网络空间安全学院（密码学院）

2023-2024 学年度第 1 学期

J00715 网络安全与系统设计 课程论文

论文题目：数字签名算法的编程实现

学生学号：20213006839

学生姓名：甄五四

学生班级：信息安全(密码学方向)理科实验班

提交时间：2024 年 1 月 20 日

授课教师：王泽群

教师评阅：

成绩：_____

教师签名：_____

目录

1 前言 4

1.1 研究背景 4

1.2 预期结果和意义 4

2 RSA 算法和 RSA 数字签名算法的基本概念和原理 5

2.1 RSA 算法的基本概念和原理 5

2.1.1 RSA 算法介绍 5

2.1.2 RSA 算法的实现原理 5

2.2 MD5 算法的介绍 6

2.3 RSA 数字签名基本概念和 RSA 数字签名算法的实现原理 7

2.3.1 RSA 数字签名基本概念 7

2.3.2 RSA 数字签名算法的实现原理 7

3 RSA 数字签名的设计与实现 9

3.1 RSA 数字签名的总体设计 9

3.1.1 RSA 数字签名所需实现的功能 9

3.1.2 RSA 数字签名软件的总体要求和设计 9

3.2 RSA 数字签名系统的设计实现 10

3.2.1 密钥产生的实现 10

3.2.2 数字签名的设计实现 11

3.2.3 验证数字签名的设计与实现 11

4 测试 12

4.1 测试内容 12

4.2 测试步骤 12

4.3 测试结果 15

结论 15

致 谢 15

参 考 文 献 16

附录 A 17

附录 B 19

数字签名算法的编程实现

摘要： 随着计算机网络的普及，网络安全问题逐渐突显，威胁着人们的生活和国家安全。网络安全的实质是信息安全，信息安全技术的核心之一是数字签名技术。数字签名作为公钥密码体制中的一种重要算法,能够提供消息的身份验证、消息的完整性和电子文件的不可否认性,在商业和金融行业等许多领域有着广泛的应用。数字签名可以解决否认、伪造、算改、冒充等问题。使用数字签名技术使得发送者发送的时候不能否认发送的报文签名，接受者不能伪造发送者的报文签名，即网络中的某一用户不能冒充另一用户。实现数字签名的算法也有很多，如 ELGamal 数字签名算法、基于 SM2 的数字签名等。RSA 算法是目前公认的在理论和实际应用中最为成熟和完善的一种公钥密码体制，它是第一个既能用于数据加密也能用于数字签名的算法，是公钥密码体制的代表，本文基于 RSA 公开密码算法实现数字签名。

关键词：RSA 算法，MD5 算法，RSA 数字签名，验证

Implementation of Digital Signature Algorithm

Abstract: With the popularization of computer networks, network security issues have gradually become prominent, threatening people's lives and national security. The essence of network security is information security, and one of the core technologies of information security is digital signature technology. As an important algorithm in public-key cryptography, digital signatures provide message authentication, message integrity, and non-repudiation of electronic documents. They have widespread applications in various fields, including business and finance. Digital signatures can address issues such as denial, forgery, tampering, and impersonation. The use of digital signature technology ensures that the sender cannot deny the signed message at the time of transmission, and the receiver cannot forge the sender's message signature. In other words, a user in the network cannot impersonate another user. There are many algorithms for implementing digital signatures, such as the ElGamal digital signature algorithm and the SM2-based digital signature. The RSA algorithm is currently recognized as the most mature and complete public-key cryptosystem in both theory and practical applications. It is the first algorithm that can be used for both data encryption and digital signatures, representing the public-key cryptosystem. This paper implements digital signatures based on the RSA public-key algorithm.

Keywords: RSA algorithm, MD5 algorithm, RSA digital signature, verification

1 前言

1.1 研究背景

随着数字化时代的到来，电子文件的广泛使用对信息安全提出了严峻的挑战。在电子政务、电子商务等领域，人们频繁进行数字文件的传输和交互，因此确保数据的完整性、真实性以及发送者身份的可验证性成为亟需解决的问题。传统手写签名在数字环境中的应用受到限制，数字签名作为一种数字化的替代方案应运而生。RSA 数字签名算法作为公钥密码学领域的重要代表，在保障信息安全和确保数字通信的可信性方面发挥着关键作用。该算法基于大数因子分解的困难性，为数字签名提供了一种安全可靠的数学基础。在网络通信的普及和数字社会的发展背景下，RSA 数字签名算法凭借其独特的数学原理、广泛的应用性以及公认的安全性逐渐成为研究和实践中的首选方案。

