

涉密信息系统工程

实验二 对称加密算法的研究与使用



中国海洋大学

2018.03-2018.04

小组分工

学号	姓名	分工
15020032003	葛畅	对称密码算法的对比及优缺点；参考文献格式整理
15020032009	刘艺语	环境搭建；程序实现及优化；测试结果及分析；
15020032010	刘渊晨	密码学及对称加密算法的发展历史、研究现状、应用现状；测试结果及分析；
15020032011	刘子悦	AES 算法研究现状、应用现状、基本原理；加解密、对称加解密基本原理；程序修改及界面设计；

目录

1 实验目的.....	1
2 实验预备知识.....	1
2.1 加密与解密的基本原理.....	1
2.1.1 密码学的发展历史、研究现状、应用现状.....	1
2.1.2 加密与解密算法的基本原理及解释密码学中基本概念.....	6
2.2 对称加密算法的基本原理.....	8
2.2.1 对称加密算法的发展历史、研究现状、应用现状.....	8
2.2.2 对称加密与解密算法的基本原理.....	12
2.2.3 对比三种对称加密算法.....	13
2.3 拟采用的加密算法原理.....	16
2.3.1 AES 算法的研究现状、应用现状及主要优缺点.....	16
2.3.2 AES 加密算法的具体原理.....	18
3 实验内容.....	19
3.1 环境搭建说明.....	19
3.1.1 选定语言或框架，搭建完整的开发环境.....	19
3.1.2 加密算法库.....	22
3.1.3 在开发环境中设置对于加密算法库的引用.....	22
3.2 加密及解密程序的实现.....	23
3.2.1 加密解密函数.....	23
3.2.2 设计 UI 界面.....	25
3.2.3 将 ui 文件转换为 py 文件.....	25
3.2.3 添加事件.....	25
3.2.4 将单个 py 文件发布为 exe 可执行文件.....	26
3.3 测试结果及分析.....	27
3.3.1 文件选择对话框在 MainWindow 下崩溃.....	27
3.3.2 基于 cryptography 库中 hazmat 层提供的底层加密基元写成的 AES+CBC 加密算法可用性不强.....	28

3.3.3 加密解密需要一次完成.....	28
3.3.4 关于 os.urandom(n)函数.....	28
3.3.5 测试结果.....	29
3.3.6 分析.....	39
参考文献.....	41

1 实验目的

1. 理解对称加密算法的基本原理；
2. 学会使用某种语言的加密算法调用方法；
3. 基于 C++、Java、或 .net 中成熟的 AES 或 SM4 加密库实现一个对称加密解密算法程序。

2 实验预备知识

2.1 加密与解密的基本原理

2.1.1 密码学的发展历史、研究现状、应用现状

2.1.1.1 密码学发展历史：

每当听到“密码学”这一词，首先映入我们脑海的可能是电子邮件的加密、网络银行、支付宝的使用、证券交易等，似乎密码学只与我们电子通讯有着密不可分的联系。实际上，密码学是一门古老的学科，历史的道路上到处点缀着密码的踪迹。我们走过了农业社会、工业社会，如今正处于信息社会的伟大时代，每个社会阶段都对应着不同的密码学发展，密码学的重要作用也日益显著。

密码学的发展大致可分为 4 个阶段^[1]：手工阶段：最早使用密码学的例子可以追溯到公元前 2000 年，当古埃及还在使用没有标准密码规则的象形文字时。可以说自从有了人类战争，就有了密码技术。自埃及时代起，在几乎所有发明了文字的文化圈中，密码学总是以各种形式存在其中。早期的密码技术在我们现在看来是非常简单的，主要以“替换”和“换位”实施密码变换，例如，古希腊的斯巴达密码棒，古罗马的凯撒（Caesar）密码，法国的维吉尼亚（Vignere）密码等等。

机械阶段：随着破译技术的不断进步，人们对密码算法的安全性要求也越来越高，相应的算法复杂度也越来越大，这使得我们开始改进加密手段，开始使用机械的方法来实现相对复杂的密码算法。从 20 世纪初到第二次世界大战后，出

现了一些专用的加密机器，比如二战时期德国使用的恩尼格玛（Enigma）密码机，日本使用的紫密加密机。

现代密码阶段：随着电子通信与计算机技术的发展，密码学得到了系统的发展。1949 年，香农（C.Shannon）发表了《保密系统的通信原理》和《通信的数学理论》两篇旷世之作，以数学方法为密码学和通信技术奠定了坚实的数学理论基础，证明了一次一密密码系统的完善保密性，标志着密码学从艺术迈上了科学的轨道^{[2]25}。

密码学的新方向：1971 年，IBM 提出了作为对称密码体系基础的分组密码思想及结构，引发了密码学家们研究对称密码标准的高潮。为了满足政府及民间对信息安全的需求，1976 年，美国国家标准局正式公布了数据加密标准 DES，对密码学发展产生了深远的影响。同年，W.Diffie 和 M.Hellman 发表了划时代的论文《密码学的新方向》，开创了公钥密码学的新纪元。从此以后，密码学以崭新的科学面貌进入了信息安全领域。

2.1.1.2 密码学研究现状

近年来，密码学已与物理学、量子力学、生物学等学科相结合，形成了“量子密码”“混沌密码”“生物密码”等新型的密码学领域，但其发展还不够成熟，各国都在努力研究和发展新型密码技术，争夺战略制高点^{[2]280}。

研究新型计算机破解密码的项目是各国科学家们正在关注的领域，我国也在不断努力研发自己的超级计算机和量子计算机。超级计算机之所以对现行的密码体制造成威胁，是它无敌的计算速度。现行的一些密码方案是属于计算安全的。是建立在困难性假设的基础上而设计的，如果计算资源和计算能力允许，它们是可以被强力破解的。所以超级计算机或量子计算机的出现，求解密码方案中的困难性问题就相当容易。面对岌岌可危的现有密码方案，只有发展新的密码技术才能解决当今密码学的忧虑。

目前，密码学界已在纷纷研究能够抵抗量子计算机工的密码方案，我们称为“后量子时代密码学”。前面提到的混沌密码技术、量子密码技术、生物特征密码技术的理论都已初见成效，但仍需一段时间，这些技术才能应用于我们日常的保密需求之中。

混沌密码技术：我们知道，密码方案设计的最基本原则是扩散与混乱，混沌

理论中的非周期性、不可预测性、连续宽频带和类噪声等特点，与实现扩散与混乱的目标不谋而合。混沌加密系统一定程度上模仿了“一次一密”的特点，它有保密性强、随机性好、密钥量大、更换密钥方便等特点，在抗干扰性、截获率、信号隐蔽和加密速度等方面具有较大优势，它有望广泛应用于图像和视频加密领域，由于混沌密码是不依赖于算法复杂度的系统，因此安全性高。但也存在着抗选择明文攻击差、低维混沌不能产生高保密性等缺点。

量子密码技术：国际上，量子密码通信日趋成熟。同样地，我国量子密码也达到了世界先进水平，甚至在某些方面已经处于领先地位。但量子密码要真正走向实用，还任重而道远，距其发展成熟、稳定可能还需 15 到 20 年。量子密码目前的主要研究包括量子密钥分配、量子认证、量子秘密共享等。量子密码学的理论基础是量子力学，它具有两个不可比拟的属性，一是能检测信道中的窃听行为，使数据不可复制和篡改；二是能实现比目前的密码体制更高的安全等级。

生物特征密码技术：目前应用最广泛的有指纹和人脸识别技术，并且已经发展了掌纹识别、掌型识别、发音识别、虹膜识别、静脉识别、签名识别等多种生物识别技术。在密码方案中，随机性强的密钥，其安全性也高，但是记忆难度也相应地增加，也就是说，密钥的安全性与方便性之间是对立的。然而生物密码技术既可以保证密钥的随机性，又能保证使用的方便性。生物密码技术与传统的密码技术相比，具有依附于人体、不易伪造、不易模仿、无需记忆等优势，是极具应用潜力的新型密码方法。

DNA 密码是近年来颇受学术界关注的一个生物密码前沿方向。DNA 密码源于 DNA 计算，DNA 计算是一种以生物分子 DNA 作为计算介质，以生物化学反应为计算手段的新的计算方法。DNA 计算具有并行性、低能耗性和存储量大的优点，有很多的数学困难问题可以通过 DNA 计算实现穷举搜索而得到结果。DNA 密码是以 DNA 为信息存储的载体，借助于 DNA 的生物化学特性实现数据安全的密码系统，具体利用 DNA 编码技术实现加密或信息隐藏技术。DNA 密码具有一些潜在特性，如大规模并行性可以实现快速加密和解密过程，生物化学特性可以实现大容量数据加密存储，不依赖于数学困难问题而设计的方式可抵抗超级计算机的攻击。随之 DNA 计算的深入发展，以 DNA 计算为基础的 DNA 密码应用前景已逐渐显现出来。

上述是一些目前正在研究与发展的新型密码技术^{[2]244}，但一个新的密码方案的提出，我们怎样才能证明它是安全的呢？目前采取的做法是：提出一种方案后，基于某种假设对方案作出具体的安全性分析，给出其安全性论断，若该方案在较长时间内不能被破解，大家就普遍接受密码方案的安全性；若在使用后发现安全漏洞，就对密码方案进行修改和改进，以继续使用。事实上，这种做法有很大的缺陷，因为我们不知道它是否能抵抗未知攻击。这就涉及到了另一个学术界关注的研究领域——可证明安全。可证明安全理论是通过证明来给出密码方案的安全性论断，证明它不仅能抵抗已知攻击，还能抵抗未知攻击，它实质上是一种证明密码方案安全性的形式化方案。现在可证明安全性方法主要有两大类：一类是一计算复杂性理论为基础，首先指定密码协议的安全目标，然后根据攻击者的能力形式化定义一个攻击模型，再采用计算复杂性理论的归约方法（将密码方案的破译归约到数学困难性问题的攻破或归约到数学问题的困难性）去分析密码协议，最后若能指出攻破密码协议的唯一方法就是破解其数学困难性假设，我们就说密码协议是可证明安全的。另一类方法称为形式化分析方法，采用逻辑、代数或自动机之类的方法描述和推理协议的安全性，像 BAN 逻辑、SVO 逻辑、CTL 逻辑、串空间、Petri 网等方法，这些方法还可以做模型检测和软件漏洞分析。近年来，学者们还研究出了一些自动检测软件工具，它们能够利用计算机自动检测来证明安全协议的安全性和可靠性。

可证明安全性理论具有重要的理论价值和应用价值，目前已成为国内外密码学界最为关注的问题之一。

2.1.1.3 密码学应用现状

伴随着信息网络的发展，病毒、木马、黑客、蠕虫、后门等术语逐步为大众所知，“棱镜门”、“心脏出血”、银行账号和隐私泄露等网络攻击事件不断发生，使网络安全渐成焦点，网络攻击威胁日益严峻，势必需要牢固的密码技术对信息保驾护航。其实，现代密码技术已悄然走到我们身边，网上购物、网络银行、第三方支付、基金证券交易等电子商务活动的不断掀起，让我们逐渐感受到密码技术的用场。我将列举以下几个密码技术的应用场景^{[2]256}：

