主要算法

1. 总体流程

证明者的工作流程如下：

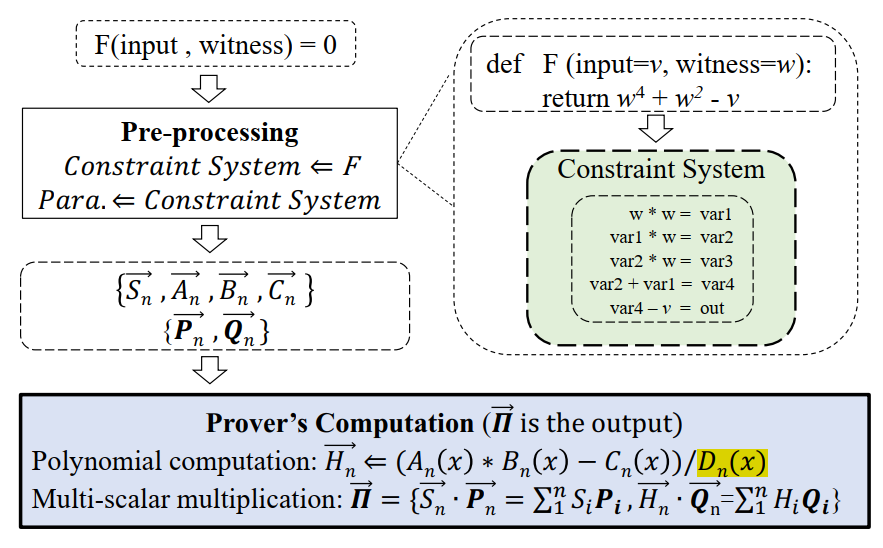


图 1证明者处理流程

证明者首先进行预处理，算数约束函数为（R1CS），包含输入和见证的多个线性或多项式方程。同时产生各种随机参数包括证明密钥。预处理输出两组数用于后续计算。

是标向量，n由约束系统决定。

是点向量。每个向量为预置椭圆曲线n个点。

利用硬件加速计算生产证明如下图。主要包括POLY和MSM。

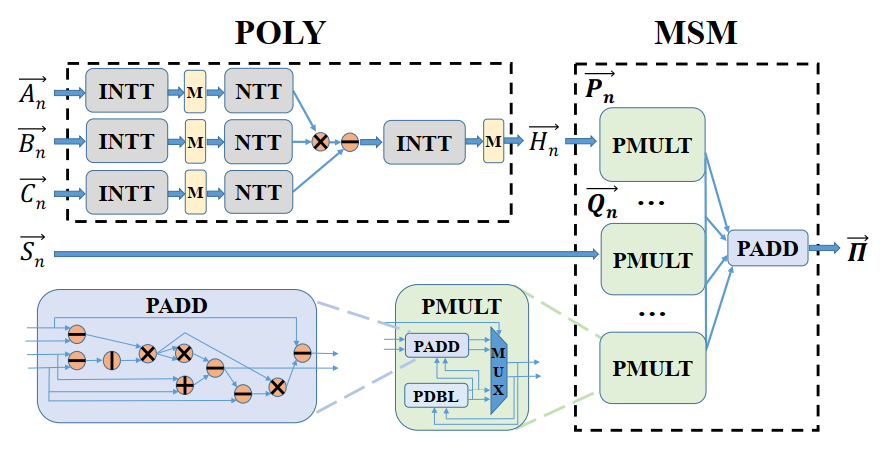


图 2硬件加速

1. MSM

包括、和**、**的椭圆曲线向量内积运算（包括**PADD,PMULT**）。

数学公式：

**算法1：**PMULT转为PADD和PDBL

1. 将ki表示为二进制，自低位到高位在二进制数每一位的位置上进行一次PDBL（相对前一位）运算（注意是每一位的位置，而不是每一位数）。
2. 将二进制数各个位上为1的PDBL结果进行相加。

手机屏幕截图

描述已自动生成

**算法2：**Pippenger算法

1. 选定一个*s*窗口，将*λ*位的二进制数标量*k*表示分成*λ/s*个块（chunk）；每块中含有个有效组（去掉0000的组）。
2. 计算每个chunk的**Gj**，黑色的钟表

