现代密码学笔记

周康韫

April 8, 2024

Contents

1 第二讲　公钥密码学：RSA 3

2 第三讲　公钥密码学：Diffie-Hellman 6

3 第四讲　公钥密码学：Signature 10

本笔记内容来自丁津泰教授《现代密码学（英)》课程。由于授课完全使用英文，笔者可能出现翻译上的错漏，烦请读者批评指正。

1 第二讲 公钥密码学：RSA

在密码学中，我们有一个重要的假设：给定一个加密系统，我们总是知道它是如何设计的。

在对称密码中，我们的核心议题是试图找到尽量“随机”的置换。从恩尼格玛机器到 AES，这个基本的思路从未改变过。

在 1960 年之前，密码主要由政府机构使用，密码本、密钥交换并不是一个很大的问题。但在六十年代之后，私人计算机出现，我们开始有了一个极其巨大的计算机网络，在这样的情况下，我们又如何才能进行密钥分发?

Steven Levy 的《Crypto》介绍了有关现代密码学出现的历史。推荐大家去看一看。

公钥密码学的思想最早由迪菲 · 赫尔曼给出。他并不是非常优秀的数学家，提出了公钥加密的思路，却不知道如何去实现。最早的公钥加密方案由 R、S、A 三人给出，一个是工程师，一个是计算机学家，一个是数学家。姚期智先生从 MIT 辞职时，Shamir 得到了他的职位，从而和 MIT 的另外两名教授（R 是他的老朋友）见面。有一天 Shamir 在图书馆中发现了迪菲的论文，告诉 R 我们应该试着做成这件事情——他们又找到 Adleman 合作。

Shamir 在论文的最后他留下了一句话：如果读者期望得到更多信息，可以写邮件给我（当时还没有电子邮件）。他把论文发表了之后就出去度假了——当他回到大学，发现自己的办公室里已经堆满了数千封邮件。他此时终于意识到自己做出了多么伟大的发明。

下面我们来系统地介绍公钥密码学。首先介绍 RSA。

假设小明希望使用 RSA 系统来进行加密通讯。他首先需要选取两个很大的质数 p, q（长度至少为 1024 比特），然后取 $n = pq$。接下来，他还需要找到一个 e，满足：

$$\gcd(e, \phi(n)) = \gcd(e, (p-1)(q-1))) = 1.$$

小明将公开他的公钥：(n, e)。他还将藏好自己的私钥:d，满足 $de = 1 \mod \phi(n)$。

在 RSA 加密方案中，明文空间为 Z_n；加密函数为：

$$m \to m^e \mod n = c \in Z_n.$$

这个加密过程是可以被持有公钥的任何人完成的——只要知道整数 n, e 就可以。接下来，得到密文 $c = m^e$ 的小明只需要计算：

$$c^d = m \mod n$$

即可得到信息 m。

我们计算一个具体的例子。选取两个质数为 $5, 11$，取 $e = 3$ 与 $\phi(n) = 40$ 互素。公钥对为 $(55, 3)$；私钥为 27。假设密文是 31，加密得到 $31^3 = 36$；解密只需计算 36^{27}。我们并不需要计算 27 次乘法——我们只需要使用快速幂算法。

$$36^{27} = (36)^{2^4} \times (36)^{2^3} \times 36^2 \times 36.$$

需要特别注意的是，当 $\gcd(m, n) > 1$ 时，解密函数并不能得到 m，但这件事发生的概率可以被认为是 0.（使用 1024 位的密钥，概率约为 10^{-300}，这可以被认为是不可能事件。）

下面我们讨论 RSA 系统的安全性。十分显然的是，如果知道 $\phi(n) = (p-1)(q-1)$，则我们可以计算出 p, q（反过来也一样）。如果我们能对大整数做素因子分解，我们可以知道 p, q，也就破解了这个加密方案。反过来，如果我们能有效地计算出 d，我们也能找到对 n 的分解（这一点不那么容易看出）。但是，值得强调的是，我们并不一定需要找到私钥才能破解密文（我们尚未证明或者证伪这件事）。因此，破解 RSA 并不等价于大整数分解问题。

