****

**本 科 毕 业 设 计**

|  |  |
| --- | --- |
| 课题名称 | 基于国密的医疗分诊系统 |
| 学 院 | 数学与信息科学学院 |
| 专 业 | 信息与计算科学 |
| 班级名称 | 信计182 |
| 学生姓名 | 杨敏铝 |
| 学 号 | 1815200055 |
| 指导教师 | 周权 |
| 完成日期 |  |

教 务 处 制

基于国密的医疗分诊系统

信息与计算科学 信计182 杨敏铝

指导教师:周权

摘要 我们身处于疫情蔓延的时代背景之下，全民对于医疗服务的需求大大增

加，为了缓解医疗体系的负担以及降低医护工作人员的工作压力，本次毕业设计将从分诊的角度出发，使用小程序提供的用户登录后收集用户信息的功能、为用户提供分诊服务的功能、为用户提供直接挂号的功能、为用户提供查询预约记录的功能四大功能来代替人工分诊，从而提高医疗系统的运转效率。除此之外，随着信息技术的发展，个人信息的泄露日益严重，分诊系统通过安全的国密算法sm2和sm9对患者和医生的个人信息进行保护，避免了由于系统的缺漏导致的个人信息泄露。

关键词 国密；分诊；SM2；SM9

**ABSTRACT** We are in the era of epidemic spread, and the demand for medical services has greatly increased. In addition, in order to alleviate the burden of the medical system and reduce the work pressure of medical staff, this graduation project will replace manual triage by using the four functions provided by the applet: the function of collecting user information after user login, the function of providing triage service for users, the function of providing direct registration for users, and the function of querying appointment records for users, So as to improve the operation efficiency of the medical system. In addition, with the development of information technology, the leakage of personal information is becoming more and more serious. The triage system protects the personal information of patients and doctors through secure national secret algorithms SM2 and SM9, avoiding the leakage of personal information caused by the lack of the system.

**KEY WORDS** Commercial Cryptography；triage；SM2；SM9

目录

1. 前言
2. 基础知识

2.1 小程序与数据安全

2.1.1 小程序

2.1.2数据安全

2.2密码应用

2.2.1密码应用及其趋势

2.2.2密码应用的必要性

3. 分诊系统

3.1分诊系统的系统架构

3.1.1前端架构

3.1.2后端架构

3.2分诊系统的部署

3.3分诊系统的功能

3.3.1分诊功能

3.3.2加密功能

3.4分诊系统的流程

4. 系统的设计与具体实现

4.1分诊功能的设计与实现

4.1.1就医人员

4.1.2管理员

4.2加密功能的设计与实现

4.2.1 国密SM2加密算法原理

4.2.2 国密SM9加密算法原理

4.3加密过程

4.3.1国密SM2加密过程

4.3.2国密SM9加密过程

5. 测试

5.1 对小程序基本功能的测试

5.2 对加密效果的测试

6. 结论

7. 参考文献

8. 致谢

前 言

我们处于疫情的时代背景之下，全民抗疫是一个漫长的过程，在这个过程中，医患之间的关系需要着重维护，只有良好的医患关系才能够让全民抗疫顺利进行，为此，本课题利用科技作为桥梁，搭建起医生与患者之间的沟通桥梁，从减轻医疗体系的负担角度出发，实现为就医人员提供智能分诊的服务，从而达到维护医患关系、提升全民抗疫的效率的目的。

随着科技的日益发展和微信的普及，小程序在慢慢地渗透入我们的生活，而轻巧便捷的小程序也提高了大众的生活水平和生活效率。现在的医疗水平也逐渐提升，信息化的医疗服务需要跟上医疗技术的进步，小程序——可以搭建起医患之间沟通的科技桥梁，能够提升医疗服务效率，节省患者时间，使得整体的医疗系统更上一层楼。对于患者而言，线上预约挂号时需要“引导者”，为了避免无效挂号，小程序需要根据患者出现的症状进行合理的分诊[1][2]。

在这个信息爆炸的时代，个人信息泄露严重，对于医疗系统，个人信息的泄露会加重医患关系的矛盾，因此本课题在开发医疗分诊系统小程序的时候，需要使用国密算法对医患双方的个人信息进行加密保护。除了防止不法分子盗取个人隐私，从中谋取利润，还可以防止黄牛倒卖号，破坏看病就医的秩序。

关于大数据隐私保护的方法有很多，而国内外对于数据隐私的保护手段，其实是可从可信硬件和密码学两个方向入手，硬件手段主要是可信执行环境(Trusted Execution Environment)，而密码学手段主要有同态加密(Homomorphic Encryption，HE)、零知识证明(Zero-knowledge Proof，ZKP)以及安全多方计算(Multi-party Computation)。而国密算法在多个领域中都有着很好的应用，例如移动网络支付中可以使用国密算法进行加密[5]，对于银行等要求高数据安全的相关机构有着基于国密算法的数据存储[6][7]，但目前国内外对分诊系统的研究都是着重研究分诊的算法，并未对医患的数据进行有效的保护。

