密码学基础实验报告

東北大學

实验名称	Vigenere 和 Column Permutation 的编程实现
班　　级	软信 2201
学　　号	20226694
姓　　名	魏董帅
日　　期	2024 年 6 月 5 日
成　　绩	
评 阅 人	

软件学院

一、 实验目的与意义

理解经典密码算法原理:

通过编程实现 Vigenère 密码和列置换密码,可以深入了解这两种经典的多表代换密码的工作原理,包括密钥生成、加密解密过程等关键步骤。这有助于加深对对称加密算法的理解。

实践加密技术:

编程实现这两种密码算法可以将理论知识转化为实践操作,培养学生的编程能力和解决实际问题的能力。这对于未来从事密码学或信息安全相关工作很有帮助。

安全性分析:

Vigenère 密码和列置换密码都存在一定的安全隐患,通过编程实现可以进一步了解其安全性问题,如密钥长度对安全性的影响、频率分析攻击等。这有助于认识经典密码算法的局限性,为学习更安全的加密算法奠定基础。

密码算法比较:

通过编程实现并比较 Vigenère 密码和列置换密码,可以让学生了解不同密码算法的特点和差异,为选择合适的加密算法提供参考。这有助于培养学生的密码算法选择和应用能力。

二、 实验环境

操作系统：Windows 操作系统
调试软件：Code::Blocks
版本号：20.03
上机地点：信息 B405

三、 实验的预习内容

Vigenere 密码是一种多表密码，使用一系列凯撒密码来加密文本。它使用一个关键字来加密文本，通过将关键字的字母与明文的字母相加来生成密文。在编程实现 Vigenere 密码时，首先需要编写一个函数来生成 Vigenere 表，然后编写加密和解密函数来使用 Vigenere 表进行加密和解密操作。

Column permutation 密码是一种置换密码，它通过对明文进行列置换来生成密文。在编程实现 Column permutation 密码时，需要编写一个函数来进行列置换操作，以及加密和解密函数来使用列置换进行加密和解密操作。

在预习阶段，我主要学习了 Vigenere 密码和 Column permutation 密码的原理和实现方式。我了解了 Vigenere 表的生成方法以及加密和解密的算法。对于 Column permutation 密码，我学习了列置换的原理和实现方式。

四、 实验的步骤与调试方法

实验的大致步骤：

编写加密和解密函数
编写测试用例,验证函数的正确性
对测试用例进行调试和修正,直到所有测试用例都通过
对实际输入进行测试,确保函数能正确处理各种输入情况

调试方法

检查输入数据是否正确:确保输入的明文和密钥都是正确的。

逐步调试代码:使用 print 语句或调试器逐步检查代码的执行过程,确保每个步骤都

正确。

测试边界情况:测试空字符串、单个字符、长度不同的密钥等边界情况,确保代码能够正确处理这些情况。

比较结果:将加密和解密的结果进行比较,确保解密后的明文与原始明文一致。

五、 实验数据与实验结果

column

明文：weidongshuai　　加密后结果：enuwohdsiiga

密文：weidongshuai　　解密后结果：dwugoeasniih

Vigenere

需要加密的文本：WEIDONGSHUAI

密钥：KEY

加密后文本：GIGNSLQWFEM

解密后文本：WEIDONGSHUAI

六、 实验用程序清单（要有注释）

column

```
#include <stdio.h>
#include <string.h>

// 加密函数
void encrypt(char *plaintext, int key[], int columns) {
    int length = strlen(plaintext);
    int rows = (length + columns - 1) / columns;
    char matrix[rows][columns];

    int k = 0;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            if (k < length) {
                matrix[i][j] = plaintext[k++];
            } else {
                matrix[i][j] = ' ';  // 填充空格
            }
        }
    }

    for (int j = 0; j < columns; j++) {
        int col = key[j];
        for (int i = 0; i < rows; i++) {
            printf("%c", matrix[i][col]);
        }
    }
}
```

```
// 解密函数
void decrypt(char *ciphertext, int key[], int columns) {
    int length = strlen(ciphertext);
    int rows = (length + columns - 1) / columns;
    char matrix[rows][columns];

    int k = 0;
    for (int j = 0; j < columns; j++) {
        int col = key[j];
        for (int i = 0; i < rows; i++) {
            matrix[i][col] = ciphertext[k++];
        }
    }

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            printf("%c", matrix[i][j]);
        }
    }
}

int main() {
    char plaintext[100];
    int key[] = {1, 0, 3, 2};  // 列置换的顺序
    int columns = 4;

    printf("请输入明文: ");
    fgets(plaintext, 100, stdin);
    plaintext[strcspn(plaintext, "\n")] = 0;  // 去除末尾的换行符
    printf("明文: %s\n", plaintext);
    printf("加密后: ");
    encrypt(plaintext, key, columns);

    char ciphertext[100];
    printf("\n\n请输入密文: ");
    fgets(ciphertext, 100, stdin);
    ciphertext[strcspn(ciphertext, "\n")] = 0;  // 去除末尾的换行符
    printf("解密后: ");
    decrypt(ciphertext, key, columns);

    return 0;
}
```

