

# RSA Encryption

Tom Davis

tomrdavis@earthlink.net

<http://www.geometer.org/mathcircles>

October 10, 2003

## 1 Public Key Cryptography

One of the biggest problems in cryptography is the distribution of keys. Suppose you live in the United States and want to pass information secretly to your friend in Europe. If you truly want to keep the information secret, you need to agree on some sort of key that you and he can use to encode/decode messages. But you don't want to keep using the same key, or you will make it easier and easier for others to crack your cipher.

But it's also a pain to get keys to your friend. If you mail them, they might be stolen. If you send them cryptographically, and someone has broken your code, that person will also have the next key. If you have to go to Europe regularly to hand-deliver the next key, that is also expensive. If you hire some courier to deliver the new key, you have to trust the courier, et cetera.

### 1.1 Trap-Door Ciphers

But imagine the following situation. Suppose you have a special method of encoding and decoding that is “one way” in a sense. Imagine that the encoding is easy to do, but decoding is very difficult. Then anyone in the world can encode a message, but only one person can decode it. Such methods exist, and they are called “one way ciphers” or “trap door ciphers”.

Here's how they work. For each cipher, there is a key for encoding and a different key for decoding. If you know the key for decoding, it is very easy to make the key for encoding, but it is almost impossible to do the opposite—to start with the encoding key and work out the decoding key.

So to communicate with your friend in Europe, each of you has a trap door cipher. You make up a decoding key  $D_a$  and generate the corresponding encoding key  $E_a$ . Your friend does exactly the same thing, but he makes up a decoding key  $D_b$  and generates the corresponding encoding key  $E_b$ . You tell him  $E_a$  (but not  $D_a$ ) and he tells you  $E_b$  (but not  $D_b$ ). Then you can send him messages by encoding using  $E_b$  (which only he can decode) and vice-versa—he encodes messages to you using  $E_a$  (which only you can decode, since you're the only person with access to  $D_a$ ).

Now if you want to change to a new key, it is no big problem. Just make up new pairs and exchange the encoding keys. If the encoding keys are stolen, it's not a big deal. The person who steals them can only encode messages—they can't decode them. In fact, the encoding keys (sometimes called “public keys”) could just be published in

a well-known location. It's like saying, "If you want to send me a private message, encode it using this key, and I will be the only person in the world who can read it." But be sure to keep the decoding key (the "private key") secret.

## 1.2 Certification

There is, of course, a problem with the scheme above. Since the public keys are really public, anyone can "forge" a message to you. So your enemy can pretend to be your friend and send you a message just like your friend can—they both have access to the public key. Your enemy's information can completely mislead you. So how can you be certain that a message that says it is from your friend is really from your friend?

Here is one way to do it, assuming that you both have the public and private keys  $E_a$ ,  $E_b$ ,  $D_a$ , and  $D_b$  as discussed in the previous section. Suppose I wish to send my friend a message that only he can read, but in such a way that he is certain that the message is from me. Here's how to do it.

I will take my name, and pretend that it is an encoded message, and decode it using  $D_a$ . I am the only person who can do this, since I am the only person who knows  $D_a$ . Then I include that text in the real message I wish to send, and I encode the whole mess using  $E_b$ , which only my friend knows how to decode.

When he receives it, he will decode it using  $D_b$ , and he will have a message with an additional piece of what looks to him like junk characters. The junk characters are what I got by "decoding" my name. So he simply encodes the junk using my public key  $E_a$  and makes certain that it is my name. Since I am the only one who knows how to make text that will encode to my name, he knows the message is from me.

You can encode any text for certification, and in fact, you should probably change it with each message, but it's easy to do. Your message to your friend would look like this:

"Attack at dawn. Here is my decoding of 'ABCDEFGH': 'JDLEODK'."

To assure privacy, for each message, change the "ABCDEFGH" and the corresponding "JDLEODK".

## 2 RSA Encryption

OK, in the previous section we described what is meant by a trap-door cipher, but how do you make one? One commonly used cipher of this form is called "RSA Encryption", where "RSA" are the initials of the three creators: "Rivest, Shamir, and Adleman". It is based on the following idea:

It is very simply to multiply numbers together, especially with computers. But it can be very difficult to factor numbers. For example, if I ask you to multiply together 34537 and 99991, it is a simple matter to punch those numbers into a calculator and 3453389167. But the reverse problem is much harder.

Suppose I give you the number 1459160519. I'll even tell you that I got it by multi-

plying together two integers. Can you tell me what they are? This is a very difficult problem. A computer can factor that number fairly quickly, but (although there are some tricks) it basically does it by trying most of the possible combinations. For any size number, the computer has to check something that is of the order of the size of the square-root of the number to be factored. In this case, that square-root is roughly 38000.

Now it doesn't take a computer long to try out 38000 possibilities, but what if the number to be factored is not ten digits, but rather 400 digits? The square-root of a number with 400 digits is a number with 200 digits. The lifetime of the universe is approximately  $10^{18}$  seconds – an 18 digit number. Assuming a computer could test one million factorizations per second, in the lifetime of the universe it could check  $10^{24}$  possibilities. But for a 400 digit product, there are  $10^{200}$  possibilities. This means the computer would have to run for  $10^{176}$  times the life of the universe to factor the large number.

