RSA

Crazy_13754

2024 年 1 月 22 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：

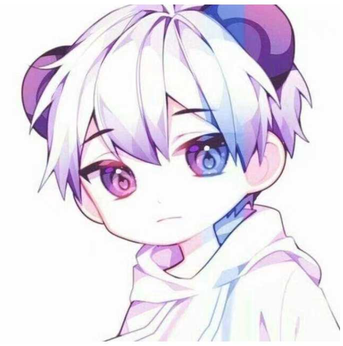

图 1: This is an inserted jpg graphic

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:

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

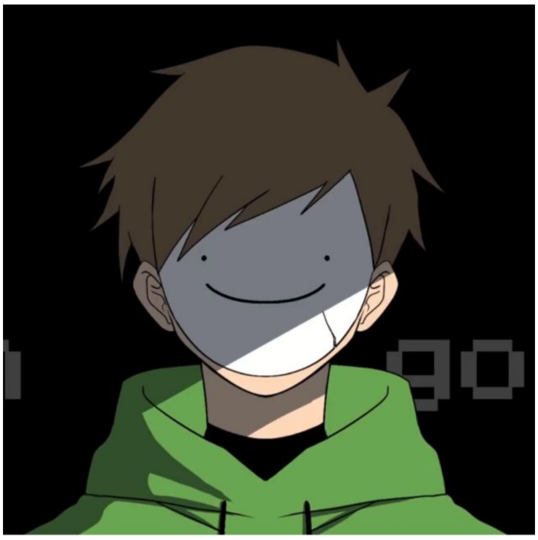

图 2: 冬の花

图 3: RickT

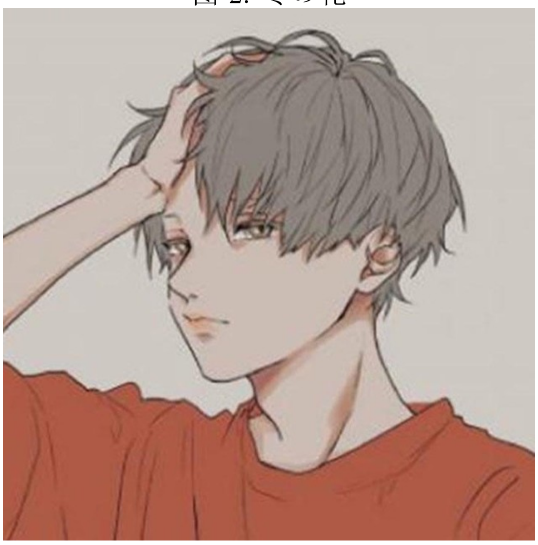

图 4: 朗格拉日

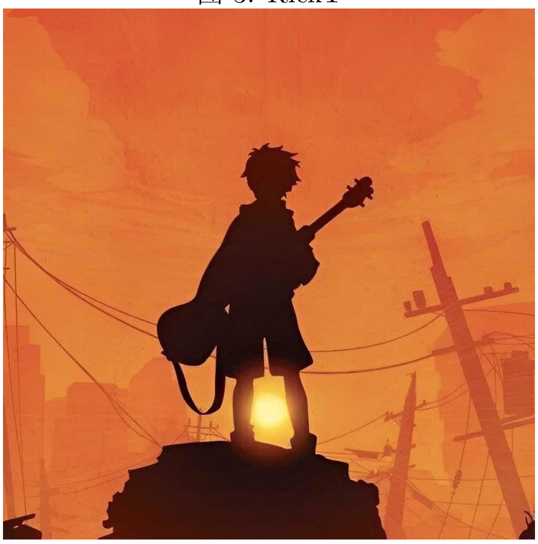

图 5: dx3906

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，我们可以用归纳法证明：

- 1、2 满足以上条件。
- 若 $n \neq p_1^{k_1}p_2^{k_2}\cdots p_r^{k_r}$，那它一定是个合数，可以分解为两个或多个小于 n 的数字乘积。

