因子分解算法

RSA的安全性被认为完全取决于分解大整数的难度。更准确地说，给定N，其中N=pq，p和q为素数，如果能确定p或q，那么就能破坏RSA。因此，人们投入了大量精力来开发高效的因子分解算法。
下面会简单介绍几种整数分解算法，首先是最简单的穷举法，即尝试根号N一下的数字直到找到满足条件的素数对。
之后，会介绍Dixon算法，以及在其基础上进行改进的二次筛选法。这种算法成功适用于对于130位左右的数字进行因子分解。

穷举法

给定一个整数N，尝试使用每个在可能范围内的整数进行尝试，首先参与尝试的数都是奇整数，因此所需要的工作量约为$\sqrt{N}/2$。
进一步情况下，筛掉所有的非质数，工作量被降低到 $\pi(\sqrt{N})$,其中的`π(x)`是计算小于或者等于x的素数数量函数，当N是一个大数，可以认为π(n) ≈ $N/\ln(N)$ (这个公式暂时无从考究)
因此可以得出结论，理想情况下对于大数N的因子分解成本在 $N/\ln(N)$

Dixon算法

同样地，我们假设需要对于大数N进行因子分解，并保证其存在一个指数对因子。我们假设存在整数x和y，满足 $N = x^2 - y^2$，即 $N = (x-y)(x+y)$ ，那么就可以说找到了N的因子，更进一步说，如果能够找到x和y满足 $x^2 - y^2$ 数倍于N：
$x^2 = y^2(modN)$
我们可以将这个倍数定义为k，即 $(x-y)(x+y) = kN$。如果我们运气不好，存在 $(x-y) = k$ 和 $(x+y) = N$，况且现将这种情况排出在外，在剩下的可能性中，通过 $\gcd(N, x-y)$ 和 $\gcd(N, x+y)$ 以获得两个因子。

举个简单的例子：
$100 = 9(mod91)$, 显而易见可以得到：
x = 10
y = 3
因而得到91的两个因子 $\gcd(91, 7) = 13$ 和 $\gcd(91, 13) = 7$
这样的x和y还有数对，例如 $34^2 = 8^2(mod91)$，这就是为什么要使用gcd的原因。由于基于欧

几里得算法，gcd的计算难度极低，因此这个想法的难处就在于如何找寻到合适的x和y。
在此基础上，稍微放宽条件，比如一个 x^2 和一个不可开方的整数，如:
$41^2 = 32(mod 1649)$
$43^2 = 200(mod 1649)$
将这两个式子的左右两边相乘，得到:
$41^2 * 43^2 = 32 * 200(mod 1649) = 80^2(mod 1649)$
接下来的一切也就顺理成章，可以得出1649的质数因子 17*97。

在上述第二个例子中，将两个非平方组合成一个平方对。原因也很简单将32和200完全结构成质数相乘的组合后会发现:
$32 = 2^5 * 5^0$
$200 = 2^3 * 5^2$
$32 * 200 = 2^8 * 5^2$
要找到"完美平方"，只需要关注这些数的质数因子的幂，只要这些幂是偶数，其就是可以被开方的，因此作出如下定义：

$$32 \rightarrow \begin{bmatrix} 5 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} (mod 2)$$

$$200 \rightarrow \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} (mod 2)$$

$$200 * 32 \rightarrow \begin{bmatrix} 8 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} (mod 2)$$

如上式所示，为了确定是否具有平方同余，我们只需要幂的(mod 2)向量。

所考虑的数字的因子分解中素数种类数量决定了向量的大小。由于算法最终想要考虑大的数字，所以必须保持向量的维度尽可能小。因此，我们选择一个界B和一组素数，其中每个素数都小于B。这组素数就是因子基。虽然因子库中的所有素数都小于B，但通常并不是每个这样的素数都包含在因子库中。

一个在给定因子基础上完全因子化的数被称为`B-smooth`，通过限制设定B，也就是限制了幂向量的大小。因子库中的元素越少，需要处理的向量就越小，与之相对应的B-smooth也就越难以发现。