网上办公系统的密码技术：网络技术的飞速发展正在全面推进无纸化办公的进程，网上办公系统是企事业内部与外部之间实现办公信息收集与处理、流动与

共享，以实现科学决策的信息系统。网上办公管理信息系统经常受到攻击，包括“密锁”病毒、“砸波”(ZBot)木马，还有 2013 年针对中国国家顶级域名系统.CN 的历史上最大规模的拒绝服务攻击(DDOS)。网络办公系统的安全性主要涉及：物理安全、网络安全、数据安全隐患。以加密技术和数字认证技术为核心的密码技术依然是网上办公系统的关键技术，加密技术主要解决数据保密性问题，包括存储加密和传输加密；数字认证主要通过数字签名、数字信封、数字摘要、数字时间戳、数字证书和生物识别等技术实现，在网络通信中建立可信安全通信信道，以确保信息通信的合法性、完整性和不可抵赖性。公钥密码学的出现推动了数字认证方法的发展，使得网络办公系统变得更为安全可靠。

电子邮件的加密与认证：目前常见的电子邮件加密方法主要有三种，一是利用对称加密算法加密邮件，这种方式目前已经很少使用；二是利用 PKI/CA 认证加密来加密邮件，这是一种基于公钥密码系统的方式；三是利用基于身份的密码技术进行电子邮件加密，基于身份的密码是 Shamir 于 1984 年提出的，其思想是将用户公开的身份信息作为用户公钥，用户私钥由一个称为私钥生成器的可信中心生成。

网上购物安全保障技术：互联网上的虚拟交易平台，一般通过第三方可信机构实现交易。对于淘宝网，支付宝充当着买卖双方的可信第三方，“支付宝网站”采用了先进的 128 位 SSL 加密技术，对于用户，支付宝采用了数字证书技术机制；对于“支付宝账户”，采用了登录口令和支付口令双重口令。类似于淘宝网，大多数购物网站采用的核心安全技术一般是 PKI/CA、数字证书、安全控件等，其最根本的基础是加密技术、数字签名技术及数字鉴别技术等。

网上支付的安全防护：网上支付是电子支付的一种形式，包括两种方式，一种是采用用户网上银行直接支付；另一种可以通过第三方与银行之间的支付接口进行支付（如支付宝）。网上支付的安全保障的核心技术仍然是加密算法和数字签名算法，具体支撑包括认证技术实现欺诈的防范、加密技术实现信息的保密、采用数据杂凑技术实现数据的完整性、利用数字签名保证交易行为的不可抵赖性、利用复杂数字签名技术实现多边支付安全。作为个人用户，则可使用 U 盾（采用 1024 位非对称密钥算法对网上数据进行加密、解密和数字签名）、数字证书、宝令、电子密码器等帮助提升账户的安全等级。

手机信息安全的防护：针对手机用户的安全威胁主要有窃听、定位跟踪、WiFi 及无线路由器漏洞和社交网络隐私泄露等。要从根源上解决手机信息安全，还得依靠密码学中的加密技术和认证技术。手机通话语音或相关敏感信息一般通过信令加密实现，信令加密有两种形式，一种是对专用控制信道上传送的信令信息的关键字段加密；另一种是采用 AES 算法，对专用控制信道和公共控制信道的部分信令进行整条加密。无线安全传输层协议 WTLS 也采用了相关加密技术，功能类似于 SSL 加密传输技术，确保信息在传输的过程中经过编码、加密处理。

密码技术还应用在云计算环境、大数据时代中，总之密码技术的应用早已渗透到我们身边的一点点滴，密码学的发展将会一直持续下去，同时需要我们大众进一步地了解密码技术，关注自身的信息安全，提升安全意识。

2.1.2 加密与解密算法的基本原理及解释密码学中基本概念

要讲解加解密算法基本原理，我们需要为参与信息交换的实体物质和抽象方法起名字，因此在本实验报告中我们用 Alice、Bob 等人名来指代这些信息交换的参与者（表 2-1）。

表 2-1 本实验报告中的主要名词一览

名称	说明
Alice	普通参与者
Bob	普通参与者
Eve	窃听者
M	明文
C	密文
K	密钥
$E(\cdot)$	加密
$D(\cdot)$	解密

下面我们引入一个场景：Alice 想 Bob 发送电子邮件。在此场景中，发出邮件的 Alice 称为发送者（sender），收到邮件的 Bob 称为接收者（receiver）。但邮件在互联网中传输时，存在窃听者（eavesdropper）窃听邮件的可能（图 2-1）。

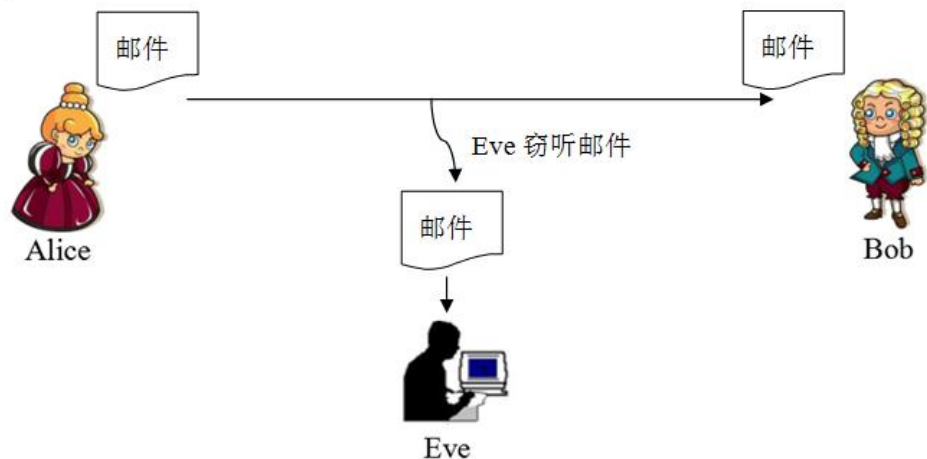


图 2-1 Alice 向 Bob 发送邮件，Eve 窃听邮件

Alice 为避免窃听者窃听邮件，将邮件使用密钥（K）进行加密（encrypt）后发送出去；Bob 接收到来自 Alice 的加密邮件，需要利用密钥（K'）对加密邮件进行解密（decrypt）。在此过程中涉及密码学的基本概念（表 2-2），同时此过程也可抽象为加密和解密的基本原理（图 2-2）。

表 2-2 密码学的基本概念

概念	说明
明文（M）	需要加密的信息；
加密	隐藏信息的过程；
加密方法（E(•)）	隐藏信息的方法；
密文（C）	加密后的明文；
解密	从密文恢复明文的过程；
解密方法（D(•)）	从密文恢复明文的方法；
密钥（K）	一种参数，它是在明文加密为密文或将密文加密为明文的算法中输入的参数，密钥分为对称密钥与非对称密钥。

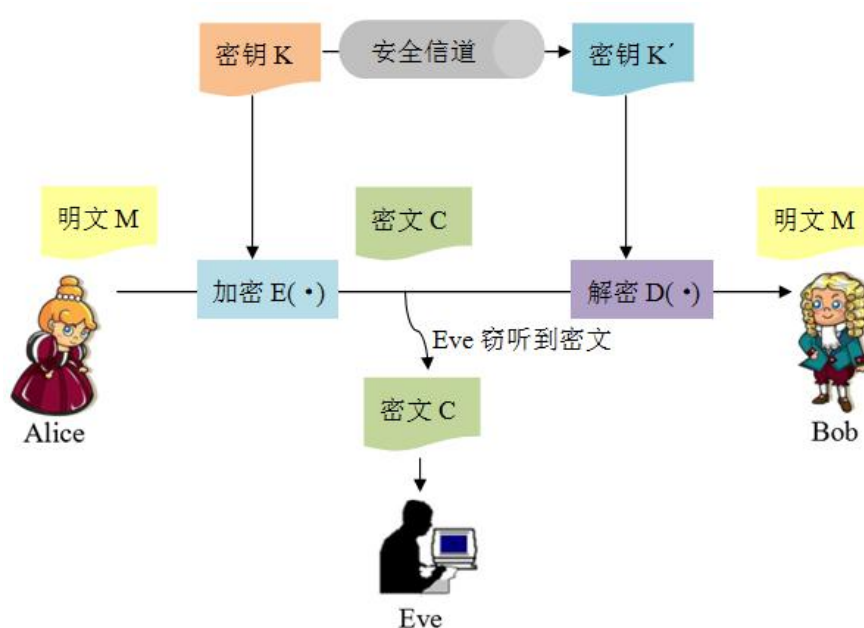


图 2-2 加密和解密的基本原理

注意加密和解密的密钥不一定相同，加密和解密涉及的基本公式：

$$\text{加密: } E_K(M) = C \quad 2-1$$

$$\text{解密: } D_{K'}(C) = M \quad 2-2$$

$$D_{K'}(E_K(M)) = M \quad 2-3$$

2.2 对称加密算法的基本原理

2.2.1 对称加密算法的发展历史、研究现状、应用现状

2.2.1.1 对称加密算法的发展历史

古典密码让我们最先感受到的其实就是对称密码的雏形，它的主要特点是加密和解密所使用的密钥是相同的，现代对称密码体制就是源于这种原理。按照对明文消息加密方式的不同。对称密码体制一般可分为分组密码和序列密码（也称流密码）两类。

1. 序列密码

早期著名的一次一密（One-Time Pad）是序列密码产生的思想来源。序列密码最早是在 1917 年由 Gilbert Vernam 发明^[3]，不过在那个时代并没有称之为序列密码。Vernam 发明了一个可以自动加密电传打字机通信的电机设备。明文以纸

带的形式输入到该设备中，而密钥序列则是作为第二条纸带输入到该设备中。所以序列密码有时也称为 Vernam 加密，而一次一密有时也称作 Vernam 密码。

在过去数年中，人们提出了很多序列密码。但与分组密码相比，序列密码受政治的影响很大，虽然也有公开设计和研究成果发表，但流密码的设计与分析成果大多还是保密的，目前可以公开见到的、较有影响的流密码方案包括 A5、SEAL、RC4、PIKE 等。

序列密码的发展有两个方面的显著特点^[4]：一方面，随着人们对非线性序列认识的深入，用于构造序列密码的序列源从简单的对线性序列进行非线性改造转向直接构造非线性序列。从基于线性反馈移位寄存器（Linear Feedback Shift Register, LFSR）的序列密码发展为基于非线性反馈移位寄存器（Non-Linear Feedback Shift Register, NFSR）、整数剩余类环上序列、带进位的反馈移位寄存器（Feedback with Carry Shift Register, FCSR）的序列密码。