   中度可信度描述已自动生成；其中bi[j]表示第j个chunk中的系数，二进制表示0000，0001，0010,…，1111（1，2…，）。
3. 将**Gj**以系数累加获得最终结果。

详细数学表示如下：

文本

中度可信度描述已自动生成

日历

中度可信度描述已自动生成

**算法3：**PADD算法（参考【4】【5】【6】）

~~椭圆曲线上，P=(), Q=(), W()=P+Q的加法规则:~~

对定义在有限域上的椭圆曲线



:

当时，，根据定义，，结果为理想点（无穷远点）。

当时，有





对于*m*，当时

（模逆至少有一次）

当时



综上，

**算法4：**PDBL算法

~~椭圆曲线上，P=(), W()=2P的2倍加法规则:~~

标量积（scalar multiplication）的定义可以从加法扩展，其中 k为正整数



通过调用已实现的加法并循环累加结果，可以直接求得点的坐标，但这样要做k-1次加法运算。因为椭圆曲线上点的加法，满足交换律和结合律，所以可以用PDBL改进。例

改进后的计算过程为

※计算 2 P = P + P，即 

※计算 4 P = 2P + 2P，即 

※计算 8 P = 4P + 4P，即 

※……

※计算 

※最后计算。

随着 k 的增大，“倍乘法”能显著提升 k P 的计算效率。

**算法5：**扩展欧几里得算法

扩展欧几里得算法应该是最优的求逆元算法之一 ，他和费马小定理具有同样的时间复杂度O(log(n))，但是费马小定理需要模数为质数，扩展欧几里得算法则不需要。

若e与m互素，满足，则d为e的逆元。

显然，有（y为任意常数），又因为，所以有，也就是



只要求得任意一组满足 不等式的d，y。d就是e的逆元。

**引理：存在 x , y 使得 gcd(a,b)=ax+by**

**用扩展欧几里德算法的过程如下：**

求exgcd(a, x)—>利用欧几里得算法不断递归直到m=1,n=0—>反向递归求出第一层的m和n，m即为e模m的逆元。

**算法6：**费马小定理



所以，，即为T的逆元。求逆元转为求T的幂模，可用蒙哥马利快速幂模算法实现。

**算法7：**蒙哥马利约简

蒙哥马利模乘约减的思路是通过变换，将需要取模的数控制到很小的范围（由[ 0 , N 2 − 2 N + 1 ] 变 为 [ 0，2 N − 1 ]，这样只需要通过最多一次减法即可完成取模运算；同时在变换过程中，通过选择除数，将除法的开销降到最小（比如在计算机中，除以2的幂通过移位即可实现，开销大大降低）

令 T ∈ Z 且 T < R × N

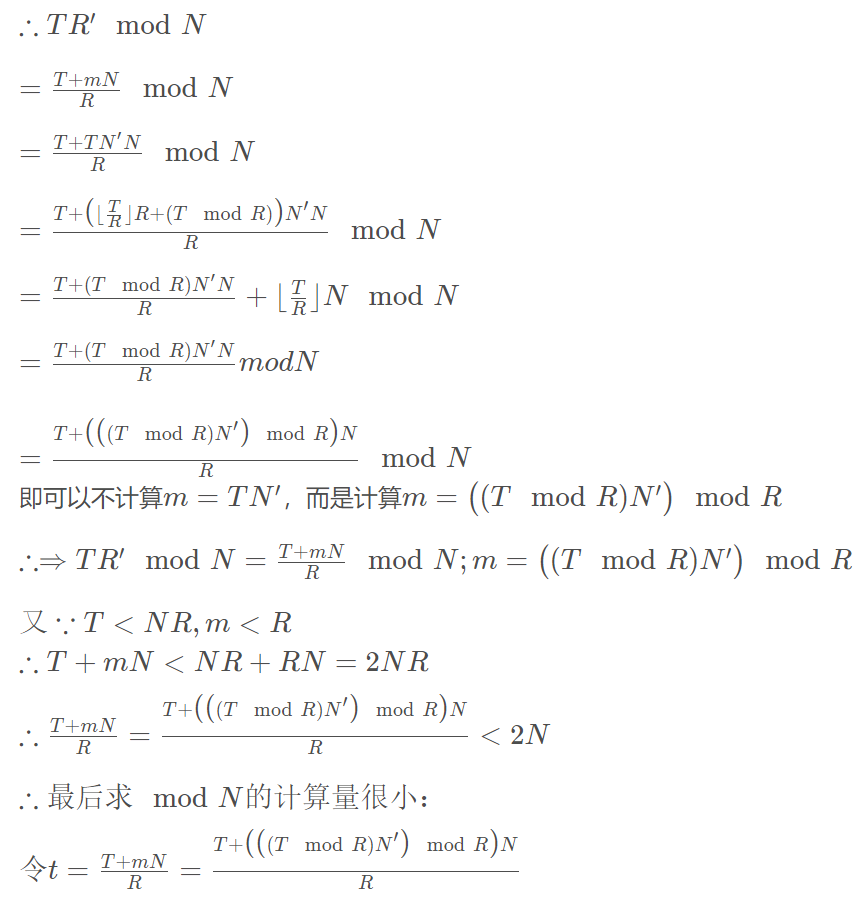
有：T = T × 1 = T ( R R ′ − N N ′ ) = T R R ′ − T N N ′

