成 绩 评 定 表

学生姓名	宁高聪	班级学号	1503130115
专　业	网络工程	课程设计题目	多功能加密应用软件
评 语	组长签字：		
成绩			
日期	2018 年 1 月 11 日		

课程设计任务书

学　院	信息科学与工程学院	专 业	网络工程
学生姓名	宁高聪	班级学号	1503130115
课程设计题目	多功能加密应用软件		

实践教学要求与任务:

1. 任务描述

通过 JAVA 编程实现移位密码、仿射密码、维吉尼亚密码和置换密码。

2. 具体要求

(1) 掌握密码学基础中古典加密体制的几种算法

(2) 了解并掌握古典加密算法的原理与实现

(3) 了解古典算法的特性及加解密规则

工作计划与进度安排:

第一阶段：开题，确定算法。

第二阶段：设计代码，调试程序。

第三阶段：书写设计报告。

第四阶段：程序验收、答辩。

指导教师： 2018 年 1月 2日	专业负责人： 2018 年 1 月 2 日	学院教学副院长： 2018 年 1 月 2 日

摘 要

本次设计是一个简易的多功能加密软件，可以实现移位密码、仿射密码、维吉尼亚密码和置换密码四种加密方式对文本的加密或解密，使用 Java 语言实现加密处理，HTML 作为前台为用户提供操作界面，并使用了 bootstrap 框架对界面进行了美化，为用户提供了友好的交互方式。

关键词 古典加密；Java；JSP

目　录

1 课程设计内容

1.1 课程设计题目

多功能加密应用软件

1.2 课程内容及设计要求

(1) 掌握密码学基础中古典加密体制的几种算法
(2) 了解并掌握古典加密算法的原理与实现
(3) 了解古典算法的特性及加解密规则

2 课程设计原理

2.1 古典加密相关知识

加密：将明文按固定长 m 分组，即每行 m 个字母，在密钥控制下按某一顺序交换列，最后按列优先的顺序依次读出，即产生了密文。解密：逆过程。以下是几种古典加密算法简要说明：

1) 置换密码

把明文中的字母重新排列，字母本身不变，但其位置改变了，这样编成的密码称为置换密码。置换密码是把明文中的字母顺序倒过来，然后截成固定长度的字母组作为密文。

2) 代换密码

代换密码是指先建立一个替换表，加密时将需要加密的明文依次通过查表，代换为相应的字符，明文字符被逐个替换后，生成无任何意义的字符串，即密文，代换密码的密钥就是其代换表 。

代换密码是将明文中的字符替代成其他字符。

3) 多表代换密码

单表代替密码的安全性不高，一个原因是一个明文字母只由一个密文字母代替。可以利用频率分析来破译。故产生了更为安全的多表代换密码，即构造多个密文字母表，在密

钥的控制下用以一系列代换表依次对明文消息的字母序列进行代换。著名的多表代替密码有 Vigenere 密码等。

3 总体方案设计

3.1 设计流程图

界面流程图如图 3.1 所示，用户可以在加密界面自由选择加密方式或者解密，然后输入文本和秘钥，点击提交后后台获取用户输入的内容，根据用户的选择转到相应的模块处理，处理完成后将结果传递回前台展示。

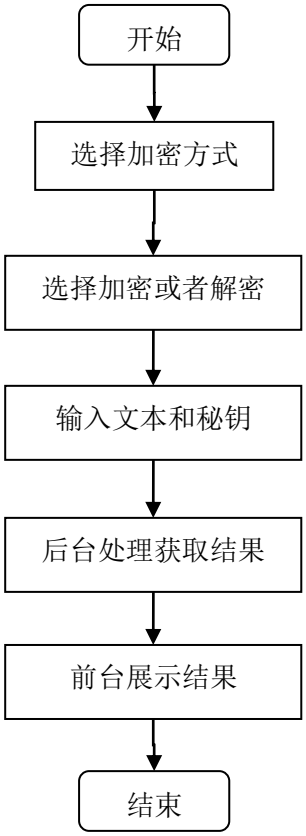

图 3.1 设计流程图

3.2 关键模块及代码实现

3.2.1 界面设计模块

界面设计使用 HTML 进行设计，共分为两个界面，分别为加密和结果的界面，加密界面使用了表单来供用户输入信息，表单包括两个下拉菜单和两个输入框，输入完成后通过提交按钮提交输入的信息。结果显示界面使用表格来显示，第一行是标题，第二行是本次加密或解密的结果，之后几行则是历史操作的信息。同时在列中输出了操作类型，加密文本，操作结果等信息。

