Learn RSA the Hard Way

IIIx

temple3x@gmail.com
templex.xyz

April 18, 2020

介绍 RSA 基本原理的文章和书籍非常多， 但没有一篇能满足我但好奇心。 我不是特别在意RSA是如何工作的， 我更关心 RSA 是如何被发现的。 当然了， 因为我已经预先知道了 RSA，无论我如何尝试都不能踏上真正的发现之旅， 不过这一路会有趣味， 不是吗?

现在， 我把这份乐趣带给你们。

---

1. 前言

1.1. 主要内容

正文内容在逻辑上分为三个主要部分:

1. 明确不可破译的密码的具体目标
2. 为达成目标进行尝试
3. 求证与总结

1.2. 阅读指南

本文对读者能力做了如下假设:

1. 掌握初等数学的基本运算和公式
2. 能使用计算机语言编写基础函数

同时对读者的好奇心做出了如下假设:

1. 有设计一套不可破译密码的野心
2. 好奇自己的好奇心究竟有多强

之所以是 “The Hard Way” 原因如下:

1. 创建一套不可破译的密码是个艰巨的任务
2. 以上过程势必包含灵光乍现的时刻， 这一方面需要不断提出好问题， 一方面也让人费解
3. 提出好问题需要不断的猜测和验证
4. 满足以上三点需要保持好奇心， 而我们通常忘了自己好奇心有多强
5. 满足以上四点需要足够的休息与停顿， 而我们通常被教育 “时间就是金钱” “金钱就是一切”

本文的阅读方法有两种:

1. 按章节顺序阅读， 请准备好纸笔与放松的心情
2. 根据附录 9.1 中的指引阅读， 如果你想要的仅仅是 “知识”

注：

1. * 表示内容超出了预设的读者能力范围， 需要我们进一步学习与探索
2. ** 表示内容没有超出能力预设， 但不得不直接引用的定理。一方面， 这是因为对这些定理进行证明会干扰整体思路； 另一方面， 在“一无所知” 的情况下得出惊人的结果是反常识的。
3. *** 表示出现了“灵光乍现” 或是“直觉”。我们不得不接受“任何创造性工作都不可避免的依赖灵感” 这一事实

2. 当我们说一套密码难以破译时我们在说什么

在我们的想象中， 密码似乎是离我们很遥远。它像从天而降的陨石一般砸进天才的大脑离， 然后天才奋笔疾书设计/破译一套密码。实际上， 密码在我们的日常生活中很常见， 如果来到一个陌生的国家， 当地人说这我们听不懂的外语， 对于我们， 他们的交流就是加密的。或许有人觉得我不够严肃， 简直是在玷污密码学高深莫测的形象。

现实中， 我们也的确这么干过。二战时， 美军曾利用纳瓦霍人的语言作为密码， 在中国温州也有关于温州话作为战争密码的民间传说。

温州话（我今晚吃了一头牛）＝吾内窝飞企鹅一兜ng鹅奥
温州话（吾内窝飞企鹅一兜ng鹅奥）＝我今晚吃了一头牛

不难发现， 破译温州话也很简单， 抓一个温州人便是了。这听上去很粗暴， 但打战的时候我们就是这么干的， 天才数学家只是备选方案。

由于开发一套新算法的代价很高， 通常跟不上被窃听的速度， 因此我们选择在原有算法的基础之上混一点东西进去， 这点东西被称之为密钥。听上去很保密， 实际上却很脆弱——我们如何保证传递密钥的通信是安全的呢？ 答案是没有办法。

所以设计不可破译密码的关键是： 允许被窃听***。

这听上去匪夷所思。

3. 两把钥匙

加密端到解密端的线路运训被窃听， 意味着必须满足如下方程***：

加密算法（我今晚吃了一头牛，公开钥匙）＝x
解密算法（x，私有钥匙）＝我今晚吃了一头牛

原本我们只有一把密钥， 如果被截获游戏便结束了。倘若钥匙变成两把， 一把只能加密， 另外一把则可以解密， 由于解密钥匙无需发送给对端， 它非常安全。与此同时， 上面的式子蕴含了两个关于密钥对的关键信息：