因此，为了简化医院就诊的流程[3][4]、优化资源的配置，提升服务质量，本研究将充分利用现有的医院信息平台，为患者提供预约排队挂号的服务，同时为了避免无效挂号、以及保护医患双方的隐私，本课题将基于国密算法SM2和SM9开发一个小程序版的医疗分诊系统。

基础知识

* 1. 小程序与数据安全

2.1.1小程序

小程序，最大的特点就体现在“小”，它是一种不需要下载安装即可使用的程序，用户只需要简单地扫一扫就可以使用，不需要像传统的应用程序一样安装、登录，传统的应用程序需要占用大量手机的存储空间，从而降低手机性能，而微信小程序依托于拥有超过12亿用户的微信平台，不需要额外下载软件。

从用户角度来说，使用小程序就能够拥有传统应用程序的大部分功能，方便快捷，用户体验感好，尤其是在疫情之下，小程序已经渗透到了我们每个人的身边，不论是做核算还是出示健康码，只需要扫一扫便可以轻松登记，就能够极大地提升整个地区的医疗服务速度。

从技术角度来说，小程序的开发相比传统应用程序，成本更低、开发更简单，能够通过接口去实现大部分传统应用程序的相关功能。

2.1.2数据安全

小程序前端的开发相比传统应用程序更简单，而后端的功能可以依靠小程序前端提供的相关接口来实现，从小程序上获得的相关信息可以通过相关接口调用存储到数据库中，因此小程序的用户数据安全其实和数据库存储安全有关。

OWASP，全称“开放式Web应用程序安全项目”是一个非营利性的组织，该组织再2003年首次出版了“Top10”，也就是10项最严重的web应用程序安全风险列表，而SQL注入常年占据OWASP排行榜前十的位置，SQL注入严重地威胁到数据安全。在信息安全管理的层面，我们应该加强防护，尽量避免SQL注入或其他的应用漏洞，但代码是人写的，总是不可避免地会出现纰漏，例如CSDN的数据库遭到攻击，攻击者爆出CSDN网站用户数据库采用的是明文存储，这严重地威胁到了用户的数据安全，因此，除了加强信息安全层面的管理，还应该使用正确、有效、合规的密码技术对数据库中的敏感数据进行加密存储，避免因为未知的隐患导致数据库信息泄露后，以明文的形式暴露在攻击者面前。

* 1. 密码应用

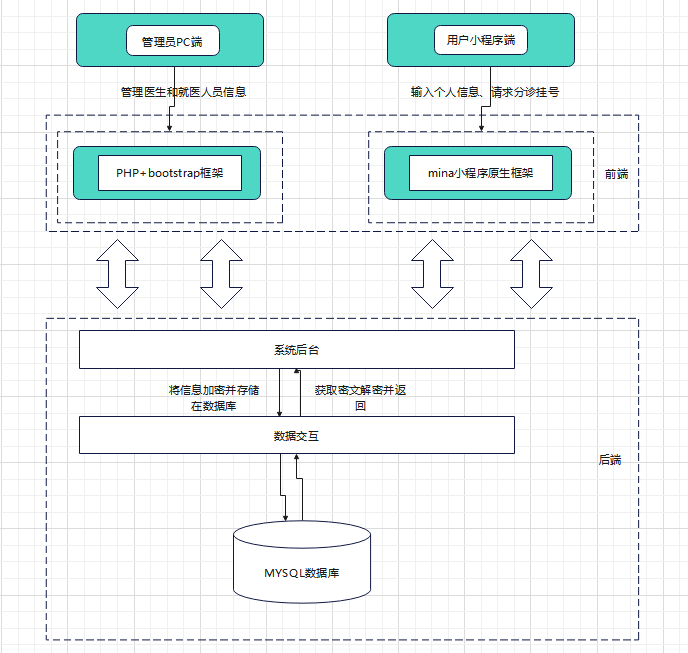
2.2.1密码应用及其趋势

密码在实现数据安全个人隐私保护方面起着至关重要的作用，它是信息系统安全的基础，也是实现个人隐私保护的重要手段之一。密码技术作为网络安全的基础和根本，是各个国家都大力追逐的领域，随着我国密码学专家王小云依次攻破了MD5算法和SHA-1算法，以及国际形式日益严峻，国际密码算法的安全性问题逐渐暴露了出来。没有网络安全就没有国家安全，实现网络安全的基石——密码技术就显得尤为重要，因此，我国密码管理与研究组织积极响应国家要求，自主研发了一套用于数据加密处理系列的国密算法，如SM2、SM3、SM4、SM9、ZUC等，从根本上实现了对密码算法的自主掌控，夯实了网络安全的基础。

