Jacob Kilver

Introduction to Information Security

Assignment 3

20 Nov 2016

Task 2 – Attack Small Key Space

Find p and q

To factor the public key N I used the algorithm based on the one outlined here: http://www2.mae.ufl.edu/~uhk/FACTORING-LARGE-COMPOSITE-NUMBERS.pdf. We know that N is the product of primes p and q:

$$N = p * q$$

We also know that the sum of two prime numbers is an even number. We can represent the sum of p and q as follows:

$$p + q = 2n$$

Where n is the average integer value of p and q. We eliminate q from the equations to solve for p

$$p^2 + 2pn + N = 0$$

$$p = n - \sqrt{n^2 - N}$$

Substituting p, we can solve for q:

$$q = n + \sqrt{n^2 - N}$$

To search for p and q, I iterated through different values of n. I started at the square root of N and worked my way up. Once I found an integer value for p using the equation above, I solved for q and exited. This seemed to efficiently find the values for p and q.

Find private key

From lecture and the textbook, we know that the private key must satisfy the following equation:

$$de \bmod \varphi(N) = 1$$

where φ(N) is the totient of public key N. I interpreted this equation to be the following equivalent statement:

$$de = k\varphi(N) + 1$$

$$d = \frac{k\varphi(N) + 1}{e}$$

where k is any positive integer. Integer k handles the fact that the modulo operator repeats. This formulation allowed me to search the private key space much more efficiently since I only had to search values of k until the k equaled e. For some reason that I am still exploring, the equation above did not yield a valid d for every value of k. I had to explicitly check if the value of d satisfied the identity in the first equation. If it did, I found the private key, and so I exited the loop.

Task 3 – Where is Waldo?

Understanding of vulnerability

The basis of RSA encryption is finding the product of 2 large prime numbers is difficult to do. Indeed, a simple brute force implementation for Task 2 with just a 64-bit key locked up my computer. However, finding the greatest common divisor of two numbers, even large ones, is a relatively simple computation. If two public keys share a common divisor (other than 1), both these keys can be compromised by decomposing them into their separate primes (p and q). The shared prime is p can be computed quickly and the q for each key can be found by simply dividing the public key by p.

Note: You need a collection of public keys to unlock this vulnerability. Also, this attack assumes there is a deficiency, a lack of entropy, when the keys are generated. If there is adequate entropy the keys should not share common divisors.

Getting the private key

To determine the "waldo" I had to find the greatest common divisor between two public keys. I used the division-based version of the Euclidean algorithm to accomplish this. If the two public keys shared a non-unary divisor, the "waldo was found.

After finding the shared prime p between the 2 public keys, the respective q's could be found by dividing the public keys by p. In other words

$$q_1 = \frac{N_1}{p}$$

$$q_2 = \frac{N_2}{p}$$

Where N_1 and N_2 are the respective public keys. The e was provided by the public key as well, so once p, q, and e were found, the private could be computed just as in task 2 before. In fact, I used the exact same code to compute the private key in task 3 as I did in task 2. Finally, I validated encryption and decryption using the provided public key and the computed private key to ensure that everything was correct.