1. 公开钥匙（公钥）对应唯一一个私有钥匙（私钥）， 反之亦然
2. 从公钥极难推导出来私钥

要满足条件1 是因为如果公钥与私钥是一对多， 或者多对一， 或者多对多的关系， 则意味着我们无法确认加密/解密者的身份是否合法。条件2 则是我们允许窃听的信心来源。

为了讨论方便， 我们用字母表示上式中的元素：

$$c = E(m, u)$$
$$m = D(c, v)$$

其中 E 为加密算法； m 为明文， 即要加密的内容； u 为公钥； c 为密文， 即加密后的内容； v 为私钥。

让我们继续对上面提出的两个关键信息进行追问。因为单凭这两条信息太过简略， 缺乏具体的行动指引， 并且如果仅仅是满足这两个条件是非常容易的， 我们只需要一个 “温州话” 的升级版本：

1. 提前准备好一对随机数 u， v
2. 加密方利用 u 加密， 解密方利用 v 解密

于是：

$$c = E(m, u)$$
$$x = D(c, v)$$

至于 x 会是什么？ 天知道！ 我们需要追加一条要求：

1. 公钥对应唯一一个私钥， 反之亦然
2. 从公钥极难推导出私钥
3. 通过私钥可以解密公钥加密的数据

这同时意味着， 私钥可以很容易的求出公钥。因为条件1 与条件3 要求公钥与私钥存在唯一的数学关系， 我们希望这个算式最好是容易求解的：

1. 公钥对应唯一一个私钥， 反之亦然
2. 从公钥极难推导出私钥
3. 通过私钥可以解密公钥加密的数据
4. 通过私钥解出公钥很容易

但是这样就够了吗？ 倘若我们设计出的密码有且只有一对密钥可以工作， 或在一定范围内只有非常有限的密钥对， 那么它将是非常脆弱的， 只能算是个数学游戏。于是：

1. 公钥对应唯一一个私钥， 反之亦然
2. 从公钥极难推导出私钥
3. 通过私钥可以解密公钥加密的数据
4. 通过私钥解出公钥很容易
5. 在一定范围内， 存在大量密钥对

我们还需要对 “极难” 做出定义。首先， 公钥肯定可以推导出私钥， 如果不满足这一点则意味着公钥与私钥毫无关联， 也就不可能满足条件3 。同时很显然的是， 不能以任何直接的数学公式通过公钥求得私钥， 因为这样过于简单， 那么从公钥到私钥的求解办法只能是穷举， 即将列出的所有可能性逐个尝试， 只要我们让可能性足够多， 穷举就真的是无穷无尽了。那么， 进一步的：

1. 公钥对应唯一一个私钥， 反之亦然
2. 从公钥只能通过穷举推导出私钥
3. 通过私钥可以解密公钥加密的数据
4. 通过私钥解出公钥很容易
5. 在一定范围内， 存在大量密钥对

重新整理一下我们目前的收获：

1. 已知私钥， 可以很容易求解唯一公钥

2. 已知公钥， 只能通过穷举求解唯一私钥
3. 在一定范围内， 存在大量密钥对
4. $m = D(E(m,u),v)$

好么， 事情看上去更复杂了。

休息一下。

4. 密钥候选者

先抛开加/解密的要求以及实践要求， 观察密钥的两大基础要求：

1. 已知私钥 v， 可以很容易求解唯一公钥 u
2. 已知公钥 u， 只能通过穷举求得唯一私钥 v

首先， 我们不妨假设 u， v 均为整数， 因为整数可以精确的在计算机中被表达， 那么 u 可以表示为**：

$$u = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} + a_n x^n$$

对于这个多项式运， 我们暂时能想到的复杂度最高的运算为指数运算， 其逆运算为对数运算，听上去不错。其中最简单的形式之一是令 $a_i = 0, 0 \leqslant i \leqslant n-1$，$a_n = 1$， 则：

$$u = x^n$$

如果已知 u 与 x 求 n， 看上去是个挺难算的问题。但细想一下， u 但因子实际上只有 x 一个，且函数随 n 递增， 比如我们只需要每次用折半的方法（已知整数解肯定存在， 我们很容易计算 n 的上限max 与下限min， 每次取 $(\min + \max)/2$ ， 若大于目标则继续与 min 折半， 若小于则与 max 折半）很快就能找到 n。这让人很是失望， 对数真的是个不错的想法。