另一方面，分组密码的兴起和繁荣对序列密码产生了较大冲击，尤其对面向软件的序列密码设计影响巨大。1987 年，里夫斯特（Rivest）为 RSA 公司设计了 RC4，这是一个以字节输出、面向快速软件实现的序列密码体制。2004 年 11 月，由 ECRYPT（European Network of Excellent for Cryptology）启动了欧洲序列密码计划—eSTREAM，其主要目的是公开征集序列密码体制，推动序列密码的研究并希望最终获胜的序列密码体制能得到广泛应用。到 2005 年 5 月，ECRYPT 共征集了 34 个候选算法。经过前两轮筛选，最终有 16 个算法进入了 eSTREAM 第三阶段的评测，分别是 CryptMT、Dragon、HC、LEX、NLS、Rabbit、Salsa20、SOSEMANUK、DECIM、Edon80、F-FCSR、Grain、MICKEY、Moustique、Pomaranch 和 Trivium，其中前 8 个算法是面向软件实现设计的，后 8 个算法是面向硬件实现设计的。

eSTREAM 的候选密码体制基本反映了当代序列密码的发展趋势。分组密码不仅对序列密码在形式上产生影响（软件化、分组化和标准化），而且其设计思想也逐步向序列密码设计渗透，特别是以字节为输出、面向快速软件实现的序列密码体制。

2. 分组密码

分组密码的研究有着悠久的历史，有些古典密码就是采用分组密码。现代分

组密码的研究始于 20 世纪 70 年代美国国家加密算法 DES 的征集。为了建立适用于计算机系统的商用密码，美国国家标准局 NBS 于 1973 年 5 月 15 日和 1974 年 8 月 27 日先后向公众发布了征求加密算法的公告。在对提交的算法进行评估后，IBM 公司提出的分组密码算法 Lucifer 中选，它于 1976 年 11 月被美国政府采用，随后被美国国家标准局和美国国家标准协会（ANSI）承认，1977 年 1 月以数据加密标准 DES（Data Encryption Standard）的名称正式向社会公布。由于 DES 密钥长度仅为 56bit，造成了被攻破的局面，1994 年 1 月，美国国家安全局（NSA）决定在 1998 年 12 月以后不再使用 DES 作为加密标准。

随后美国标准技术研究所（NIST）又提出了一种三重 DES 标准，称为 3DES（Triple DES），并于 1998 年被正式定义。1990 年国际数据加密算法 IDEA（International Data Encryption Algorithm）正式公布，它是由瑞士联邦学院的 Xuejia Lai 和 James Massey 研制的一个对称密码分组，曾经是有希望替代 DES 的众多算法的一种。它是在 DES 的基础上发展出来的，类似于三重 DES，但它同样具有密钥太短的缺点。

1997 年，NIST 开始征集新的数据高级加密标准 AES（Advanced Encryption Standard）。2000 年 10 月 2 日，在大会的投票表决中，Rijndael 算法脱颖而出作为新的 AES。2002 年 5 月 26 日，高级加密标准 AES 成为有效的标准。如今，AES 已然成为了对称密钥加密中重要的国际标准之一，并逐渐取代了 DES 的地位，走向前台。

2012 年 3 月 21 日国家密码管理局发布第 23 号公告，将 SM4 算法确定为国内官方公布的第一个商用密码算法，标志着我国也有了自己的对称密码标准，并逐步走向国际，也体现了我国的分组密码算法设计已达到国际先进水平。

近些年，应用需求推动了轻量级分组密码的产生。轻量级分组密码最显著的特点是密钥长度相对较短，密码算法结构简单，这一点也是它广泛应用于微型设备的根本原因。目前典型的轻量级密码算法主要有 PRESENT、MIBS、TWIS、PRINT cipher、KLEIN 和 LED。PRESENT 算法是由 Bogdanov 等人在 2007 年的顶级国际学术会议 CHES 上提出来的，该算法的功耗和面积与安全性达到了很好的平衡，可以说它是一种超级轻量级密码算法。MIBS 是 M.Izadi 等人在国际会议 CANS 2009 上提出的，LED 算法时 Guo 等人在 2011 年 CHES 会议提出的。

2.2.1.2 对称加密算法的研究现状:

序列密码当中,级数较低的比较简单的 LFSR 结构在实际应用中的安全系数不高,所以针对此模型当前的研究的主要思路是构建相对复杂的反馈函数,共有三个方向:一是在一个或多个 LFSR 中添加非线性元素;二是考虑 LFSR 的级数 n 足够大,三是考虑使用非线性反馈移位寄存器 NLFSR。此外,如何生成与 m 序列(满足 Golomb 的 3 个随机性公设的序列)不同的伪随机序列也是当今流密码研究的一个重要方向,目前所拓展的伪随机序列的周期只是整个自然数中的一小部分,研究者们希望看到更多新的周期加入进来^{[1][20]}。

分组密码的早期研究基本上是围绕 DES 进行的,进入 21 世纪以来,随着 AES 算法的出现,对分组密码的研究又掀起了新的高潮。虽然已有了广泛的应用,目前分组密码还有很多理论问题和实际问题有待继续研究和完善,包括:如何设计可证明安全的密码算法,如何测试密码算法的安全性,分组密码的实现研究,包括软件优化、硬件实现和专用芯片等等。

轻量级密码技术的研究也是当前很受关注的一个方向,相应地,轻量级安全认证协议也是一个重要的研究领域,例如轻量级的 MAC 等。传统的公钥基础设施 PKI/CA 体系过于庞大、成本高和技术复杂,对于计算能力受限的应用环境不适用,而轻量级认证协议的主要目标是解决 RFID 系统中存在的用户安全和隐私问题,通过开销小的计算实现认证功能^{[2][24]}。

2.2.1.3 对称加密算法的应用现状:

流密码以其易于实现、加/解密快速、无错误传播、应用协议简单等优点,在政府、军事、外交等重要部门的保密通信及各种移动通信系统中被广泛应用。在某些情况下,序列密码实现比分组密码实现所需的资源(比如代码大小或芯片面积)少,所以对受限的环境而言(比如手机),序列密码更具优势。目前应用较为广泛的流密码体制有 A5 算法和 RC4 算法。全球移动通信系统,即 GSM 是欧洲提出的移动通信标准,A5 是 GSM 中执行加密运算的流密码算法,它用于从用户手机到基站的连接加密。RC4 是应用最广泛的基于软件实现的序列密码体制,已被应用于 MS Windows、Lotus Notes、Oracle SQL 及使用安全套接字层 SSL 协议的 Internet 通信等方面。2004 年,3GPP 启动长期演进计划(LTE)的研究,

即 4G 国际通信标准。由我国自主设计的加密算法 128-EEA3 和完整性算法 128-EIA3 也参与了 LTE 通信加密标准的申报工作，最终这两个算法的核心——祖冲之算法（ZUC）被 3GPP 初步定为 LTE 国际标准，并在 SA#53 会议上被正式通过成为国际标准^{[1][29-33]}。

在绝大多数领域中，分组密码的使用比序列密码更广泛。在过去近二十年的时间里，DES 被广泛应用于各行业的加密领域，成为国际上最常用的商用加密算法。在我国，DES 曾被用于三金工程，尤其是金卡工程，DES 算法在 POS 机、ATM 机、智能卡（IC 卡）、加油站、高速公路收费站等领域被广泛使用。自从我国的对称密码标准 SM4 颁布后，国内就基本使用自己的标准了，国家密码管理局已将 SM4 定为我国无线局域网产品的加密算法标准，并逐步被应用于商用密码领域，如基于 SM4 算法的 DSP 和智能卡已广泛应用于银行、保险交通等领域^{[2][106]}。AES 算法具有安全性、高效性、易实现性和灵活性等优点，是一种较 DES 更好的算法，目前已成为商用密码领域的主要加密方法，AES 的产品在我们日常生活中也随处可见，如我们使用的电脑、我们周围的无线信号、我们使用的无线路由器等。

轻量级分组密码主要适用于计算能力弱、存储小、能耗小、吞吐量要求不高、安全级别适中的环境，主要通过硬件实现，例如适用于 RFID 标签、智能卡、非接触 IC 卡、U 盾、U 盘、SIM 卡等设备。

2.2.2 对称加密与解密算法的基本原理

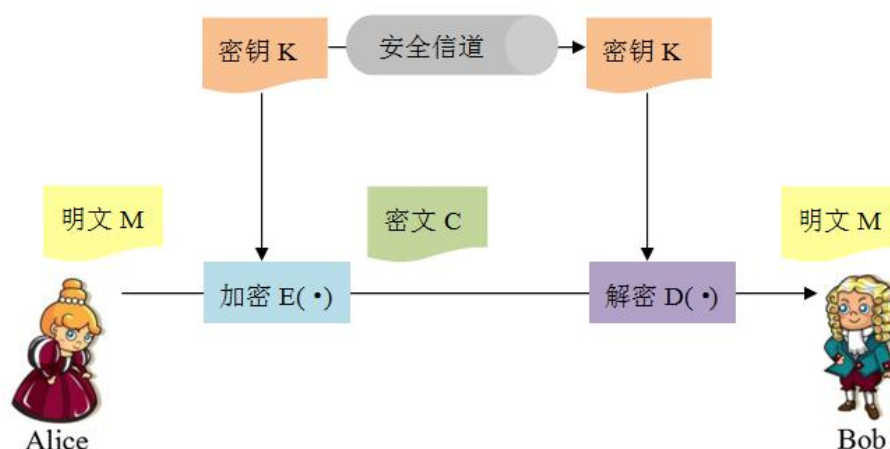


图 2-3 对称加密和解密算法的基本原理

注意加密和解密的密钥相同（与图 2-2 区分），加密和解密涉及的基本公式：

$$\text{对称加密: } E_K(M) = C \quad 2-4$$

$$\text{对称解密: } D_K(C) = M \quad 2-5$$

$$D_K(E_K(M)) = M \quad 2-6$$

2.2.3 对比三种对称加密算法

2.2.3.1 DES 算法

数据加密标准 (Data Encryption Standard, DES)。DES 是一种分组密码体制，它将明文按 64 位一组分成若干组，使用长度为 56 个数据位的密钥，利用变换的组合和迭代产生的分散、错乱的相互作用将明文中的各分组变为密文组的块加密算法。

1. 优点

- 1) 运算速度快。由于 DES 算法仅使用最大为 64 位的标准算术和异或，置换，代换，移位操作四种基本运算，运算速度相对较快；
- 2) 密钥产生容易。DES 算法通过移位、循环移位和置换等操作生成密钥，这些操作对于计算机来说是简单可行的；
- 3) 应用场景广泛。适合于在当前大多数计算机上用软件方法实现，同时也适合在专用芯片上实现^[5]，现在广泛应用于计算机网络通信、电子资金传送系统、保护用户文件、用户识别等场景。

2. 缺点

- 1) 密钥长度过短。由于 DES 算法中只用到 64 位密钥中的 56 位，而第 8、16、24、...、64 位并未参与 DES 运算。换句话说，DES 的安全性是基于除了 8、16、24、...、64 位外的其余 56 位的组合变化才得以保证。不幸的是，随着时间推移，各种能够明显降低成本，且通过暴力破解来发现 DES 密钥的捷径已被发现。况且，随着计算机的运行速度变得越来越快，56 位密钥被认为对于高安全性的程序是不够长的；
- 2) 存在弱密钥。在 DES 算法中，一共存在 12 个半弱密钥和 4 个弱密钥。由于在子密钥的产生过程中，密钥被分成了 2 个部分。假设这 2 个部分恰好分成全 0 或全 1，那么每轮产生的子密钥都是相同的。当密钥是全 0 或全 1，或