随着响应国家的要求，如今实施的商用密码应用安全性评估推动了国产密码的应用，使得我国重要机构逐步实现使用我国自给自足的商用密码，大大增加了网络安全防护。因此，本课题使用了商用密码SM2和SM9进行医疗体系中医患双方的隐私保护，能够极大地提高医疗体系的数据安全。

分诊系统

3.1 分诊系统的系统架构



小程序架构分为前端架构和后端架构，图中所示是系统架构以及前后端数据进行交互的方式。对于管理人员，使用的是pc端的浏览器，通过向系统后台发起请求数据，系统收到请求数据后，调用数据库，获取密文信息，解密后向前端返回请求数据；就医人员使用小程序，输入个人信息后，通过分诊选择对应的科室，向系统后台提交挂号请求，系统收到请求数据后，调用数据库，获取医生相关密文，后台系统解密后向小程序返回医生信息，用户自行选择医生进行挂号，挂号时向系统后台提交挂号请求，系统收到请求数据后，调用数据库，将相应的用户ID和医生ID发送给数据库，建立医患之间的联系。

3.2 分诊系统的部署

3.3 分诊系统的功能

3.3.1分诊功能

为了方便就医人员看病就诊，在网上进行预约排队挂号，本课题设计了一种医疗分诊系统。分诊系统的主要功能分为四大类：用户登录后收集用户信息的功能、为用户提供分诊服务的功能、为用户提供查询预约记录的功能。

该系统在分诊界面可以根据身体的不同部分出现的不同症状进行查询分诊，得到分诊推荐科室后，选择医生进行挂号，从而达到分诊挂号的目的

3.3.2加密功能

信息设计到每个人的财产安全甚至是人身安全，因此保护信息的安全有位重要。本课题使用国密算法SM2和SM9对不同的信息进行加密，以保证就医人员信息和医生信息为重要目的，确保攻击者无法获取到有效的个人信息，实现对个人信息的有效保护。如今在医院中还存在黄牛倒卖号的现象，这会使得本已挂好号排好队的就医人员收到不公正的对待，破坏看病就医的秩序。除此之外，也存在对就医人员信息盗取的非法行为，不怀好意的人将信息卖给其他人来获利，侵犯了就医人员的隐私，甚至还可能威胁到就医人员的财产安全和人身安全。对于大型医院面对大量就医人员，一个系统是否符合的国家商用密码应用安全性标准，因此，用国密算法进行信息加密可以有效地保护信息的安全，给就医人员带来一定的安全感。

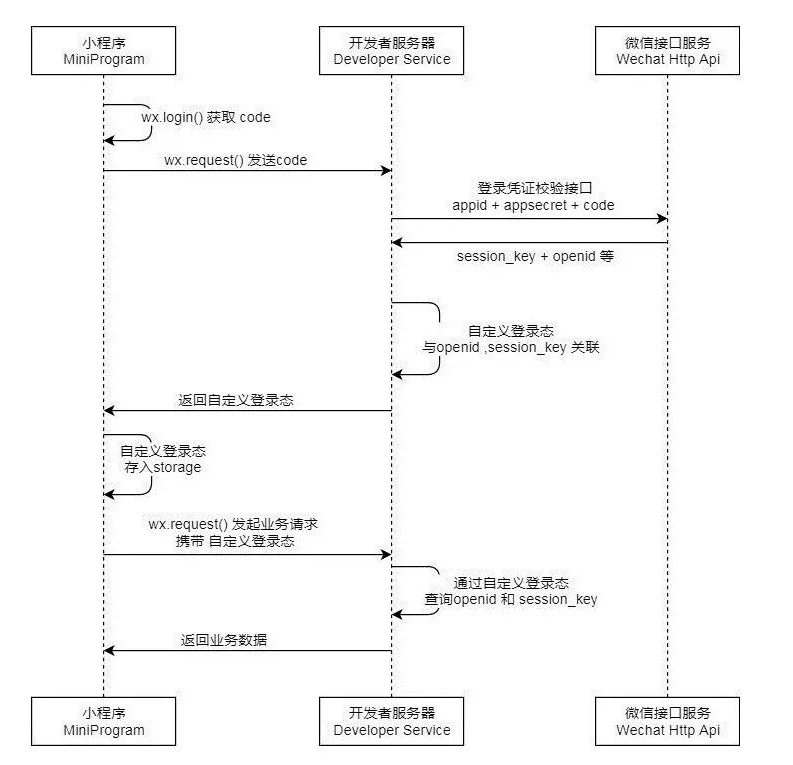
3.4 分诊系统的流程

分诊系统的设计与具体实现

4.1 分诊功能的设计与实现

4.1.1就医人员

本课题首先对就医人员和医生的信息分别选用不同的加密算法进行加密并存储在数据库中，在分诊界面，用户通过输入相应症状，提交到后台后，后台返回建议用户选择的科室，用户请求挂号科室后，后台返回科室对应医生数据，包括姓名、擅长领域等信息，用户选择医生进行挂号后，前端将医生和就医人员的信息返回后台，后台再存入数据库。



