RSA

Crazy_13754

2024 年 1 月 21 日

摘要

写了些关于 rsa 的东西

目录

1 引言

实际上，写这篇的时候发现有论文写的很清楚了[1]。此外，网上各种博客内容已经一应俱全，本来把它们丢上来就可以了。你问我为什么还要写这篇文章？请看这个图 1：

图 1: This is an inserted jpg graphic

如果你还是没有明白我想说什么，请再看看这个表 1：

如果（虽然几乎是当然的）你还是不理解，那就看看这些图2345:

图 2: 冬の花

图 3: RickT

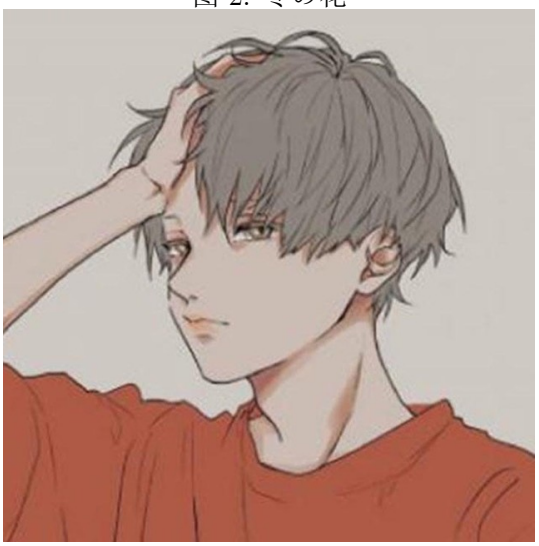

图 4: 朗格拉日

图 5: dx3906

对写奇奇怪怪东西的看法	可以理解	不可理喻
支持	0.1%	0.0%
不支持	0.2%	99.7%

表 1: Table to test captions and labels

这显然把事情弄得更糟。好吧，只是我在学 LaTeX，而你—我的朋友—浪费了你生命中的宝贵时间来看刚刚的内容，而且你接下来看到的东西也都可以在网上找到。不知道这篇文章所帮助你减少的在浏览器上搜索的时间能否弥补它所浪费的时间。或许，你需要的是及时止损，现在就放弃阅读？

公钥密码系统的观点是由 Diffie 和 Hell man 在 1796 年首次提出的, 它是密码学发展史上具有里程碑意义的一件大事。与传统对称密码体制（即加、解密密钥相同）相比, 公钥系统使用两个密钥：加密密钥可以公开, 称为公钥；解密密钥保密, 为私钥。产生公钥体制的内在动力有两个：

(1) 传统对称体制下密钥的存储和分配问题；

(2) 消息鉴别问题, 就是指用来检验消息来自于声称的来源并且没有被修改。

公钥体制的基础是陷门（单向函数）, 即某种实际处理过程的不可逆性。目前的公钥思想基于两种：一是依赖于大数的因数分解的困难性；二是依赖于求模 p 离散对数的困难性。RSA 密码算法就是基于大数的因数分解的困难性。

RSA 加密算法是一种非对称加密算法，在公开密钥加密和电子商业中被广泛使用。RSA 是由罗纳德·李维斯特 (Ron Rivest)、阿迪·萨莫尔 (Adi Shamir) 和伦纳德·阿德曼 (Leonard Adleman) 在 1977 年一起提出的。当时他们三人都在麻省理工学院工作。RSA 就是他们三人姓氏开头字母拼在一起组成的。1973 年，在英国政府通讯总部工作的数学家克利福德·柯克斯 (Clifford Cocks) 在一个内部文件中提出了一个与之等效的算法，但该算法被列入机密，直到 1997 年才得到公开。

这篇文章介绍了因子分解的相关算法，然而，这其中的很多方法对 RSA 来说并不适用。对于 RSA 的“暴力”攻击来说，普通数域筛法 (GNFS)应该是最优的。如果我能看懂这个算法的话，会加上对它的介绍的……

2 数学基础

2.1 欧拉函数

对于 $n \in \mathbb{Z}^*$，欧拉函数 $\varphi(n)$ 表示小于等于 n 的正整数中与其互质的数字个数。其可表示为：

$$\varphi(n) = \begin{cases} 1, & n = 1 \\ n(1-\frac{1}{p_1})(1-\frac{1}{p_2})\cdots(1-\frac{1}{p_r}), & n = p_1^{k_1}p_2^{k_2}\cdots p_r^{k_r}, p_i \text{ 为素数，} k_i \in \mathbb{Z}^* \end{cases}$$

n 一定属于以上某条件，因为大于 1 的正整数都可以写成一系列质数的积：若 n 是质数，则显然成立；若 n 不是质数，那它一定是个合数，用归纳法可以证明。

当 $n = p^k$ 时，不包含 p 的数才能与其互质。而包含 p 的数字有 $p, 2\cdot p, \ldots, p^{k-1}\cdot p$ 共 p^{k-1} 个，因此此时 $\varphi(n) = p^k - p^{k-1} = p^k(1-\frac{1}{p})$。