1.2 预期结果和意义

数字签名技术满足网络安全的目标即身份真实性、信息机密性、信息完整性、服务可用性、不可否认性、系统可控性、系统易用性、可审查性等等。特别是其身份鉴别、数据完整性和不可抵赖性在电子商务、电子政务等应用领域中有很重要的作用。目前关于数字签名的研究主要集中点是基于公钥密码体制的数字签名。在公钥密码体制中，解密和加密密钥不同，解密和加密可分离，通信双方无须事先交换密钥就可建立起保密通信，因此它较好地解决了传统密码体制在网络通信中出现的问题。手写签名的每一项业务都是数字签名的潜在用场。当需要对某一实体进行认证、传输具有有效性的密钥以及进行密钥分配时，便可以借助数字签名来完成任务。数字签名技术在身份识别和认证、数据完整性、不可否认等方面具有其它技术无法替代的作用。而在公钥体制中，RSA 是一个较为完善的公钥密码算法，不仅能够同时用于加密和数字签名，而且易于理解和操作，是被广泛研究的公钥密码算法。因此，基于 RSA 的数字签名具有较强的研究性和实际应用意义。通过 MD5 算法对明文消息计算哈希值得到数字摘要，之后使用 RSA 的私钥 e 和 n 对数字摘要加密得到数字签名。实际中我们会把数字签名附着在消息后面发送给对方，如果对方使用公钥 d 解密后得到的数字摘要与 MD5 再次计算明文消息得到的数字摘要相同，则验证成功。为了提高安全性，可以对明文消息加密后再计算哈希摘

要，密钥也需要进行安全交换。为了突出数字签名算法的实现，本课设简化了部分具体实现。

2 RSA 算法和 RSA 数字签名算法的基本概念和原理

2.1 RSA 算法的基本概念和原理

2.1.1 RSA 算法介绍

RSA 算法是一种公钥密码算法，由 Rivest、Shamir 和 Adleman 三位专家于 1977 年提出，并在 1978 年首次发表。这一算法具有独特之处，因为它是第一个既能用于加密又能用于数字签名的算法，并且易于理解和操作。RSA 的安全性基于大数分解问题的困难性，但目前尚未有理论证明破解 RSA 是否等同于大数分解。尽管 RSA 有一些缺点，如密钥生成复杂、安全性依赖于大数分解等问题，但它已经得到了广泛的应用。在实际应用中，RSA 在硬件和软件两个方面都取得了显著的成果。在硬件方面，技术成熟的集成电路已广泛应用于各种消费类电子产品，为 RSA 提供了强大的硬件支持。而在软件方面，RSA 主要应用于 Internet 上，特别是在加密连接、数字签名和数字证书的核心算法中，RSA 发挥着关键作用。著名的开源工具包 OpenSSL 等广泛使用 RSA 实现签名和密钥交换。RSA 在网络安全中也发挥了重要作用，解决了大规模网络应用中密钥的分发和管理问题。采用公开密钥密码体制，RSA 能够实现安全的密钥交换，提高了网络通信的安全性。此外，RSA 的数字签名机制也为网络中表征个体或机构的真实性提供了有效手段。其数据具有惟一性和私有性，这使得 RSA 在数字签名服务方面具有独特的优势。尽管 RSA 存在一些缺点，如产生密钥复杂、安全性依赖于大数分解等方面的问题，但作为最早并且最重要的公开密钥算法之一，RSA 在各领域的应用不胜枚举。在电子安全领域形成了完备的国际规范，而 RSA 作为其核心算法在现代密码学中持续发挥着不可替代的作用。

2.1.2 RSA 算法的实现原理

1) 密钥生成

1. 随机选取两个素数，作为 p 和 q.

2. 计算$n = p * q, \varphi(n) = (p-1) * (q-1)$

3. 随机选取 e,使 e 与$\varphi(n)$互质

4. 利用扩展欧几里得算法，计算 d 使$d * e = 1 mod(n)$

5. 得到公钥(e, n)和私钥(d, n)

2) RSA 加密算法

1. 将明文消息 M 转换为整数 m

2. 计算密文 C：$C = m^e mod(n)$其中 e 是公钥中的指数，n 是公钥中的模

3. C 是加密后的消息

3) RSA 解密算法

1. 计算明文 m：$m = C^d mod(n)$其中 d 是私钥中的指数，n 是私钥中的模

2. m 是解密后的消息

2.2 MD5 算法的介绍

MD5（Message-Digest Algorithm 5）是一种广泛使用的密码学哈希函数，用于生成数据的消息摘要，以确保数据的完整性和真实性。MD5 算法是由 Ronald Rivest 于 1991 年设计并发布的，是 MD 系列算法中的一种。MD5 算法对任意长度的信息进行压缩，将其转换成固定比特数的格式，从而显著提高了信息存储的效率。其设计理念类似于 SP 网络中的扩散作用，作为信息摘要算法，MD5 可视为一种压缩算法，由于原信息长度为任意长，最终摘要的比特长度通常小于原信息长度。哈希计算具有不可逆性的特点使得即使知道摘要信息，要确定原文也是极其困难的。MD5 算法的填充方式以及得到的哈希值长度均为 128 比特，然而，由于一些设计上的缺陷，其产生冲突的概率并没有理想中那么优越。为了提升安全性，在 MD4 算法的基础上，MD5 进行了一轮迭代的增加，形成四轮的结构。尽管在近十几年的应用中，MD5 算法相对来说是相对安全的。hashlib 模块为 Python 提供了一种简便的方式来使用各种哈希算法，如 MD5、SHA-1、SHA-256 等，本设计使用 hashlib 中的 md5 来计算哈希值。