It is, however, not too hard to check to see if a number is prime—in other words to check to see that it cannot be factored. If it is not prime, it is difficult to factor, but if it is prime, it is not hard to show it is prime.

So RSA encryption works like this. I will find two huge prime numbers,  $p$  and  $q$  that have 100 or maybe 200 digits each. I will keep those two numbers secret (they are my private key), and I will multiply them together to make a number  $N = pq$ . That number  $N$  is basically my public key. It is relatively easy for me to get  $N$ ; I just need to multiply my two numbers. But if you know  $N$ , it is basically impossible for you to find  $p$  and  $q$ . To get them, you need to factor  $N$ , which seems to be an incredibly difficult problem.

But exactly how is  $N$  used to encode a message, and how are  $p$  and  $q$  used to decode it? Below is presented a complete example, but I will use tiny prime numbers so it is easy to follow the arithmetic. In a real RSA encryption system, keep in mind that the prime numbers are huge.

In the following example, suppose that person A wants to make a public key, and that person B wants to use that key to send A a message. In this example, we will suppose that the message A sends to B is just a number. We assume that A and B have agreed on a method to encode text as numbers. Here are the steps:

1. Person A selects two prime numbers. We will use  $p = 23$  and  $q = 41$  for this example, but keep in mind that the real numbers person A should use should be *much* larger.
2. Person A multiplies  $p$  and  $q$  together to get  $pq = (23)(41) = 943$ . 943 is the “public key”, which he tells to person B (and to the rest of the world, if he wishes).
3. Person A also chooses another number  $e$  which must be relatively prime to  $(p - 1)(q - 1)$ . In this case,  $(p - 1)(q - 1) = (22)(40) = 880$ , so  $e = 7$  is fine.  $e$  is also part of the public key, so B also is told the value of  $e$ .

4. Now B knows enough to encode a message to A. Suppose, for this example, that the message is the number  $M = 35$ .
5. B calculates the value of  $C = M^e \pmod{N} = 35^7 \pmod{943}$ .
6.  $35^7 = 64339296875$  and  $64339296875 \pmod{943} = 545$ . The number 545 is the encoding that B sends to A.
7. Now A wants to decode 545. To do so, he needs to find a number  $d$  such that  $ed = 1 \pmod{(p-1)(q-1)}$ , or in this case, such that  $7d = 1 \pmod{880}$ . A solution is  $d = 503$ , since  $7 * 503 = 3521 = 4(880) + 1 = 1 \pmod{880}$ .
8. To find the decoding, A must calculate  $C^d \pmod{N} = 545^{503} \pmod{943}$ . This looks like it will be a horrible calculation, and at first it seems like it is, but notice that  $503 = 256 + 128 + 64 + 32 + 16 + 4 + 2 + 1$  (this is just the binary expansion of 503). So this means that

$$545^{503} = 545^{256+128+64+32+16+4+2+1} = 545^{256} 545^{128} \dots 545^1.$$

But since we only care about the result  $\pmod{943}$ , we can calculate all the partial results in that modulus, and by repeated squaring of 545, we can get all the exponents that are powers of 2. For example,  $545^2 \pmod{943} = 545 \cdot 545 = 297025 \pmod{943} = 923$ . Then square again:  $545^4 \pmod{943} = (545^2)^2 \pmod{943} = 923 \cdot 923 = 851929 \pmod{943} = 400$ , and so on. We obtain the following table:

$$\begin{aligned} 545^1 \pmod{943} &= 545 \\ 545^2 \pmod{943} &= 923 \\ 545^4 \pmod{943} &= 400 \\ 545^8 \pmod{943} &= 633 \\ 545^{16} \pmod{943} &= 857 \\ 545^{32} \pmod{943} &= 795 \\ 545^{64} \pmod{943} &= 215 \\ 545^{128} \pmod{943} &= 18 \\ 545^{256} \pmod{943} &= 324 \end{aligned}$$

So the result we want is:

$$545^{503} \pmod{943} = 324 \cdot 18 \cdot 215 \cdot 795 \cdot 857 \cdot 400 \cdot 923 \cdot 545 \pmod{943} = 35.$$

Using this tedious (but simple for a computer) calculation, A can decode B's message and obtain the original message  $N = 35$ .

## 2.1 RSA Exercise

OK, now to see if you understand the RSA decryption algorithm, suppose you are person A, and you have chosen as your two primes  $p = 97$  and  $q = 173$ , and you have chosen  $e = 5$ . Thus you told B that  $N = 16781$  (which is just  $pq$ ) and you told him that  $e = 5$ .

He encodes a message (a number) for you and tells you that the encoding is 5347. Can you figure out the original message? Hint—well, not really a hint, but a check of your final answer: it is a four-digit number that is a pattern of digits.