者一半是 1 或 0 时, 就会产生弱密钥或半弱密钥, DES 算法的安全性就会变弱;

- 3) 加密单位太小。作为分组密码, DES 的加密单位仅有 64 位二进制, 这对于数据传输来说太小, 因为每个分组仅含 8 个字符, 而且其中某些位还要用于奇偶校验或其他通讯开销;
- 4) DES 不能对抗差分和线性密码分析。迄今为止, DES 算法中的 S 盒 8 个选择函数矩阵的设计原理因美国政府方面的干预, 不予公布。从这一方面严格地讲 DES 算法并不是一个真正的公开加密算法。S 盒设计中利用了重复因子, 致使加密或解密变换的密钥具有多值性, 造成使用 DES 合法用户的不安全性。而且, 在 DES 加密算法的所有部件中, S 盒是唯一的具有差分扩散功能的部件 (相对于逐位异或), 其它都是简单的位置交换, 添加或删减等功能, 毫无差分扩散能力。这样, DES 的安全性几乎全部依赖于 S 盒, 攻击者只要集中力量对付 S 盒就行。

2.2.3.2 IDEA 算法

国际数据加密算法 (IDEA) 是上海交通大学教授来学嘉与瑞士学者 James Massey 联合提出的。IDEA 是作为迭代的分组密码实现的, 使用 128 位的密钥和 8 个循环。IDEA 算法的安全性主要是通过混乱与扩散来对数据进行操作而得到保证的, 普遍认为在目前的计算条件下 IDEA 算法是安全的从理论上讲, IDEA 属于“强”加密算法, 至今还没有出现对该算法的有效攻击算法。

1. 优点:

- 1) IDEA 算法的密码强度很高, 十分安全。IDEA 算法的分组长度为 64 比特, 足够抵抗统计分析, 同时, IDEA 所使用的密码反馈操作方式也进一步加强了密码的强度。IDEA 的密钥长度为 128 比特, 足够抵抗密钥穷尽攻击, 可以想象, 在将来的很长时间内, IDEA 在密码强度方面都是很安全的;
- 2) IDEA 的设计考虑了方便硬件和软件的实现。IDEA 使用 16 比特的自分组, 非常适合在 16 位的机器上运行。同时密码操作也使用了简单的加法、移位等基本操作编程实现, 其中最困难的模乘法也可以容易地用简单的基本操作构成。实现上通常用超大规模集成电路进行的硬件实现和设计目标是取得高速度, 而软件实现则有灵活和低价的优点。

2. 缺点:

- 1) IDEA 算法相对来说是一个比较新的算法, 其安全性研究也在不断进行之中。在 IDEA 算法公布后不久, 就有学者指出: IDEA 的密钥扩展算法存在缺陷, 导致在 IDEA 算法中存在大量弱密钥类, 但这个弱点通过简单的修改密钥扩展算法(加入异或算子)即可克服。目前尚未出现新的攻击算法, 一般认为攻击 IDEA 算法唯一有效的方法是穷尽搜索 128bit 的密钥空间。

2.2.3.3 RC4 算法

RC4 加密算法是大名鼎鼎的 RSA 三人组中的头号人物 Ron Rivest 在 1987 年设计的密钥长度可变的流加密算法簇。RC4 算法是一种在电子信息领域加密的技术手段, 用于无线通信网络, 是一种电子密码, 只有经过授权(缴纳相应费用)的用户才能享受该服务。

1. 优点:

- 1) 算法简单; 算法主要包括初始化算法和伪随机子密码生产算法两大部分, 核心就是随机数生成器和异或运算, 易于编程实现;
- 2) 运行速度快; 该算法的速度可以达到 DES 加密的 10 倍左右;
- 3) 具有很高级别的非线性, 对差分攻击和线性攻击具有免疫能力;
- 4) 运用广泛; 起初用于保护商业机密, 后来该算法被发布在互联网中, 扩大了使用范围, 现在广泛应用于代码加密。因其可以自动为 Html, JavaScript, Css 代码加密, 不留下加密痕迹, 且不改变代码量大小, 不影响运行效率也不会修改开发者的任何代码^[6]。

2. 缺点:

- 1) 由于 RC4 算法采用的是异或运算, 所以, 一旦子密钥序列出现了重复, 密文就有可能被破解。Golic 描述的线性统计弱点^[7], 这个弱点表明, 利用 $2^{6n-7.8}$ 个输出值, 就能够使得的输出值与随机串区分开。后来, Fluher, Mantin 和 Shamir 在其论文《Weaknesses in the Key Scheduling Algorithm of RC4》中提出了两种分析算法的有效方法: Invariance Weakness 和 IV Weakness 分析(合称 FMS 分析方法)。FMS 分析方法的问世, 使得整个 802.11 无线市场存在着不安全的风险。

2.3 拟采用的加密算法原理

2.3.1 AES 算法的研究现状、应用现状及主要优缺点

2.3.1.1 AES 的研究现状

DES 的安全性和应用前景受到挑战, 因此需要设计一个高保密性能的、算法公开的、全球免费使用的分组密码算法, 用于保护敏感信息, 并希望以此新算法取代 DES 算法, 称为新一代数据加密标准, 取名为高级数据加密标准 (AES)。AES 算法并不是一个具体的算法, 而是一个算法的标准, 1997 年 4 月 15 日美国国家标准技术研究所发起征集 AES 算法的活动。2000 年 10 月 2 日正式公布比利时 Rijmen 和 Daemen 设计的 Rijndael 算成为 AES 算法。

目前, 其研究一方面在 AES 的性能分析及其优化, 另一大方面是与密码学其他知识相结合应用于现实场景等, 比如, 以 AES 为核心工具, 采用 CBC 工作模式设计了一个伪随机数生成器^[8]、改进 AES 超混沌加密算法用以图像加密算法的安全性能, 有效抵抗统计攻击和差分攻击^[9]。

2.3.1.2 AES 的应用现状

802 无线局域网草案已经曾将 AES 算法列为必要部件, IETF RFC 草案曾要求为传输层安全标准包含 AES 算法。AES 算法应用大方向的主要是软件实现和硬件实现。

从 AES 高级加密标准算法实现的角度上看, 核心算法的优化是当前主要的研究方法。同时提高算法针对不同应用环境的适应性, 以期在实际应用中取得良好的加密应用效果, 也是各行业研究的主题。当前最吸引各行业的部分, 也是成果较集中的部分是 S 替换盒的研究与优化, 轮变换过程的研究与优化, 密钥扩展的研究与优化这几个方面。

从软件实现的角度上看, AES 加密算法的应用领域是很广泛的, 语音信息加密, 视频信息加密, 数据库数据加密, 电子商务等各种需要电子加密的应用场合, 对它都给予厚望。AES 算法在软件实现上可利用的平台很多, VB, C, C#, Java 等。比如, 在文件的输入和输出过程中完成数据流的加密和解密^[10]。在硬件实现中, AES 加密算法密码芯片执行效率提高的一个关键就是轮变换的优化。

郑行为防止国外 AES 硬件产品中可能存在的“陷门”,开展 AES 硬件实现的自主研究^[11],为了降低面积复杂度,减少资源占用,采用复合域组合逻辑来实现非线性的字节代换和逆字节代换;进行了轮单元的 7 级流水划分。在此基础上,完成了完全环展开与反馈模式下的循环迭代两种 AES 硬件实现方案。在密钥扩展方面,采用复合域算法和 7 级流水线设计,实时为 AES 提供加密轮密钥。基于消息认证,进行可用于电子商务交易系统的安全协议研究。也完成了支持协议的终端硬件设计,可实现高速数据认证、加密、解密、数字签名与完整性检测。

对 AES 加密算法的研究不论从理论上还是从实现上讲,都已经取得了一定的成果,理论意义和实践意义都较强,同时也激起了社会各界研究开发的热情。国内在实现上的研究主要集中在 IC 卡,智能卡,加密硬盘等方面。国内有爱国者和朗科等公司在移动加密硬盘和优盘有实际的应用。

2.3.1.3 AES 的主要优缺点

1. 优点:

- 1) 无论使用反馈模式还无反馈模式,在广泛的计算环境的硬件和软件实现件能都始终有着优秀的表现;
- 2) 它的密钥建立时间极短,且灵敏性良好;
- 3) 极低的内存需求使它非常适合于在存储器受限的环境中使用;
- 4) AES 是一个分组迭代密码,分组长度和密钥长度设计灵活。可变分组长度,分组长度可设定为 32 比特的任意倍数,最小值为 128 比特,最大值为 256 比特;
- 5) AES 的密钥长度比 DES 大,它也可设定为 32 比特的任意倍数,最小值为 128 比特,最大值为 256 比特,所以用穷举法是不可能破解的。在可预计的将来,如果计算机的运行速度没有根本性的提高,用穷举法破解 AES 密钥几乎不可能;
- 6) AES 算法的设计策略是宽轨迹策略(WideTrail Strategy, WTS)。WTS 是针对差分分析和线性分析提出的,可对抗差分密码分析和线性密码分析。

2. 缺点:

- 1) 关于 AES 的缺点,目前为止还未发现,并且 AES 预计将在未来几十年里代替 DES 在各个领域中得到广泛应用。总之, AES 算法汇聚了安全性、效率高、易实现性和灵活性等优点,是一种较 DES 更好的算法。

2.3.2 AES 加密算法的具体原理

AES 的关键在于加密和解密的运算资源共享和 S-Box 的设计。这两点的决定最终实现性能的关键因素。根据 AES 的算法标准，得到加解密的结构图（图 2-4），加密算法一个分组的一轮（图 2-5）和解密算法一个分组的一轮（图 2-6）