2.3 RSA 数字签名基本概念和 RSA 数字签名算法的实现原理

2.3.1 RSA 数字签名基本概念

RSA 数字签名体制是基于 RSA 算法的一种数字签名技术，旨在确保数字数据的完整性、真实性和不可否认性。RSA 算法在实践中已被证明具有较高的安全性，因此在众多安全标准中得到广泛应用，包括 ISO/IEC 9796、ANSI X9.30-199X 以及美国联邦信息处理标准 FIPS 186-2 等，将 RSA 作为推荐的数字签名标准算法之一。

RSA 数字签名算法包含签名算法和验证签名算法。该算法利用 RSA 算法的加密和解密原理，通过哈希函数（在本设计中采用 MD5 算法）生成消息摘要 MD，并将其用作所需加密的对象。数字签名的特点在于代表了消息的特征，一旦消息发生改变，数字签名的值也将随之改变，不同的消息将得到不同的数字签名。通过这种方式，接收方能够确保消息确实来自发送方，并得到数字签名的真实性保证。

在实践中，数字签名通过认证技术来辨认真伪，包括数字签名认证、身份认证和公开密钥证明等机制。数字签名认证提供对数字签名的鉴别方法，身份认证提供辨别和确认通信双方真实身份的方式，而公开密钥证明机制则用于验证密钥。在网络时代，数字签名的应用使人们能够确定通信对象的身份，验证文件是否被篡改，成为网络安全中不可或缺的一部分。

数字签名类似于手写签名，具有可验证签名产生者身份、证实被签消息内容以及由第三方验证等特性。为了实现数字签名的以上性质，必须满足签名可信、不可伪造、不可复制、不可改变和不可抵赖等要求。数字签名的原理在于通过某种算法对原始消息进行运算，生成消息摘要，然后用私钥对摘要进行加密，形成数字签名。接收方通过使用公钥解密数字签名，再通过相同的算法计算消息摘要，从而验证数字签名的真实性。这一过程确保了消息的完整性和发送方身份的真实性。数字签名必须满足三个性质:1)接受者能够核实并确认发送者对信息的签名，但不能伪造签名。2)发送者事后不能否认和抵赖对信息的签名。3)当双方关于签名的真伪发生争执时，能找到一个公证方做出仲裁，但公证方不能伪造这一过程

2.3.2 RSA 数字签名算法的实现原理

对于公钥密码体制，其中的加密变换与解密变换必须是可交换的互逆变换，才

能应用于数字签名。在本系统中选择的 RSA 算法满足这一要求。数字签名的生成过程如下：1)使用 MD5 算法对待签名的原始文件进行运算，得到一个唯一的报文摘要（Message Digest）。不同的原始文件将产生不同的摘要，但对于相同的原始文件，摘要是唯一的。2)发送方使用自己的 RSA 私钥对生成的报文摘要进行加密，形成发送方的数字签名。3)数字签名作为附件与原始文件一同发送给接收方。4)接收方首先对接收到的原始文件使用相同的算法计算新的报文摘要。接着，使用发送方的 RSA 公钥对数字签名进行解密。最后，比较两个报文摘要。如果值相同，接收方确认该数字签名确实是由发送方生成的。

RSA 用于数字签名系统的实现：

1) 签名算法：

(1) 消息摘要计算： 在签名之前，原始消息首先经过 MD5 计算，生成一个 128 位的消息摘要。MD5 函数是一种单向散列函数，能够将任意长度的消息压缩成固定长度的摘要。MD5 具有单向性和抗碰撞性，使其适用于信息完整性检验。该算法的设计不依赖于任何假设或密码体制，且执行速度较快，因此被广泛认可。

(2) RSA 加密： 使用签名者的私钥对生成的消息摘要进行 RSA 加密，得到加密后的字符串，即数字签名。

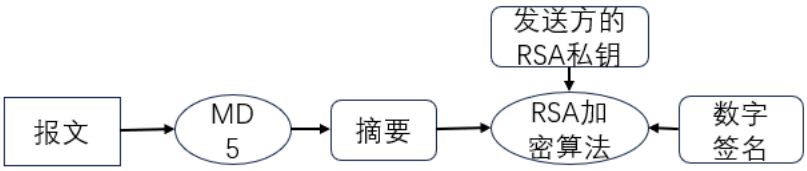

图 1 RSA 数字签名过程

2) 验证签名算法：

(1) RSA 解密： 数字签名实际上是加密的消息摘要，通过 RSA 解密方法，使用签名者的公钥对加密的消息摘要进行解密。解密的结果应为 128 位的消息摘要。

(2) 消息摘要计算和比较： 验证者对原始消息使用 MD5 算法重新计算，得到验证者自己计算的消息摘要。接着，验证者比较解密得到的消息摘要和自己计算的消息摘要。如果两者相同，验证成功，确认消息的完整性，并确认签名确实为签名者的。否则，验证失败，表明签名可能被冒充或者消息被篡改。