∴ T + T N N ′ = T R R ′

令m = T N ′

∴ T + m N = T R R ′

∴ 对 任 意 T < N R , ∃ m 使 ( T + m N ) / R 是 整 数。



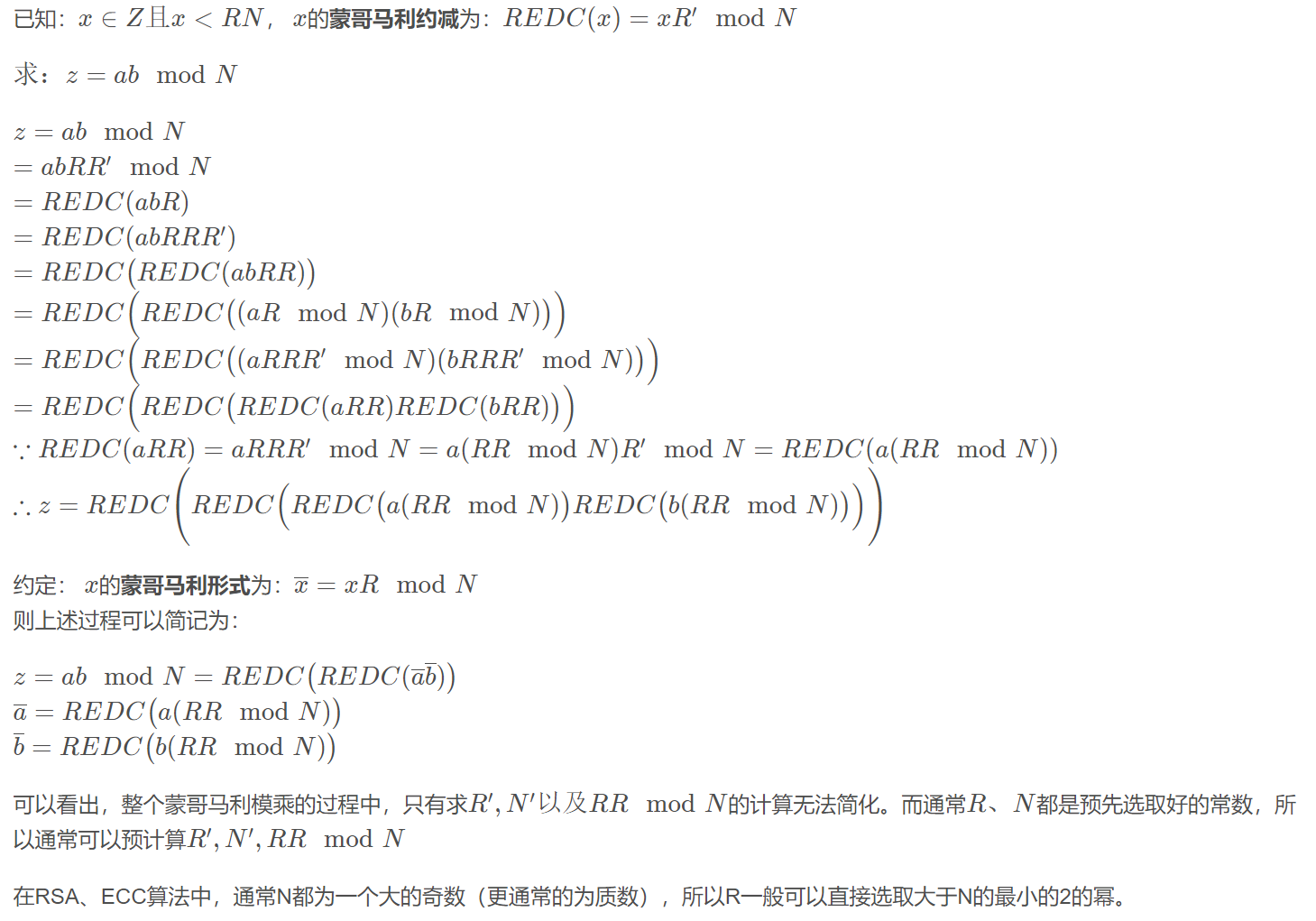
同时，R选取满足gcd(R,N)=1且是2的幂的整数，除以R的运算转变为移位运算，对R求模的运算变为与运算。