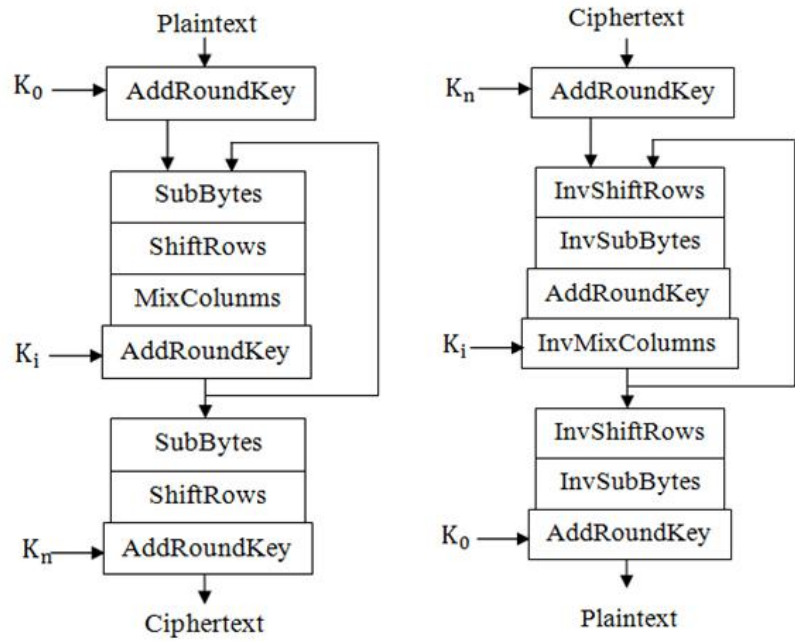


图 2-4 AES 加密（左）、解密（右）结构图

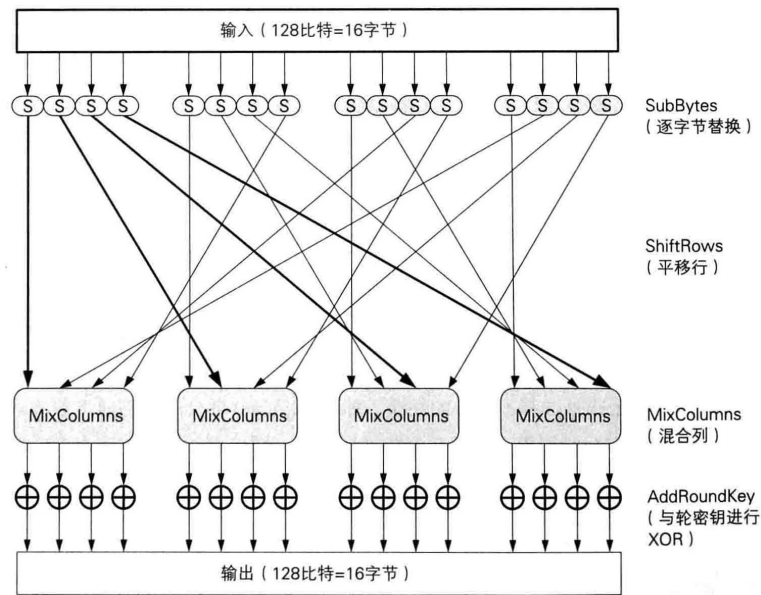


图 2-5 AES 加密算法一个分组的一轮

AES 是分组算法的一种，每个分组为 128bit，即 16 字节。加密过程首先需要逐个字节地对 16 字节的数据进行 SubBytes 处理。SubBytes 是以每个字节的值

(0-255 中任意值) 为索引，从一张有 256 个值的替换表 (S-box) 中查找出对应值的处理，相当于一个字节的值根据 S-box 替换成另一个字节的值 (类似于简单替换密码的原理)。之后进行 ShiftRows 的是将 SubBytes 的输出以字节为单位进行打乱。下一步的 MixColumns 是对一个 4 字节的值进行比特运算，将其变为另外一个 4 字节的值。最后，要将 MixColumns 的输出与轮密钥进行 AddRoundKey 处理。以上为一轮，实际上 AES 要重复上述过程 10-14 轮计算。

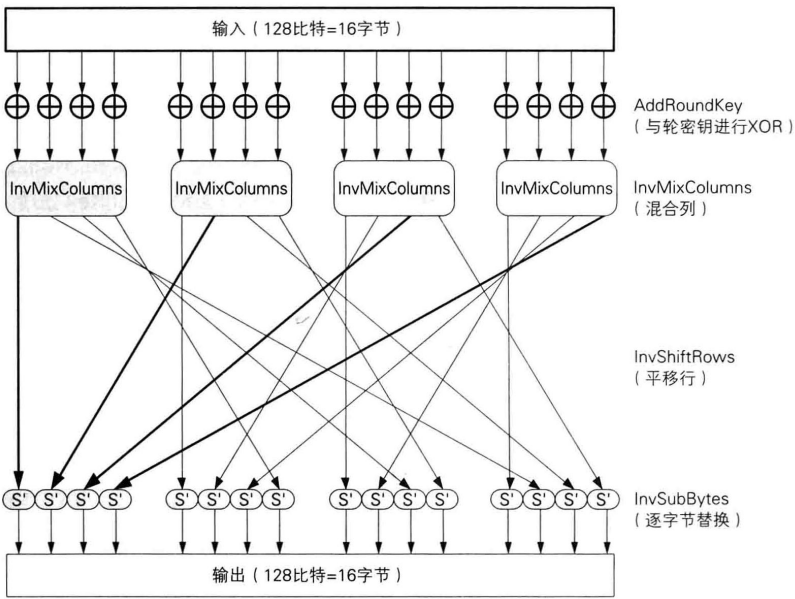


图 2-6 AES 解密算法一个分组的一轮

解密过程是加密过程的逆过程，不再赘述。

3 实验内容

3.1 环境搭建说明

考虑到生产环境的规范性，保证实验免受外部因素干扰，我们选择搭建虚拟环境。

3.1.1 选定语言或框架，搭建完整的开发环境

选择 Python 语言，并使用 Anaconda 搭建和管理虚拟环境。

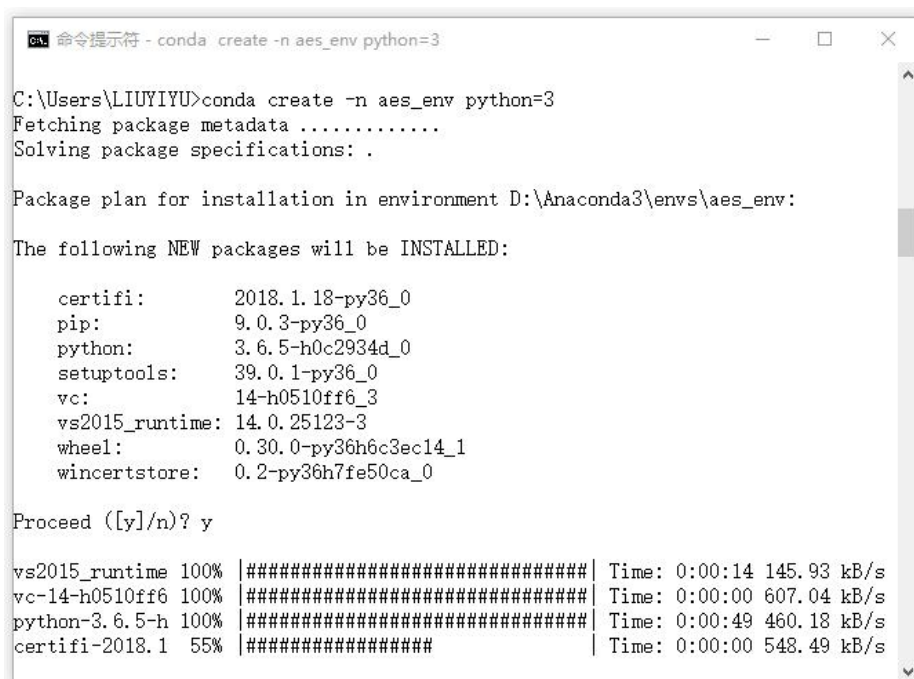
3.1.1.1 下载 Anaconda

从清华大学开源软件镜像站下载 Anaconda，安装时注意勾选配置路径或之后手动配置，直至 cmd 下输入 conda 关键字有效。

地址：<https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/>

3.1.1.2 创建新的虚拟环境

使用命令：conda create -n aes_env python=3，创建名为 aes_env 的虚拟环境，且配置 python 版本为 python3（图 3-1）。



```
命令提示符 - conda create -n aes_env python=3

C:\Users\LIUYIYU>conda create -n aes_env python=3
Fetching package metadata .....
Solving package specifications: .

Package plan for installation in environment D:\Anaconda3\envs\aes_env:

The following NEW packages will be INSTALLED:

  certifi:         2018.1.18-py36_0
  pip:             9.0.3-py36_0
  python:          3.6.5-h0c2934d_0
  setuptools:      39.0.1-py36_0
  vc:              14-h0510ff6_3
  vs2015_runtime:  14.0.25123-3
  wheel:           0.30.0-py36h6c3ec14_1
  wincertstore:    0.2-py36h7fe50ca_0

Proceed ([y]/n)? y

vs2015_runtime 100% |#####| Time: 0:00:14 145.93 kB/s
vc-14-h0510ff6 100% |#####| Time: 0:00:00 607.04 kB/s
python-3.6.5-h 100% |#####| Time: 0:00:49 460.18 kB/s
certifi-2018.1 55%  |#####| Time: 0:00:00 548.49 kB/s
```

图 3-1 创建名为 aes_env 的虚拟环境，且配置 python 版本为 python3
创建成功后，可以用 activate wechat_env 激活虚拟环境（图 3-2），也可以使用 Anaconda Navigator 在可视化界面中查看刚刚创建的虚拟环境（图 3-3）。

```
C:\Users\LIUYIYU>activate aes_env
(aes_env) C:\Users\LIUYIYU>
```