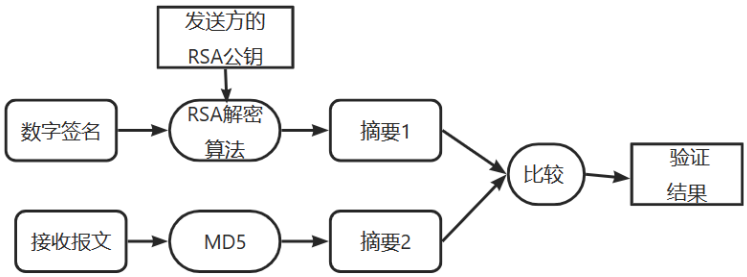

图 2　RSA 数字签名验证过程

3　RSA 数字签名的设计与实现

3.1 RSA 数字签名的总体设计

3.1.1　RSA 数字签名所需实现的功能

在本软件中，需要实现以下功能：

1) 生成 RSA 密钥：生成一对 RSA 密钥，其中包括公钥$ke = (e, n)$和私钥$kd = (d, n)$。

2) 利用 MD5 算法计算消息摘要 MD：使用 MD5 算法对给定消息进行摘要计算，生成消息摘要 MD。

3) 数字签名的实现： 利用私钥 d 对消息摘要 MD 进行加密计算，采用 RSA 算法中的加密方法生成数字签名。

4) 验证数字签名： 使用公钥 e 对数字签名进行解密计算，采用 RSA 算法中的解密方法。得到解密结果后，与步骤 2)中计算得到的消息摘要 MD 进行比较。如果两个消息摘要一致，则判定签名成功。

3.1.2　RSA 数字签名软件的总体要求和设计

本软件的总体要求包括以下几点：

1) 按照要求生成一对非对称密钥，包括公钥和私钥。

2) 输入的任意消息字符串（明文信息）或者文件，利用 MD5 算法生成相应的消息摘要 MD。

3) 利用生成的私钥 e，按照 RSA 算法的加密原理对消息摘要 MD 进行加密运算，生成数字签名。

4) 利用生成的公钥 d，按照 RSA 算法的解密原理对加密的消息摘要（数字签名）进行解密运算，得到对应的消息摘要。然后，将得到的消息摘要与生成的消息摘要 MD 进行比较，以验证数字签名者的身份真实性。

5) 确保软件具有友好的用户交互，提供清晰完整的提示信息，使操作过程舒适且图形界面设计美观。

整体设计基于 python 开发环境，采用 PyCharm Community Edition 2023.2.2 的运行环境，以保障软件的高效性和稳定性。

3.2 RSA 数字签名系统的设计实现

3.2.1 密钥产生的实现

通过调用 sympy.randprime 随机生成两个素整数 p，q，本设计默认两个素数为 512 比特，可以计算$n = p * q$，通过欧拉函数计算$\varphi(n) = (p-1) * (q-1)$。再随机生成一个素数 e，由于公钥密码系统安全性不依赖于公钥，以及为了加快密钥生成速度、数字签名速度，所以默认素数 e 为 16 比特。e 需要与$\varphi(n)$互质才可以使用，可以使用欧几里得算法求最大公约数是否为 1 判断互质，本系统直接调用 sympy.gcd 来判断公约数是否为 1。再通过扩展欧几里得算法求得 d，即$e * d = 1 mod\ (n)$。则得到私钥$kd = (d, n)$和公钥$ke = (e, n)$。

部分关键代码如下：

1) 随机生成素整数

```
def generate_prime(bits=512):
    """生成一个指定位数的素数,默认 512 比特"""
    return sympy.randprime(2**(bits-1), 2**bits)
```

2) 生成公钥，计算 n

```
def generatePublicKey():
    p = generate_prime()
    q = generate_prime()
    n = p * q
    phi_n = (p-1)*(q-1)
    e = generate_prime(16)
    while sympy.gcd(e, phi_n) != 1:
        e = generate_prime(16)
    return e, n, phi_n
```

3) 扩展欧几里得求私钥 d

```
def generatePrivateKey(a, p, b=1):
        k = 1
        r = gcd(a, p)
        if r != 1:
            p = p // r
        while True:
            if (b + p * k) % a == 0:
                return (b + p * k) // a
            k += 1
```

