Senior Math Circles – Cryptography and Number Theory Week 3

Dale Brydon

Feb. 16, 2014

1 RSA Encryption

RSA encryption was first publicly proposed by Rivest, Shamir, and Adelman in 1977. We now have enough background to describe how it works. RSA is what is known as a *public key* cryptosystem, since encryption is done using a single key that is public information. Decryption is done using the private key, which only the receiver has access to.

The first stage in RSA encryption, is key generation. If Alice wants to be able to receive messages, she first needs to generate keys. To do so she picks two large random primes, p and q, with the same number of bits (the exact size is determined by her desired security level). Then she computes $n = pq$, $\varphi(n) = (p-1)(q-1)$, and a random number $0 < e < \varphi(n)$ such that e is invertible mod $\varphi(n)$. Finally, she computes $d \equiv e^{-1} \bmod \varphi(n)$ such that $0 < d < \varphi(n)$ (i.e. she finds the unique integer d so that $ed \equiv 1 \pmod{\varphi(n)}$ and $0 < d < \varphi(n)$). Her public key is the pair (n, e) and her private key is d.

To send a message, m, to Alice, Bob must first obtain her public key, (n, e). We assume that Bob is somehow able to encode his message as a number and that when he does so $m < n$ and $\gcd(m, n) = 1$. To encrypt the message, Bob computes $c = m^e \bmod n$ and sends c to Alice. To decrypt, Alice computes $r = c^d \bmod n$. We now need to verify that $r = m$.

Proof. Notice that $r \equiv c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}$. We know by choice of d, that $ed \equiv 1 \pmod{\varphi(n)}$. This implies that there exists some integer q so that $ed = q\varphi(n) + 1$. But then,

$$\begin{aligned} r &\equiv m^{(q\varphi(n)+1)} \pmod{\varphi(n)} \\ &\equiv (m^{\varphi(n)})^q m \pmod{\varphi(n)} \\ &\equiv 1^q m \pmod{\varphi(n)} \\ &\equiv m \pmod{\varphi(n)}, \end{aligned}$$

where the simplification to get to the second last line comes from the fact that

$$m^{\varphi(n)} \equiv m^{(p-1)(q-1)} \equiv 1 \pmod{n},$$

by the earlier proposition, since $\gcd(m, n) = 1$. □

As a toy example, suppose $p = 11$ and $q = 3$. Then $n = 33$ and $\varphi(n) = 20$. Suppose also $e = 7$. Using the Extended Euclidean Algorithm, this gives that $d = 3$. Let's use this setup to encrypt the message $m = 4$. We compute

$$4^7 \equiv 4^3 4^3 4 \equiv 64(64)4 \equiv (-2)(-2)(4) \equiv 16 \pmod{33}.$$

Hence, $c = 16$. Decrypting we can see that $16^3 = (4^2)^3 = (4^3)^2$. Hence, we get that

$$(4^3)^2 \equiv (64)(64) \equiv (-2)(-2) \equiv 4 \pmod{33},$$

as expected.

The next question we must face is to wonder whether all the computations can be performed efficiently. In particular, can the modular exponentiations be computed quickly, since, in practice, the exponents will be 300-digit numbers or bigger. Certainly, standard exponentiation by repeated multiplication will not work. But, we can make use of a clever trick together with the fact that we can reduce intermediate results mod n to make a fast algorithm.

Square-and-Multiply Algorithm Output: $c = m^e \bmod n$

1. Write e in binary as $e = b_k b_{k-1} \cdots b_1 b_0$.
2. Set $r_0 = e$ and compute $r_1 = e^2 \bmod n$.
3. Compute $r_2 = e^4 \bmod n$ as $r_1^2 \bmod n$, $e^8 \bmod n$ as $r_2^2 \bmod n$, and so on, up to $r_k = e^{2^k} \bmod n$.
4. Compute c as
$$c = (r_k)^{b_k}(r_{k-1})^{b_{k-1}} \cdots (r_1)^{b_1}(r_0)^{b_0} \bmod n.$$

As an example let's compute $11^{18} \bmod 21$. First we notice that $18 = 16 + 2$ and so in binary is 10010. Now we can compute

$$\begin{aligned}
11^2 \bmod 21 &= 121 \bmod 21 = 16 \\
11^4 \bmod 21 &= (11^2)^2 \bmod 21 = 16^2 \bmod 21 = 256 \bmod 21 = 4 \\
11^8 \bmod 21 &= (11^4)^2 \bmod 21 = 4^2 \bmod 21 = 16 \bmod 21 \\
11^{16} \bmod 21 &= (11^8)^2 \bmod 21 \bmod 21 = 16^2 \bmod 21 = 4 \bmod 21
\end{aligned}$$

So then

$$\begin{aligned} 11^{18} \bmod 21 &= (4^1)(16^0)(4^0)(16^1)(11^0) \bmod 21 \\ &= 4(16) \bmod 21 = 64 \bmod 21 = 1. \end{aligned}$$

Another thing we should consider are the various assumptions we made about m. Obviously there is some way to encode words as numbers, since this is how computers communicate. Unfortunately, this encoding method certainly does not guarantee that $m < n$. For example, a 1024-bit string contains at most 128 characters under the most common character encoding. We can get around this problem by splitting up the message into parts and encoding those parts so that each encoded part is smaller than n. Assuming the parts are $m_1, \cdots, m_k$, encryption is done by encrypting each part m_i to c_i. Finally, we assumed that $\gcd(m, n) = 1$. This is a fair assumption, since the positive divisors of n are 1, p, q, and n. Since $0 < m < n$, we know $\gcd(m, n) < n$. So if $\gcd(m, n) \neq 1$, it must equal to p or q. But then by finding such a message Bob can easily factor n, something we assume he cannot do. Further, there are only $q - 1$ messages where p is a divisor and $p - 1$ messages where q is a divisor. The chances of randomly picking such a message is, hence, $(q-1+p-1)/n \approx (q+p)/n = 1/p+1/q$. Since p and q are extremely small, this probability is essentially 0 (less than the chance of winning the lottery every week for 4 months.)

