Boston University
Summer I 2008

MA341 Number Theory
Allan Maymin

The application of prime numbers to RSA encryption

Prime number definition: Let us begin with the definition of a prime number p. The number p, which is a member of the set of natural numbers N, is considered to be *prime* if and only if the number p has exactly two divisors: 1 and p. It is important to note that the number one (1) is not considered a prime because it only has one divisor.

Numbers that are not prime are defined as *composite*. From this definition, we can see that a prime number can never be even, since it would have the divisor of 2. Of course, the only exception to that rule is the prime number 2, which has no other divisors but itself. Therefore, we can extend the definition to say that every prime number greater than 2 must be odd. From the above definition, we can also see that a prime number can never end in 0 or 5, since such numbers would have divisors of 5.

Primality testing: Given an integer N, we want to find the factors of N. If no such factors of N exist, then we can state that N is a prime number. The most complete solution to this problem is to perform an exhaustive search over all possible factors. This is called the full trial division algorithm and it is defined below:

> Given a composite integer n , trial division consists of trial-dividing n by every prime number less than or equal to $\sqrt{n}$. If a number is found which divides evenly into n, that number is a factor of n. (Wikipedia)

Let us take two large prime integers p and q. If we take the composite integer pq, we now have a very large composite, called a *semi-prime*, with only two large prime divisors: p and q. This creates an integer factorization problem of trying to find large prime divisors of an even larger composite number. This problem is at the heart of cryptography and we will explore it in more detail.

RSA Algorithm: This assumption forms the base of the Rivest-Shamir-Adleman algorithm, also known as the *RSA algorithm*. This algorithm is used to encrypt messages that are sent between two parties. Let us call one party a sender, and the other a receiver.

First, the receiver runs the RSA Algorithm to generate a public key and a private key. The public key is used by the sender to encrypt the message, and the private key is used by the receiver to decrypt the message. The algorithm below will create a private key consisting of two integers:

1. Choose two distinct large random prime numbers *p* and *q*
2. Compute $n = pq$
   - n is used as the modulus for both the public and private keys
3. Compute the totient: $\varphi(n) = (p-1)(q-1)$.
4. Choose an integer *e* such that $1 < e < \varphi(n)$, and *e* and $\varphi(n)$ are coprime
   - *e* is released as the public key exponent
5. Compute *d* to satisfy the congruence relation $de \equiv 1 \pmod{\varphi(n)}$; i.e.

$de = 1 + k\varphi(n)$ for some integer *k*.
- *d* is kept as the private key exponent

(Wikipedia)

Now, we show the application of this algorithm:

Encryption

Alice transmits her public key (n, e) to Bob and keeps the private key secret. Bob then wishes to send message **M** to Alice.

He first turns **M** into a number $m < n$ by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext c corresponding to:

$$c = m^e \mod n$$

Bob then transmits c to Alice.

Decryption

Alice can recover m from c by using her private key exponent d by the following computation:

$$m = c^d \mod n.$$

Given m, she can recover the original message **M**. (Wikipedia)

As you can see, if you could factor n into p and q, you can easily find d by solving the congruence:

$$ed \equiv 1 \pmod{(p-1)(q-1)}$$

Example: Let's pick P = 7, Q = 19. This means N = P * Q = 133. Let m= (P-1) * (Q-1) = 6 * 18 = 108. Now, we need to find an integer e, that is coprime to m.

e = 2 => gcd(e, 108) = 2 (no)
e = 3 => gcd(e, 108) = 3 (no)
e = 4 => gcd(e, 108) = 4 (no)
e = 5 => gcd(e, 108) = 1 (yes!)

Now, we find d, such that d * e = 1 (mod m).

This means a number s exists for which de – 1 = m * s. From this formula, we can get d = (m * s + 1) / e.

s = 0, d = (108 * 0 + 1) / 5 = 1/5 (no)
s = 1, d = (108 * 1 + 1) / 5 = 109/5 (no)
s = 2, d = (108 * 2 + 1) / 5 = 217/5 (no)
s = 2, d = (108 * 3 + 1) / 5 = 325/5 = 65 (yes!)

The public key is defined as {n,e} = {133, 5}. The private key is defined as {n, d} = {133, 65}.

Encryption Example: Suppose Alice has a public key of {133,5} and a private key of {133,65}. She transmits the public key to Bob.

Bob wishes to send the message M = 6, which is some arbitrary code. Bob must encode his message into ciphertext c, from the following formula:

$$c = m^e \mod n$$

In this case, c = 6^5 (mod 133). We calculate this to c = 62.