3.2.2 数字签名的设计实现

首先 from hashlib import md5，从 hashlib 中导入 md5 函数，用于计算消息摘要。对明文信息使用 md5 计算消息摘要并转为 16 进制。调用 sign 函数对消息摘要进行数字签名，通过$Sig(m) \equiv m^d \equiv c \ mod \ n$,得到消息摘要 m 的数字签名 c。

部分关键代码如下：

1) 对消息计算消息摘要、签名

```
message = self.mesText.get(1.0, END).rstrip()
hashMes = md5(message.encode('utf-8')).hexdigest()
signature = sign(hashMes, self.privateKeyText.get(1.0, END).rstrip(),
                 self.pubKeySecondPartText.get(1.0, END).rstrip())
self.signatureText.insert(1.0, hex(signature)[2:])
```

2) 签名函数 sign

```
def sign(hash, privKey, secondPartPubKey):
        hashInt = int(hash, 16)
        privKey = int(privKey, 16)
        secondPartPubKey = int(secondPartPubKey, 16)
        signature = pow(hashInt, privKey, secondPartPubKey)
        return signature
```

3.2.3 验证数字签名的设计与实现

首先需要对消息通过 md5 计算消息摘要，并转为 10 进制。对已经签名的消息调用 check 函数解密数字签名得到另一个消息摘要，与 md5 计算得到的消息摘要比较，如果相等则验证通过。即对于$Ver(m, c)$,如果$m \equiv c^e \ mod \ n$，则验证成功。

部分关键代码如下：

1) 对消息验证

```
hashMes = md5(mesText.encode('utf-8')).hexdigest()
hashInt = int(hashMes, 16)
```

```
calculatedHash = check(signatureText, pubKeyFirstText, pubkeySecondText)
if calculatedHash == hashInt:
    showinfo("信息", "签名有效！", parent=self)
else:
    showinfo("信息", "签名无效！", parent=self)
```

2) 验证函数 check

```
def check(signature, firstPartPubKey, secondPartPubKey):
        signature = int(signature, 16)
        firstPartPubKey = int(firstPartPubKey, 16)
        secondPartPubKey = int(secondPartPubKey, 16)
        hash = pow(signature, firstPartPubKey, secondPartPubKey)
        return hash
```

4 测试

4.1 测试内容

测试系统的签名结果和验证结果，通过生成公私钥，私钥对文本的消息摘要进行数字签名，然后再用对应公钥验证数字签名是否有效，来判定测试结果成功还是失败。还可以通过伪造消息摘要、使用不对应的公私钥等测试数字签名是否有效。

4.2 测试步骤

步骤一：生成密钥(n，e，d)

打开 RSA 数字签名系统-->点击生成密钥按钮-->进入生成密钥页面-->点击生成密钥对按钮生成公钥、私钥-->点击保存私钥、保存公钥分别保存

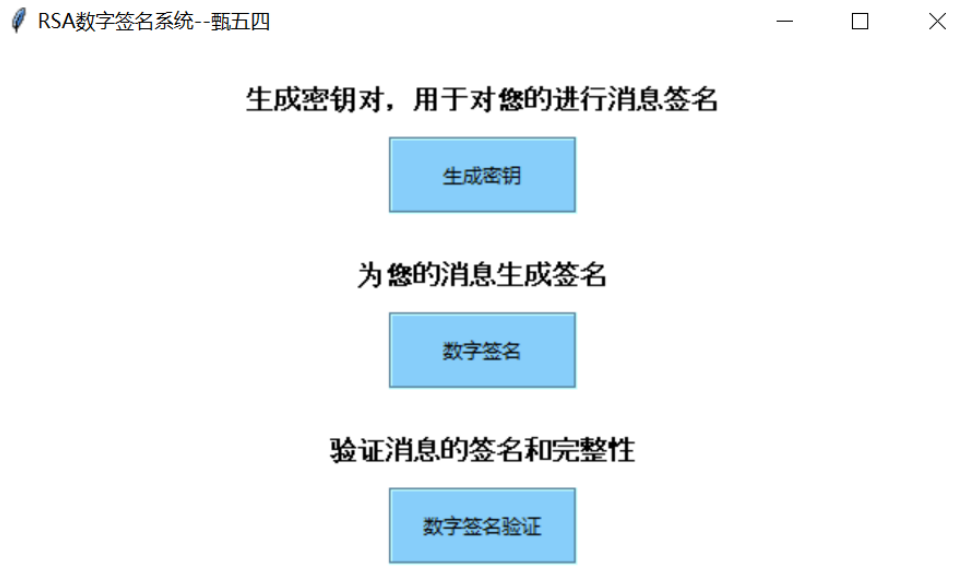

图 3　RSA 数字签名系统主页面

图 4 生成 RSA 密钥对

步骤二：签名模块进行签名操作

RSA 数字签名系统主页面-->点击数字签名按钮-->进入签名页面-->点击需要数字签名的文件(也可以直接输入消息)-->点击加载刚才生成的私钥文件-->点击签名得到数字签名，点击保存可以保存数字签名文件