This naturally brings us into a discussion of the security of the scheme. Clearly, if an adversary can factor n, then they can compute $\varphi(n)$ and then d. But, there does not seem to be an algorithm on a classical computer that can factor large numbers quickly. It can be shown that finding d necessarily requires factoring n. However, it is not known that performing a decryption without the private key requires factoring n. That being said, there is no known method to efficiently perform these decryptions and it is believed to be computationally hard.

2 Signatures

RSA encryption allows 2 parties to communicate privately without having shared any secret information in advance. However, for Bob and Alice to communicate, Bob still needs to get a copy of Alice's public key. Supposing he doesn't know or can't meet with Alice, how can he make sure what he gets is really her public key and not the public key of some attacker, Eve? He needs some way to authenticate the public key to make sure it really comes from Alice. So public key encryption systems replace the need for a secure channel prior to meeting with an authenticated channel.

How do we authenticate people in real life? Many different methods are used, but a common one is a signature. Bob can happily receive Alice's public key, as long as it comes with a certificate of authenticity signed by someone he trusts. So then, how might we go about producing a digital signature? It should have the property that only a single person can produce their own signature, but everyone else can verify that the person in question has indeed signed a particular message.

RSA can also be used as a method to sign messages. Key generation is exactly the same as for encryption. To sign a message, m, Alice simply computes $s = m^d \bmod n$. To verify Alice's signature, s, on a message, m, Bob computes $r = s^e \bmod n$ and checks that $r = m$.

As a simple example, suppose $n = 35$, $d = 7$, and $m = 3$. We would sign the message by noting $3^7 = 3^4 3^3 = (81)(27)$. Since $81 \bmod 35 = 11$, we get that $3^7 \bmod 35 = 11(27) \bmod 35 = 17$. In this case, $e = 7$ as well and I leave it to you to verify that $11^7 \bmod 35 = 3$.

The security properties of this scheme are very similar to those of RSA encryption. In particular you can see that if RSA is hard to decrypt it is likely hard to come up with a fake signature s so that $s^e = m$ for some choice of message $m \neq 1$.

Now that we know about signatures and encryption, we can discuss how secure communications occur over the web. Let's look at some of the steps involved in visiting a website using a secure connection. First you take your browser to the page. At this point your browser receives a certificate from the website that contains the website's public key and a signature on that key from a certificate authority. Your browser has the public key of the certificate authority hard-coded into it and so is able to verify the signature on the key. (You can inspect the certificate by clicking on the lock icon that appears when you visit a secure page.) Once it has verified the public key, it generates a random key to be used in a symmetric encryption scheme (such as a one-time pad) and encrypts that with the website's public key. It sends the encrypted key to the website and so now there is a shared secret key that can be used to exchange encrypted messages both ways.

3 Problem Set

1. Suppose you are sent the ciphertext $c = 8$, encrypted with the RSA public key where $n = 35$ and $e = 5$. Determine the decryption key and decrypt the ciphertext.

2. Suppose the message $m = 6$ has been signed with the RSA private key corresponding to the public key $n = 55$ and $e = 3$, to produce the signature $s = 39$. Is this a valid signature for the message?

3. In this question you'll show how learning n and $\varphi(n)$ allows you to factor n, when $n = pq$ is the product of two primes.

   (a) Express $\varphi(n) - (n + 1)$ in terms of p and q.

   (b) Using part (a), explain how knowing n and $\varphi(n)$ allows you to expand the polynomial $(x - p)(x - q)$.

   (c) Using part (b), come up with a method to find p and q given n and $\varphi(n)$.

   (d) Use your method from part (c) to factor $n = 247$ given that $\varphi(n) = 216$.

4. (Note: this problem is somewhat laborious.) Suppose your RSA private key is 11 and your modulus is $n = 899$ and you receive the ciphertexts

   $$c_1 = 593,\ c_2 = 140,\ c_3 = 728,\ c_4 = 101,\ c_5 = 134,\ c_6 = 126, c_7 = 508,\ c_8 = 255.$$

   Decrypt the ciphertexts and then use the encoding $A = 2, B = 3, \cdots, Z = 27$ to determine the message.

5. (a) Suppose you know that n is the product of two consecutive primes, (e.g. 7 and 11). Explain how to factor n. You may use the fact that there is an efficient algorithm to find the smallest prime greater than a number.

   (b) 2491 is the product of two consecutive primes. Factor 2491.

6. (Challenge Problem) A *decryption exponent* for an RSA public key (n, e) is an integer d with the property that $a^{de} \equiv 1 \pmod{n}$ for all integers a that satisfy $\gcd(a, n) = 1$. Suppose that Eve has a magic box that creates decryption exponents for (n, e) for a fixed modulus, n, and for a large number of different encryption exponents, e. Explain how Eve can use her magic box to try to factor n. [Note: this is problem 3.9 of *An Introduction to Mathematical Cryptography* by Hoffstein, Pipher, and Silverman.]