In[8]:= **PNP =**

```
PNP =
947 247 806 476 271 646 660 443 698 107 028 073 252 397 481 758 015 020 137 732 882 767 954 165 ⋱
338 379 903 567 765 249 477 543 857 038 469 778 782 688 273 687 701 107 719 862 005 082 701 ⋱
006 676 383 952 668 942 595 233 770 641 066 878 087 656 475 006 725 087 863 521 188 908 280 ⋱
123 681 221 617 022 280 485 987 353 493 866 887 341
```

Out[8]=

```
947 247 806 476 271 646 660 443 698 107 028 073 252 397 481 758 015 020 137 732 882 767 954 165 ⋱
338 379 903 567 765 249 477 543 857 038 469 778 782 688 273 687 701 107 719 862 005 082 701 006 ⋱
676 383 952 668 942 595 233 770 641 066 878 087 656 475 006 725 087 863 521 188 908 280 123 681 ⋱
221 617 022 280 485 987 353 493 866 887 341
```

In[19]:=

```
ScientificForm[N[PNP, 249]]
```

Out[19]//ScientificForm=

```
9.47247806476271646660443698107028073252397481758015020137732882767954165338379900⋱
  35677652494775438570384697787826882736877011077198620050827010066763839526 6⋱
  8942595233770641066878087656475006725087863521188908280123681221617022228048⋱
  5987353493866887341 × 10^248
```

N is fixed. We now guess q,where N = q * p. (It is usually p * q, but for the sake of keeping all work uniform q is the smallest factor.)

I am guessing with a guess of 1/8th the value. (Most LCF’s are usually small.) This will give me a view if the magnitude is greater or higher than my N guess. N guess is bigger than the actual N my guess of q is too high. If N guess is too small, my guess of q is too small. But I am looking for a q that will yield the closes value of were N = N guess or 0 = 0. But because of the error in the equation it is just a guess.

So into this equation:

(Sqrt[((((q^2 * PNP^4 + 2* PNP^2 * q^5) + q^8) / PNP^4) * ((PNP^2/ q^2)))])

Not the simpllified equaiton I gave you earlier. That simplification did not follow the rules of parathesis. Also not there are at least four other equations we could use. But we will use the one above.(I will share those extra equations. They are in my book which is free until April 5th.)

Also note because calling a variable “N” in *Mathematica* conflics with the keyword PNP will be the variable for N.

In[9]:=

```
q = PNP * 0.125
```

Out[9]= 1.18406×10^{248}

In[13]:=
```
NumberForm[q, 249]
```
Out[13]//NumberForm=
```
1.18405975809534 × 10^248
```

In[14]:=
```
NumberForm[1.18406 × 10^248, {249, 248}]
```
Out[14]//NumberForm=
```
1.1840597580953400000000000000000000000000000000000000000000000000000000000000000
   000000000000000000000000000000000000000000000000000000000000000000000000000
   000000000000000000000000000000000000000000000000000000000000000000000000000
   0000000000000000000 × 10^248
```

Place this q into the equation for the first test. q had to have as many sinificant firgures as PNP. Not pretty, but it doesn’t matter on the first guess.

In[15]:=
```
(Sqrt[((((q^2 * PNP^4 + 2 * PNP^2 * q^5) + q^8) / PNP^4) * ((PNP^2 / q^2)))])
```
Out[15]=
```
1.752496888425992 × 10^495
```

Look at scientific notation of the PNP given versus PNP guess. PNP guess is much too high in magnitude. For a second guess we will use an even smaller q guess. We will make it very small because we are not concerned with accuracy yet, only magnitude. We will manipuate the exponent.

In[26]:=
```
q =
 1.1840597580953400000000000000000000000000000000000000000000000000000000000000000
   00000000000000000000000000000000000000000000000 × 10^126
```
Out[26]=
```
1.1840597580953400000000000000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000 × 10^126
```

New q back into the equation:

In[29]:=

```
PNP
q
Sqrt[(((q^2 * PNP^4 + 2 * PNP^2 * q^5) + q^8) * PNP^2) / (PNP^4 * q^2)]
```

Out[29]= 947 247 806 476 271 646 660 443 698 107 028 073 252 397 481 758 015 020 137 732 882 767 954 165 338 379 903 567 765 249 477 543 857 038 469 778 782 688 273 687 701 107 719 862 005 082 701 006 676 383 952 668 942 595 233 770 641 066 878 087 656 475 006 725 087 863 521 188 908 280 123 681 221 617 022 280 485 987 353 493 866 887 341

Out[30]= $1.18405975809534000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 \times 10^{126}$

Out[31]= $9.472478064762716466604436981070280732523974817580150201377328827679541653383799035677652494775438570384697787826882736894 5360 \times 10^{248}$

Right magnitude. Now to guess at the accuracy. If my q guess is correct and varialbles have been cleared with proper values. The number looks to close to PNP given but it does differ. Now I would take PNP guess minus PNP given and divide the result by q and add it to q for more error.

Thanks for looking at my problem. It must not work. But I would like to share with you the four other equations that seem to work. I have yet to find a relationship between them. But if you follow the link I gave previously my book is free till April 5th.

Again thanks for the interest in my problem. I needed someone with expert math knowledge to review it. But I don't think my *Mathematica* variables are clearning. I shouldn't have gotten such a close N given equals N guess in 2 guesses. If anything I learned a lot of cryptography researching this problem. It is time to retire this problem and do some other coding.

Thanks again.

Test equation:

In[32]:=

```
y = ((PNP^2 / q) + q^2) / PNP
```

Out[32]= $7.9999999999999970158638203339546517228100393918863688345659344863842804962990066698648035624486852957811949654378031897353747 3 \times 10^{122}$

In[33]:= **y * q**

Out[33]= 9.4724780647627164666044369810702807325239748175801502013773288276795416533837990⋱
356776524947754385703846977878268827368945360 × 10^{248}