最后，我们讨论如何找出两个大素数 p。根据素数定理：

$$\lim_{N\to\infty} \frac{\pi(N)}{N/\log N} = 1.$$

因此，如果我们使用长度约为 $exp(300)$ 的素数，在大奇数中随便取一个，总有约 1/150 的概率找到素数。我们只需要随便取一些数，然后检验它们是否为素数——如果是，我们就很高兴地采用它们。同样地，尽管我们有多项式时间内判定任何整数是否为素数的方式，我们并不会采用它（耗时太长，不经济）。我们可以采用费马检验法：

$$y^{x-1} \neq 1 \mod x \Rightarrow x \in \mathbb{Z}/\mathbb{P}.$$

我们还有成功率更高（可以计算成功率）的素性检验法（Miller-Robin test），在实践中我们一般采用这些方法。

下一个有关安全性的问题是，什么样的素数是安全的？我们可以给出一些不安全的素数的例子：如果 p, q 都是梅森素数，那么我们可以很轻松地破解这个加密方案。具体方法如下：随便选一个整数，希望它是 $Z_n^\times$ 的一个原根，对其反复进行（模 n 的）平方计算，寄希望于某一次计算之后得到 1——从而得到 $Z_n^\times$ 的阶数。这样尝试的成功率很高（接近 1/2）。类似地，如果我们选择一些太容易被分解（素因子太小、太少）的 $p - 1 = 2^a 3^b 5^c ...$，攻击者同样可以对随机的整数进行快速幂运算，并指望在很短的时间内尝试出一个结果。

最后，我们介绍通过平方筛法破解一般的 RSA 的方法。如果我们已经找到一组非平凡的 $x, y \in Z_n$，满足

$$x^2 = y^2 \mod n.$$

那么我们就可以通过计算 $\gcd(x \pm y, n)$ 来得到 p, q。下面我们使用平方筛法来找到一组这样的 x, y。定义函数 f:

$$f(x) = (x - m)^2 - n,$$

其中 m 常取为 $\sqrt{n}$ 取整。我们计算一系列的 $\{f(x_i)\}$，保留那些只有较小素因子的:

$$f(x_i) = (-1)^{a_{0,i}} \times 2^{a_{1,i}} \times 3^{a_{2,i}} \times ... \times {p_k}^{a_{k,i}}$$

我们需要找到一组 $\{f(x_i)\}$，使得 Z_2 上向量组 $\{(a_{0,i}, a_{1,i}, ..., a_{k,i})\}$ 线性相关。定义矩阵

$$A = [a_{j,i}]_{i,j=0,1,...,n}$$

，由于其行向量线性相关，必可以找到 Z_2 上的线性方程组 $A^T v = 0$ 的非零解。这组非零解给出了:

$$Y^2 := \prod_{i=1}^{n} r_i f(x_i) = (-1)^{2d_1} \times 2^{2d_2} \times ... \times p_k^{2d_k}.$$

其中 r_i =0 或 1。则有以下等式:

$$Y^2 = \prod_{i=1}^{n} [r_i((x_i - m)^2 - n)] = (\prod_{i=1}^{n} r_i(x_i - m))^2 \mod n = X^2.$$

以上的等式不能保证得到一组非平凡的 Y, X（例如，如果运气非常糟糕，我们可能会得到一组 $\mod n$ 同余于 ± 1 的 X, Y，那么我们将得不到任何有用的结果），但从概率上来说，我们相当有可能通过以上的方法得到一组有效的 X, Y 解。

在实践中，对较大的 n（我记不清最大记录是多少了...）使用平方筛法所产生的矩阵 A 行/列数往往高达 2^{30}，存储数据和解方程都带来了不小的技术挑战。

2 第三讲 公钥密码学：Diffie-Hellman

本课程第一个 Project：自行编写一个 RSA 加密方案，要求 p, q 在 128 比特以上。可以使用 Magma/C++ 或任何语言。(5 分)

回到课程内容。我们介绍一种加速 RSA 解密的方式。对于解密过程

$$c^d = m \mod n,$$

我们希望计算能够更快。由于解密时持有私钥 p, q，根据中国剩余定理，有环同构：

$$\mathbb{Z}_{p\times q} = \mathbb{Z}_p \times \mathbb{Z}_q.$$

我们只需计算

$$c^d = x_1 \mod p;\ \ c^d = x_2 \mod q.$$

在进行这两次计算中，c, d 的位数将会降低一半（p, q 的位数一般相同或非常接近），这可以极大地降低计算时间。注意到求解 $\mod p, q$ 的同余方程时只需使用辗转相除法求逆，这比求较高的幂次要快得多。关于 RSA 的介绍就到这里。

下面我们介绍 DH 密钥交换方案。这个方案的原理非常简单，即使是高中生也能听懂。在 DH 方案应用的情形中，我们假设两位通信者，A 和 B，需要在公开的网络上通讯，也就是说，他们之间交换的任何信息都可能被攻击者截获。