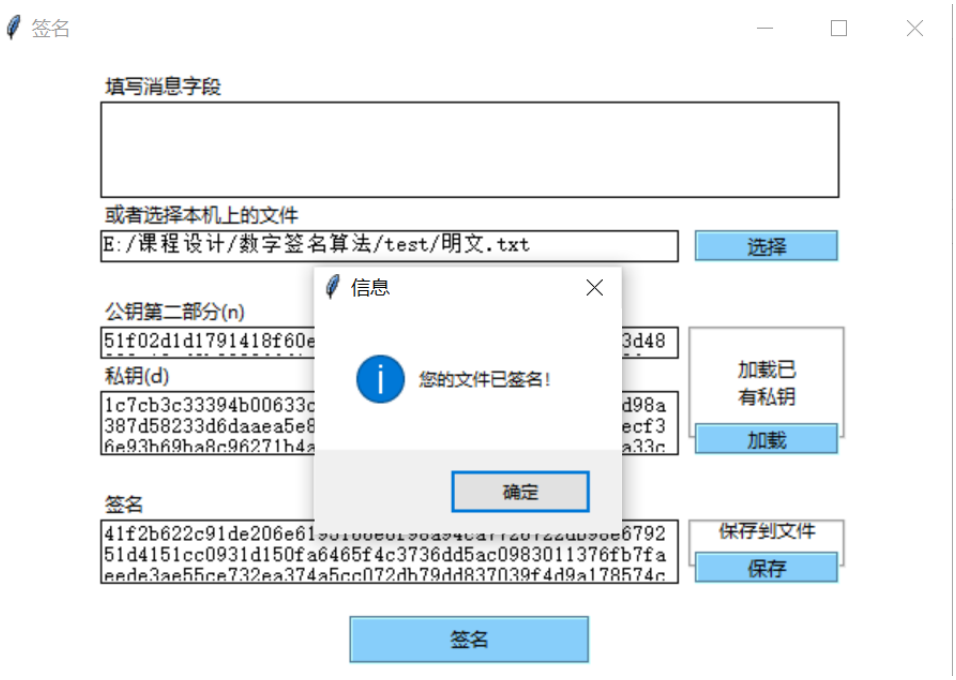

图 5 签名模块进行签名操作

步骤三：验证模块进行验证操作

RSA 数字签名系统主页面-->点击数字签名验证按钮-->进入验证签名页面-->点击选择按钮选择明文文本-->点击加载按钮加载公钥文件-->点击签名处的加载按钮加载步骤二得到的数字签名文件-->点击验证、验证签名是否有效

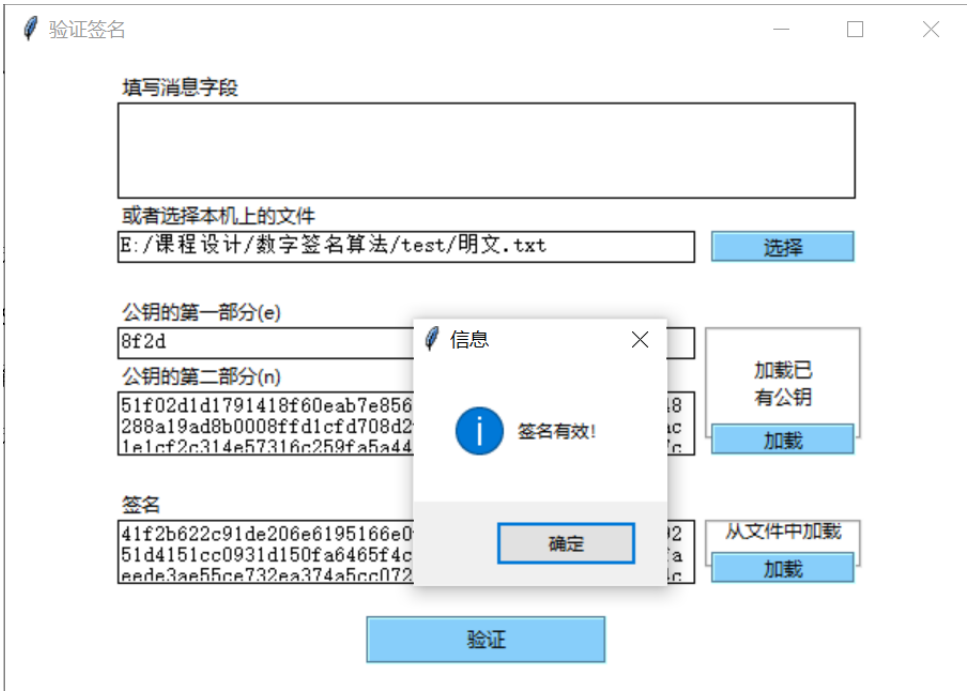

图 6 验证模块进行验证操作

4.3 测试结果

数字签名结果验证有效，测试成功！验证阶段如果选择其他公钥、明文和数字签名文本不对应等都会显示签名无效，再次验证数字签名算法安全性。

结论

数字签名作为现代密码学中的一个主要研究领域，在信息安全、身份认证、数据完整性、不可否认性以及匿名性等方面具有广泛的应用。特别是在大型网络安全和电子商务系统中，数字签名占据着重要的地位，已成为计算机网络中不可或缺的安全措施之一，是确保数据可靠性和实现认证的重要工具。本课设设计的数字签名系统采用 RSA 数字签名方案，利用 RSA 算法作为非对称加密方法，通过对使用 MD5 算法生成的报文摘要进行加密得到数字签名，测试结果表明数字签名系统设计可行。