## 3 How It Works

RSA cryptography is based on the following theorems:

**Theorem 1 (Fermat's Little Theorem)** *If  $p$  is a prime number, and  $a$  is an integer such that  $(a, p) = 1$ , then*

$$a^{p-1} = 1 \pmod{p}.$$

**Proof:** Consider the numbers  $(a \cdot 1), (a \cdot 2), \dots, (a \cdot (p - 1))$ , all modulo  $p$ . They are all different. If any of them were the same, say  $a \cdot m = a \cdot n \pmod{p}$ , then  $a \cdot (m - n) = 0 \pmod{p}$  so  $m - n$  must be a multiple of  $p$ . But since all  $m$  and  $n$  are less than  $p$ ,  $m = n$ .

Thus  $a \cdot 1, a \cdot 2, \dots, a \cdot (p - 1)$  must be a rearrangement of  $1, 2, \dots, (p - 1)$ . So modulo  $p$ , we have:

$$\prod_{i=1}^{p-1} i = \prod_{i=1}^{p-1} a \cdot i = a^{p-1} \prod_{i=1}^{p-1} i,$$

so  $a^{p-1} = 1 \pmod{p}$ .

**Theorem 2 (Fermat's Theorem Extension)** *If  $(a, m) = 1$  then  $a^{\phi(m)} = 1 \pmod{m}$ , where  $\phi(m)$  is the number of integers less than  $m$  that are relatively prime to  $m$ . The number  $m$  is not necessarily prime.*

**Proof:** Same idea as above. Suppose  $\phi(m) = n$ . Then suppose that the  $n$  numbers less than  $m$  that are relatively prime to  $m$  are:

$$a_1, a_2, a_3, \dots, a_n.$$

Then  $a \cdot a_1, a \cdot a_2, \dots, a \cdot a_n$  are also relatively prime to  $m$ , and must all be different, so they must just be a rearrangement of the  $a_1, \dots, a_n$  in some order. Thus:

$$\prod_{i=1}^n a_i = \prod_{i=1}^n a \cdot a_i = a^n \prod_{i=1}^n a_i,$$

modulo  $m$ , so  $a^n = 1 \pmod{m}$ .

**Theorem 3 (Chinese Remainder Theorem)** *Let  $p$  and  $q$  be two numbers (not necessarily primes), but such that  $(p, q) = 1$ . Then if  $a = b \pmod{p}$  and  $a = b \pmod{q}$  we have  $a = b \pmod{pq}$ .*

**Proof:** If  $a = b \pmod{p}$  then  $p$  divides  $(a - b)$ . Similarly,  $q$  divides  $(a - b)$ . But  $p$  and  $q$  are relatively prime, so  $pq$  divides  $(a - b)$ . Consequently,  $a = b \pmod{pq}$ . (This is a special case with only two factors of what is usually called the Chinese remainder theorem but it is all we need here.)

### 3.1 Proof of the Main Result

Based on the theorems above, here is why the RSA encryption scheme works.

Let  $p$  and  $q$  be two different (large) prime numbers, let  $0 \leq M < pq$  be a secret message<sup>1</sup>, let  $d$  be an integer (usually small) that is relatively prime to  $(p - 1)(q - 1)$ , and let  $e$  be a number such that  $de = 1 \pmod{(p - 1)(q - 1)}$ . (We will see later how to generate this  $e$  given  $d$ .) The encoded message is  $C = M^d \pmod{pq}$ , so we need to show that the decoded message is given by  $M = C^e \pmod{pq}$ .

**Proof:** Since  $de = 1 \pmod{(p - 1)(q - 1)}$ ,  $de = 1 + k(p - 1)(q - 1)$  for some integer  $k$ . Thus:

$$C^e = M^{de} = M^{1+k(p-1)(q-1)} = M \cdot (M^{(p-1)(q-1)})^k.$$

If  $M$  is relatively prime to  $p$ , then

$$M^{de} = M \cdot (M^{p-1})^{k(q-1)} = M \cdot (1)^{k(q-1)} = M \pmod{p} \quad (1)$$

By the extension of Fermat's Theorem giving  $M^{p-1} = M \pmod{p}$  followed by a multiplication of both sides by  $M$ . But if  $M$  is not relatively prime to  $p$ , then  $M$  is a multiple of  $p$ , so equation 1 still holds because both sides will be zero, modulo  $p$ .

By exactly the same reasoning,

$$M^{de} = M \cdot M^{q-1} = M \pmod{q} \quad (2)$$

If we apply the Chinese remainder theorem to equations 1 and 2, we obtain the result we want:  $M^{de} = M \pmod{pq}$ .

Finally, given the integer  $d$ , we will need to be able to find another integer  $e$  such that  $de = 1 \pmod{(p - 1)(q - 1)}$ . To do so we can use the extension of Fermat's theorem to get  $d^{\phi((p-1)(q-1))} = 1 \pmod{(p - 1)(q - 1)}$ , so  $d^{\phi((p-1)(q-1))-1} \pmod{(p - 1)(q - 1)}$  is a suitable value for  $e$ .

---

<sup>1</sup>If the message is long, break it up into a series of smaller messages such that each of them is smaller than  $pq$  and encode each of them separately.