所以通过选择R以及上述变换，可以快速求取TR′mod N。

上述变换，即为蒙哥马利约减算法Montgomery Reduction，记为REDC

即：R E D C ( T ) = T R ′ m o d    N（利用蒙哥马利求逆要求T<2^(length（N）)）

**算法8：**蒙哥马利模乘



详细请见“【1】【2】【3】”。

1. NTT

3.1 NTT的表示

对于，若的值互不相同，则称g为p的原根。

FFT利用单位根的性质实现分治优化多项式乘法，原根也有类似性质。NTT利用原根代替FFT的。这样可以不用复数实现取模运算。

设g是p的原根，令，其中要求能被N整除，原根的四个性质：









**NTT的原根表示：**

NTT: 

INTT: 

这里的指的是在模M下的逆元。

3.2 蝴蝶变换（参考【7】【8】）

对于；按照下标奇偶分组



令，

可得。

当，时，（左半部分）

（折半定理）

当，时，（右半部分）



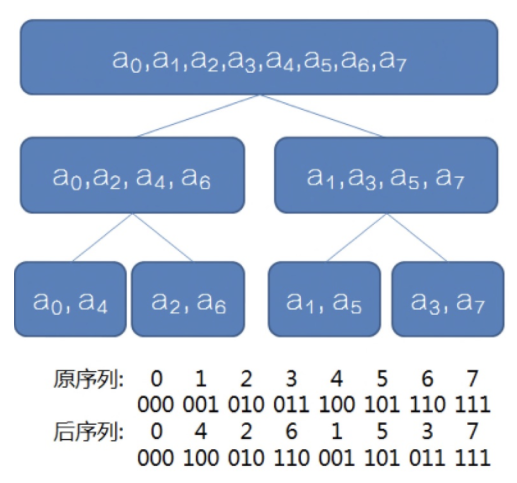
其中，（折半定理）

由消去定理，，所以



结论，若已知和在的取值，即可求出的值。而和为的一半规模，显然可转化为子问题递归求解。

下图为递归求解的蝴蝶变化示意，。可以看到最后得到的序列就是原序列每个二进制的反转（bit-inverse操作）。



求反转后序列的位逆操作为：

首先，从小到大求，此时是已知的。因此



3.3 NTT并行算法（参考文中的[21][49]和【9】）

可以将规模为N的NTT分解成相互独立的规模更小的NTT以便于并行实现，分解也使得内存管理变得更加灵活。将输入向量看成一个二维的向量（m × M），这里，N , I , J都是2的指数次方量级的。那么，a的各个成分就可以表示为：