举个例子，假设N=1829， B=15

因此因子基包含: {-1, 2, 3, 5, 7, 11, 13}

由于我们想要因子较小的数字，因此处理介于-N/2和N/2之间的模数是有利的，而不是在0到N-1的范围内。

接下来选择一个随机数r，并判断 $r^2(modN)$ 是否是B-smooth，重复这一过程，直到获得足够多数量的B-smooth。

同时也可以使用一种更效率的方法：选择值 $\lfloor kN \rfloor$ 和 $\lceil kN \rceil$，对于k=1,2,3,4, 测试它们的平方是否为B-smooth。

$$
\begin{aligned}
42^2 &= 1764 = -65 = -1 \cdot 5 \cdot 13 (mod 1829) \\
43^2 &= 20 = 2^2 * 5 (mod 1829) \\
60^2 &= 1771 = -58 = -1.2 \cdot 29 (mod 1829) \\
61^2 &= 63 = 3^2 * 7 (mod 1829) \\
74^2 &= 1818 = -11 = -1 \cdot 11 (mod 1829) \\
75^2 &= 138 = 2 * 3 * 23 (mod 1829) \\
85^2 &= 1738 = -91 = -1 * 7 * 13 (mod 1829) \\
86^2 &= 80 = 2^4 * 5 (mod 1829)
\end{aligned}
$$

除了60和75外均为B-smooth，对于其他的因数我们可以定义一个七维的向量，从上到下分别代表因子-1,2,3,5,7,11和13.

$$
42^2 = -65 \to \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} 43^2 = 20 \to \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} 61^2 = 63 \to \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}
$$

$$74^2 = -11 \rightarrow \begin{bmatrix}1\\0\\0\\0\\0\\1\\0\end{bmatrix} 85^2 = -91 \rightarrow \begin{bmatrix}1\\0\\0\\0\\1\\0\\1\end{bmatrix} 86^2 = 80 \rightarrow \begin{bmatrix}0\\0\\0\\1\\0\\0\\0\end{bmatrix}$$

这些向量的任意组合的mod 2都应该是一个零向量，所以我们得到:

$$\begin{bmatrix}1\\0\\0\\1\\0\\0\\1\end{bmatrix} \bigoplus \begin{bmatrix}0\\0\\0\\1\\0\\0\\1\end{bmatrix} \bigoplus \begin{bmatrix}0\\0\\0\\0\\1\\0\\0\end{bmatrix} \bigoplus \begin{bmatrix}1\\0\\0\\0\\1\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\0\\0\\0\\0\end{bmatrix}$$

从而呈现的结果为：

$$\begin{aligned} 42^2 * 43^2 * 61^2 * 85^2 &= (-65) * 20 * 63 * (-91) \\ &= (-1 * 5 * 13) * (2^2 * 5) * (3^2 * 7)(-1 * 7 * 13) \\ &= 2^2 * 3^2 * 5^2 * 7^2 * 13^2 (mod 1829) \end{aligned}$$

也就是

$(42 * 43 * 61 * 85)^2 = (2 * 3 * 5 * 7 * 13)^2 (mod 1829)$

即

$1459^2 = 901^2 (mod 1829)$

易得 $1829 = 59 * 31$

这个例子提出了一些问题。例如，我们能非常确定地获得解决方案吗？如果是这样的话，在我们确定获得解决方案之前，需要经过多少尝试？这些问题可以用一些基本的线性代数得到肯定的回答。

更系统的方法是存在的：假设矩阵M，这个矩阵实质上也就是上面那六个B-smoooth向量组合而成，存在一个解x使得Mx=0，也就是说，这个算法实际要解决的问题，已经被缩小到求解向量x：

$$\begin{bmatrix}1&0&0&1&1&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\1&1&0&0&0&1\\0&0&1&0&0&0\\0&0&0&1&0&0\\1&0&0&0&1&0\end{bmatrix}\begin{bmatrix}x0\\x1\\x2\\x3\\x4\\x5\end{bmatrix}=\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\end{bmatrix}(mod2)$$

通常，如果n是因子库中的元素数（包括-1），那么n也是每个列向量中的元素数量，因此矩阵M有n行。

众所周知在线代中，如果矩阵M中至少有n+1列，那么我们一定可以找到列的线性相关集合。并且这个计算过程非常简单。

也就是说，对于n+1个或更多个B-smoooth关系，将一定能获得平方的同余，从而很有可能获得n的因子。

```
// Given integer N, find a nontrivial factor
Select B and factor base of primes less than B
n = number of elements in factor base (including -1)
// Find relations
m = 0
while (m < n)
    y = r2 (mod N) // r can be selected at random
    if y factors completely over the factor base
        then Save mod 2 exponent vector of y
        Save r2 and y
        m=m+1
    end if
end while// Solve the linear system
M = matrix of mod 2 exponent vectors
Solve Mx = 0 (mod 2) for vector x = (xo,x1,...,xn)
I = {i | xi = 1}
```

我们得到平方的同余 $\prod_I r_i^2 = \prod_I y_i(modN)$
计算所需的gcd并得到合理的质因子对。

值得强调的是，通过增加B，我们可以更容易地找到B-smoooth，但向量的维度会增加，从而使由此产生的线性代数问题更难解决。

但是这个算法的优势在于，计算B-smooth是可以并行的，通过给定k台不同的计算机，每台计算机都可以测试随机值以获得的不同的B-smooth。尽管最后，线性方程的求解是不平行的。

