应用密码学（CSE 539） MCS 作品集报告

美国*亚利桑那州立大学*坦佩分校 Chirag Bhansali
cbhansal@asu.edu

I. 导言

这是 2019 年秋季学期选修的应用密码学（CSE 539）课程的作品集报告。在这门课程中，我们必须完成 6 个项目/作业，每个项目/作业都要求我们使用常规课堂上讲授的主题和概念。

在项目 1 中，我们需要创建一种新的/修改过的算法，并使用 C、C++ 或 Java 实现该算法，使用 S-box（替换）和 P-box（置换）对 32 位的明文数据块（txt、pdf 或 jpeg 格式）进行加密和解密，从而模糊明文和密码之间的关系，并使用蛮力技术尝试找出用于进行转换的密钥，前提是我们知道明文和密码。我们还需要记录蛮力技术找到密钥所需的时间，并评论所用时间与文件类型之间是否存在任何关系，以及任何可能导致更快破解加密的弱点。

在项目 2 中，我们获得了用 C 语言编写的文件加密和解密代码，我们要对加密算法的设计、是否是自包含算法发表评论，并在 只知道要解密的文件类型（jpg、pdf、txt）的情况下，使用两种不同的技术（暴力法和密码分析法）来发现加密过程中使用的密钥，报告加密过程中的弱点、可能采取的修复步骤，并记录两种技术的运行时间以进行比较。

项目 3 给我们提供了 10 个哈希值，我们必须在密码未加盐、为字母数字且长度为 4 个字符的情况下获取密码。我们被要求使用黑客常用的两种方法来破解密码哈希值，即暴力破解和彩虹表攻击。我们要为上述两种技术各编写一个程序，记录它们的运行时间，并对创建彩虹表所涉及的实现步骤、大小、所使用的还原函数以及破解哈希值所用方法中的任何不足之处发表评论。

在项目 4 中，我们被要求使用项目 2 中的加密算法和项目 3 中的散列算法来创建 MAC（消息验证码）、

用于维护信息的完整性和真实性。如果入侵者试图修改信息，整个数据的哈希值就会发生变化，接收者就能知道数据的真假，因为发送的哈希值和使用 MAC 程序计算的哈希值并不相似。我们要编写的是 C 语言程序，命令行需要两个输入，即包含信息的文件路径和用于加密的密钥。

在项目 5 中，我们使用 CA（证书颁发机构）颁发的数字证书。使用 Python3 密码学库，我们必须获取主体的公钥和模数，以便双方进行安全通信；验证主体证书是否有效；从根证书和主体证书中提取主体名称、序列号、加密算法、公钥模数、公钥指数、私钥指数等信息。最后，我们必须使用主体证书中的公钥加密信息，并使用主体的私钥解密验证。

在最后一个项目中，我们开发了一个 CPP 程序，用于实现 "挑战响应 "方案（该方案用于验证被挑战方是否知道特定挑战的正确答案）、通过向构造函数提供参数来测试给定的 RSA 类，以 及 "盲签名"（发送方从接收方处获取已签名的文件，但不透露接收方的身份）。我们将提供执行上述每种技术的步骤。

II. 解决方案说明

A. 项目 1

在加密部分，我们修改了 Vigenere 密码，这是一个使用密钥和双输入表的多字母替换系统，但我们的密码不使用字母，而是使用十六进制字符，这样就减少了输入表的大小。然后，我们通过重复使用密钥，使密钥和信息长度相 同 ， 直到两者长度相同。在经过替换循环（将明文和密钥相加，然后按字符顺序取模 16）后，中间密码会经过 3 个不同的 P-Box，以使密码更加模糊，并且难以找到密码和明文之间的模式。

解密时，我们通过 3 个 P-Boxes 密码，然后通过修改后的维基解密，将密钥和密文值相减，再按字符顺序取一个模，最后得到明文。

提出算法后，我们将其逻辑转化为 CPP 语言程序，用于加密和解密。在暴力破解部分，我们修改了解密代码，由于我 们 知道明文和密文，所以一开始密钥为 0x00，然后以 1 为单位递增，直到解密结果和明文的值不一样为止。

B. 项目 2

我们对加密程序代码进行了密码分析，仔细研究了其中的每一个步骤，了解了这些步骤的目的，以及它们是如何将明文和密钥转化为密文的。我们还分析了所提供的 MD5（信息加密）代码的工作原理，以了解其目的。在分析过程中，我们发现它并不是一个自包含的代码，它 "作弊 "的方式是不使用 MD5 来加密明文。