4.1.2管理人员

管理人员通过PC端，通过互联网

通过查看微信小程序官方文档，通过login进行登录的流程如图所示，由于测试号无法在微信公众平台中进行管理，因此本课题的数据库中以用户的昵称作为标识（但实际应用中使用openid识别唯一的用户）

图 7

数据库选择的是mysql数据库，使用的管理软件是Navicat for MySQL。总体数据库表如下：



图 8

User（用于存放就医人员的个人信息）：由于SM2算法加密后的密文长度等于明文长度+96B，长度随明文长度而变，因此将加密后的

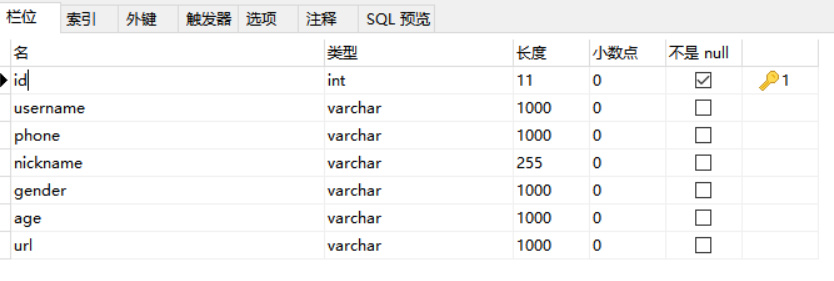


图 9

Department（用于存放科室信息）

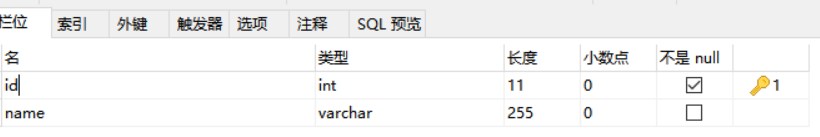


图 10

Doctor（用户存放医生的个人信息）



图 11

Doctor对应外键：



图 12

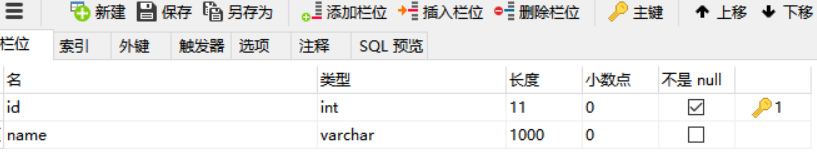


图 13

Line(用于存放医生与就医人员的关系）

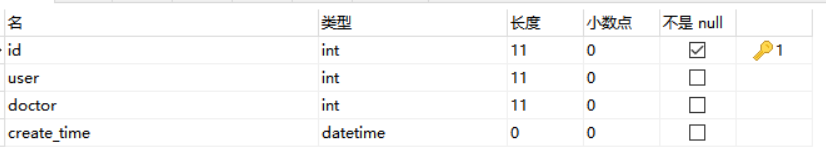


图 14

4.2 加密功能的设计与实现

4.2.1国密SM2加密算法原理

SM2为非对称加密，基于ECC。该算法已公开。由于该算法基于ECC，故其签名速度与秘钥生成速度都快于RSA。ECC 256位（SM2采用的就是ECC 256位的一种）安全强度比RSA 2048位高，但运算速度快于RSA。  
国家密码管理局公布的公钥算法，其加密强度为256位

