In-Class Exercise

CSCI3381-Cryptography

October 9, 2014

On one level, this is just a bunch of routine calculations to make sure you understand the number-theoretic concepts and algorithms we've been talking about for the past week or two. When you've finished, we'll talk about what this has to do with cryptography.

You can and should use a computer or a calculator, but, with the exception of the last question, use it only to compute products, quotients and remainders, and sequence the higher-level calculations by hand (this applies especially to Problems 4 and 6, where there is something more to do than just multiply and divide.) This is to make sure that you understand all the underlying algorithms in detail.

1. Set $p = 43$, $q = 59$. (These, of course, are primes.) Compute pq. Call this value N.

2. Compute $(p - 1)(q - 1)$. Call this value m.

3. Find the smallest odd positive integer $e > 1$ such that $\gcd(e, m) = 1$. You can really do this by simple inspection, testing $e = 3, 5, \ldots,$ in turn, without having to resort to Euclid's algorithm. (This is because you know small factors of m.)

4. Find $d = e^{-1} \bmod m$. We know this exists because e was chosen relatively prime to m. (You have to use the extended Euclid algorithm for this, but it's especially easy in this case.)

5. Pick an integer P between 0 and $N - 1$. Try to 'act random' when you pick it.

6. Compute $C = P^e \bmod N$. Do not use the built-in `pow` function in Python for this. Instead, write out all the multiplications and divisions you have to perform to compute this by repeated squaring (there should be three multiplications and three divisions).

7. Now compute $C^d \bmod N$. Since d is rather large, this will be tedious to compute by hand, so you can use the `pow` function in Python. But try to estimate the number of multiplications and divisions the repeated squaring algorithm performs to evaluate this. Compare your result to the number you chose in 5. If they're not equal, go back and check your work.