在暴力破解过程中，我们修改了解密代码，将密钥值设为 0x00，并以 1 为单位递增，直到密文前 4 个字符与密钥的 XOR 值没有产生与我们试 图 找到的解密版本相匹配的特定十六进制文件签名。对于密码分析的结果，我们只需创建一个程序，在密文的前 4 个字符和特定的十六进制文件签名之间进行 XOR 运算，即可得到所需的密钥值。

C. 项目 3

我们首先生成所有 32 位长字母数字密码，并将其存储在一个文件中。为了进行暴力破解，我们读取了包含密码的文件，逐个读取每个密码，使用项目说明中提供的散列函数，将其与给出的散列值进行比较，并打印出找到匹配的密码。

对于彩虹表，我们选择的表大小为 62 * 238328，即选择 62 个不同的密码，使用散列和还原函数运行链式过程 238328 次，然后将结果散列和密码存储在一个文件中。然后，我们将给出的哈希值逐一使用还原函数和哈希函数，最多运行 238328 次，直到生成的中间哈希值与我们存储的 62 个哈希值中的任何一个不匹配为止。一旦找到匹配值，就会提取存储的密码，然后进行链式处理，直到找不到给定的哈希值为止，这样我们得到的给定哈希值的密码就是我们想要的结果。

D. 项目 4

在创建 mac.c 文件时，如说明所述，我们首先从命令行接收 1024 字节文件的路径和密钥作为参数。我们首先使用加密命令对给定的 1024 字节文件数据进行加密。

我们使用项目 2 中的哈希函数，使用 32 位长密钥，创建了一个 1028 字节的结果，然后删除前 4 个字节，将其与密钥集中在一起，并使用项目 3 中的哈希函数对集中后的值进行哈希运算，得到结果 MAC。

E. 项目 5

为了创建 Python 文件，我们创建了一些函数，用于从给定的主体和根证书以及主体的私钥中收集和显示各种要求的信息，我们使用了 Python3 密码学库，如 `load_certificate`、`get_signature_algorithm()`、`get_notAfter()`、`get_notBefore()`、`load_pem_public_key`, `dump_publickey` 等库来获取所需的信息。我们还使用 `encrypt()` 对信息进行了加密，并使用 `decrypt()` 函数对密码进行了解密。

F. 项目 6

在 CPP 文件中，我们按照 RSA 的说明创建了随项目提供的 RSA 类实例，通过向构造函数提供 0 或 1 或 2 个参数来检查加密和解密函数，并通过向构造函数提供非质数作为参数来检查是否只对质数有效。

在 "挑战-回应 "方案中，我们创建了两个 RSA 类实例，并将其中一个实例的公钥和模数分配给另一个实例。生成随机信息，使用公钥加密，使用私钥解密，并检查随机数和解密值是否相似。

对于 "盲签名"，我们按照项目说明中给出的步骤，创建了 RSA 类的 Alice 和 Bob 实例，它们相互通信，Alice 发送信息，由 Bob 签名，但 Bob 不知道 Alice 的身份。

III. 结果描述

A. 项目 1

借助修改后的维基内尔密码，我们获得了由于密钥和移位变化而导致的同一字符的不同值，从而使其能够抵御基于频率分析的攻击；借助 3 个 PBox，我们缓解了密钥长度预测问题，并使密码与明文相比尽可能模糊。

我们发现文件类型与算法之间没有关系，所有 3 种类型的暴力破解所需的时间相同。该算法的一个弱点是密钥长度较小，因此暴力破解过程花费的时间较少，解密也更容易。

B. 项目 2

首先，我们发现加密过程没有使用项目提供的 MD5 算法来处理明文，而是对密钥进行加密并生成密文
程序在明文和密钥之间做 XOR 运 算 。

我们发现，通过密码分析所花费的时间远远少于暴力破解所花费的时间，而且对于不同的文件，有不同的文件签名，这些签名对于解密过程和获得所需的解密文件非常有用， 因此每种文件格式获得解密文件所花费的时间都不同。为了消除这一弱点，我们建议对明文而不是密钥进行 MD5 运算，或者使用比 XOR 运算产生更复杂结果的运算。

C. 项目 3

在这个项目中，我们在进行暴力破解时找到了给定哈希值的所有密码，但在彩虹表攻击中，我们没有找到其中一个给定哈希值的密码。暴力破解的执行时间几乎是彩虹表的 10 倍，而且我们发现，对于 4 个字符长的小密码，计算结果的时间不到 2 分钟，因此，密码长度必须大于 6 个字符，这两种攻击才不可行。