（1）加密原理

设需要发送的消息为比特串*M*，为 *M*的比特长度。

为了对明文*M*进行加密，作为加密者的用户A应实现以下运算步骤：

A1：用随机数发生器产生随机数；

A2：计算椭圆曲线点，(表示）将的数据类型转换为比特串；

A3：计算椭圆曲线点，若*S* 是无穷远点，则报错并退出；

A4：计算椭圆曲线点，将坐标、的数据类型转换为比特串；

A5：计算，若为全0比特串，则返回 A1；

A6：计算；

A7：计算；

A8：输出密文。

（2）解密原理

设为密文中的比特长度。

为了对密文进行解密，作为解密者的用户B应实现以下运算

步骤：

B1：从*C* 中取出比特串，将的数据类型转换为椭圆曲线上的点，验证是否满足椭圆曲线方程，若不满足则报错并退出；

B2：计算椭圆曲线点，若*S*是无穷远点，则报错并退出；

B3：计算，将坐标、的数据类型转换为比特串；

B4：计算，若为全0比特串，则报错并退出；

B5：从*C* 中取出比特串，计算；

B6：计算，从*C* 中取出比特串，若，则报错 退出；

B7：输出明文。

（3）安全参数设置

随机数 k 和私钥最好大点，以上比较安全。

4.2.2国密SM9加密算法原理

SM9 为标识加密算法（Identity-Based Cryptography），非对称加密，标识加密将用户的标识（如微信号、邮件地址、手机号码、QQ 号等）作为公钥，省略了交换数字证书和公钥过程，使得安全系统变得易于部署和管理，适用于互联网应用的各种新兴应用的安全保障，如基于云技术的密码服务、电子邮件安全、智能终端保护、物联网安全、云存储安全等等。这些安全应用可采用手机号码或邮件地址作为公钥，实现数据加密、身份认证、通话加密、通道加密等。在商用密码体系中，SM9 主要用于用户的身份认证，据新华网公开报道，SM9 的加密强度等同于 3072 位密钥的 RSA 加密算法。

（1）加密算法

设需要发送的消息为比特串*M* ，为 *M* 的比特长度，为分组密码算法中密钥的比特长度， 为函数,Z) 中密钥的比特长度。

为了加密明文*M*给用户B，作为加密者的用户 A 应实现以下运算步骤：

A1：计算群中的元素；

A2：产生随机数；

A3：计算群中的元素，将的数据类型转换为比特串；

A4：计算群中的元素；

A5：计算群*T* 中的元素，按将的数据类型转换为比特串；

A6：按加密明文的方法分类进行计算：

a) 如果加密明文的方法是基于密钥派生函数的序列密码算法，则

1) 计算整数，然后计算。 令为最左边的比特，为剩下的比特，若为全0比特串，则返回 A2；

2) 计算。

b) 如果加密明文的方法是结合密钥派生函数的分组密码算法，则

1) 计算整数，然后计算。 令为最左边的比特，为剩下的比特，若为全 0 比特串，则返回 A2；

2) 计算。

A7：计算；

A8：输出密文。

（2）解密算法

设为密文中的比特长度，为分组密码算法中密钥的比特长度，为函数中密钥的比特长度。

为了对进行解密，作为解密者的用户应实现以下运算步骤：

B1：从*C* 中取出比特串，将的数据类型转换为椭圆曲线上的点，验证是否成立，若不成立则报错并退出；

B2：计算群中的元素，将的数据类型转换为比特串；

B3：按加密明文的方法分类进行计算：

a) 如果加密明文的方法是基于密钥派生函数的序列密码算法，则计算

1) 计算整数，然后计算。 令为最左边的比特，为剩下的比特，若为全0比特串，则报错并退出；

2) 计算。

b) 如果加密明文的方法是结合密钥派生函数的分组密码算法，则计算

1) 计算整数，然后计算。 令为最左边的比特，为剩下的比特，若为全0比特串，则报错并退出；

2) 计算,)。

B4：计算，从*C* 中取出比特串，若不等于，则报错并退出；

B5：输出明文。

4.3加密过程

4.3.1国密SM2加密过程

关键代码：

# sm2的公私钥

SM2\_PRIVATE\_KEY = '00B9AB0B828FF68872F21A837FC303668428DEA11DCD1B24429D0C99E24EED83D5'

SM2\_PUBLIC\_KEY = 'B9C9A6E04E9C91F7BA880429273747D7EF5DDEB0BB2FF6317EB00BEF331A83081A6994B8993F3F5D6EADDDB81872266C87C018FB4162F5AF347B483E24620207'

sm2\_crypt = sm2.CryptSM2(public\_key=SM2\_PUBLIC\_KEY, private\_key=SM2\_PRIVATE\_KEY)

# 加密

def encrypt(info):

    encode\_info = sm2\_crypt.encrypt(info.encode(encoding="utf-8"))

    encode\_info = b64encode(encode\_info).decode()  # 将二进制bytes通过base64编码

    return encode\_info

# 解密

def decrypt(info):

    decode\_info = b64decode(info.encode())  # 通过base64解码成二进制bytes

    decode\_info = sm2\_crypt.decrypt(decode\_info).decode(encoding="utf-8")

    return decode\_info

if \_\_name\_\_ == "\_\_main\_\_":

    action = sys.argv[1]  # 取命令中的加解密动作

    contact\_info = urllib.parse.unquote(sys.argv[2])  # 取命令中需要加解密的内容

    # print(action)

print(contact\_info)

    if action == "encrypt":

        encrypted\_contact\_info = encrypt(contact\_info)

        print(encrypted\_contact\_info)

    if action == "decrypt":

        decrypted\_contact\_info = decrypt(contact\_info)

        print(decrypted\_contact\_info)