致　谢

首先，我要衷心感谢王泽群老师对我课程设计论文的耐心和专业指导，以及对于数字证书、消息摘要等内容的讲解，还要许多网络安全学院的老师关于网络安全的课程讲解。在他们的指导下，我顺利完成了这篇课程设计论文。虽然我深知一篇课程设计论文不能完全代表我在网络安全方面的水平，但我将以老师们为榜样，不断学习、勤奋钻研，在网络安全领域不断进取，将理论知识转化为实际应用，成为该领域的有力人才，以回报老师的悉心教诲。

其次，我要感激网络上众多博主们。他们在网络安全领域的付出，留下了许多珍贵的学习资料，分享许多关于数字签名、RSA、MD5 等细致的分析、讲解。给我的课程设计带来了许多帮助，助力我完成这次课程设计。

最后，我要由衷地感谢我的家人。是他们多年来对我的坚定支持，让我得以走到今天，成功完成课程设计。他们的支持是我不断努力的强大动力，也是我前行路上的坚实后盾。我将怀着对家人的深深感激之情，迎接未来的挑战。

参 考 文 献

[1]顾婷婷,李涛,尹鹏,等.RSA 和 RSA 数字签名的实现[J].网络安全技术与应用,2001(7):34-36.

[2]黄硕.RSA 数字签名算法在软件加密中的应用[J].网络安全技术与应用,2018(6):37-37,52.

[3]谭毓安,王佐,曹元大.RSA 数字签名算法在软件加密中的应用[J].计算机系统应用,2004(8):33-35.

[4]弋改珍.RSA算法的研究与实现[J].现代计算机(专业版)，2018（30)：12-14，30.

[5]司红伟,汤彬.RSA 应用现状及其在文件加密中的应用[J].电脑与电信,2009(6):76-77,80.

[6]王静文,吴晓艺,袁康霖.基于 3DES_RSA 算法数字签名的设计与实现[J].福建电脑,2010(5):92-93.

[7]张建伟,李鑫,张梅峰.基于 MD5 算法的身份鉴别技术的研究与实现[J].计算机工程,2003(4):118-119,145.

[8]白永祥, 何林, 陈逸怀. 基于 Python 的 RSA 密码算法的设计与实现[J]. 电子设计工程, 2021.

[9]叶萍.基于RSA的数字签名算法的设计实现[J].科技信息,2009(30):-I0021.

[10]耿媛媛. 几类特殊的数字签名研究[D]. 黑龙江:哈尔滨师范大学,2014. DOI:10.7666/d.D678524.

[11]余勇.浅析 RSA 数字签名算法的软件加密应用[J].太原城市职业技术学院学报,2016(8):187-188.

[12]司应硕,杨文涛,张森.一种基于MD5与RSA算法的数字签名系统设计与实现[J].新乡学院学报,2011(1):44-46.

[13]顾婷婷,李涛,尹鹏,等.RSA 和 RSA 数字签名的实现[J].网络安全技术与应用,2001(7):34-36.

附录 A

为了验证本课程设计的数字签名系统的安全性，做了一些伪造消息摘要的测试。具体来说，现在有明文 1、明文 2 两个明文，现在有一对公钥 1、私钥 1。同时对两个明文进行数字签名得到签名 1、签名 2，如果对于明文 1 和签名 2 验证结果为签名无效则系统安全。之后对于明文 1 使用公钥 2、私钥 2(可生成多个密钥对测试)进行数字签名，看能否得到相同的消息摘要。