图 3-2 激活虚拟环境

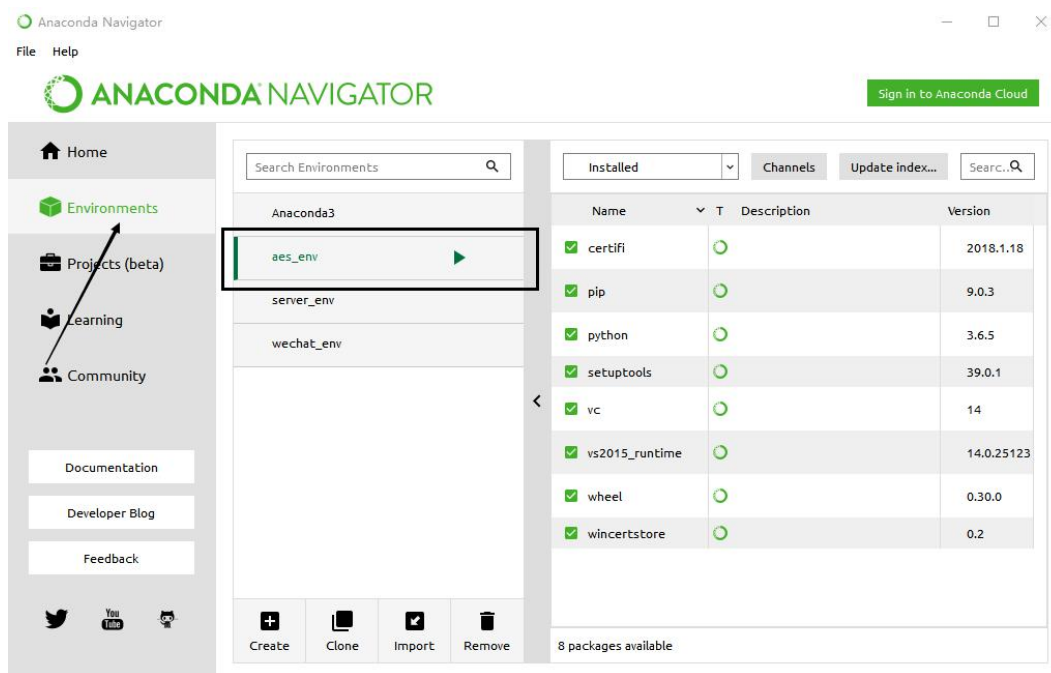


图 3-3 利用 Anaconda Navigator 查看创建的虚拟环境

3.1.1.3 安装 PyQt5

PyQt 的版本必须与 python 的版本一致，本实验使用的版本为 python3.6，PyQt5。安装 PyQt5 时会自动搜索 python 的安装路径。

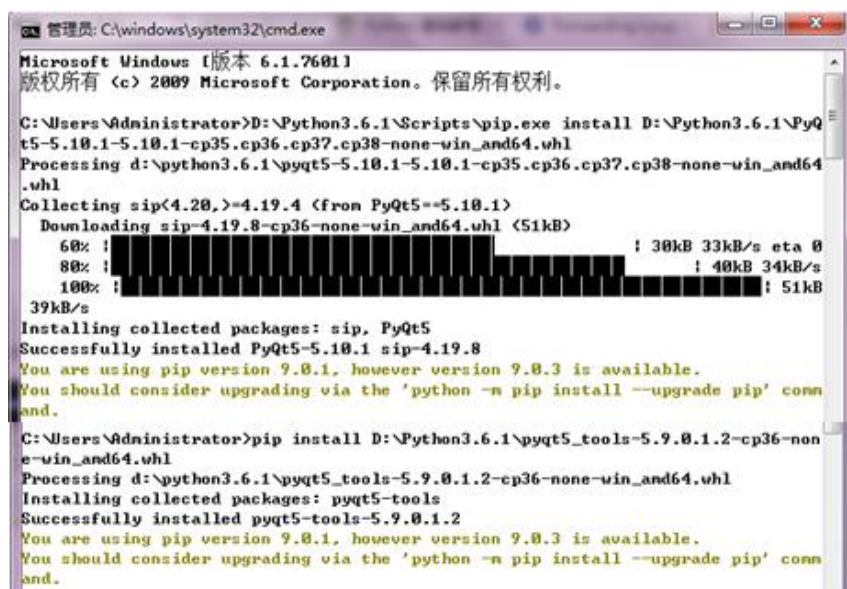


图 3-4 安装 PyQt5

可以测试一下 PyQt 的 ui 文件转换工具即可知道环境变量是否配置完成，打开命令行，输入 pyuic5，如果没有提示“不是内部或者外部命令”，说明安装成功。

3.1.2 加密算法库

我们选择使用 Cryptography 加密算法库。Cryptography 是 python 语言中非常著名的加解密库，在算法层面提供了高层次的抽象，使用起来非常简单、直观，同时还保留了各种不同算法的低级别接口，保留灵活性。Cryptography 的目标是建立一个标准 Python 加密库，支持 Python 2.7, Python 3.4+, and PyPy 5.3+。现有的 Python 密码库（如 M2Crypto, PyCrypto, or PyOpenSSL 等），虽然功能已经比较完善，市面上也有不少应用，但仍存在一些问题：

- 1) 缺少 PyPy 和 Python 3 支持；
- 2) 缺少维护；
- 3) 使用了差评的算法实现（例如旁路攻击 side-channel attacks）；
- 4) 缺少高级（易于使用）的 APIs；
- 5) 缺少 AES-GCM 和 HKDF 等算法；
- 6) 经不住测试。
- 7) APIs 不严谨，错误百出

因此我们选用的 Cryptography 密码库，Cryptography 分为两个层，方法层（cryptographic recipes）层和危险底层（cryptographic hazardous primitives，简称 hazmat）。方法层提供用于适当的对称加密，hazmat 层提供底层的加密基元。cryptographic recipes，直译为“密码学菜谱”。cryptographic primitives，即为密码学原语，也就是基本的密码学概念，如加密、签名、Hash 等算法。但是直接使用密码学原语容易出错，在实际应用中无法保证安全性。基于这一点，该库对密码学原语进行了安全集成，形成了更高层次的“密码学菜谱”。因此该库对于初学者十分友好，我们可以直接按照说明开发想要的工具。

3.1.3 在开发环境中设置对于加密算法库的引用

输入命令 `conda install cryptography`，下载 Cryptography 密码库。

```
(aes_env) C:\Users\LIUYIYU>conda install cryptography
Solving environment: done

## Package Plan ##

environment location: D:\Anaconda3\envs\aes_env

added / updated specs:
- cryptography

The following packages will be downloaded:
```

package	build	
cryptography-2.2.1	py36hfa6e2cd_0	515 KB
asn1crypto-0.24.0	py36_0	155 KB
openssl-1.0.2o	h8ea7d77_0	5.4 MB
cffi-1.11.5	py36h945400d_0	213 KB
Total:		6.2 MB

图 3-5 下载 Cryptography 密码库

```
11 from cryptography.fernet import Fernet
12 from cryptography.hazmat.backends import default_backend
13 from cryptography.hazmat.primitives import hashes
14 from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
```

图 3-6 引用 Cryptography 密码库

3.2 加密及解密程序的实现

Cryptography 密码库实现了一个集成的对称密码函数，称为 Fernet。它可以保证信息无法被篡改和破解。Fernet 模块包括了对数据的加解密以及签名验证功能，以及密钥过期机制。

该模块采用如下定义：

- 加解密算法为 AES，密钥位长 256，前 128 位用于加密，后 128 位用于签名；
- 采用 CBC 模式，填充标准 PKCS7；
- 签名算法为 SHA256 的 HMAC；
- 密钥可以设置过期时间。

3.2.1 加密解密函数

在这一部分，我们需要调用 Fernet 模块。

3.2.1.1 密钥生成

密钥可以随机生成，也可以是用户自定义的。如果使用自定义的密钥，则需

要调用 PBKDF2 函数处理密钥，它的基本原理是通过一个伪随机函数（本实验使用 HMAC 函数），把明文和一个盐值作为输入参数，然后重复进行运算，最终产生密钥。如果重复的次数足够大，破解的成本就会变得很高。而盐值的添加也会增加“彩虹表”攻击的难度。

随机生成密钥的函数为 `Fernet.generate_key()`。其算法内核是 `os.urandom(n)` 函数，这个函数的原理将在 3.3 部分具体说明。

用户自定义密钥后，需要设定 PBKDF2 算法的参数。

```
1. password = b"password"
2. salt = os.urandom(16)
3. kdf = PBKDF2HMAC(
4.     algorithm=hashes.SHA256(),
5.     length=32,
6.     salt=salt,
7.     iterations=100000,
8.     backend=default_backend()
9. )
10. key = base64.urlsafe_b64encode(kdf.derive(password))
```

3.2.1.2 初始化 Fernet

使用密钥初始化 Fernet，初始化时将检查密钥长度是否合法。具体调用方式为 `cipher_suite = Fernet(key)`。

3.2.1.3 加密

加密对应的方法是 `encrypt(data)`，`data` 为待加密字节串。加密时，首先获取当前系统时间，然后使用 `os.urandom(n)` 函数生成 128 位的初始向量，将当前系统时间和初始向量一起作为参数加密生成密文字节串。具体调用方式为 `cipher_text = encrypt(data)`。

3.2.1.4 解密

解密对应的方法是 `decrypt(data)`，`data` 为待解密字节串。解密时要检查数据格式是否为字节串，同时要检测过期时间 TTL（TTL 默认为 None），还会将加密前填充的字符删去。具体调用方式为 `plain_text = cipher_suite.decrypt(data)`，由于

解密后得到的数据格式仍为 byte 型字节串，因此要使用 `decode()` 函数将其转换为字符串。

3.2.2 设计 UI 界面

打开 Qt Designer，根据实验要求设计窗体。主要功能包括选择明文文件、密文文件、密钥文件，提取明文、密文结果显示在文本框内，导出明文、密文文件等等。



图 3-7 界面设计

3.2.3 将 ui 文件转换为 py 文件

窗体设计好后另存为文件 `AES.ui`，我们要使用 `PyUIC` 将 `.ui` 文件转化为 `py` 文件。打开命令行，输入 `pyuic5 -o D:\AES.py D:\AES.ui` 回车。其中，`-o` 后的参数为输出文件的名称 `-o` 后第二个参数为生成的 `ui` 文件的名称。

```
(aes_env) C:\Users\LIUYIYU>pyuic5 -x -o D:\AES.py D:\AES.ui
```

图 3-8 将 ui 文件转换为 py 文件的命令

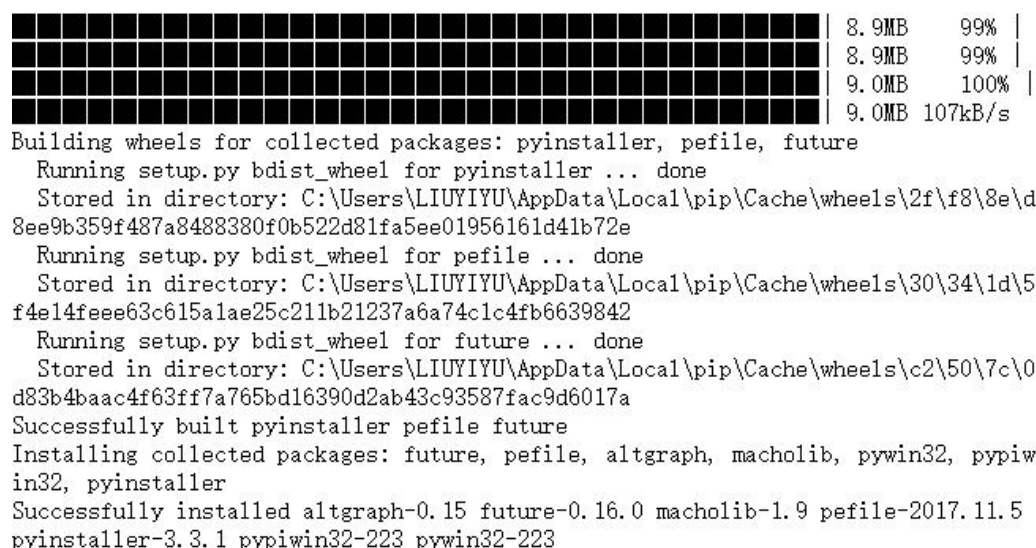
3.2.3 添加事件

最初生成的 `py` 文件只有关于窗体及窗体内各部件的描述。我们需要给各个按钮添加事件。Qt 采用信号与槽机制，槽函数可以和一个信号相连接，当这个信号发生时，它就被自动调用。连接信号与槽的是 `connect()` 函数。调用方法为

self.btnChooseEnFile.clicked.connect(self.chooseEnFile)。括号中的函数是我们需要自主实现的，主要包括弹出文件选择框、加密、解密、保存文件。

3.2.4 将单个 py 文件发布为 exe 可执行文件

将.py 文件发布为 exe, 需要 pyinstaller。首先在虚拟环境下运行命令 `pip install pyinstaller`



```
Building wheels for collected packages: pyinstaller, pefile, future
Running setup.py bdist_wheel for pyinstaller ... done
Stored in directory: C:\Users\LIUYIYU\AppData\Local\pip\Cache\wheels\2f\8e\d
8ee9b359f487a8488380f0b522d81fa5ee01956161d41b72e
Running setup.py bdist_wheel for pefile ... done
Stored in directory: C:\Users\LIUYIYU\AppData\Local\pip\Cache\wheels\30\34\1d\5
f4e14feee63c615a1ae25c211b21237a6a74c1c4fb6639842
Running setup.py bdist_wheel for future ... done
Stored in directory: C:\Users\LIUYIYU\AppData\Local\pip\Cache\wheels\c2\50\7c\0
d83b4baac4f63ff7a765bd16390d2ab43c93587fac9d6017a
Successfully built pyinstaller pefile future
Installing collected packages: future, pefile, altgraph, macholib, pywin32, pypiwin32, pyinstaller
Successfully installed altgraph-0.15 future-0.16.0 macholib-1.9 pefile-2017.11.5
pyinstaller-3.3.1 pypiwin32-223 pywin32-223
```