Vigenere

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

// 使用密钥对文本进行加密
char* vigenere_encrypt(char* text, char* key) {
    int textLen = strlen(text);
    int keyLen = strlen(key);
    char* encrypted = (char*)malloc(textLen + 1);
    int i;
    for (i = 0; i < textLen; i++) {
        // 对每个字符进行加密
        encrypted[i] = ((text[i] - 'A' + key[i % keyLen] - 'A') % 26) + 'A';
    }
    encrypted[i] = '\0';
    return encrypted;
}

// 使用密钥对文本进行解密
char* vigenere_decrypt(char* text, char* key) {
    int textLen = strlen(text);
    int keyLen = strlen(key);
    char* decrypted = (char*)malloc(textLen + 1);
    int i;
    for (i = 0; i < textLen; i++) {
        // 对每个字符进行解密
        decrypted[i] = ((text[i] - 'A' - (key[i % keyLen] - 'A') + 26) % 26) + 'A';
    }
    decrypted[i] = '\0';
    return decrypted;
}

int main() {
    char text[100]; // 明文
    char key[100]; // 密钥

    printf("请输入要加密的文本(大写): ");
    scanf("%s", text);

    printf("请输入密钥: ");
    scanf("%s", key);

    // 加密文本
```

```
    char* encryptedText = vigenere_encrypt(text, key);
    printf("加密后的文本: %s\n", encryptedText);

    // 解密文本
    char* decryptedText = vigenere_decrypt(encryptedText, key);
    printf("解密后的文本: %s\n", decryptedText);

    free(encryptedText);
    free(decryptedText);

