);
6.      return jfc.getSelectedFile();
7.  }
8.
9.  //从文件中读取明文或密文
10. public void ReadFile() {
11.     try {
12.         int temp;
13.         fis = new FileInputStream(chooseFile);
14.         while((temp = fis.read()) != -1) {
15.             sb.append((char)temp);
16.         }
17.     } catch (IOException e) {
18.         e.printStackTrace();
19.     } finally {
20.         try {
21.             fis.close();
22.         } catch (IOException e) {
23.             e.printStackTrace();
24.         }
25.     }
26. }
27.
28. //将明文或密文写回文件
29. public void WriteFile() {
30.     try {
31.         fw = new FileWriter(chooseFile);
32.         fw.write(sb.toString());
33.     } catch (IOException e) {
34.         e.printStackTrace();
35.     } finally {
36.         try {
37.             fw.close();
38.         } catch (IOException e) {
39.             e.printStackTrace();
40.         }
41.     }
42. }
```

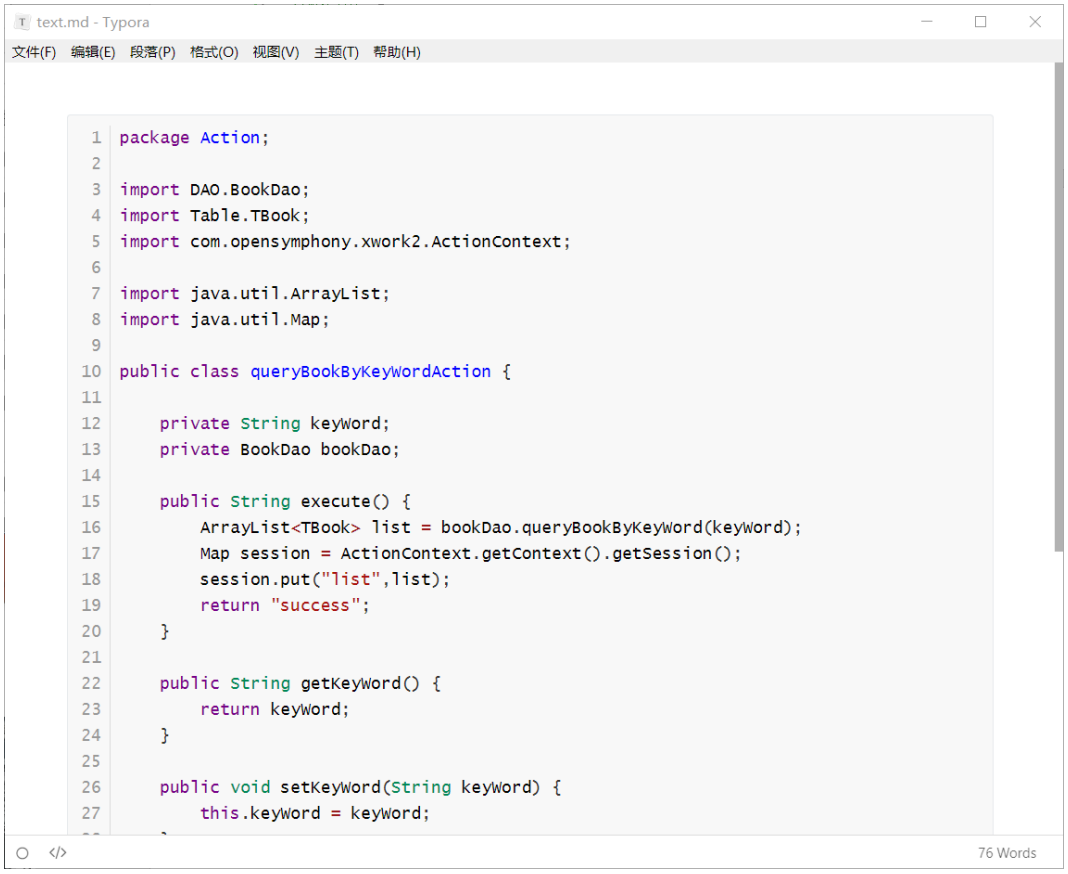
2）解密方法：

- 首先通过 OpenFile 方法打开一个待解密的文件；
- 调用 ReadFile 方法从待加密文件中获取全部密文，记录；
- 使用 CBC 工作模式，将密文通过密钥按照 7 位一组进行解密，并把得到的数据与初始化数进行亦或运算；初始化数变更为上一组的最初的密文，依次解密，得到一轮解密数据；
- 使用 ECB 工作模式，利用密钥对明文再进行一轮解密，得到正确的明文；
- 调用 WriteFile 方法将加密后的明文写回文件。