可以看到，这里的j指标变得更快，而J指标变得更慢。

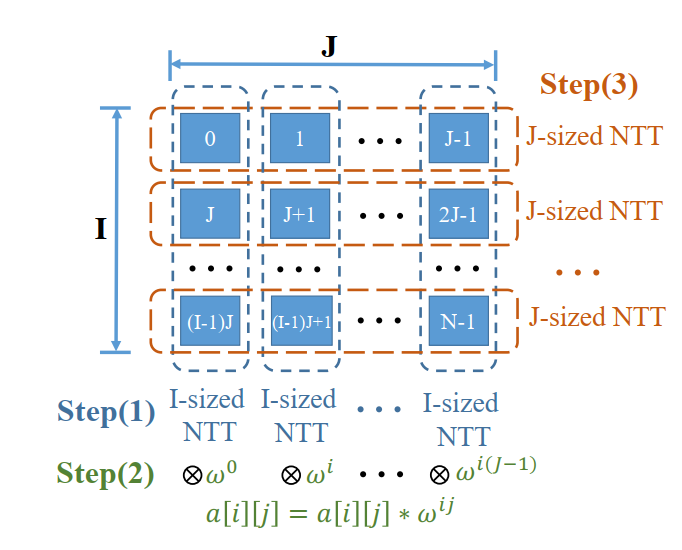
原来的NTT就可以写为:



这里的k和K决定了结果的下标。做恒等变换得到



从这里发现，这个分解可以得到如下的并行实现的步骤（参照论文中的图）：



步骤1：把输入的待变换向量表示为的矩阵。首先对矩阵的每一列作长度为的NTT变换；

步骤2：列变换后的矩阵元素乘以各自的**旋转因子**，此处的为整个向量的单位原根，即上文中的；

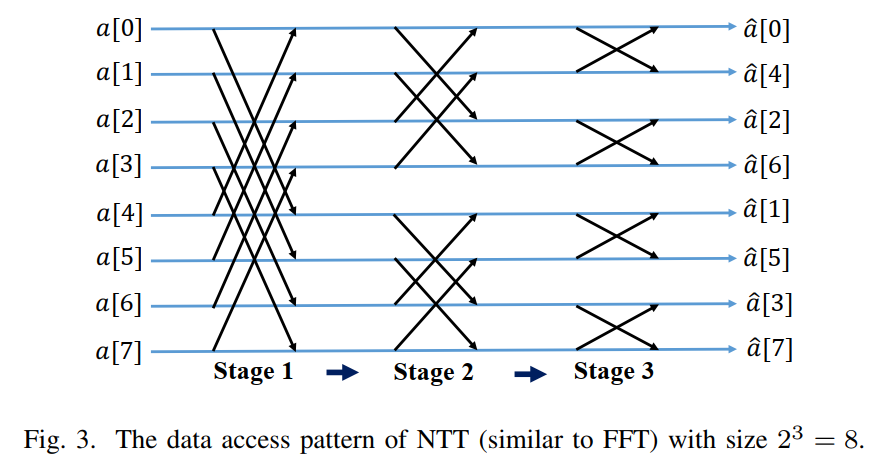
步骤3：乘完旋转因子的矩阵再对每一行进行长度为的NTT变换，并对变换后的矩阵进行按列排序输出，即为最终的NTT结果。