首先，A 和 B 需要确定（并公开发布）二元组 (p, g)，其中 p 是大素数；$2 \le g \le p-2$，且 $g \mod p$ 的阶数充分大（理想的情况是，g 是 p 的原根）。

之后，A、B 执行如下操作：

- A 取定整数 $a \in \{0, 1, ..., p-2\}$，取 $\alpha = g^a \mod p$，并公开发布 α 的值。
- B 取定整数 $b \in \{0, 1, ..., p-2\}$，取 $\beta = g^b \mod p$，并公开发布 β 的值。
- B 接收 α 之后，计算 $\alpha^b = g^{ab} \mod p$；同理，A 接收 β 之后，计算 $\beta^a = g^{ab} \mod p$。$k = g^{ab}$ 就是 A、B 共有的密钥（例如，可以将其作为 AES 等私钥加密方案中的密钥）。

DH 的安全性由以下两个问题的困难性保证：

Definition 2.1 (CDH). *在 Diffie-Hellman 框架中，已知 (p, g, g^a, g^b)，试在多项式时间内求 g^{ab}。*

Definition 2.2 (离散对数问题，DL)**.** 给定 $p \in \mathbb{P}, n, a \in \mathbb{Z}^{\times}$；$a$ 是 p 的一个原根。已知 a, b 的值，试在（关于 p 的位数）的多项式时间内求出唯一的 $i \in Z_p$，使 $a^i = b \mod p$。即：在有限域 F_p 内对给定的原根 a 和非零的 b，试在多项式时间内求离散对数 $\log_a(b)$。

其中离散对数问题至少不比 CDH 问题简单，但我们尚未知道这两个问题是否等价，但 DL 的解显然可以解决 CDH。

既然我们现在手头有了两个不同的公钥加密方案（密钥交换方案），究竟哪个更好呢? 在实践中，RSA 具有一些 Forward Security Problem。斯诺登曾经披露过美国政府有一个巨大的数据库，在其中存放了所有互联网上的 RSA 加密通讯（以及它们的公钥）。由于在实践中，同一个用户往往一直使用相同的一组 RSA 密钥 (n, e, d)，一旦对于某个特定的 n，$n = pq$ 被破解，则这名用户的所有通讯都可以被破解。相反地，在使用 DH 密钥交换方案时，我们显然会在每次使用不同的 a, b（否则每次都得到的是相同的密钥，这毫无意义），相比于 RSA 这会具有更好的安全性。

一个对 DH 非常自然的推广是，我们并不总需要在 $\mathbb{Z}_p$ 这个群上进行运算。只要我们找到了某一个有限群，在其中做幂运算十分快速，但是求对数非常困难，我们同样可以在其上运行 DH。一个最常用的例子是椭圆曲线群。

下面我们介绍 El Gamal 公钥加密方案——这是一个基于 DH 问题，但达到与 RSA 相似功能的加密方案。

Definition 2.3 (El Gamal 加密方案)**.** *El Gamal* 加密方案由如下元素构成：

- 公钥：三元组 (p, g, A)，其中 $p \in \mathbb{P}$；g 是 p 的原根；$A = g^a \mod p$，其中 $a \in \{0, ..., p-2\}$。
- 私钥：a。
- $P = \{0, 1, ..., p-1\}$。

给定 (p, g, A)，加密如下进行：选取整数 $b \in \{1, ..., p-2\}$，计算 $B = g^b \mod p$。对明文 m 进行如下加密：$c = A^b m \mod p$。最终，传输的密文是二元组 (B, c)，因此 $|C| = 2|P|$。

解密过程是直接的：$B^a = g^{ab} \mod p$；令 $x = p - 1 - a$，有：

$$B^x c = g^{b(p-1-a)} A^b m = A^{-b} A^b m = m \mod p.$$

只需计算 x 和 $B^x c$ 即可。

El Gamal 的核心思想是，在进行加密时，事实上是使用公钥给信息加上了噪音 (noise)，而只有持有私钥的人才能去除信息上的噪音。这个加密方案和 RSA 还有一点不同:RSA 中加密是双射，而 El Gamal 中加密函数仅仅是单射。我们往往并不需要加密过程是双射——我们完全可以把明文集合映射到一个比它大得多的集合。

下面我们进入公钥密码学的一个重要主题：身份验证 (Authentication)。在进行通讯时，我们希望能够获得一种对通信人身份的确证——没有人能冒充他人进行通讯，也不能假装自己没有发送过某一条信息。