之后的二次筛使用了一种更有效且并不复杂的方法。它是Dixon算法的改进。其用于分解高达约110至115位小数的大整数。

二次筛选法

二次筛分解算法本质上是Dixon算法的进一步发散，相比于Dixon算法，其在寻找B-smooth上得到了进一步的加强，这两者求解最后的结果过程中，线性代数计算部分是相同的。
在Dixon算法中，给定一个界限B和相应的因子基，由此为了得出结果我们必须找到对应的B-smooth。

算法首先给出一个多项式:
$Q(x) = (\lfloor \sqrt{N} \rfloor + x)^2 - N$

使用这个式子来生成测试B-smooth的值，并由此式命名其为二次筛算法。

为了获得B-smooth，选择一个包含0的区间，比如[-M, M]，并对于其中的每一个整数x我们都要计算 $y = Q(x)$ ，接下来对N求模，有$y = \widetilde{x}^2$,即$\widetilde{x} = \lfloor \sqrt{N} \rfloor + x$
在讨论二次筛选算法中使用的筛子之前，我们先回顾一下Eratosthenes筛选法。假设需要找出所有小于31的素数。首先，我们列出从2到30的所有数字。

- 首先从4开始划掉所有的偶数，从6开始划去所有3的倍数，以此类推。这个序列中，我们从每一个素数出发，向后划去它们的倍数。

这个筛子不仅向我们提供了如何筛选素数，还提供了相当的非素数的信息，在筛选过程中将背个素数的倍数都打上对应的标记，筛选完成后我们便能够在每个非素数查看其所用有的标记。

于是二次筛的做法如下:

- 从2开始，对每个整除2的将每个数字除以2。
- 接下来从3开始，做同样的动作。
- 然后是5，接下来是7。假设我们在这一点上停下来。那么，与这个数组中的1现在占据的位置相对应的数字是7平滑的，也就是说，它们没有大于7的素数。然而，与非1相对应的一些数字也是7平滑的，例如28。

在QS算法中遵循一些重要的计算规则，以获得B光滑关系。通过试验划分来测试每个候选Q(x)的B光滑性是昂贵的。假设我们发现，素数p，其中p在因子基中，除以Q(x)。那么验证这一点就很容易了。
一旦我们确定Q(x)可以被p整除，我们就知道以下每一个的Q也可以被p整除:
$...x - 2p$，$x - p$，x，$x + p$，$x + 2p...$

通过对因子库中的T和其他素数的其他选择重复这一点，我们最终可以“筛选”出区间$[-M, M]$

中的B光滑整数。这个过程与上面讨论的Eratosthenes的筛选有点相似。

有几个技巧可以用来加快这个过程。例如，假设y=Q(z)可被p整除。那么y=0（mod p），根据Q的定义，我们得到:
$(\lfloor\sqrt{N}\rfloor + x)^2 = N(modp)$

因此，在我们的因子库中给定p，我们可以计算 $N(modp)$ 的平方根，比如说，s_p，和$p - s_p$，并使用它们立即确定[-M，M]的值序列，使得相应的Q（x）可被p整除。

由于存在一种有效的算法（Shanks-Tonelli算法）来实现这种思路，因此这种方法是有效的。
实际筛选过程如下。创建一个包含值Q(z)的数组，其中x=-M，-M+1,...-1,0,1,..., M-1，M。对于因子基中的第一个素数p,生成可被p整除的x∈[-M，M]的序列，如前一段所述。对于其中的每一个，我们都知道对应的数组元素可以被p整除，所以确定p的最高幂，该幂除以数组元素，并将该幂以模2的形式存储在对应于给定数组元素的幂向量中。还要将数组元素除以这个p的最高幂。对因子库中剩余的每个素数p重复此过程。
当完成筛选后，那些为1的数组元素将完全分解到因子基t上，而这些正是B-smooth。对于B-smooth，保留t的模2次方向量，并丢弃所有剩余的元素和向量。

筛分过程有一些相当明显的改进。然而也有几个不那么明显但至关重要的改进在实践中使用。最重要的是，可以使用廉价的近似“对数”计算来避免昂贵的除法运算。结果只是近似值，因此筛选的幸存者将需要进行二次测试，以确定它们是否真的是B-smooth的。
总之，QS算法可以被视为Dixon算法的改进版本。由于筛选而产生的加速比是显著的，并且对于多个多项式，筛选间隔[-M，M]可以小得多。这能够保证更好的并行实现。