既然对数不满足需求（或是暂时不满足）， 我们看看其他可能性， 然而怎么看单调性都是不可避免的。但也不能这么快放弃， 我们现在的需求是打破单调性， 准确的说为了满足需求， 最好让 u 的值随着 x 的增加而 “随机漫步”。

如何做到这一点呢——让 u 在一个特定的集合内***。

做到这一点最直观， 也是最简单的运算是模运算***。我们可以假设：

$$u \equiv x^n \pmod{m}$$

这似乎非常令人满意， 然而怎么确定 x， n 与 m 呢？ 它们和私钥 v 的关系又是什么呢？ 毕竟 u 当中肯定蕴含着得到 v 的潜能。一时实在是很难想到， 暂时保留猜测吧。

在一筹莫展的情况下， 我们不妨看看为什么直觉告诉我们对数是个好主意， 这或许对我们会有启发。我们之所以会觉得对数是个好主意， 是因为：

1. 保证了唯一性
2. 正向计算十分容易
3. 逆运算（原以为）需要穷举

对于原因3 来说， 主要是其中蕴含的大量乘法吸引了我们， 而大量乘法之所以沦为简单是由于暴露了唯一的因子 x。可是， 谁又规定了 x 一定是 “一个数” 而不是一个表达式呢？ ***

小学一年级的数学课程里， 最重要的内容就是乘法分解了。把一个数分解成若干质数的乘积是我们从小就开始的训练， 那么， 同样的， 在这里我们为什么要拘泥于 x 就是 “x”， 而不是若干质数的乘积呢？ 有意思的是， 若我们假设 v 是一个合数：

1. $n=1$， 这是因为我们通过乘法已经隐藏了 x， n 看上去无关紧要了（暂时保留意见）
2. 在 1 的基础上， 我们希望 u 是若干不同质数的乘积， 正向计算很容易
3. 质数并不容易找， 尤其是当数字很大时*， 因而反向计算非常难

我们不妨假设 u 是两个不同质数的乘积。这是最简单的设想， 行不行还不知道， 但这是一个不错的开始。这里对条件3 进行一点必要的补充： 判断一个数是不是质数是相对简单的， 最直接的方法是根据质数的定义判断， 但是在一个很大的数字范围内精确的找到唯一的那一对质数似乎就非常痛苦了， 我们要验证的数字太多， 比如我们可以让 u 比可观测宇宙的原子数量（不过10^{82}而已）还大个宇宙原子数量倍或者再大一点。但是这样做固然是安全， 可我们还有没有能力去做加密呢？ 这个问题我们先放一放， 毕竟我们连具体算法都不知道， 又如何知道怎么优化呢？

另外， 别忘了我们为了打破单调性做出的尝试， 那么模运算是唯一的办法吗？ 或许有其他有意思的思路*。可能是我们现在所掌握的知识还不够多， 先把这个问题留在心里吧。

虽然现在还有很多疑问， 但我们手上的线索并不多， 先试一试能不能以质因数分解为基础构造加/解密函数吧。

5. 初尝构造

我们的构造必须满足：

$$m=D(E(m,u),v)$$

好家伙！ 一看到这个公式， 刚才的喜悦就化作鸟兽散尽了。理性的直觉告诉我， 这太难了。

那干脆勇敢一点？ 要知道 RSA 的作者们也是做了一年多的尝试， 最终在一个夜晚蹦出了 RSA。不过我也不是说放手瞎搞， 先再看一遍目前的假设吧：

1. $v=f(p,q)$， p q 为两个不同的质数
2. $u=p\cdot q$
3. $E(m,u)=c$
4. $D(c,p,q,v)=m$

E 目前有两个参数 m， u， 我们很难仅根据两个参数构造一个复杂的逆运算。另外， 其中 m 为明文与 D 无关， u 尽管由 p， q 得到， 但目前看上去也与 D 的计算过程无关。这里似乎了少了重要的东西——即 E 与 D 的关联（暂时无法确定是不是真少了）， 若没有这个关联， 解密将是无稽之谈。或许， 我们需要新加一个参数， 试试看吧：