4.3.2国密SM9加密过程

关键代码：

#加密

    def encrypt(self, r\_str='0000AAC0541779C8FC45E3E2CB25C12B5D2576B2129AE8BB5EE2CBE5EC9E785C', msg\_bytes='4368696E 65736520 49424520 7374616E 64617264', mode='block'):

        try:

            a\_b\_c\_d = self.de\_str.split('\n')

            self.deB = construct\_BNPoint2(a\_b\_c\_d[0], a\_b\_c\_d[1], a\_b\_c\_d[2], a\_b\_c\_d[3])

            x\_y = self.Ppub\_str.split('\n')

            self.P\_pub\_e = construct\_BNPoint(x\_y[0], x\_y[1])

            if not dll.P2\_isOnBNTCurve(self.deB):

                print('deB is not on the curve.\n')

                return

            if not dll.P\_isOnBNCurve(self.P\_pub\_e):

                print('P\_pub\_e is not on the curve.\n')

                return

            msg\_bytes=bytes(str\_to\_hex\_list(msg\_bytes))

            mLen = len(msg\_bytes)

            dll.Get(byref(self.r), r\_str.encode(), 16)

            K2\_len = 256

            if mode == 'stream':

                K2\_len = K2\_len / 8

                klen = int((mLen + K2\_len) \* 8)

                K = self.char\_arr(klen / 8)

                dll.PKC\_kem(byref(K), byref(self.C1), self.\_\_id\_str.encode(), self.P1, self.P2, self.P\_pub\_e, int(klen),

                            self.r)

                K1 = self.char\_arr(mLen)

                for i in range(mLen):

                    K1[i] = K[i]

                K2 = self.char\_arr(K2\_len)

                for i in range(int(K2\_len)):

                    K2[i] = K[i + mLen]

                self.C2 = self.char\_arr(mLen)

                dll.Bytes\_XOR(byref(self.C2), byref(K1), msg\_bytes, mLen)

                self.C3 = self.char\_arr(32)

                dll.MAC(byref(self.C3), byref(self.C2), mLen, byref(K2), int(K2\_len))

                return (BNPoint\_byte(self.C1).upper() + Byte\_str(self.C3, 32).upper() + Byte\_str(self.C2,

                                                                                                 mLen).upper())

            elif mode == 'block':

                K1\_len = 0x80

                klen = K1\_len + K2\_len

                K = self.char\_arr(klen)

                dll.PKC\_kem(byref(K), byref(self.C1), self.\_\_id\_str.encode(), self.P1, self.P2, self.P\_pub\_e, int(klen),

                            self.r)

                K1\_len = int(K1\_len / 8)

                K2\_len = int(K2\_len / 8)

                K1 = self.char\_arr(K1\_len)

                for i in range(K1\_len):

                    K1[i] = K[i]

                K2 = self.char\_arr(K2\_len)

                for i in range(int(K2\_len)):

                    K2[i] = K[i + K1\_len]

                if mLen % 16==0:

                    block\_size = mLen/16

                else:

                    block\_size = (math.floor(mLen/16)+1)\*16

                input = self.char\_arr(block\_size)

                for i in range(block\_size):

                    if i < mLen:

                        input[i] = msg\_bytes[i]

                    else:

                        input[i] = 0x0C

                self.C2 = self.char\_arr(block\_size)

                dll.sm4\_setkey\_enc(byref(self.ctx), K1)

                dll.sm4\_crypt\_ecb(byref(self.ctx), 1, block\_size, byref(input), byref(self.C2))

                self.C3 = self.char\_arr(32)

                dll.MAC(byref(self.C3), byref(self.C2), 32, byref(K2), K2\_len)

                return (BNPoint\_byte(self.C1).upper() + Byte\_str(self.C3, 32).upper() + Byte\_str(self.C2, block\_size).upper())

        except Exception as e:

            logging.debug(e)

#解密

    def decrypt(self, cipher = '2445471164490618E1EE20528FF1D545B0F14C8BCAA44544F03DAB5DAC07D8FF42FFCA97D57CDDC05EA405F2E586FEB3A6930715532B8000759F13059ED59AC0BA672387BCD6DE5016A158A52BB2E7FC429197BCAB70B25AFEE37A2B9DB9F3671B5F5B0E951489682F3E64E1378CDD5DA9513B1C',

                mode='block'):

        try:

            if not self.\_\_get\_cipher(cipher,mode):

                return

            a\_b\_c\_d = self.de\_str.split('\n')

            self.deB = construct\_BNPoint2(a\_b\_c\_d[0], a\_b\_c\_d[1], a\_b\_c\_d[2], a\_b\_c\_d[3])

            x\_y = self.Ppub\_str.split('\n')

            self.P\_pub\_e = construct\_BNPoint(x\_y[0], x\_y[1])

            if not dll.P2\_isOnBNTCurve(self.deB):

                print('deB is not on the curve.\n')

                return

            if not dll.P\_isOnBNCurve(self.P\_pub\_e):

                print('P\_pub\_e is not on the curve.\n')

                return

            dll.Pairing\_opt(byref(self.w), self.deB, self.C1)

            msg\_len = int((len(cipher)-128-64)/2)

            idLen = len(self.\_\_id\_str)

            len\_msg = 64 + 384 + idLen

            msg = self.char\_arr(len\_msg)

            dll.PtoByte(byref(msg), self.C1)

            msg1 = self.char\_arr(384)

            dll.F12toByte(byref(msg1), self.w)

            for i in range(384):

                msg[64 + i] = msg1[i]

            for i in range(idLen):

                msg[64 + 384 + i] = self.\_\_id\_str[i].encode()