    return 0;
}
```

七、 思考题（必需回答）写明如下问题

1. Vigenere 密码的原理是什么？

   加密过程:

   将明文和关键词对齐,关键词可以重复使用直到与明文等长。

   对于明文中的每个字母,找到其在字母表中的位置。

   将该字母的位置与关键词对应字母的位置相加(模 26),得到密文字母的位置。

   将得到的位置转换为对应的字母,即为密文字母。

   解密过程:

   将密文和关键词对齐,关键词可以重复使用直到与密文等长。

   对于密文中的每个字母,找到其在字母表中的位置。

   将该字母的位置与关键词对应字母的位置相减(模 26),得到明文字母的位置。

   将得到的位置转换为对应的字母,即为明文字母。

2. Vigenere 密码的主要缺陷有哪些？

   周期性:

   Vigenère 密码使用关键词重复加密,这导致密文存在周期性。

   如果攻击者能够发现密文中的周期性,就可以尝试破解密码。

   密码强度有限:

   Vigenère 密码的强度主要取决于关键词的长度和复杂度。

   如果关键词太短或太简单,密码就很容易被破解。

   频率分析攻击:

   由于 Vigenère 密码仍然使用字母频率,因此频率分析攻击仍然有效。

   攻击者可以分析密文中字母的出现频率,从而猜测关键词的长度和内容。

   字典攻击:

   如果关键词是常见的单词或短语,那么字典攻击就很容易破解密码。

   攻击者可以尝试使用常见的关键词来破解密码。

   密钥管理:

   Vigenère 密码需要在发送方和接收方之间共享关键词,这增加了密钥管理的复杂性。

如果关键词被泄露,整个加密系统就会被破坏。

3. 对 Vigenere 密码的分析方法主要思想?

   频率分析:

   分析密文中字母的出现频率,并与自然语言的字母频率进行比较。

   这可以帮助确定关键词的长度,并为猜测关键词提供线索。

   周期性分析:

   寻找密文中的周期性模式,这可能暴露关键词的长度。

   可以使用自相关函数或其他统计方法来检测密文的周期性。

   字典攻击:

   使用常见的单词或短语作为猜测的关键词,尝试破解密码。

   这种方法对于使用简单关键词的 Vigenère 密码很有效。

   暴力破解:

   穷举所有可能的关键词,直到找到正确的关键词。

   这种方法对于长关键词的 Vigenère 密码很耗时,但可以保证找到正确的关键词。

4. 对 Vigenere 密码的有效改进方法是什么?

   使用更长和更复杂的关键词:

   关键词的长度和复杂度是 Vigenère 密码安全性的关键因素。

   使用长度更长、包含数字和特殊字符的关键词可以大大提高密码的强度。

   使用动态密钥:

   在加密过程中,可以使用动态变化的密钥,而不是固定的关键词。

   这可以进一步增加密码的复杂度,降低频率分析和周期性分析的效果。

   结合其他加密技术:

   可以将 Vigenère 密码与其他加密技术(如流密码或块密码)结合使用。

   这种混合加密方法可以提高整体的安全性,降低单一加密方法的缺陷。

   使用随机密钥生成器:

   使用随机密钥生成器来生成关键词,而不是使用人工选择的关键词。

   这可以确保关键词的随机性和复杂性,提高密码的安全性。

5. Column permutation cipher 的原理是什么?

   将明文消息划分为等长的块,每个块包含 n 个字符。

   将每个块排列成 n 列,每列包含一个字符。

   根据事先约定的列置换顺序,对这些列进行重新排列。

   将重新排列的列依次读出,形成密文。

6. 给定关键字为“experiment”,加密矩阵将包括几列,以及列置换的次序是什么?

7. Column permutation cipher 的安全性增强方法是什么?

   使用双重列置换:

   在基本的 Column Permutation Cipher 中,增加一个额外的列置换步骤。

   即先按照一个置换顺序对列进行重排,然后再按照另一个置换顺序进行重排。

   这样可以大大增加密码的复杂度和抗攻击能力。

   结合其他加密方法:

   将 Column Permutation Cipher 与其他加密算法如 Substitution Cipher、Transposition Cipher 等结合使用。

   先使用 Column Permutation Cipher 对明文进行预处理,然后再应用其他加密算法。

这样可以充分利用不同加密方法的优势,提高整体的安全性。

使用可变的列置换顺序:

不使用固定的列置换顺序,而是采用可变的置换顺序。

可以根据明文、密钥或其他因素动态生成置换顺序,增加破解的难度。

增加矩阵的维度:

不仅对列进行置换,还可以对行进行置换。

即将明文划分为 m×n 的矩阵,对行和列都进行置换操作。

这样可以大大增加密码空间,提高安全性。

八、 结束语

通过本次学习，我学会了 Vigenere 密码和 Column permutation 密码的基本原理和编程实现方法。对于 Vigenere 密码，我了解了如何使用关键字对明文进行加密和解密；对于 Column permutation 密码，我学会了如何按照特定顺序对明文进行列重新排列以生成密文，并且如何根据密钥进行解密操作。

在学习过程中，我深刻认识到密码学在信息安全领域中的重要性，了解了密码学的基本原理和常见算法。这让我对信息安全有了更深入的了解，也为我今后从事信息安全工作打下了坚实的基础。

我对本课程的内容和安排非常满意，觉得课程内容丰富，实用性强，让我收获颇丰，对密码学有了更深入的了解。希望未来能够增加更多实际案例的分析和实践操作，让学习更加生动有趣

九、 参考文献

1.Richard J.Spillman ：《 CLASSICAL AND CONTEMPORARY CRYPTOLOGY》，清华大学出版社，2005-7.

2.William Stallings. Cryptography and Network Security: Principles and Practice. 8th ed. Prentice Hall, 2020.

3. Ling Dong, Kefei Chen. Cryptographic protocol. Security analysis based on trusted freshness. 高等教育出版社，2011-10.

4. 胡向东.《应用密码学：第 4 版》， 电子工业出版社，2019-6.

实验成绩

考查内容	分数	得分
做好实验内容的预习，写出预习报告	10	
了解实验题目的调试方法	10	
按实验要求预先设计好程序	10	
认真记录实验数据并分析实验结果	10	
实验后按要求书写实验报告，记录实验用数据及运行结果	30	
创新能力强，在实验中设计的程序有一定的通用性，算法优化	20	
实验过程中，具有严谨的学习态度，认真、踏实、一丝不苟的科学作风	10	