加密界面代码

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>加密</title>
    <!-- 最新版本的 Bootstrap 核心 CSS 文件 -->
    <link rel="stylesheet"
          type="text/css"
          href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css"

integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
          crossorigin="anonymous">

<script
            src="http://code.jquery.com/jquery-3.2.1.min.js"
            integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4="
            crossorigin="anonymous"></script>
</head>
<body>
<script type="text/javascript">
    $(document).ready(function (value) {
        var text = "${text}";
```

```
            if (text.trim() === "") {
                console.log("null");
            } else {
                console.log("not null");
                $("#text").attr("value", text);
            }
        })
</script>
<br><br><br>
<div class="row">

    <div class="col-xs-4 col-sm-4 col-md-4 col-lg-4">
    </div>

    <div class="col-xs-4 col-sm-4 col-md-4 col-lg-4">
        <nav class="navbar navbar-default">
            <div class="container-fluid">
                <!-- Brand and toggle get grouped for better mobile display -->
                <div class="navbar-header">
                    <a class="navbar-brand" href="/encrypt"><p class="text-primary">加
密</p></a>
                </div>
                <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                    <a class="navbar-brand" href="/result">结果</a>
                </div>
            </div>
        </nav>
    </div>

    <div class="col-xs-4 col-sm-4 col-md-4 col-lg-4">
```

```
    </div>
</div>
<div><br><br><br><br><br></div>
<div class="row">
    <div class="col-xs-4 col-sm-4 col-md-4 col-lg-4"></div>

    <div class="col-xs-4 col-sm-4 col-md-4 col-lg-4">
        <form class="form-horizontal" action="/encrypt" method="post">
            <div class="form-group">
                <label class="col-md-3 control-label">加密方式</label>
                <div class="col-md-9">
                    <select class="form-control" name="type" onclick="selectItem(this)">
                        <option value="0" selected>请选择</option>
                        <c:forEach items="${types}" step="1" var="item">
                            <option value="${item.order}" >${item.note}</option>
                        </c:forEach>
                    </select>
                </div>
            </div>
            <div class="form-group">
                <label class="col-md-3 control-label">加密|解密</label>
                <div class="col-md-9">
                    <select class="form-control" name="actionType"
onclick="selectEncryptType(this)">
                        <option value="0" >加密</option>
                        <option value="1" >解密</option>
                    </select>
                </div>
            </div>
```

```
            <div class="form-group">
                <label for="text" id="text_label" class="col-md-3 control-label">明文
</label>
                <div class="col-md-9">
                    <input name="text" class="form-control" id="text" placeholder="明文
">
                </div>
            </div>

            <div class="form-group" id="tip_group" style="display: none">
                <div class="col-xs-3 col-sm-3 col-md-3 col-lg-3">
                </div>

                <div class="col-xs-9 col-sm-9 col-md-9 col-lg-9">
                    <c:forEach items="${types}" step="1" var="item">
                        <div id="tip${item.order}" class="alert alert-info" style="display:
none">${item.keyTip}</div>
                    </c:forEach>
                </div>
            </div>

            <div class="form-group" id="key_group">
                <label for="key" id="key_label" class="col-md-3 control-label">秘钥
</label>
                <div class="col-md-9">
                    <input name="key" class="form-control" id="key" placeholder="秘钥
">
                </div>
            </div>
```

```
                <div class="form-group">
                    <div class="col-xs-6 col-sm-6 col-md-6 col-lg-6"></div>
                    <div class="col-xs-2 col-sm-2 col-md-2 col-lg-2">
                        <button type="submit" id="submit" class="btn btn-primary">加密
</button>
                    </div>
                    <div class="col-xs-4 col-sm-4 col-md-4 col-lg-4"></div>
                </div>
            </form>
        </div>