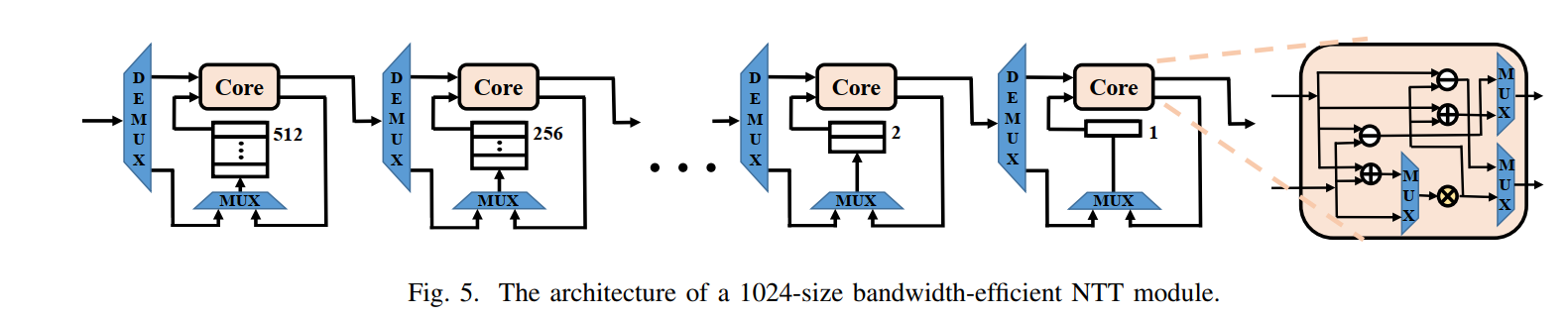
可以看到，并行处理方法不存在位逆操作。

下图为n = 16的NTT蝶形变换示意图。由图可以看到，其实并行处理本质上是将蝶形变换从中间截断。以4×4为例，阶段位置为第二级变换位置。即先分别对列向量序号为0，4，8，12；2，10，6，14；1，9，5，13；3，11，7，15；的元素分别进行一次NTT，然后乘以旋转因子，再进行后续的行NTT得到最终的结果。



3.4小尺寸NTT流水线设计（参考文中的[34]）





文中图五是1024尺寸NTT的pipeline优化设计，它包含10个阶段。每个阶段有一个NTT核在两个元素间进行蝶形运算。并且产生两个新的元素给下一阶段使用。核中的数学运算包含13个周期延迟。FIFO的深度匹配每个阶段的NTT步长：第一阶段FIFO深度为512，第二阶段FIFO深度为256，以此类推。Pipeline模块每个周期从外部存储器中读取一个元素，在第一个阶段的第一个512周期，512个元素保存到 FIFO中，在下一个512周期启动NTT核，用新读取的元素和pop出的顶部元素作为两个输入，两个元素的期望步长是512。这样就可以用FIFO而不是多路复用器来正确的执行跨距。每个周期NTT核心产生两个输出元素，其中一个直接发送到下一阶段，另一个放入本级FIFI缓冲区（尾部进入）。下一阶段遵循同样的规则，但是利用不同的FIFO深度来实现不同的步长。最后一级将输出写回存储器。

3.5 费马素数

数论变换中，要求p是素数且N是p-1的因子。由于N是2的幂，所以构造的素数。通常P为费马素数。

|  |  |  |  |
| --- | --- | --- | --- |
| r⋅2k+1 | r | k | g |
| 3 | 1 | 1 | 2 |
| 5 | 1 | 2 | 2 |
| 17 | 1 | 4 | 3 |
| 97 | 3 | 5 | 5 |
| 193 | 3 | 6 | 5 |
| 257 | 1 | 8 | 3 |
| 7681 | 15 | 9 | 17 |
| 12289 | 3 | 12 | 11 |
| 40961 | 5 | 13 | 3 |
| 65537 | 1 | 16 | 3 |
| 786433 | 3 | 18 | 10 |
| 5767169 | 11 | 19 | 3 |
| 7340033 | 7 | 20 | 3 |
| 23068673 | 11 | 21 | 3 |
| 104857601 | 25 | 22 | 3 |
| 167772161 | 5 | 25 | 3 |
| 469762049 | 7 | 26 | 3 |
| 998244353 | 119 | 23 | 3 |
| 1004535809 | 479 | 21 | 3 |
| 2013265921 | 15 | 27 | 31 |
| 2281701377 | 17 | 27 | 3 |
| 3221225473 | 3 | 30 | 5 |
| 75161927681 | 35 | 31 | 3 |
| 77309411329 | 9 | 33 | 7 |
| 206158430209 | 3 | 36 | 22 |
| 2061584302081 | 15 | 37 | 7 |
| 2748779069441 | 5 | 39 | 3 |
| 6597069766657 | 3 | 41 | 5 |
| 39582418599937 | 9 | 42 | 5 |
| 79164837199873 | 9 | 43 | 5 |
| 263882790666241 | 15 | 44 | 7 |
| 1231453023109121 | 35 | 45 | 3 |
| 1337006139375617 | 19 | 46 | 3 |
| 3799912185593857 | 27 | 47 | 5 |
| 4222124650659841 | 15 | 48 | 19 |
| 7881299347898369 | 7 | 50 | 6 |
| 31525197391593473 | 7 | 52 | 3 |
| 180143985094819841 | 5 | 55 | 6 |
| 1945555039024054273 | 27 | 56 | 5 |
| 4179340454199820289 | 29 | 57 | 3 |

注：NTT预先计算参数

给的数据中是253位，即；

需要预计算的参数包括：

1. 素数的逆元，其中， 而。
2. 原根的逆元，其中.
3. 的逆元，其中.
4. 参数。