$$E(m,u,n)=c$$
$$u=f_1(p,q)$$
$$n=f_2(p,q)$$

由于本文一直以 u 作为公钥， 目前公钥还有一部分未确定， 所以将 $p\cdot q$ 赋值给 n， 暂时保留 u。并且 u 必须与 v 有函数关系***， 所以：

$$n=p\cdot q$$
$$v=f_3(p,q,u)$$

重新整理假设：

1. 公钥：$u = f_1(p, q, v), n = p \cdot q$
2. 私钥：$p, q, v = f_2(p, q, u)$
3. $E(m, u, n) = c$
4. $D(c, p, q, v) = m$

现在我们最需要的是休息， 明天见。

6. 关于加密的进一步假设

目前， 我们有：

1. 公钥：$u = f(p, q, v)$ $n = q \cdot q$
2. $E(m, u, n) = c$

其中对 E 对期望是已知 E， c， u， n 只能通过穷举得到 m。针对已有的有限参数， 我们能构造出的运算很有限， 但是别忘了类似的问题在“4. 密钥候选者”中我们讨论过一次， 我们已经提出了一种可能性的构造似乎很适合现在的需求， 它有如下形式：

$$c \equiv m^u \pmod{n} \qquad (1)$$

更加友好的是， 通过前面的假设我们只需确认 u 即可完成构造（至少只有一个未知量了， 好消息， 不是吗？ ）。我们现在看看如何找到正确的 u 来满足：

$$D(c, p, q, v) = m \qquad (2)$$

我们将（1）展开：

$$m^u - n \cdot \lambda = c \qquad (3)$$

可以肯定是， c 中蕴含向后得到 m 的潜能， 这意味着通过解密运算 E， 与 m 无关的都会被约去， 对上式来说要约去 n。什么运算可以在约去 n 的情况下， 保持（3）继续成立呢？ ——模 n***。

于是我们可以得到如下式子以满足我们的要求：

$$\alpha(m^u - n\lambda)^x \equiv \beta m^\gamma \pmod{n} \qquad (4)$$

由于是模 n， 所以包含 n 的项可以约去， 于是由（4）得：

$$\alpha(m^u)^x \equiv \beta m^\gamma \pmod{n}$$

以上是关于解密E 的一般化构造， 不难发现的是对于 βm^γ 的指数 γ 来说它若不为 1 则意味着解密过程需要开 γ 次方来求得 m， 这显得没有意义， 加密算法的安全性不由这里保证， 我们可以抹去凭空增加的复杂度， 同理我们可以假设 α，β 也均为 1， 则：

$$m^{ux} \equiv m \pmod{n} \qquad (5)$$

由（5）可知， x 显然为我们苦苦寻觅的私钥（只有这么一个没有名字的可怜虫了）， 于是：

$$m^{uv} \equiv m \pmod{n} \qquad (6)$$

若 $u \cdot v = 1$ 则上式显然成立， 但也失去了加密的意义。这是个好的开始， 至少我们找到了一种满足等式的解。

由（6）我们容易想到欧拉定理**:

$$m^{\varphi(n)} \equiv 1 (\mathrm{mod}\, n)$$

当 m 与 n 互质， 我们有 $u \cdot v - 1 = \lambda \cdot \varphi(n)$， 即:

$$v = \frac{\lambda \varphi(n)}{u} + \frac{1}{u}$$

当 $\varphi(n)$ 与 u 互质，λ， u， v 有唯一解。否则， 我们有可能找到多组 λ， u 满足同一个 v。

我们找到公钥与私钥了！

且慢， 因为欧拉定理要求 m 与 n 互质， 这似乎对明文提出了不合理对要求。但别忘了， 我们还有其他约束条件:

$$uv - 1 = \lambda \varphi(n)$$

这能帮助我们对任意正整数 m 满足(6) 吗? 我们必须得试一试， 因为一旦满足， 我们就大功告成了！

休息时间到了。休息， 休息。

7. 功亏一篑?