图 7 公钥 1 对明文 1 签名得到签名 1

图 8 公钥 1 对明文 2 签名得到签名 2

图 9 明文 1 对应签名 1 验证有效

图 10 明文 1 对应签名 2 验证无效

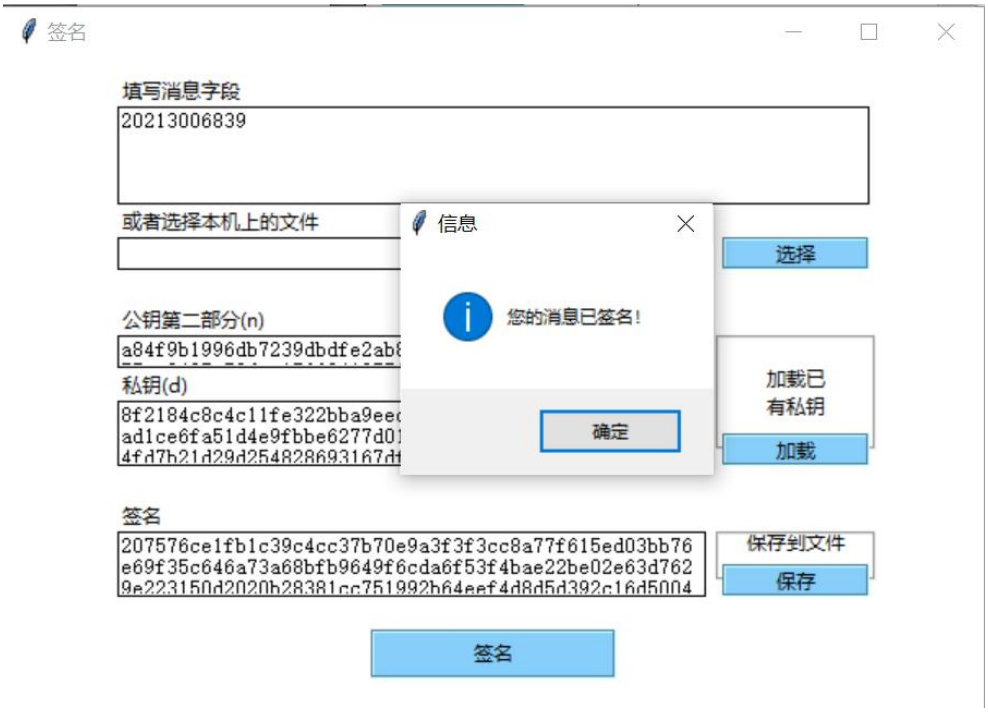

图 12 私钥 2 对明文 1 数字签名得到不同摘要

附录 B

本系统具有友好的交互性，对于用户的错误操作也能正常处理，给出提示。以下给出部分可能出现的异常情况处理效果。

1) 没有填写消息字段或者选择明文文件就直接签名

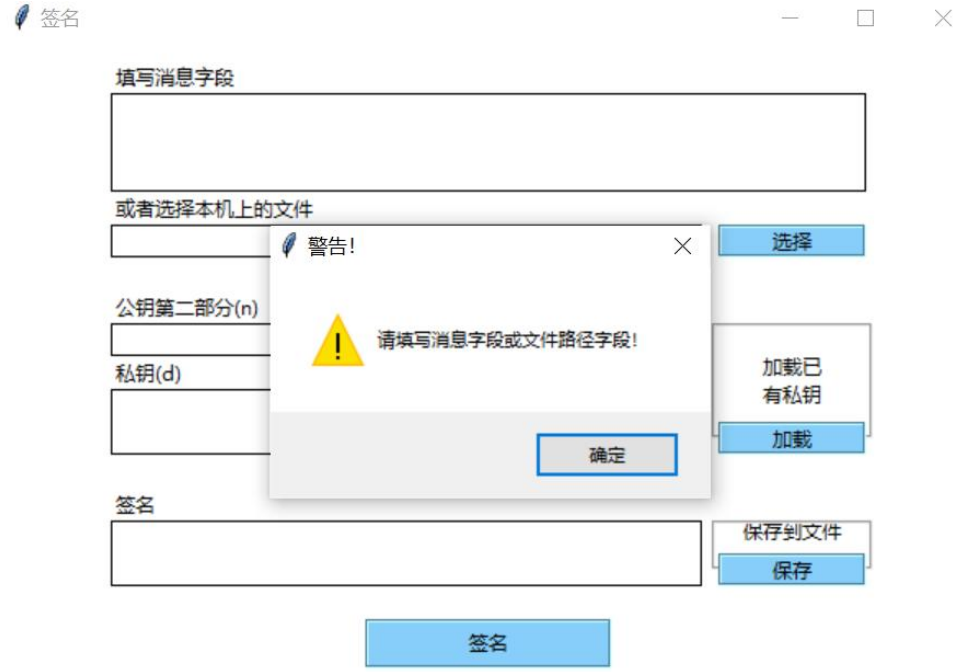

图 13 没有明文签名报错

2) 没有填写私钥就直接签名

图 14 没有密钥签名报错

3) 对于数字签名阶段加载错误文件为私钥、数字签名验证阶段加载了错误公钥文件作为公钥以及错误的签名文件给出错误信息，这主要是通过在密钥生成阶段以及签名阶段保存文件时以在文件中添加额外字符实现，如果是直接复制或输入的公私钥或签名则默认不报错，只是密钥格式(非 16 进制)错误无法得到签名，签名格式错误没有反馈信息。

图 15 加载公钥文件当私钥显示无效

4) 对于选择或者加载了不存在的文件

图 16 加载不存在的文件