Math From Scratch Lesson 22: The RSA Encryption Algorithm

W. Blaine Dowler

January 2, 2012

Contents

1 What Is Encryption?

Encryption is designed to render data inaccessible to people who are not trusted. The idea is to encode the information in some way, and provide the recipient with a means to decode it so that it can be read. The encoding and decoding steps need to be designed in a manner that makes it difficult to decode the message if you are not the intended recipient.

The simplest encryptions are substitution encryption schemes. For example, replacing a with b, replacing b with c, etc, replacing each letter in a message with the next letter of the alphabet. This would encode "hello world" as "ifmmp xpsme," which is not particularly easy to read. However, if the message is long enough, then knowledge of the English language can help decode the message without an official key. The letter m above appears frequently, so it is a likely substitute for one of the more common letters in English. It also appears as a double letter, further reducing the number of choices. If messages are several paragraphs long, any substitution scheme gets far too easy to crack.

Another risk to encryption is that anyone who has access to the decryption method can read any encrypted information. For example, if two countries are at war and one side uses a single encryption/decryption key for all secret transmissions, then the other side can gain unlimited access to secret data if they can obtain the key in question. If this is a single, shared key, then it must be shared somehow, which increases the risk of exposure.

2 What Is RSA Encryption?

RSA encryption, named for creators Ron Rivest, Adi Shamir and Leonard Adleman, uses the power of math to make decryption very difficult for those who aren't meant to have access to the encrypted data. It is based on totient functions and modular arithmetic, as detailed in the past few lessons.

The RSA method increases security first by using multiple encryption keys for each entity involved. Imagine two people, Reed and Tony, want to exchange secret data. Tony wants to send a message to Reed, so Reed generates two keys. The public key is published anywhere and everywhere, and is used to encode messages intended for Reed. Tony can encode a message using the public key, but he cannot decode it. It can only be decoded by Reed's private key, which Reed does not share. This is the basic concept; now let's add the math.

2.1 Key Generation

We start by finding a large N to use as a modulus for our calculations. The more prime factors this N has, the easier it will be to crack the encryption by brute force, as more combinations of multiplication will generate the same N. Typically, this N is generated by taking the product of two large prime numbers, P and Q, such that $N = PQ$.

We now need the totient of N, or $\phi(N)$. As P and Q are coprime and the only factors of N, we find that $\phi(N) = \phi(PQ) = \phi(P)\,\phi(Q) = (P-1)(Q-1)$.

Finally, we choose an integer e which is relatively prime to (and smaller than) $\phi(N)$. This will be a part of the public key used for encoding messages. Finally, we search for and select an integer d to be part of the private key used for decoding, such that $de = 1\,(mod\ \phi(N))$.

The public key is the pair (N, e) and the private key is the pair (N, d). The public key is freely distributed, as a single key alone is not sufficient to encode and decode messages.

2.2 Encoding

To encode a message, we use some scheme to transform it into an integer m. Anything stored on a computer is stored as a series of binary numbers, so this step is certainly possible. However, certain integers are easily decoded by brute force attacks without the appropriate keys, so there are methodologies in place to make the message difficult to decode in this fashion. Those "padding" schemes, which pad the original message with unrelated data designed to complicate inappropriate decryption, are not going to be discussed here. The RSA encryption doesn't actually depend on them, and works without them, but may be less secure.

We take the public key pair (N, e) and apply it to the message m to create the encrypted cyphertext c as

$$c \equiv m^e \, (mod \; N)$$

2.3 Decoding

To decode, we take the private key (N, d) and use it as follows:

$$m \equiv c^d \, (mod \; N)$$

This is the same as computing

$$m \equiv c^d \equiv (m^e)^d \equiv m^{de} \, (mod \; N)$$

This works on the condition that

$$m^1 \equiv m^{de} \, (mod \; N)$$

This condition holds, which can be proven as follows. By construction, $\phi(N) = (P-1)\,(Q-1)$, so we can guarantee that $P-1$ divides into $\phi(N)$. We have also chosen d and e such that $de \equiv 1 \, (mod \; \phi(N))$, or $de = k \cdot \phi(N) + 1$ for some k.

Euler's theorem, established in our previous lesson, states that $m^{\phi(N)} \equiv 1 \, (mod \; N)$ when m and N are coprime. For the moment, let us constrain our attention to $m^{P-1} \equiv 1 \, (mod \; P)$. This is Euler's theorem for a prime P if $1 < m < P$ holds. Since $P-1$ divides $\phi(P)$, we can say

$$m^{k \cdot \phi(P)} \equiv 1 \, (mod \; P)$$

or

$$m^{k \cdot \phi(P)+1} \equiv m \, (mod \; P)$$

Again, by construction, $\phi(P)$ divides into $\phi(N) = \phi(PQ) = \phi(P)\,\phi(Q)$, so we can choose a k which is a multiple of $\phi(Q)$ to transform our statement into

$$m^{k \cdot \phi(N)+1} \equiv m \, (mod \; P)$$

Identical arguments can be made relative to Q, which can be combined to show that

$$m^{de} \equiv m \, (mod \; PQ) \equiv m \, (mod \; N)$$

All of this logic depended upon the notion that m was relatively prime to its exponents. If this is not the case, then the entire argument reduces to $0 \equiv 0 \, (mod \; N)$, which is obviously true. So, RSA works.

2.4 A Fully Worked Example

Let us choose small, more manageable numbers for our worked example. In the real world, we'll want to choose the largest P and Q we can manage to make decryption as difficult as possible. Let us begin with $P = 11$ and $Q = 13$. With these choices, $N = 143$ and $\phi(N) = 120$.

We produce our public encryption key by choosing a third prime e satisfying $1 < e < N$ and $\gcd(e, N) = 1$. The prime 17 satisfies these criteria.

Now we need our private key. We want $de \equiv 1 \, (mod \; \phi(N))$, or $17d \equiv 1 \, (mod \; 120)$. Using methods from previous lessons, we find that $d = 233$ satisfies this criteria. We are now equipped with both public and private keys.

Let us now choose a message. Assume the message Tony wishes to send Reed is 42. Tony encodes this message by calculating $c = m^e \, (mod \; N) = 42^17 \, (mod \; 143) = 48$. Tony sends this message to Reed.

Reed now decodes the message by calculating $m = c^d \, (mod \; N) = 48^233 \, (mod \; 143) = 42$, Tony's original message.

2.5 Bidirectional Security

You may have noticed that the fundamental step involves m^de. The multiplication of d and e is commutative, meaning that it doesn't actually matter

which is applied first. Therefore, if Reed wants to send Tony a message in a manner that ensure Tony knows it came from Reed, then Reed can encrypt the message with his *private* key before sending it. Tony can then decode the message using Reed's *public* key. Anyone will be able to decode it, true, but only Reed could have encoded it, so the source is assured. When both parties need to have a private conversation that could not have originated with anyone else, they can apply both methods. Reed can encode a message with his own private key and Tony's public key in some predetermined sequence. Tony can then use his own private key and Reed's public key to decode the message in the appropriate sequence, and read a message for Tony's eyes only that is assured to have originated from Reed.

The most common practice for verifying the identity of the sender is not to encrypt the entire message, but to encrypt only a standard signature based on a private key, and then providing information in the public key indicating what the signature should be when decoded. This allows for the digital signing of documents for legal reasons, without requiring the recipient to have the complete software package installed to read the message.

3 Next

Math from Scratch will take a break for July and August while the Bureau's Summer School (about Einstein's relativity) comes out on a weekly basis. Upon returning in September, Math From Scratch will work specifically towards the goal of defining algebraic fields with an eye to defining the rational numbers.