            K2\_len = 256

            if mode == 'stream':

                K2\_len = K2\_len / 8

                klen = int((msg\_len + K2\_len) \* 8)

                K\_ = self.char\_arr(klen / 8)

                dll.KDF(K\_, msg, len\_msg, int(klen))

                K2\_len = int(K2\_len / 8)

                K1\_ = self.char\_arr(msg\_len)

                for i in range(msg\_len):

                    K1\_[i] = K\_[i]

                K2\_ = self.char\_arr(K2\_len)

                for i in range(int(K2\_len)):

                    K2\_[i] = K\_[i + msg\_len]

                M\_ = self.char\_arr(msg\_len)

                dll.Bytes\_XOR(byref(M\_), K1\_, self.C2, msg\_len)

                return Byte\_str(M\_, msg\_len).upper()

            elif mode == 'block':

                K1\_len = 0x80

                klen = K1\_len + K2\_len

                K\_ = self.char\_arr(klen / 8)

                dll.KDF(K\_, msg, len\_msg, int(klen))

                K1\_len = int(K1\_len / 8)

                K2\_len = int(K2\_len / 8)

                K1\_ = self.char\_arr(K1\_len)

                for i in range(K1\_len):

                    K1\_[i] = K\_[i]

                K2\_ = self.char\_arr(K2\_len)

                for i in range(K2\_len):

                    K2\_[i] = K\_[i + K1\_len]

                block\_size=len(self.C2)

                M\_ = self.char\_arr(block\_size)

                dll.sm4\_setkey\_dec(byref(self.ctx), K1\_)

                dll.sm4\_crypt\_ecb(byref(self.ctx), 0, block\_size, byref(self.C2), M\_)

                return Byte\_str(M\_, msg\_len).upper()

        except Exception as e:

            logging.debug(e)

if \_\_name\_\_ == '\_\_main\_\_':

    sm9\_enc = CryptSM9\_Encryption()

    Ppub, de = sm9\_enc.generate\_key()

    action = sys.argv[1]

    if action == "encrypt":

        message = urllib.parse.unquote(sys.argv[2])

        print(message)

        # message = '杨敏铝'

        message = message.encode('utf-8')

        message1 = message.hex()

        cipher= sm9\_enc.encrypt(mode='block', msg\_bytes=message1)

        print(cipher)

    if action == "decrypt":

        cipher = sys.argv[2]

        plaintext=sm9\_enc.decrypt(cipher=cipher, mode='block')

        print(plaintext)

        plaintext = plaintext.replace('0C', '')

        print(plaintext)

        result = codecs.decode(plaintext.encode('utf-8'), "hex").decode('utf-8')

        print(result)

测试

5.1 对小程序基本功能的测试

5.1.1测试资源与测试环境

|  |  |  |  |
| --- | --- | --- | --- |
| 关键项 | 数量 | 性能要求 | 期望到位阶段 |
| 测试PC机 | 一台 | CORE I7，内存16G | 实现系统后台功能 |
| 微信开发者工具 | 一个 | / | 实现小程序前端功能 |

|  |  |
| --- | --- |
| 资源名称/类型 | 配置 |
| 操作系统环境 | Win10 64位 |
| 支撑平台 | 微信开发者工具 |
| 测试方法 | 手工测试 |
| 软件运行工具 | Pycharm、Navicat for MySQL、微信开发者工具、phpstudy |

5.1.2对小程序基本功能的测试

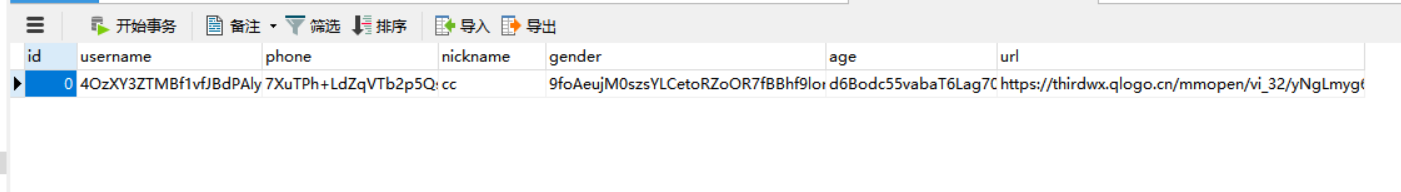
|  |  |
| --- | --- |
| 测试范围 | 检索时间与非法攻击 |
| 测试目标 | 检索速度毫秒级以上，可以抵抗唯密文攻击 |
| 采用技术 | Debug调试与黑盒测试 |
| 工具与方法 | 手工测试 |
| 开始标准 | 功能测试完成 |
| 完成标准 | 检索数据毫秒级反应以及无安全漏洞 |
| 测试重点与优先级 | 根据实际需求设定 |
| 需考虑的特殊事项 | 根据实际需求设定 |

