Project Euler 182

Study Crypto

0x01 题设

题目描述:
PE182-中文翻译站

0x02 Solution

首先我们先去遍历可能的($(e, \phi(N)) = 1$)的所有e，然后借助于一条很好的结论来统计所有符合要求的e，并进行累加，最后得到的就是答案。

什么结论呢？

> Thm.1
> 如果这种(e,m)符合$m \equiv m^e(mod\ N)$，我们称之为一个暴露组合，那么在e固定的情况下，暴露组合的个数为:$[1 + (p-1, e-1)] * [1 + (q-1, e-1)]$
> （这里的p,q为N的两个素因子）

接下来给出的，就是这条结论的证明过程

这里要用到数论中有关同余与原根的一些理论

> Thm.2
> $\forall 1 \le i \le n, a \equiv b(mod\ m_i) \Leftrightarrow a \equiv b(mod\ [m_1, m_2, \ldots, m_n])$
>
> Thm.3
> 若$(a, m) = 1$，则$1 = a^0, a^1, \ldots, a^{ord_m(a)-1}$ 这$ord_m(a)$个数模m两两不同余。特别地，当a是模m的原根($ord_m(a) = \phi(m)$)时，这$\phi(m)$个数是模m的一个既约剩余系。(注意：m不一定为素数,a一般选取为模m的一个既约剩余系。)

现在，我们假设g为N的一个原根，也可以称之为生成元，对于任意一个在模p的既约同余系中的整数，都有存在一个x使得$g^x \equiv m(mod\ p)$。

而且$m \equiv m^e(mod\ N)$可以根据**Thm.2**转换成下列同余方程组:

$$\begin{cases} m \equiv m^e (mod\ p) - (1) \\ m \equiv m^e (mod\ q) - (2) \end{cases}$$

只讨论式(1)，式(2)同理：

将m用g^x进行替换

$$\begin{aligned} (g^x)^e &\equiv g^x (mod\ p) \\ g^{xe} &\equiv g^x (mod\ p) \\ xe &\equiv x (mod\ \phi(p)) \\ xe &\equiv x (mod\ p-1) \\ x(e-1) &\equiv 0 (mod\ p-1) \qquad (3) \end{aligned}$$

> 引理1:
> 同余方程$ax \equiv b(mod\ n)$有解当且仅当$(a, n)|b$
> 引理2:
> 同余方程$ax \equiv b(mod\ n)$**在模n的意义下**有$d = (a, n)$个解或者无解。

这两条引理证明之后给出= =

不过有一点很显然的是，对于每一个方程，都至少存在三个解，
理由如下（以p为例）：

因为$\phi(N)$为偶数，那么e必为奇数，e-1为偶数，又p-1为偶数，所以二者的最大公约数至少为2，最后再加1为3。

一般会为哪三个必存在的解呢？

Ans: `0`, `1`, `p-1|q-1|N-1`

对于第三个解存在性的证明，相当于证
$(X-1)^{2k+1} (mod\ X) \equiv (X-1)\ (k = 1, 2, 3, \ldots)$
切记： $m \equiv m^e (mod\ N) \nLeftrightarrow 1 \equiv m^{e-1} (mod\ N)$

请再次注意引理2，为什么是在模n的意义下，结合下文，一般情况下，`0`, `1`都是暴露组合的解，显而易见，`p-1`也是，但在这里模数是`p-1`所以，式(3)的解数为
$(e-1, p-1) + 1$.

不妨定义`Ans(p)`, `Ans(q)`如下：

$$\begin{cases} Ans(p) = 1 + (e-1, p-1) \\ Ans(q) = 1 + (e-1, q-1) \end{cases}$$

那么接下来便是合并计算个数的问题，或者可以说是一个从模p(或q)的剩余系映射到模N的剩余系中，求解暴露组合的个数。

这里就要引用中国剩余定理(CRT)来解决：

> 引理3: 同余方程组中子方程解的个数的乘积为该同余方程组的解的个数

具体证明见参看内容[1].本人之后会补充= =

因此答案为：

$$Ans(N) = Ans(p) * Ans(q)$$

参考内容:

[1]. Messages which equal their encryptions. – Solutions to Problem Set 3
[2]. Handbook of Applied Cryptography p.290

有时间可以去阅读另外一篇文章,与本题关系不是很大，但很有意思：
跨越千年的RSA算法