我们现在的问题是:

已知

$$uv - 1 = \lambda \cdot \varphi(n) \qquad (1)$$

求证

$$m^{uv} \equiv m (\mathrm{mod}\, n) \qquad (2)$$

m 与 n 互质， 即$(n.m) = 1$时， （2）为欧拉定理， 得证。

当 $(n.m) = n$ 时， 即 $m = \alpha n$, 上式显然成立。

由 $n = pq$ 可知， 还需证 $(n.m) = p$ 或 $(n.m) = q$ 的情况， 这两种情况求证一种等价于另一种情况， 所以我们不妨设 $(n, m) = p$。此时 $(q, m) = 1$， 由欧拉定理可知:

$$m^{\varphi(q)} \equiv 1 (\mathrm{mod}\, q) \qquad (3)$$

由欧拉引理可知:

$$\varphi(n) = \varphi(pq) = \varphi(p)\varphi(q) \qquad (4)$$

由 (1) 与 (4) 得:

$$m^{\,\mathrm{uv}} = m^{\lambda\varphi(n)+1} = m^{\lambda\varphi(p)\varphi(q)+1}$$

带入（3）得:

$$m^{\,\mathrm{uv}} = m^{\lambda\varphi(p)\varphi(q)+1} \equiv m (\mathrm{mod}\, q) \qquad (5)$$

由 (5) 得 $q \mid (m^{uv} - m)$， 因 $p \mid m$， 所以 $p \mid (m^{uv} - m)$， 又因为 $(q, p) = 1$， 所以 $n \mid (m^{uv} - m)$。证毕。

功亏一篑？ 大功告成！

8. 回顾

首先， 我们分析了原有加密方案的最大缺陷——不允许在非加密线路上进行加密通信， 可谁又来给非加密线路加密呢？ 由于这个问题无解， 我们干脆尝试在非加密线路上安全通信。这又要求我们将原本的密钥分为公钥和私钥。

为了实现上面的要求， 我们希望私钥可以求出公钥， 但公钥极难得到私钥。通过对公钥的多项式展开， 我们发现了公钥可能的构造， 并以这个可能性为基础尝试打造难以逆运算的加密算法（在不知道私钥的情况下）。

在上一步的基础上， 我们选择了模幂运算。但不确定是否能满足我们的性质要求， 通过若干证明， 终于求得了我们梦寐以求的加密算法。

运气是不是太好了点?

可我们别忘了我们尽管幸运的得到了最终算法， 可在这个过程中还有有些有意思的启发没有得到进一步的探索， 或许 RSA 不是我们唯一的选择?

另外， 我们对 RSA 安全性的一些直觉判断真如我们所设想的吗?

9. 附录

9.1. Learn RSA the Easy Way

这里介绍一种如何更全面的掌握 RSA 相关知识的路径:

1. 阅读 RSA 作者的论文《A Method for Obtaining Digital Signatures and Public-Key Crypotosystems》， 这篇论文结构清晰简单， 从需求开始到 RSA 算法到必要的数学证明与算法实现， 可以帮助我们较快的掌握 RSA 的核心内容
2. 《Introduction to Algorithms》的 The RSA public-key cryptosystem， 在这本书中较为完整的展示了 RSA 算法的运算步骤， 帮助我们获得更具体的关于 RSA 实践的认识
3. 接下来属于进阶知识（具体实现与攻防数学原理）:
    i. 根据对 RSA 具体算法步骤的了解， 对参数选择的攻击与防御， 包括但不限于:
        a) 共模攻击
        b) 循环攻击
    ii. 对 RSA 依赖的因式分解的攻击与防御， 包括但不限于:
        a) 证明计算解密指数 v 的任何算法都近似等价于分解 n
        b) 分析质因数分解的难度
        c) 证明高斯素数定理
        d) 素数的筛选与选择
    iii. 对加速 RSA 运算的分析与实现