当 $n = p_1^{k_1}p_2^{k_2}\cdots p_r^{k_r}$ 时，可以通过两种方式证明该式。

2.1.1 容斥原理

先考虑 $n = p_1^{k_1}p_2^{k_2}$ 的情况。在计算 $\varphi(n)$ 过程中应当剔除 $p_1, 2p_1, \ldots, \lfloor\frac{n}{p_1}\rfloor p_1$，同理剔除 $p_2, 2p_2, \ldots, \lfloor\frac{N}{p_2}\rfloor p_2$，并根据容斥原理（计数时重复计数的部分要扣除）还需要补回 $p_1p_2, 2p_1p_2, \cdots, \lfloor\frac{n}{p_1}\rfloor p_2$ 的部分。这证明了 $\varphi(n) = n(1-\frac{1}{p_1})(1-\frac{1}{p_2})$。

同理，$\forall m, n \in \mathbb{Z}^*$，有 $\varphi(mn) = mn(1-\frac{1}{p_{m1}})(1-\frac{1}{p_{m2}})\cdots(1-\frac{1}{p_{mr}})(1-\frac{1}{p_{n1}})(1-\frac{1}{p_{n2}})\cdots(1-\frac{1}{p_{nk}}) = \varphi(m)\varphi(n)$，这就证明了 $n = p_1^{k_1}p_2^{k_2}\cdots p_r^{k_r}$ 时，$\varphi(n) = n(1-\frac{1}{p_1})(1-\frac{1}{p_2})\cdots(1-\frac{1}{p_r})$。

2.1.2 中国剩余定理

不是很能看懂。直接贴个链接。（大概理解意思，但是不理解“第二种证明方法有问题”，显然是没真明白）

2.2 欧拉定理

欧拉定理指的是，如果两个正整数 a 和 n 满足 $\gcd(a,n)=1$，则有：

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

证明……先等等吧……

3 加密与解密过程

以下是算法描述：

- 取两个非常大的素数 p 和 q。
- 令 $N=p\cdot q$，$\varphi(N)=(p-1)\cdot(q-1)$。
- 找到一个素数 $e<\varphi(N)$，且要求 e 与 $\varphi(N)$ 互素，即有 $gcd(e,\varphi(N))=1$。
- 计算 e 在模 $\varphi(N)$ 上的逆元 d，即求 d 满足 $e\cdot d \bmod \varphi(N)=1$。
- (N,e) 作为公钥 (pk, Public key) 发给对方，(N,d) 作为私钥 (pt, Private key) 保存。
- 对方接受公钥后，将原文 (pt, Plaintext) 通过预先设定好的方式转换成数字，记为 p，则密文 (ct, Ciphertext) 满足 $ct=p^e \bmod N$（也就是 $ct\equiv p^e(\mathrm{mod} N)$），对方将密文发回。
- 接收密文，并使用私钥解密：$p=ct^d \bmod N$，也就是 $p\equiv ct^d(\mathrm{mod} N)$。

4 RSA 数字签名

现在设想这样一个情形：小唐想要和小姜通信，于是小姜制作了一份美味的密钥并把公钥发给小唐。然而攻击者小张劫持了通信网络，把自己做的公钥发给了小唐，然后用自己的公钥解密信息，并用小姜的公钥重新加密。显然，小唐和小姜几乎没有可能发现信息泄露了—因为攻击者小张并没有留下什么痕迹。

为了解决这个问题，人们需要 RSA 数字签名来证明公钥体制下的信息发送者。

其过程为：将信息通过哈希函数生成一个散列值（哈希函数是防碰撞的，也就是难以找到具有两个相同结果的输入）然后将这个数值通过私钥加密。接收者接收公钥和私钥加密的信息，经过相同的哈希算法验证散列值是否与解密的散列值一样。你可能发现了—这个签名功能能验证信息是否在传输过程中发生了部分丢失，但似乎并不能解决上述问题。

参考文献

[1] 陈传波 and 祝中涛. Rsa 算法应用及实现细节. 计算机工程与科学, 28(9):13–14, 2006.