        <div class="col-xs-4 col-sm-4 col-md-4 col-lg-4"></div>
</div>
<script type="text/javascript">
        function selectItem(select) {
            var index = select.selectedIndex;
            console.log(index);
            <c:forEach items="${types}" var="item">
                var $tip = $("#tip" + ${item.order});
                if (index === ${item.order}) {
                    $("#tip_group").css("display", "block");
                    $tip.css("display", "block");
                } else {
                    console.log(index === ${item.order});
                    console.log("order: " + ${item.order});
                    $tip.css("display", "none");
                }
            </c:forEach>
        }
```

```
    function selectEncryptType(select) {
        var index = select.selectedIndex;
        var $submit = $("#submit");
        var $text = $("#text");
        var $plaintext_label = $("#plaintext_label");
        if (index === 1) {
            $submit.text("解密");
            $plaintext_label.text("密文");
            $text.attr("placeholder", "密文");
        } else {
            $submit.text("加密");
            $plaintext_label.text("明文");
            $text.attr("placeholder", "明文");
        }
    }
</script>
</body>
</html>
```

结果显示界面主要代码

```
<table class="table table-hover">
    <thead>
    <tr>
        <th>#</th>
        <th>加密类型</th>
        <th>状态</th>
        <th>文本</th>
        <th>秘钥</th>
        <th>结果</th>
    </tr>
```

```
</thead>
<tbody>
<tr class="success">
    <td>${info.id}#解析结果</td>
    <td>${info.eType.note}#${info.actionType == 0 ? "加密" : "解密"}</td>
    <td>${info.success ? "成功" : "失败"}</td>
    <td>${info.text}</td>
    <td>${info.key}</td>
    <td>${info.actionType == 0 ? info.ciphertext : info.plaintext}</td>
</tr>

<tr class="active">
    <td>#</td>
    <td>#</td>
    <td>#</td>
    <td>#</td>
    <td>#</td>
    <td>#</td>
</tr>

<c:forEach items="${infoList}" begin="0" end="${infoList.size()}" step="1" var="item">
    <tr class="info">
        <td>${item.id}</td>
        <td>${item.eType.note}#${item.actionType == 0 ? "加密" : "解密"}</td>
        <td>${item.success ? "成功" : "失败"}</td>
        <td>${item.text}</td>
        <td>${item.key}</td>
        <td>${item.actionType == 0 ? item.ciphertext : item.plaintext}</td>
    </tr>
</c:forEach>
```

```
    </tbody>
</table>
```

3.2.2 后台流程控制模块

后台处理模块使用 Spring MVC，通过拦截器拦截从前台到达的请求，在控制器中进行处理后将结果回传到前台。

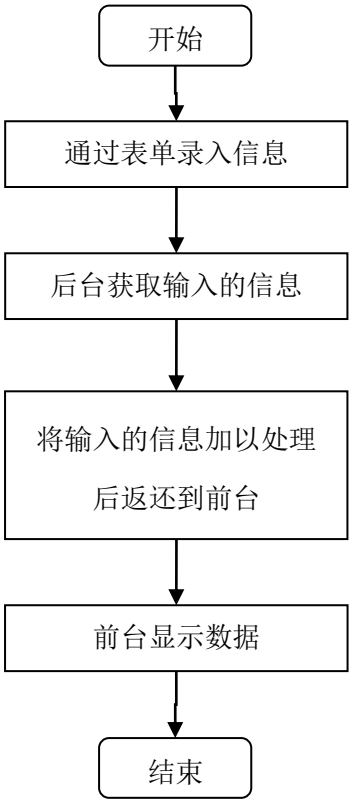

图 3.1 设计流程图

模块实现代码

```
@Controller
public class EncryptController implements IGson {

    private IEncrypter encrypter;
    /**
     * 储存加密结果的集合
     */
```

```
    private static List<Info> resultList = new ArrayList<>();
    private static int count = 0;

    /**
     * 转到结果界面
     * @return
     */
    @RequestMapping(value = "/result", method = RequestMethod.GET)
    public ModelAndView show(Model m) {
        ModelAndView mav = new ModelAndView("result");
        mav.addObject("infoList", resultList);
        return mav;
    }