我们首先非常简短地介绍哈希函数。哈希函数是将任何信息映射到固定长度信息的函数:

$$F : \{0,1\}^* \to \{0,1\}^n,$$

其中 n 是一个定值（例如 256)。对于任意信息 X，计算 $F(X) = Y$ 非常快速，但是计算 Y 的原像 $F^{-1}(Y)$ 非常困难。我们事实上对哈希函数有更强的要求：给定 F，我们无法在多项式时间内找到任何一对 $X_2 \neq X_1$，使得 $F(X_1) = F(X_2)$。这个性质被称为 Collision Resistant。

如何设计一个哈希函数? 我们首先设计一个压缩函数，它是将固定长度的信息 “压缩” 到更短长度的函数。我们可以给出一个例子（尽管不太实用）：随机选定一组 $\mathbb{Z}_2$ 上的三次多项式:

$$[f_i(x_1, ..., x_m)], i = 1, 2, ..., n$$

对于信息 $X = x_1x_2...x_m$，压缩函数将 X 映到 $f_1(X), ..., f_n(X)$。

现在我们手上有了一个压缩函数，很容易想到如何用压缩函数构造一个哈希函数——多压缩几次就行了（如果需要，就在信息末尾加 0)。很简单，但是很有效。

另一种有效构造哈希函数的方式是借助任何一种被认为是安全的加密函数 $E_k(m)$，其中 k 是密钥，m 是明文。考虑信息 $X = (X_1, X_2)$，压缩函数可以构造如下:

$$C(X_1, X_2) = E_{X_1}(X_2) \oplus X_2;$$

$$C(X_1, X_2) = E_{X_1 \oplus X_1}(X_2) \oplus X_1;$$

这样的构造方式往往都是比较安全的。

真正使用的哈希函数往往都有非常琐碎复杂的构造，我们不多加介绍。下面我们介绍一种使用 RSA 构造签名系统的方法。

给定信息 m，我们使用一个（公开的）哈希函数 H，计算 $H(m)$；像在 RSA 中一样，我们选定大质数 p, q，公开公钥对 n, e，藏好私钥 d——这将是我们的“印章”，用来在信息上刻上“这确实由我发送”的证据。在给别人发送信息时，我们还需要发送一个签名：

$$s = H(m)^d \mod n.$$

收到信息的人会同时收到二元对 (m, s)。验证签名的方式是，计算 $H(m)$ 和 $s^e \mod n$。如果二者相同，说明发出信息的人确实具有私钥 d——签名成立了。

电子签名系统在现实中有着极其重要的应用。我们每个人都在使用的私人电脑上的系统都需要常常更新，但我们怎么能保证系统的更新包确实是由微软发出的，而非某些恶意的、希望摧毁你的电脑的冒充者？答案是，微软公司发布了一个 2048 比特长度的 RSA 公钥用作自己的 RSA 签名（这个非常大的 $n = pq$ 就存放在我们的电脑上），在发送信息时，我们的电脑会自己验证信息的签名是否能对得上，从而判断信息究竟是否真正由微软发出。

一个更具有浪漫气息的描述是，如果你能够分解这个 2048 位的大素数，那你就能轻易地攻破这个世界上所有使用 windows 系统的电脑。

电子签名系统的另一个重要应用是在密钥交换方案中。我们将在下一讲中介绍应用身份验证机制的密钥交换方案（Authenticated Key Exchange, AKE）。

3 第四讲 公钥密码学：Signature

我们首先定义所谓的 One-way function.

$$F : \mathbb{F}_q^n \to \mathbb{F}_q^m$$

对给定的 x，求 $F(x)$ 十分简单，但对给定的 y，求 $x\ s.t. F(x) = y$ 是计算上困难的。

究竟什么是“困难”？这是一个基于现实世界中计算用时的模糊定义：究竟是平均的计算用时很长，还是相当大一部分的计算用时很长，还是最糟糕的情况（最容易计算的参数选取下）也极其困难？在这门课程中，我们将不会涉及这些细节上的概念。

我们提到另一个问题：如何选取 RSA 的两个素数 p, q。我们说过希望 $p-1, q-1$ 具有较大的素因子，但总体来说完全随机地指定素数是可行的（需要额外进行一些简单的检验，例如确保 $p-1$ 不能被 10^6 以下的素数整除）。

现在我们讨论电子签名系统。首先，我们定义另一种身份验证方案。假设 A 持有私钥 x；A 向全世界宣布公钥 (g, g^x)。A 希望向全世界证明自己知道 x 是什么，但又不希望在证明的过程中泄露有关 x 究竟是什么的任何信息。从这个思路出发，我们可以构建一种与 RSA 不同的身份验证方案。