图 3-9 下载 pyinstaller

然后使用命令 `pyinstaller -F -w D:\AES.py`，即可生成.exe 文件。

```
(aes_env) C:\Users\LIUYIYU>pyinstaller -F -w D:\AES.py
191 INFO: PyInstaller: 3.3.1
191 INFO: Python: 3.6.5
192 INFO: Platform: Windows-10-10.0.10586-SP0
198 INFO: wrote C:\Users\LIUYIYU\AES.spec
202 INFO: UPX is not available.
205 INFO: Extending PYTHONPATH with paths
['D:\', 'C:\Users\LIUYIYU']
206 INFO: checking Analysis
207 INFO: Building Analysis because out00-Analysis.toc is non existent
208 INFO: Initializing module dependency graph...
212 INFO: Initializing module graph hooks...
215 INFO: Analyzing base_library.zip ...
```

图 3-10 生成可执行文件

3.3 测试结果及分析

3.3.1 文件选择对话框在 QMainWindow 下崩溃

为了实现读写文件功能，我们需要在窗体中添加一个按钮，点击后弹出文件选择对话框。在查阅相关文档和示例代码后，我整合了相关函数，但是在主窗体 (QMainWindow 类) 下运行会发生异常，程序崩溃。我在 stack overflow 上看到了类似的问题，找到了出错的原因。

I try to use QFileDialog for choose a file on PyQt but it don't run



图 3-11 stack overflow 关于该问题的说明

这里涉及到 Qt 常用窗体 QWidget、QDialog、QMainWindow 之间的关系。QWidget 类是所有用户界面对象的基类。QWidget 继承于 QObject 和 QPaintDevice，QDialog 和 QMainWindow 则继承于 QWidget，QDialog、QMainWindow 两者之间没有直接关系。

本程序的主窗体是 QMainWindow 类，令程序崩溃的语句是

```
1. fname, ftype = QFileDialog.getOpenFileName(self, "选择文件")
```

其中第一个参数表明了父窗体，在这里指 QFileDialog 是基于 QMainWindow 弹出的，其实二者并无直接关系。因此这条语句应将第一个参数改为 None，即 QFileDialog 的父窗体应为空。如果主窗体是 QWidget 类，则 QFileDialog 的父窗体参数既可以使用 self，也可以使用 None。

```
2. fname, ftype = QFileDialog.getOpenFileName(Nones, "选择文件")
```

3.3.2 基于 cryptography 库中 hazmat 层提供的底层加密基元写成的 AES+CBC 加密算法可用性不强

实验初期,我们选择 cryptography 加密库后实现了一个简单的 AES+CBC 加解密程序,为保证足够的安全性,密钥是随机生成的,若使用自定义的密钥,则不可避免地会遇到彩虹表攻击的风险,安全性大大降低。要提高安全性,则需要用到某种 HASH 算法,实现较为复杂,不如选用 Cryptography 密码库集成的对称密码函数 Fernet, Fernet 使用起来更方便也更安全。

3.3.3 加密解密需要一次完成

实验中我们测试到 Fernet 的一个缺陷:若加密后先关闭程序,然后重新运行程序进行解密,则会出现解密失败的情况,这是由于程序每次运行都会随机生成一个新的盐值(用于 PBKDF2 算法导出密钥)。因此,即使密钥不变,最后初始化的 Fernet 模块也会不同,因此无法得到解密结果。

Fernet 使用 PBKDF2 算法处理密钥其实是基于安全性考虑的。如果每次哈希加密都使用相同的盐值,那么这样加盐的方式是做无用功,因为两个相同的密码会得到相同的哈希值。攻击者可以使用反向查表法对每个值进行字典攻击,只需要把盐值应用到每个猜测的密码上再进行哈希即可。

但是每次生成新的盐值的方法又对一些小型应用不够友好。针对这一问题的解决办法是储存每次加密的盐值,这种方法在服务端已经有类似的应用,但是随之又产生了一些新问题,比如保护盐值也要浪费一定的资源,同时也降低了整体的安全性。在一些不需要与服务端交互的本地应用上,只能将盐值硬编码到软件中。这也有一定的风险,因为攻击者可以专门为这个软件制作查询表和彩虹表,那么破解它生成的哈希值就变得很简单了。

3.3.4 关于 os.urandom(n)函数

生成随机密钥和随机向量的核心是 os.urandom(n) 函数,它是标准库提供的产生随机字串的函数,这个函数将会返回一个适用于加密的 n 个字节长度的(由参数 n 指定)随机字符串(而非字符串,因为可能结果并不能被任何一种字符编码所解释,所以不能称其为字符串)

这个函数将会调用系统提供的随机源，依赖于程序运行时的当前系统，在 Windows 上会使用 `CryptGenRandom()`，而在类 UNIX 系统上将会使用 `/dev/urandom`。如果当前系统没有提供随机源，那么将会抛出一个 `NotImplementedError`。

3.3.5 测试结果

我们最终的实验程序导出成了 .exe 可执行文件（图 3-10）。

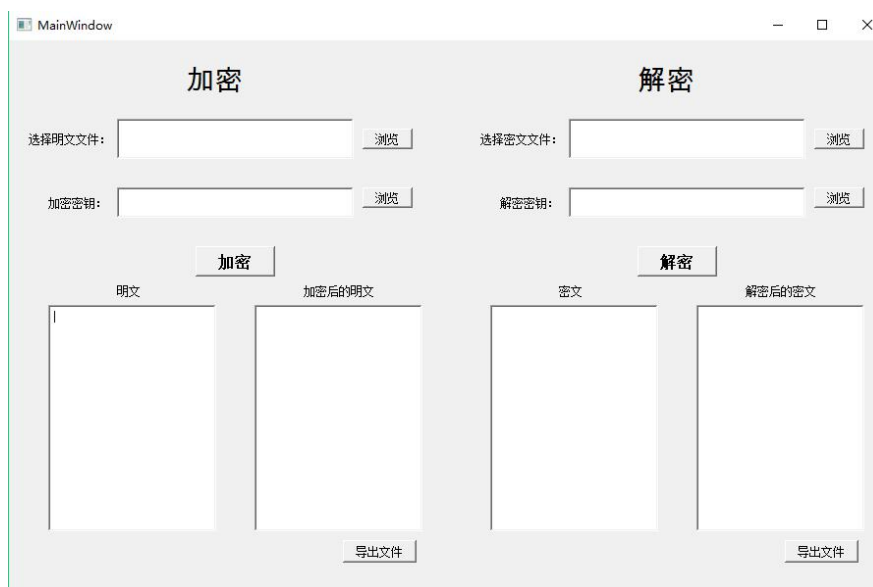


图 3-12 程序界面

对程序进行测试，我们一共用到了三个密钥。其中，密钥 `password1` 和 `password2` 较为复杂，两个密钥仅有一处不同，`password1` 的倒数第 6 个字符为小写字母 `l`，`password2` 的倒数第 6 个字符为数字 `1`。而 `password3` 则较为简单且较有规律，分别如图 3-11、3-12、3-13 所示。



图 3-13 密钥 password1



图 3-14 密钥 password2



图 3-15 密钥 password3

我们一共对 3 组明文进行了测试，每一组中的明文差异较小，而各个组的明文差异较大。明文 plaintext2 只将 plaintext1 中的中文数字“九”改成了阿拉伯数字“9”；明文 plaintext4 只将 plaintext3 中的中文冒号“：”改成了英文冒号“:”；明文 plaintext6 是 plaintext5 0 和 1 取反的形式。

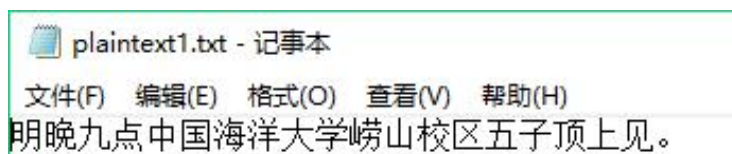


图 3-16 明文 plaintext1

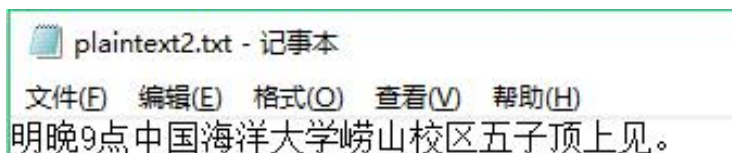


图 3-17 明文 plaintext2

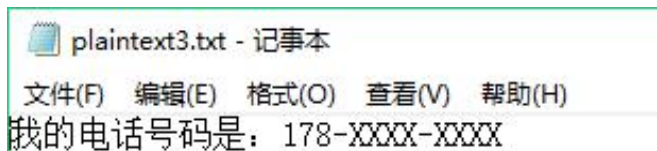


图 3-18 明文 plaintext3

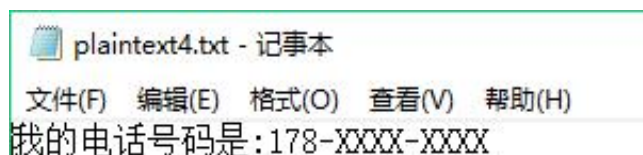


图 3-19 明文 plaintext4

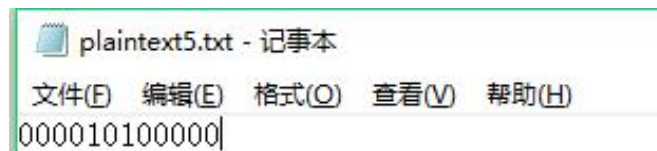


图 3-20 明文 plaintext5

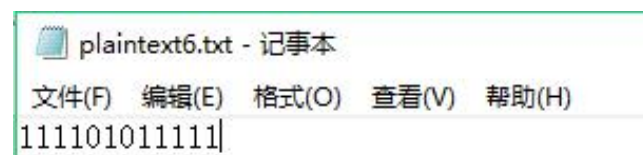


图 3-21 明文 plaintext6

1. 第一次测试，我们的明文选择 plaintext1，密钥选择 password1 进行加密。我们将加密后的明文导出为 ciphertext1.txt 文件，再选择该密文文件，密钥选择 password1 进行解密。成功解密。

加密后的明文为：

gAAAAABazKVfo6vcxD5cKRnswCrzl0QUbnpaEQqvFNiHIH1bdfdVPof5m9sPJs_4
7EQ89vYXdgN-HrXvk2VG3rvxyx5xdjSGLvq15_K5vDboRsb-UbZ40Uk797GknKl
pCrcQwkP5L1Y6ngQVpUKiZx2z2vUGp6ExAw==