    /**
     * 转到加密界面，加密后转到结果界面
     * @param info  表单信息
     */
    @RequestMapping(value = "/encrypt", method = {RequestMethod.POST,
RequestMethod.GET})
    public ModelAndView view(HttpServletRequest request, @ModelAttribute("info") Info
info){

        String path_result = "result";
        String path_encrypt = "encrypt";
        ModelAndView mav = new ModelAndView();
        System.out.println("info: " + gson.toJson(info));
        String type = info.getType();
        String key = info.getKey();
        //设置初始状态为成功
```

```
info.setSuccess(true);
if (type == null || "0".equals(type)) {
    mav.setViewName(path_encrypt);
    mav.addObject("types", EType.types);
    System.out.println("====================");
    return mav;
}
try {
    //开始加密过程
    switch (type) {
        case "1":
            encrypter = new Shifting(Integer.parseInt(key));
            break;
        case "2":
            encrypter = new Replacement(parseStringToIntArr(key));
            break;
        case "3":
            encrypter = new Vigenere(parseStringToIntArr(key));
            break;
        case "4":
            int[] keyArr = parseStringToIntArr(key);
            if (keyArr == null || keyArr.length < 2) {
                throw new IllegalArgumentException("秘钥长度非法");
            } else {
                encrypter = new AffineCipher(keyArr[0], keyArr[1]);
            }
            break;
    }

    if (encrypter == null) {
```

```
                encrypter = new ErrorEncrypter(ErrorEncrypter.ERROR_NULL_KEY);
                info.setSuccess(false);
                info.setMsg("秘钥不合法");
            }
        } catch (Exception e) {
//              e.printStackTrace();
            info.setSuccess(false);
            encrypter = new ErrorEncrypter("秘钥不合法");
        }
        String ciphertext = "";
        String plaintext = "";
        if (info.getText() == null || "".equals(info.getText())) {
            info.setSuccess(false);
            encrypter = new ErrorEncrypter("未输入明文");
        }

        try {
            if ("1".equals(info.getActionType())) {
                //解密
                plaintext = encrypter.decode(info.getText());
            } else {
                //加密
                ciphertext = encrypter.encrypt(info.getText());
            }
        } catch (IllegalArgumentException e) {
            plaintext = ciphertext = e.getMessage();
        }
        System.out.println("ciphertext: " + ciphertext);
        info.setPlaintext(plaintext);
        info.setCiphertext(ciphertext);
```

```
            info.setId(count++);
            info.seteType(EType.get(Integer.parseInt(info.getType())));
            resultList.add(info);
            mav.addObject("info", info);
            mav.addObject("infoList", resultList);
            System.out.println(gson.toJson(resultList));
            mav.setViewName(path_result);
            System.out.println("====================");
            return mav;
        }

        private int[] parseStringToIntArr(String key) {
            String[] keySplit;
            int[] keyArr;
            keySplit = key.split(" ");
            keyArr = new int[keySplit.length];
            for (int i = 0; i < keySplit.length; i++) {
                try {
                    keyArr[i] = Integer.parseInt(keySplit[i]);
                } catch (NumberFormatException e) {
                    return null;
                }
            }
            return keyArr;
        }
}
```

3.2.3 加密模块

加密模块设计了 IEncrypter 接口，接口中有加密和解密的方法，具体的操作类通过实现 IEncrypter 接口并实现相应的方法来进行加密或解密操作。

接口定义类

```
public interface IEncrypter {
    default String encrypt(String plaintext) {
        return encrypt(plaintext.toCharArray());
    }

    String encrypt(char[] plaintext);

    default String decode(String ciphertext) {
        return decode(ciphertext.toCharArray());
    }

    String decode(char[] ciphertext);
}
```

维吉尼亚加密类

```
public class Vigenere implements IEncrypter {
    private final int[] key;

    public Vigenere(int[] k) {
        if (k == null || k.length == 0) {
            throw new IllegalArgumentException("秘钥不能为空");
        }

        this.key = k;
    }

    @Override
```

```
    public String encrypt(char[] plaintext) {
        for (int i = 0; i < plaintext.length; ) {

