Week 6 - Programming Assignment [Optional: Extra Credit]

4 questions

Submit Quiz

1 point

1.

Your goal in this project is to break RSA when the public modulus N is generated incorrectly. This should serve as yet another reminder not to implement crypto primitives yourself.

Normally, the primes that comprise an RSA modulus are generated *independently* of one another. But suppose a developer decides to generate the first prime p by choosing a random number R and scanning for a prime close by. The second prime q is generated by scanning for some other random prime also close to R.

We show that the resulting RSA modulus $N = pq$ can be easily factored.

Suppose you are given a composite N and are told that N is a product of two relatively close primes p and q, namely p and q satisfy

$$|p - q| < 2N^{1/4} \quad (*)$$

Your goal is to factor N.

Let A be the arithmetic average of the two primes, that is $A = \frac{p+q}{2}$. Since p and q are odd, we know that $p + q$ is even and therefore A is an integer.

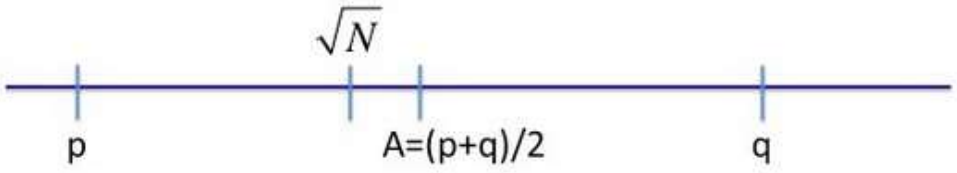
To factor N you first observe that under condition (*) the quantity $\sqrt{N}$ is very close to A. In particular, we show below that:

$$A - \sqrt{N} < 1$$

But since A is an integer, rounding $\sqrt{N}$ up to the closest integer reveals

the value of A. In code, $A = \text{ceil}(\text{sqrt}(N))$ where "ceil" is the ceiling function.

Visually, the numbers $p, q, \sqrt{N}$ and A are ordered as follows:



Since A is the exact mid-point between p and q there is an integer x such that $p = A - x$ and $q = A + x$.

But then $N = pq = (A - x)(A + x) = A^2 - x^2$ and therefore $x = \sqrt{A^2 - N}$.

Now, given x and A you can find the factors p and q of N since $p = A - x$ and $q = A + x$. You have now factored N !

Further reading: the method described above is a greatly simplified version of a much more general result (http://dl.acm.org/citation.cfm?id=1754517) on factoring when the high order bits of the prime factor are known.

In the following challenges, you will factor the given moduli using the method outlined above. To solve this assignment it is best to use an environment that supports multi-precision arithmetic and square roots. In Python you could use the gmpy2 (http://readthedocs.org/docs/gmpy2/en/latest/mpz.html#mpz-methods) module. In C you can use GMP (http://gmplib.org/).

Factoring challenge #1:

The following modulus N is a products of two primes p and q where $|p - q| < 2N^{1/4}$. Find the smaller of the two factors and enter it as a decimal integer in the box below.

```
N = 17976931348623159077293051907890247336179769789423065727343008115 \
773267580550562068698537944921298295958550138753716401571013985 86
```

```
\
478337786069255834975410851965916151280575759407526350074759352 88
\
710823649949940771895617054361149474865046711015101563940680527 54
\
007158456087857766374304008634074285527854909258 1
```

For completeness, let us see why $A - \sqrt{N} < 1$. This follows from the following simple calculation.

First observe that
$A^2 - N = \left(\frac{p+q}{2}\right)^2 - N = \frac{p^2+2N+q^2}{4} - N = \frac{p^2-2N+q^2}{4} = (p-q)^2/4.$

Now, since for all x, y : $(x-y)(x+y) = x^2 - y^2$ we obtain
$A - \sqrt{N} = (A - \sqrt{N})\frac{A+\sqrt{N}}{A+\sqrt{N}} = \frac{A^2-N}{A+\sqrt{N}} = \frac{(p-q)^2/4}{A+\sqrt{N}}.$

Since $\sqrt{N} \leq A$ it follows that $A - \sqrt{N} \leq \frac{(p-q)^2/4}{2\sqrt{N}} = \frac{(p-q)^2}{8\sqrt{N}}.$

By assumption (*) we know that $(p-q)^2 < 4\sqrt{N}$ and therefore $A - \sqrt{N} \leq \frac{4\sqrt{N}}{8\sqrt{N}} = 1/2$ as required.

Enter the answer for factoring challenge #1 in the box below:

Enter answer here

1 point

2.

Factoring challenge #2:

The following modulus N is a products of two primes p and q where $|p-q| < 2^{11} N^{1/4}$. Find the smaller of the two factors and enter it as a decimal integer.

Hint: in this case $A - \sqrt{N} < 2^{20}$ so try scanning for A from $\sqrt{N}$ upwards, until you succeed in factoring N.

```
N =648455842808071669662824265346772278726343720706976263060439070378 7 \
973086180811164627140152760614175691955873218402545206554249067198 9 \
242884484183935328197298853131051173864896596258282150250499026445 2 \
100885281673303711142296421027840289307657458645233683357077834689 7 \
158386460882396402368662522117900857878 77
```

Enter the answer for factoring challenge #2 in the box below:

Enter answer here

1 point

3.

Factoring challenge #3: (extra credit)

The following modulus N is a product of two primes p and q where $|3p - 2q| < N^{1/4}$. Find the smaller of the two factors and enter it as a decimal integer.

Hint: first show that $\sqrt{6N}$ is close to $\frac{3p+2q}{2}$ and then adapt the method in challenge #1 to factor N.

```
N =720062263747350425279564435525583738338084451473999841826
5305798191 \
635569018833779042340866418766393848517526499401789708352407
1356868 \
774411551320151882793318123090919962463618968365736431191740
4961348 \
524639707885238799396839230364676670221627018353299443241192
7381272 \
92761475307485973021927513757393879 29
```

Enter the answer for factoring challenge #3 in the box below:

Enter answer here

1 point

4.
The challenge ciphertext provided below is the result of encrypting a short secret ASCII plaintext using the RSA modulus given in the first factorization challenge.

The encryption exponent used is $e = 65537$. The ASCII plaintext was encoded using PKCS v1.5 before the RSA function was applied, as described in PKCS (https://www-origin.coursera.org/learn/crypto/lecture/JwjDq/pkcs-1).

Use the factorization you obtained for this RSA modulus to decrypt this challenge ciphertext and enter the resulting English plaintext in the box below. Recall that the factorization of N enables you to compute $\varphi(N)$ from which you can obtain the RSA decryption exponent.

```
Challenge ciphertext (as a decimal integer):
220964518674103817763065611348834180174100697878928310717318391436761356001205380042823296504735094243439462197515122564658399679428894607645420405815647489880137348641204523252293201764879166664029975091887299716905260832220677716000193292608700095799937240774589677736978175712672299511486629596279347915540
```

After you use the decryption exponent to decrypt the challenge ciphertext you will obtain a PKCS1 encoded plaintext. To undo the encoding it is best to write the decrypted value in hex. You will observe that the number starts with a '0x02' followed by many random non-zero digits. Look for the '0x00' separator and the digits following this separator are the ASCII letters of the plaintext.

(note: the separator used here is '0x00', not '0xFF' as stated in the lecture)

Enter answer here

4 questions unanswered

Submit Quiz