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。在计数 $\varphi(n)$ 的过程中不计数 $p_1, 2p_1, \ldots, \lfloor\frac{n}{p_1}\rfloor p_1$,同理对 $p_2, 2p_2, \ldots, \lfloor\frac{N}{p_2}\rfloor p_2$ 也不应该计数,并根据容斥原理(计数时重复计数的部分要扣除)还需要在结果中加上 $p_1p_2, 2p_1p_2, \cdots, \lfloor\frac{n}{p_1}\rfloor p_2$ 的数量。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：

- 小发想要给小收发一条需要保密的信息。
- 小收取两个非常大的素数 p 和 q，并令 $N=p\cdot q$，$\varphi(N)=(p-1)\cdot(q-1)$。
- 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 $e\cdot d \bmod \varphi(N)=1$。
- 小收将 (N,e) 作为公钥 (pk, Public key) 发给对方，(N,d) 作为私钥 (pt, Private key) 保存。
- 小发接受公钥后，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。
- 小收接收密文，并使用私钥解密：$p=ct^d \bmod N$，也就是 $p\equiv ct^d(\mathrm{mod}N)$。

4 RSA 数字签名与数字证书

现在设想这样一个情形：小收公布公钥之后，收到了一条信息要求他给小发转账。于是小收转了 50，但第二天，小收却声明他从未发过类似于“vivo50”这样的信息。原来，小攻，作为一名黑客，冒充小发发了一条信息。小收发现了这个公钥体制的问题——没有办法证明信息的发送者。

同一时间，另一头的小攻也在为另一个问题苦恼——他发送的信息明明是“vivo500”，可是他仅仅收到了 50！原来，信息在传输的过程中被他的对手小黑（另一名黑客），改成了 vivo50！他咬牙切齿，因为小收的消息系统太烂了——根本没办法知道信息发送过程中有没有被篡改。

本章叙述的内容就是为了证明发信人是其自身，以及消息的完整性。

4.1 散列函数

验证信息没有被篡改很简单：使用散列函数，对原文进行“摘要”。散列函数应当满足这样的性质：当输入发生微小的变化的时候结果变化很大，并且难以找到有相同输出的输入，这样，当消息发生变化，用相同算法生成的“摘要”也就会发生变化。

4.2 数字签名与数字证书

RSA 算法原理保证了其有这样的性质：即使使用私钥加密，用公钥也可以解密。如果对此不明白，你应该重读第 3 章。因此，在上文的例子中，小收也可以制作一份密钥，并将公钥公布。在发送信息的时候，用自己的私钥对摘要进行加密。如果小攻想要修改信息，那么他必须知道小收使用的私钥——不然小发得到的消息和摘要将会不一致。

然而，小攻真名张荣宸，十分滴聪明狡猾。他用小收的密钥加密了“vivo500”的假消息，然后自己制作了一份公钥发到网上，用私钥对消息的摘要加了密。小收又上了当——这份消息看起来同样是小发的消息，也有“小发的签名”。

这时候需要由数字证书解决这个问题。证书授权 (CA, CertificateAuthority) 由相应的 CA 机构颁发给小收（经过线下验证），包括了小收的公钥、小收的身份信息、CA 的签名。这样，小发就可以找到真正的小收公钥与对应消息。

验证证书的合法性是套娃的过程。小收的公钥与小收的身份信息对应的摘要，应该与 CA 的解密后签名一致。CA 根证书是安装系统内置在系统或浏览器中的，这样如果还想信息造假，需要修改用户的系统或浏览器文件，而这通常是不可能的。

或许你在考虑一种情况，即小攻完全劫持了网络，不仅有能力发送与劫持信息，还有能力修改网络上已有的信息（虽然这概率本身很低）。但小攻会发现这是白费力气（在他没有能力分解公钥的前提下），因为任何消息的改变都会导致对应摘要的改变，而且我们在第 **??** 节中提到了摘要难以一致的特性。因此，这种情况下，小攻仍然无法完成欺骗。

因此，在层层保护的加持下，小攻更需要的，可能是线下潜入某位仁兄的房间，修改他们的电脑数据——但如果这样，他直接用他们的账户给自己转账似乎更合理些。

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.