Colin Dilworth
June 3, 2011
Math 480A – Sage Software

Abstract

RSA encryption is a very valuable tool for sending information in a safe, clever way, and is also valuable for signing messages. As was said in class, the problem of being able to send messages without "the enemy" knowing what you are saying or doing has been a quandary as long as warfare has existed. Computer scientists Rivest, Shamir and Adleman invented RSA in the 70's. It is still used today for signing messages as well as encryption. The mathematical principles involved in RSA encryption are the manipulation of large primes and modular arithmetic, making it a particularly interesting topic for number theorists, as well as organizations interested in security, such as the CIA and NSA. RSA has also, at least partly, sparked the race to find large primes because the use of larger primes in the encryption algorithm makes the messages much harder to decrypt by those who don't have the correct key.

Personal Motivation

I, as an aspiring mathematician, am not particularly interested in sending secret messages to soldiers on the front lines, or organizing a coup of a foreign totalitarian regime by activating a militia with a secret password. I was first introduced to RSA encryption last year in a number theory class I took at Colgate University. The professor had just finished teaching us about modular arithmetic (ice. Chinese Remainder Theorem, congruence relations, etc.) when she told us that we would spend the last half of class learning about something fun. I thought that the class had been pretty fun and interesting up to that point, so I was excited. She showed us how RSA worked (which I will explain later), and it was very curious that you could transport information in such a way as to keep it completely secret from anyone hearing what you're sending. I was hooked. As noted above, though, I wasn't particularly interested in using it for anything, and didn't know if it would even come up in any of my classes again. I'm very glad that it did, though in a more indirect way. I do not claim to be a seasoned computer scientist, but I find this system of encryption pretty amazing, and not too difficult to implement, even for an amateur mathematician or number theorist. The main catch with the system, I found, is that the algorithm you write needs to be able to handle large primes.

History[1]

Before I explain how the process of RSA encryption works, let me first explain a brief history of RSA encryption and the people who brought it about. Ronald Rivest, Aid Shamir and Leonard Adleman were the three computer scientists who came up with the encryption system that bears their last initials (**R**ivest, **S**hamir, **A**dleman). They first publicly presented the encryption system at MIT in 1978. They went on to receive a patent for the system in 1983. There has been some controversy about the system, however, because a British mathematician named Clifford Cocks also came up with a similar system in the early 1970's. If he had come out with his conclusions, RSA encryption would probably not have come about in the way that it did.

The Procedure

Step 1: Compute two distinct prime numbers. They must be distinct, or the algorithm won't work. This was a major bug early on in the development of my algorithm. So we have

$$p, q \text{ such that no number in the integers divides } p \text{ or } q.$$

Step 2: The modulus for both the public and private key is found by getting multiplying p and q. So we have:

$$n = pq$$

---

[1] This information is from the Wikipedia page on RSA, as well as the pages on Rivest, Shamir and Adleman.

Step 3: We now compute the totient[2] of pq, which we will use to compute the other parts of our private and public keys. Since all numbers less than p and q are coprime to p and q, and p and q are coprime to each other, we have:

$$\varphi(pq) = \varphi(p)\ \varphi(q) = (p-1)\ (q-1)$$

For convenience, we'll call this number t.

Step 4: We now find e, our public key, such that e is less than and coprime to the totient of pq. So we have:

$$e < t$$

How do we know if e is coprime to (p-1)(q-1)? If we pick e to be prime and see that it doesn't divide the totient of pq that will suffice. So we have:

$$\text{e is prime; e does not divide t}$$

Step 5: We now find the multiplicative modular inverse of e modulo t. That is, we find the number d such that ed = 1 modulo t. So we have:

$$\text{d such that } t \mid ed - 1$$

d can be very hard to find in this case. In researching this problem, I found that it's very tedious, if not impossible without the use a computer. My friend and I worked on a formula for d for hours, but came up with very little of any substance. So, the easiest way to find d is with a simple while loop.

Step 6: Pretend Arthur has come up with all of these calculations, and he sends e and t to Patsy, who is aware of how this system works. Patsy then comes up with a secret message, m,

---

[2] Also called Euler's phi function, the totient of n computes the number of integers less than and coprime to n. For instance, the totient of 8 is 5 because 1, 3, 5, 6 and 7 are all coprime to 8. This information is from the course in number theory that I took last year.

for Arthur. Patsy hides m by taking m^e(mod n) and sending that number to Arthur[3]. So Arthur gets sent:

$$c = m^e (\mathrm{mod}\ n)$$

Part 7: Arthur then decodes c by taking c^d(mod n), so he can see that:

$$m = c^d (\mathrm{mod}\ n)$$

Arthur now knows how to trick the knight at the Bridge of Death.

Conclusion

This is a very powerful tool for use in computer science and security. Aside from that, it is also just an interesting for mathematicians to study. How does it work? Well, we have that

$ed = 1 + kt$
$a^t \equiv 1 (\mathrm{mod}\ n)$

Then $m^{ed} = m^{1+kt} = m(m^t)^k = m(\mathrm{mod}\ n) = m$[4] (That is a footnote, not m to the fourth power.).[5]

---

[3] Different schemes are used for hiding messages in numbers. An obvious one would be to have the letters of the alphabet correspond to the numbers 1 through 26.

[4] Voytko, J. (2008, January 8). "Why Does RSA Work?" Jake Voytko. http://www.jakevoytko.com/blog/2008/01/06/why-does-rsa-work/#euler_corollary, and the Wikipedia page on RSA.

[5] Note also that the text is bigger for readability of exponents.