图 3-22 第一次测试结果

2. 第二次测试，我们的明文选择 plaintext2，密钥仍选择 password1 进行加密。
- 我们将加密后的明文导出为 ciphertext2.txt 文件，再选择该密文文件，密钥选择 password1 进行解密。成功解密。

加密后的明文为：

```
gAAAAABazKYTkSZ8zwb9rh8ncLpR-LtIu203XcHIJ5lVqX3x3D3PQdwgGFcpGN
G7LxRrhvBpsERCBh_hJPpre5PY_u36TyTeF9RjOOuLh0xB5WmoZ0egvyBYROz8
Eu8CaWWNyOimAqIPmBExD8OISBveLVtb3PSFKg==
```



图 3-23 第二次测试结果

3. 第三次测试，我们的明文选择 plaintext1，密钥选择 password2 进行加密。我们将加密后的明文导出为 ciphertext3.txt 文件，再选择该密文文件，密钥选择 password1 进行解密。会提示密钥错误，解密失败。

加密后的明文为：

gAAAAABazKc81Yeqhj8m1fT-j-kFE-JUacJIFXvGN0TqofR0PigOHq7uZyeYPFQz
Wz5cgU-y0jebJBnElr6BZ44h27dSXbAMTLk4Vi_453KcxN41o20EbNEYCbTrbWp
T1sSTuop5uD7vBNwuUaXzxQHSPfaLvQcw==



图 3-24 第三次测试结果

4. 第四次测试，我们的明文选择 plaintext1，密钥选择 password3 进行加密。我们将加密后的明文导出为 ciphertext4.txt 文件，再选择该密文文件，密钥选择 password1 进行解密。成功解密。

加密后的明文为：

```
gAAAAABazKjFonlrzAMZVKQnuxXpgX0yMyhRHbZHXCE2CbLyanZewurlC1g  
vyUIBq-9cSQMD3VnP5zNa_jxhwwiCfQLFtdz2xfgxDDwNTswlTwBprB8WQTzpX  
dg8qzWc_Au1VuF3Di9u136_NgPUBSRWO8awhjLFoQ==
```



图 3-25 第四次测试结果

5. 第五次测试，我们选择的明文与密钥与第四次测试一致，只是再次点击了加密按钮，并将加密后的明文导出为 ciphertext5.txt 文件，进行解密。

加密后的明文为：

gAAAAABazKoKRNH29RbkigXsnAsuvP6n0V-6RhxD-1A80mF7TqNcYaDQA13V
xcKIetC4bJYlG_JRMA809QjIqxvymAOuIDQQ9Q8VIRbAWRE0KDcPh6dMDMZg
cOHGoskLjixU4PExdb6OIbIHU-q0xfyzYQiDjM-FXQ==



图 3-26 第五次测试结果

6. 第六次测试，我们的明文选择 plaintext3，密钥选择 password2 进行加密。我们将加密后的明文导出为 ciphertext6.txt 文件，再选择该密文文件，密钥选择 password2 进行解密。成功解密。

加密后的明文为：

gAAAAABazKsG1Us8shVoAo26lpywzvukMJsa1yt1MRF6iVcKDbLRPN5MXesrRt
VkdA301ntMjTRqTpMnBqaDUH5s3zmM3ZS1J8dDHmg40JnHMOseQvbl3aGoRP
Bg-6zuqRCp3GdjBAa_



图 3-27 第六次测试结果

7. 第七次测试，我们的明文选择 plaintext4，密钥仍选择 password2 进行加密。
我们将加密后的明文导出为 ciphertext7.txt 文件，再选择该密文文件，密钥选择 password2 进行解密。成功解密。

加密后的明文为：

gAAAAABazKw-Ni6tAa_PNHxTUshnXzGWWEFANbxe-1DSIUGIYgY5XVas6y8
qlWJWGhpbhn3ZUwx7cxgaH01YI2zCXiJG6Jjkwgbri3_wVpS_8--lra9Nye-nbnJi1vq
DeKBxxM-jUK1a



图 3-28 第七次测试结果

8. 第八次测试，我们的明文选择 plaintext5，密钥选择 password3 进行加密。我们将加密后的明文导出为 ciphertext8.txt 文件，再选择该密文文件，密钥选择 password3 进行解密。成功解密。

加密后的明文为：

gAAAAABazK3gSAMD4l_cftL419vgIrEUrxDnJGYwx4F39H5fyXzpjROQ-brewH
ENkOTzOI-vU-WVT6AZKpfnJ-T-pM06iv46g==



图 3-29 第八次测试结果

9. 第九次测试，我们的明文选择 plaintext6，密钥选择 password3 进行加密。我们将加密后的明文导出为 ciphertext9.txt 文件，再选择该密文文件，密钥选择 password3 进行解密。成功解密。

加密后的明文为：

gAAAAABazK475pRFLbdKGxtUHLQSSQNcPgc15J3SCYOMrffskgVuwQrMFz_d
HoC-K1pfAu94W9_tOVpolQUUrlFmu-F4OR4lHQ==

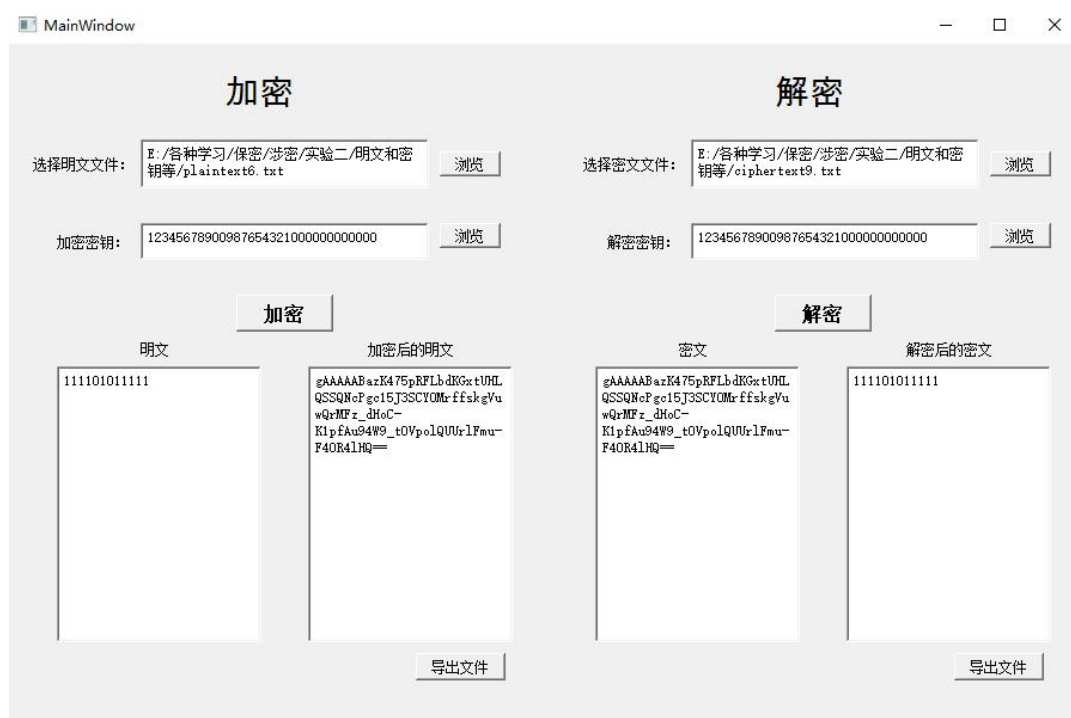


图 3-30 第九次测试结果

3.3.6 分析

根据以上的测试结果，我们可以得出以下几个结论：

1. 明文越长，加密得出的密文也越长；
2. 根据测试三，当加密和解密的密钥不同时，不能正确解密。我们在程序中添加了抛出异常处理，否则当加密和解密密钥不同时，解密会导致程序出错终止，优化程序后，会进行解密错误提示；
3. 以上 9 次测试结果，加密后的明文的前几位是相同的，都是“gAAAAABazK”，在其他同学电脑上进行测试，前几位相同的字符是“gAAAAABay”，说明前八位的字符“gAAAAABa”一直不变，后面 1-2 位字符会有区别，这是加密函数本身的问题；

4. 我们的加密函数使用了 base64 编码方式, 所以加密后的密文都是由字母和数字组成, 且一般情况下最后会出现两个等号 “==”;
5. 根据测试 1 和测试 2 的比较, 或测试 6 和测试 7 的比较, 当明文只有微小差距, 密钥相同时, 加密得到的密文完全不一致;
6. 根据测试 1 和测试 4 的比较, 当明文一致, 而密钥不同, 且其中一个为简单密钥时, 加密得到的密文也完全不一致;
7. 根据测试 4 和测试 5 的比较, 即使明文和密钥完全一致, 两次加密的结果也完全不一致, 这是因为每次加密时, 初始向量 (图 3-29 中 iv) 都会发生改变^[12];

```
48     def encrypt(self, data):
49         current_time = int(time.time())
50         iv = os.urandom(16)
51         return self._encrypt_from_parts(data, current_time, iv)
```

图 3-31 程序中 cryptography 库 encrypt 函数的具体实现

8. 根据测试 8 和测试 9 的比较, 当两个明文有相应的对应规律时, 即使密钥相同, 加密得到的结果也没有任何联系;
9. 根据测试 4、8、9, 即使我们使用的密钥具有规律且较为简单时, 得到的密文也很复杂, 没有规律可言。

参考文献

- [1] 郑东, 李祥学, 黄征等. 密码学——密码算法与协议[M]. 北京: 电子工业出版社, 2009.
- [2] 彭长根. 现代密码学趣味之旅[M]. 北京: 金城出版社, 2015.
- [3] PAAR C, PELZL J. 深入浅出密码学——常用加密技术原理与应用[M]. 北京: 清华大学出版社, 2012.
- [4] 戚文峰. 序列密码发展现状[J]. 北京: 中国计算机学会通讯, 2007, 3(09).
- [5] 刁晓红. 信息加密技术在电子商务中的应用[J]. 北京: 中国管理信息化, 2005(04): 44-47.
- [6] 黄少青. RC4 算法的安全性分析[D]. 北京: 北京邮电大学, 2009.
- [7] Golic. Linear statistical weakness of alleged RC4 keystream generator. In EUROCRYPT: Advances in Cryptology: Proceedings of EUROPCRYPT 1997.
- [8] 张羊羊. 基于 AES 的伪随机数生成器的设计与实现[D]. 南京: 南京师范大学, 2017.
- [9] 陈在平, 蔡鹏飞, 董恩增. 基于超混沌 AES 图像加密算法[J]. 吉林: 吉林大学学报(信息科学版), 2013, 31(02): 158-164.
- [10] 赵雪梅. AES 加密算法的实现及应用[J]. 常熟: 常熟理工学院学报, 2010, 24(02): 105-110.
- [11] 郑行. AES 算法的硬件优化实现及应用研究[D]. 厦门: 厦门大学, 2014.
- [12] Chris Wolfe. fernet.py[CP]. American: Paul Kehrer, 2017-10-19[2018-4-10].
<https://github.com/pyca/cryptography/blob/master/src/cryptography/fernet.py>.