            //按秘钥长度分组加密
            for (int j = 0; j < key.length; j++) {
                if ((i + j) >= plaintext.length) {
                    return String.valueOf(plaintext);
                }

                char c =
CharAndNumber.numberToChar((CharAndNumber.charToNumber(plaintext[i + j]) + key[j]) %
26);
                plaintext[i + j] = c;
            }
            //跳过已经加密过的数据
            i += key.length;
        }

        return String.valueOf(plaintext);
    }

    @Override
    public String decode(char[] ciphertext) {
        for (int i = 0; i < ciphertext.length; ) {
            //按秘钥长度分组加密
            for (int j = 0; j < key.length; j++) {
                if ((i + j) >= ciphertext.length) {
                    return String.valueOf(ciphertext);
                }
                int n = (CharAndNumber.charToNumber(ciphertext[i + j]) - key[j]) % 26;
```

```
                n = n < 0 ? n + 26 : n;
                char c = CharAndNumber.numberToChar(n);
                ciphertext[i + j] = c;
            }
            //跳过已经加密过的数据
            i += key.length;
        }

        return String.valueOf(ciphertext);
    }
}
```

仿射密码加密类

```
public class AffineCipher implements IEncrypter {
    private int keyA;
    private int keyB;
    private int gcdKeyA;

    public AffineCipher(int kA, int kB) {
        this.keyA = kA;
        this.keyB = kB;
        gcdKeyA = gcd(kA);
        if (gcdKeyA == 0) {
            throw new IllegalArgumentException("秘钥 A 不合法");
        }
    }

    @Override
    public String encrypt(char[] text) {
        final int length = text.length;
        ArrayList<Character> list = new ArrayList<>(length);
```

```
        for (int i = 0; i < length; i++) {
            list.add(text[i]);
        }

        for (int i = 0; i < length; i++) {
            //字符转换成 0-25 的数字
            int plaintext = CharAndNumber.charToNumber(list.get(i));
            int ciphertext    = (keyA * plaintext + keyB) % 26;
            list.set(i, CharAndNumber.numberToChar(ciphertext));
        }

        StringBuilder result = new StringBuilder();
        for (Character character : list) {
            result.append(character);
        }
        return result.toString();
    }

    @Override
    public String decode(char[] text) {
        final int length = text.length;
        ArrayList<Character> list = new ArrayList<>(length);
        for (int i = 0; i < length; i++) {
            list.add(text[i]);
        }

        for (int i = 0; i < length; i++) {
            //字符转换成 0-25 的数字
            int plaintext = CharAndNumber.charToNumber(list.get(i));
            int ciphertext    = (plaintext - keyB) * gcdKeyA % 26;
```

```
                ciphertext = ciphertext < 0 ? ciphertext + 26 : ciphertext;
                list.set(i, CharAndNumber.numberToChar(ciphertext));
            }

            StringBuilder result = new StringBuilder();
            for (Character character : list) {
                result.append(character);
            }
            return result.toString();
        }

        private int gcd(int keyA) {
            for (int i = 0; i < 26; i++) {
                if (i * keyA % 26 == 1) {
                    return i;
                }
            }