D. 项目 4

在这个项目中，我们收到的结果是信息的 MAC 值，它是由包含密钥和加密信息的浓缩字符串的哈希值形成的。

E. 项目 5

我们使用各种 python3 库验证了子对象的证书，并打印了有关证书的各种数据，如主体名称、发行者、序列号、加密算法、之前无效、之后无效、公钥模数（n）、公钥和私钥指数、十六进制签名。最后，我们使用主体的公钥加密了一条信息，解密时使用了主体的私钥。

F. 项目 6

我们可以看到，加密和解密函数只有在质数作为构造函数的参数时才能工作。在盲签名中，爱丽丝用公钥加密从鲍勃处获得的随机数，并与爱丽丝想要签名的信息相乘，然后发送给鲍勃，鲍勃解密后发送给爱丽丝，爱丽丝将其与随机数的倒数相乘，得到想要签名的信息。

IV. 我的贡献说明

A. 项目 1

在这个项目中，我负责创建 CPP 程序和算法，设计用于加密和解密过程的改良维基解密器和 3 个 Pbox，以及报告撰写部分。

B. 项目 2

修改用于获取所需输出密钥的解密代码，撰写报告并记录暴力分析和密码分析的输出结果，同时建议使用比 XOR 运算结果更模糊的更强算法。

C. 项目 3

我想出了适当的还原函数，这有助于覆盖大部分哈希值，并对哈希函数进行代码修改，以便在暴力破解情况下读取文件，并将计算出的哈希值与给定的哈希值进行比较，在彩虹表情况下，进行链式处理，找出中间哈希值是否与 62 个存储哈希值匹配。

D. 项目 4

创建 C 语言程序，使用项目 2 的加密函数对文件中的数据进行加密，然后将结果字符串和密钥的组合值发送到哈希函数，并获得信息的 MAC。

E. 项目 5

研究要使用的所有库、它们的实施示例以及如何在项目中使用它们。为每个部分创建了不同的 python 函数，并编写了如何使用和从程序中收集必要信息的自述。

F. 项目 6

创建 CPP 程序，为每个部分创建 RSA 类实例，并创建方法来检查解密函数和纯文本（随机数）的结果是否相似。创建挑战-响应和盲签名函数，创建显示 Alice 和 Bob 之间通信的单独函数。

V. 掌握新技能

在整个课程和项目执行过程中，我学到了许多新技能和技巧，这些技能和技巧对于完成本课程和在实际场景中使用非常有用。

1) 加密和解密过程、加密和解密步骤、S-Box 和 P-Box 的使用、使用它们创建可逆解密算法、目的

最后是蛮力，即在我们知道算法和明文并试图找到密钥的情况下，破译密码信息所需的时间和精力。

2) 蛮力和密码分析技术之间的区别，为什么蛮力可能无效，以及如何分析代码、密码文本和明文以发现其中的模式，并利用这些模式找 出 密码算法中的缺陷和瑕疵，加以利用，从而获得用于通信的对称密钥。
3) 彩虹表，大多数黑客用来破解密码的哈希值的技术。了解彩虹表的创建和使用，以及暴力破解和彩虹表所需时间和内存的关键区别，以及为彩虹表选择正确的还原函数的重要性，以便最大限度地覆盖密码及其哈希值。
4) MAC（信息验证码），它的重要性，以及它如何通过使用以前学过的哈希算法和信息加密技术来帮助维护信息的完整性和真实性。
5) Python3 库及其实现，用于验证给定根证书和主体证书的数字证书，并从中提取信息。同时使用 PKI（公钥基础设施），即用于加密和解密信息的公钥和私钥。
6) 用于安全传输数据的 RSA 加密技术，用于身份验证的挑战回应技术。还有盲签名，用 于 在不认识发件人的情况下获取某人的签名。

VI. 团队成员

所有项目都是两人一组完成的，另一位组员是 Shivank Tiwari。

参考资料

[1] Tutorialspoint.com. 公共 密钥 加密 - Tutorialspoint. [在线] 可登录 https://www.tutorialspoint.com/cryptography/public_key_encryption.htm

[2] Tutorialspoint.com. 费斯特尔 区块 密码 - Tutorialspoint. [在线] 可登录 https://www.tutorialspoint.com/cryptography/feistel_block_cipher.htm

[3] Pyopenssl.org. crypto - 通用加密模块 - pyOpenSSL 19.1.0 文档。[在线] 网址： https://pyopenssl.org/en/stable/api/crypto.html

[4] X.509 参考资料。网址： https://cryptography.io/en/latest/x509/reference/

[5] En.wikipedia.org. 盲人 签名。 [在线] 可登录 网址： https://en.wikipedia.org/wiki/Blind_signature