```
1.  //解密方法
2.  public void Decode() {
3.      chooseFile  = OpenFile("请选择待解密文件");
4.      ReadFile(); //从文件中读取密文
```

```
        //CBC
        int init = initialization, next;
        for(int i = 0;i < sb.length();i++) {
            next = sb.charAt(i);
            sb.setCharAt(i, (char) ((sb.charAt(i) + 128 - key) % 128));
            sb.setCharAt(i, (char) (init ^ sb.charAt(i)));
            init = next;
        }
        //ECB
        for(int i = 0;i < sb.length();i++) {
            sb.setCharAt(i, (char) ((sb.charAt(i) + 128 - key) % 128));
        }
        WriteFile();     //将明文写回文件
    }
```

调用解密方法对被加密后的测试文件进行解密后的源明文，如图 7 所示。




```
package Action;

import DAO.BookDao;
import Table.TBook;
import com.opensymphony.xwork2.ActionContext;

import java.util.ArrayList;
import java.util.Map;

public class queryBookByKeyWordAction {

    private String keyWord;
    private BookDao bookDao;

    public String execute() {
        ArrayList<TBook> list = bookDao.queryBookByKeyWord(keyWord);
        Map session = ActionContext.getContext().getSession();
        session.put("list",list);
        return "success";
    }

    public String getKeyWord() {
        return keyWord;
    }

    public void setKeyWord(String keyWord) {
        this.keyWord = keyWord;
```

图 7. text.md 解密后的源明文

3）可视化界面：

我为本次实验添加了可视化界面，以优化使用者体验，做成了一个小工具的样式。有欢迎语、不同的操作按钮、操作结果反馈等信息，做到让使用者打开此工具就会使用的效果。

```
//信息安全小工具主窗体，省略不重要代码部分
public class Window extends JFrame {

```

```
4.      private JLabel welcome;      //欢迎语
5.      private JLabel result;       //加解密结果
6.      private JButton encrypt;     //加密
7.      private JButton decode;      //解密
8.      private DESAlgorithm des;    //DES 算法
9.
10.     public Window() {
11.         super("信息安全小工具");    //标题
12.     }
13.
14.     private void initWindow() {
15.         //设置窗体大小、位置、窗体大小固定、关闭操作、可见性、使窗体可见
16.         //加入中间容器（自适应窗体大小）
17.     }
18.
19.     //该窗体的中间容器类
20.     class MyJPanel extends JPanel {
21.         public void initMyJPanel() {
22.             //欢迎语
23.             welcome = new JLabel("欢迎使用加解密小工具", JLabel.CENTER);
24.             add(welcome);
25.
26.             //加密
27.             encrypt = new JButton("加密");
28.             encrypt.addActionListener(new ActionListener() {
29.                 @Override
30.                 public void actionPerformed(ActionEvent e) {
31.                     des = new DESAlgorithm();
32.                     des.Encrypt();
33.                     result.setText("加密成功!");
34.                 }
35.             });
36.             add(encrypt);
37.
38.             //解密
39.             decode = new JButton("解密");
40.             decode.addActionListener(new ActionListener() {
41.                 @Override
42.                 public void actionPerformed(ActionEvent e) {
43.                     des = new DESAlgorithm();
44.                     des.Decode();
45.                     result.setText("解密成功!");
46.                 }
47.             });
48.             add(decode);
49.
50.             //加解密结果
51.             result = new JLabel("", JLabel.CENTER);
52.             add(result);
53.         }
54.
55.         @Override
56.         protected void paintComponent(Graphics g) {
57.             //背景
58.         }
59.     }
60. }
```

该信息安全小工具的初始化界面，如图 8 所示。有美观的界面设计、亲切的欢迎语、顺序排列的按钮等。

图 8. 小工具初始化界面

该信息安全小工具的功能操作都会有相应的反馈信息，如图 9 所示。

图 9. 小工具使用加密功能的反馈信息

六、实验心得：

通过本次实验，加深了我对分组密码的六种工作模式的理解，以及各种常用加密算法（如 DES、AES、RC5、RC6、RSA 等），并学会结合多种工作模式来实现加密算法，使加密程序更加安全可靠。

本次实验，我采用的是结合电子密码本(ECB)和密码块链接(CBC)两种分组密码工作模式实现的 DES 加密程序，并且我为其装配了小巧精美的可视化界面使其成为开箱即用的小工具。在开发过程中，须熟练掌握 6 大工作模式和多种常用加密算法，并思考模式和加密算法的选择，并把它们成功结合在一起。本次实验使我对信息安全学科有了进一步的认识，并且掌握了常用的基本加密解密算法，以及能够自行编写相应的加密解密程序。