            return 0;
        }
}
```

4 系统运行结果与分析

（1） 加密运行结果

加密界面如图 4.1 所示,界面中的四个输入框，第一个输入框可以选择需要使用的加密类型；第二个输入框可以选择进行的是加密还是解密；第三个则是输入的文本，如果是加密，就输入明文，如果是解密，则输入密文；第四个输入框输入秘钥；输入完成后点击加密按钮，即可进行操作。

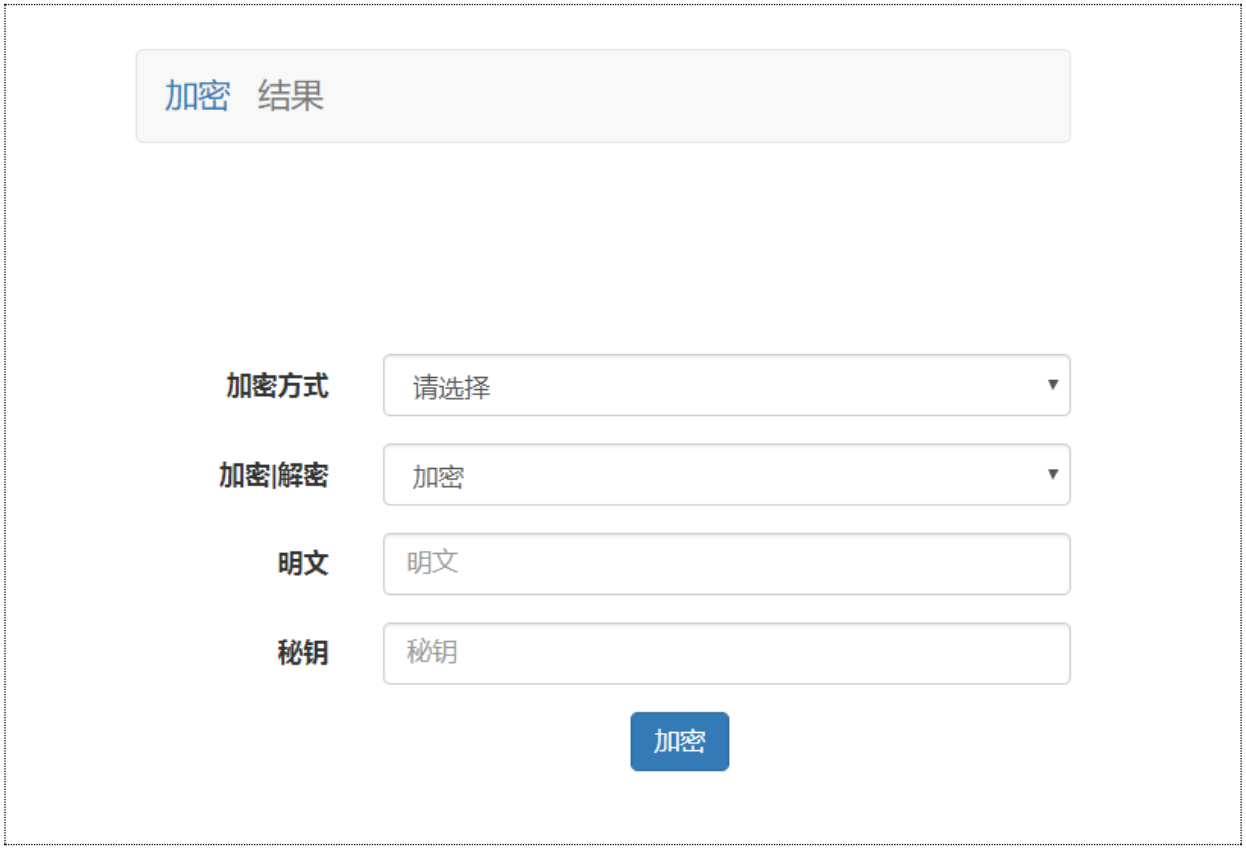

图 4.1 初始加密界面

图 4.2 仿射解密界面

（2）加密结果界面

加密结果如图 4.2 所示：加密结果的显示不仅显示了当前加密的结果，并且显示了历史加密的结果，显示值全为#的一行将当前结果与历史结果分隔开，其余每一行显示一个加密的数据信息。第一列显示的是序列号码，第二列显示加密或者解密及操作类型，第三列显示加密的状态，第四列为需要操作的文本，第五列是使用的秘钥，第六列则是最终的结果，如果是加密，结果就显示密文，解密结果则显示明文，如果出现错误，结果将显示错误信息。

图 4.4 加密结果界面

5 总结

本设计经过近一周的努力，基本满足了一个多功能加密软件冲的基本要求。完成后的程序实现了移位加密，置换加密，维吉尼亚加密和仿射加密的四种加密方式，可以对输入的文本和秘钥按照指定的加密方式进行加密或者解密，并输出加密后的结果。课程设计期间，不仅加深了对课程上学到的知识的印象，还学到了许多课本上没有的知识，积累了更多实践经验，动手能力和解决问题的能力。此次课程设计使我受益匪浅。

参考文献

[1] 付永刚.计算机信息安全[M].北京：清华大学出版社，2012.
[2] 张红旗，王鲁等.信息安全技术[M].北京：高等教育出版社，2008.
[3] 沈昌祥.信息安全导论[M].北京：电子工业出版社，2009.
[4] 徐春香.现代密码学[M].成都：电子科技大学出版社，2008.