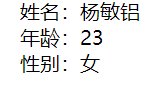
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 疼痛部位 | 操作步骤 | 预期输出 | 执行时间 |
| 01 | 头 | 点击 | 成功预约 | 35ms |

5.2 对加密效果的测试

5.2.1对sm2加密效果的测试

以对患者个人信息的进行sm2加密为例，用个人账号登录并填写信息后，查看后台数据库的信息，可以发现用户个人信息已经经过了sm2的加密（nickname和url不需要加密）。

在小程序的用户界面，通过读取数据库信息并解密后得到的明文：



**5.3 应用前景**

2019 年 3 月 21 日，国家卫生健康委员会就信息化质控与智慧医院建设工作 有关情况举行了专场发布会。重点开展以下几方面工作：一是以电子病历为核 心推动医疗机构信息化建设。二是落实《国务院办公厅关于促进“互联网+医疗健康”发展的意见》，实施进一步改善医疗服务行动计划，运用信息化手段解决人民群众看病就医过程当中的“难点”、“堵点”问题。三是加强智慧医院

建设，出台医院智慧服务分级评估标准体系，推动医院运用智能化、信息化手

段，提高医疗质量和效率，提升精细化、信息化管理水平。四是应用信息化手

段加强医疗管理。

展望国内外医疗卫生事业的发展前景，信息化、智能化已经成为医疗技术 的发展趋势。在国内，医疗卫生事业的信息化建设已经成为新一轮医疗体制改 革的重要方面，并且对促进经济转型发挥了积极作用。智慧医疗分诊系统基于 小程序采用 Python和PHP 语言进行系统开发。借助数字化、可视化模式，实现对病 人账户以及医生个人敏感信息的保护，并开发了智慧分诊以及医生叫号的功能， 将有限的医疗资源让更多人共享。

结论

通过本次课题设计了基于结合国密算法SM2和SM9的智慧医疗分诊系统，为人们看病就医提供更加方便安全的网上预约挂号，有助于在疫情这个时代背景之下减轻医疗系统的负担。

通过初步的设计、编码、测试后，本课题已经实现并满足以下功能：

1. 可以对就医人员的个人信息进行加密存储在数据库中。
2. 对医生的关键敏感信息进行加密保护。
3. 就医人员使用分诊或直接选择科室的方式进行挂号。
4. 医生可直接查看名下就医人员。

通过SM2和SM9国密算法加密，数据库中的数据可以防止极大程度的唯密文攻击，在如此繁琐的加密和解密的模式下，该系统的功能与时间效率仍然不受影响，检索速度可以达到毫秒及以上。

参考文献

[1]旷珊珊. 面向科室分诊及医生推荐的智能导诊系统研究与设计[D].河北工程大学，2021.

[2]陈家驹，翁昌晶，刘谦，羊海锋.基于全流程的急诊智慧医疗服务体系建设探讨[J].中国医疗设备，2021，36(08):134-137+156.

[3]沈珩. 基于Android的医生随诊系统的设计与实现[D].北京邮电大学，2018.

[4]王嘉宁，苏翀，任瞳.医院智能导诊系统设计与实现[J].医学信息学杂志，2018，39(08):29-32.

[5]毛宇. 移动网络支付中的国密算法研究[D].哈尔滨理工大学，2018.

[6]许小波.基于国密算法的设备安全认证系统设计[J].电子产品世界，2021，28(03):36-38+56.

[7]杨春雷.国密算法在国库信息系统中的应用研究[J].金融科技时代，2021，29(07):53-57.

[8]胡景秀，杨阳，熊璐，吴金坛.国密算法分析与软件性能研究[J].信息网络安全，2021，21(10):8-16.

致谢

时光匆匆，大学四年就要过去，这篇课题成为了我大学四年的句号。十分感谢在完成课题的过程中给予我莫大帮助的周权老师，也很感谢父母在生活上无条件对我的支持，在精神上支持我的朋友们：邓小英、罗思源、蒲杨、赵孟浩、吴振、胡志航、张裔天，是她们在我十分迷茫的时候给予我鼓励，让我的大学生活更加充实！