Alice receives the cipher-text c = 62. She uses the following formula to decipher the text:

$$m = c^d \mod n.$$

$M = C^d \% n$
$= 62^{65} \% 133$
$= 62 * 62^{64} \% 133$
$= 62 * (62^2)^{32} \% 133$
$= 62 * 3844^{32} \% 133$
$= 62 * (3844 \% 133)^{32} \% 133$
$= 62 * 120^{32} \% 133$

We now repeat the sequence of operations that reduced 62^{65} to 120^{32} to reduce the exponent down to 1.

$= 62 * 36^{16} \% 133$
$= 62 * 99^8 \% 133$
$= 62 * 92^4 \% 133$
$= 62 * 85^2 \% 133$
$= 62 * 43 \% 133$
$= 2666 \% 133$
$= 6$

As we can see, 6 is the correct message.

However, the semi-prime n = 133 can be quickly deduced to 133=7*19, making P=7,Q=19 obvious to any attacker. One can even use the trial division methodology for obtaining the prime numbers for such small n.

Now, if p and q are known, then we can quickly deduce d from the following formula:

$$ed \equiv 1 \pmod{(p-1)(q-1)}$$

This means there exists an integer s for which ed -1 = (p-1)*(q-1) * s.

D = [(p-1)*(q-1)*s + 1] / e
= [(7-1) * (19-1) * s + 1] / e
= [6 * 18 * s + 1] /5

s = 2, d = (108 * 3 + 1) / 5 = 325/5 = 65 (yes!)

And, as a result, d is found to be 65 by any malicious attacker.

As we just showed, it is extremely important that we choose large enough p and q to make a factorization attack very expensive. The goal is to make the factorization of n too computationally intensive and thus render that attack impractical. A good rule of thumb is to use p, q such that n is represented as a 1024 bit binary number.

Impracticality of large semi-prime factorization: Let us first define the function O(n) as a description of how the size of input data affects a particular algorithm's usage of computing power. As an example, consider the Trial division algorithm. In the worst-case scenario, the algorithm will perform divisor tests on all prime integers up to sqrt(n). Thus one could say that this will run in O(sqrt(n)) time.

The current best operating time is still not within the domain of polynomial time. For reference, in 2005, a team of scientists in Germany factored a 200 decimal digit number into two primes. That factorization took several months of computer time on 80 top of the line computers. It is estimated to be equivalent to more than 55 years of computational power.[1]

Relation between Φ(n) and n: If we are given Φ(n) and n, we can immediately deduce p and q. We know that

$\varphi(n) = (p-1)(q-1)$, and $n = pq$

As a result, we can easily solve this system of equations for p,q. Therefore, it is imperative to destroy all intermediary calculations during the development of the private key. This includes p and q.

Conclusion: In this paper we have shown the definition of a prime number. We then tested the primality of a sample integer by using the trial division method. Next, we explained the RSA algorithm and showed the application of encryption/decryption. Finally, we explored the impracticality of a n-factorization attack on the algorithm and stressed the importance of destroying everything but the public key and private key.

References:

1) F. Bahr, M. Boehm, J. Franke, T. Kleinjung. "rsa200". May 9 2005. http://www.crypto-world.com/announcements/rsa200.txt

2) Wikipedia.
http://en.wikipedia.org/wiki/Prime_number
http://en.wikipedia.org/wiki/RSA
http://en.wikipedia.org/wiki/Integer_factorization

Proof of RSA:

The decryption procedure works because first

$$c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}.$$

Now, $ed \equiv 1 \pmod{(p-1)(q-1)}$, and hence

$ed \equiv 1 \pmod{p-1}$ and

$ed \equiv 1 \pmod{q-1}$

which can also be written as

$ed = k(p-1)+1$ and $ed = h(q-1)+1$

for proper values of k and h. If m is not a multiple of p then m and p are coprime because p is prime; so by Fermat's little theorem

$$m^{(p-1)} \equiv 1 \pmod{p}$$

and therefore, using the first expression for ed,

$$m^{ed} = m^{k(p-1)+1} = (m^{p-1})^k m \equiv 1^k m = m \pmod{p}$$

.

If instead m is a multiple of p, then

$$m^{ed} \equiv 0^{ed} = 0 \equiv m \pmod{p}.$$

Using the second expression for ed, we similarly conclude that

$$m^{ed} \equiv m \pmod{q}.$$

Since p and q are distinct prime numbers, they are relatively prime to each other, so the fact that both primes divide $m^{ed} - m$ implies their product pq divides $m^{ed} - m$, which means

$$m^{ed} \equiv m \pmod{pq}.$$

Thus, $c^d \equiv m \pmod{n}$. (Wikipedia)