首先，A 随意地选取 r 并将 $t = g^r$ 发送给 B。B 随机选取一个数 c，并将 c 发给 A。A 再发送给 B 一个数：$r + cx$。由于 B 拥有 g^r, 他可以从 $z = r + cx$ 中计算出 g^x：

$$g^z = g^{r+cx} = g^r \times (g^x)^c. \mod p$$

（注意到我们始终在 Z_p 中进行整个过程，所以最后一步是简单的，而不是 RSA 问题）这个签名系统与 DH 方案非常相似，如果 DL 问题被解决，那么它将不再安全。

现在我们对一个签名系统进行理论上的总结。Completeness, soundness, 0-knowledge，这是我们希望一个签名系统具有的三个特质：如果知道私钥，总能正确地回答；如果不知道私钥，总不能正确地回答；在问答的过程中不会泄露关于私钥的任何信息。下面我们讨论以上定义的签名系统的 0-knowledge。

如果我们允许攻击者在两次使用不同的 c 时，A 返回了相同的 r，那么签名系统不攻自破：

$$\frac{z - z'}{c - c'} = x.$$

另一个问题是，这个签名系统的验证过程是可以“伪装”的。我们完全可以首先决定好 z, c，然后计算出对应的 r，在验证时却首先发送 g^r，然后再发送 c 和 z。在公众眼里，A' 和

B 已经完成了一次签名验证，但是 A' 或许根本不是持有私钥的 A，而是 B 自己的一个小号。

从这两个例子里，我们可以看出签名系统的安全性可能受到各种各样的挑战。

下面，我们介绍基于哈希函数的签名系统。给定信息 m，我们计算：

$$H(g^x, t, m) = c.$$

其中 g^x 是公钥三元组 (p, g, g^x) 的一部分（所有计算都在 Z_p 中进行）；t 是一个“承诺”（由我发送这条信息）；m 是发送的信息。在发送信息 m 之后，我们附上签名 $(t, z) = (g^r, r + cx)$。这个签名保证这条信息能，且仅能是由我一个人（持有私钥 x 的人）发出的。在某种程度上，这是一个“自我挑战”的机制：使用哈希函数，我给自己出了一道题：给出 $z = r + cz$ 的值，这件事只有我一个人能做到，不知道 x 的人无法计算 $z = r + cx$。

验证的过程如下：通过公钥 g^x 和信息 m、承诺 t 之后，任何人都可以计算 $c = H(g^x, t, m)$，从而验证 $g^z = g^{r+cx} = t \times (g^x)^c$ 是否成立。值得注意的是，验证者无法从这个过程中还原出 x：如果要从 $z = r + cx$ 中还原 x，无疑需要知道怎么计算 $r = \log_g t$，这又是一个离散对数问题。

现在我们可以稍微讨论一下，基于 RSA、DH 加密的公钥加密方案还可以怎么改变？最简单的回答是椭圆曲线群。要得到一种不同的公钥加密方案，我们只需要（在本质相同的有限群中）改变乘法/加法计算的定义并继续沿用 RSA/DH 或者其他方案，就能得到一种新的方案。剩下的问题是，究竟有那些群运算具有 Trapdoor 函数的性质？在此基础之上，我们希望这些群具有可调整的大小（至少要能够任意大），从而加密长度不同的信息。我们会（并不那么惊讶地）发现，已知可能具有这些性质的群是很少的，我们并没有多少选择（常用的只有 Z_p 和椭圆曲线群）。

下面讨论 AKE。类似于 DH 方案这样的密钥交换方案对中间人攻击来说是非常脆弱的，在密钥交换时选择合理的电子签名系统可以解决这个问题。SSL、TLS 就是常用的 AKE 方案。

我们还对 PKI 系统进行简单的介绍。一切签名系统都需要有一个“信任的源头”——你声称这是你的密钥，其他人又怎么确认这件事情？需要有某些机构来背书。在现实世界中，各个国家具有自己的公钥，被称为 root，是信任的源头；国家会使用这个密钥来对大型单位，例如清华大学的官方密钥进行电子签名（国家认证这是清华大学的公钥，这件事情可以由任何人核实）；清华大学再使用自己（经认证）的密钥来给教授和学生们的公钥进行签名，认证这确实是清华大学教授/学生的公钥。这种机制构成了一个信任的链条。我们有时会见到某个网页“不具有安全证书”，这实际上是说这个网站的公钥没有被更高层的机构签名认证过，因此不能保证信息传输是安全的。

一个是公钥加密方案，一个是电子签名，这两种密码系统构成了整个现代社会通讯的基础。一个最著名的例子是比特币。（关于比特币的介绍在此处不进行叙述）