LEMMA If $n = pq$ (two different primes), then $\boxed{a^{\phi(n)+1} \equiv a \pmod{n}.}$

COROLLARY $a^{k\phi(n)+1} \equiv a$.

Chapter 6

The RSA Algorithm

W. Trappe, L. Washington: "Introduction to Cryptography with Coding Theory."

6.1 The RSA Algorithm

Alice wants to send a message to Bob, but they have not had previous contact and they do not want to take the time to send a courier with a key. Therefore, all information that Alice sends to Bob will potentially be obtained by the evil observer Eve. However, it is still possible for a message to be sent in such a way that Bob can read it but Eve cannot.

With all the previously discussed methods, this would be impossible. Alice would have to send a key, which Eve would intercept. She could then decrypt all subsequent messages. The possibility of the present scheme, called a **public key cryptosystem**, was first publicly suggested by Diffie and Hellman in their classic paper [Diffie-Hellman]. However, they did not yet have a practical implementation (although they did present an alternative key exchange procedure that works over public channels; see Section 13.1). In the next few years, several methods were proposed. The most successful, based on the idea that factorization of integers into their prime factors is hard, was proposed by Rivest, Shamir, and Adleman in 1977 and is known as the RSA algorithm.

It had long been claimed that government cryptographic agencies had discovered the RSA algorithm several years earlier, but secrecy rules prevented them from releasing any evidence. Finally, in 1997, documents re-

leased by CESG, a British cryptographic agency, showed that in 1970, James Ellis had discovered public key cryptography, and in 1973, Clifford Cocks had written an internal document describing a version of the RSA algorithm in which the encryption exponent e (see the discussion that follows) was the same as the modulus n.

Here is how the RSA algorithm works. Bob chooses two distinct large primes p and q and multiplies them together to form

$$n = pq.$$

He also chooses an encryption exponent e such that

$$\gcd(e, (p-1)(q-1)) = 1.$$

He sends the pair (n, e) to Alice but keeps the values of p and q secret. In particular, Alice, who could possibly be an enemy of Bob, never needs to know p and q to send her message to Bob securely. Alice writes her message as a number m. If m is larger than n, she breaks the message into blocks, each of which is less than n. However, for simplicity, let's assume for the moment that $m < n$. Alice computes

$$c \equiv m^e \pmod{n}$$

and sends c to Bob. Since Bob knows p and q, he can compute $(p-1)(q-1)$ and therefore can find the decryption exponent d with

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

As we'll see later,

$$m \equiv c^d \pmod{n},$$

so Bob can read the message.

We summarize the algorithm in the following table.

The RSA Algorithm

1. Bob chooses secret primes p and q and computes $n = pq$.
2. Bob chooses e with $\gcd(e, (p-1)(q-1)) = 1$.
3. Bob computes d with $de \equiv 1 \pmod{(p-1)(q-1)}$.
4. Bob makes n and e public, and keeps p, q, d secret.
5. Alice encrypts m as $c \equiv m^e \pmod{n}$ and sends c to Bob.
6. Bob decrypts by computing $m \equiv c^d \pmod{n}$.

Example. Bob chooses

$$p = 885320963, \quad q = 238855417.$$

Then

$$n = p \cdot q = 211463707796206571.$$

Let the encryption exponent be

$$e = 9007.$$

The values of n and e are sent to Alice.

Alice's message is *cat*. We will depart from our earlier practice of numbering the letters starting with $a = 0$; instead, we start the numbering at $a = 01$ and continue through $z = 26$. In the previous method, if the letter a appeared at the beginning of a message, it would yield a message number m starting with 00, so the a would disappear.

The message is therefore

$$m = 30120.$$

Alice computes

$$c \equiv m^e \equiv 30120^{9007} \equiv 113535859035722866 \pmod{n}.$$

She sends c to Bob.

Since Bob knows p and q, he knows $(p-1)(q-1)$. He uses the extended Euclidean algorithm (see Section 3.2) to compute d such that

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

The answer is

$$d = 116402471153538991.$$

Bob computes

$$c^d \equiv 113535859035722866^{116402471153538991} \equiv 30120 \pmod{n},$$

so he obtains the original message. ∎

There are several aspects that need to be explained, but perhaps the most important is why $m \equiv c^d \pmod{n}$. Recall Euler's theorem (Section 3.6): If $\gcd(a, n) = 1$, then $a^{\phi(n)} \equiv 1 \pmod{n}$. In our case, $\phi(n) = \phi(pq) = (p-1)(q-1)$. Suppose $\gcd(m, n) = 1$. This is very likely the case; since p and

q are large, m probably has neither as a factor. Since $de \equiv 1 \pmod{\phi(n)}$, we can write $de = 1 + k\phi(n)$ for some integer k. Therefore

$$c^d \equiv (m^e)^d \equiv m^{1+k\phi(n)} \equiv m \cdot (m^{\phi(n)})^k \equiv m \cdot 1^k \equiv m \pmod{n}.$$

We have shown that Bob can recover the message. If $\gcd(m, n) \neq 1$, Bob still recovers the message. See Exercise 13.

What does Eve do? She intercepts n, e, c. She does not know p, q, d. We assume that Eve has no way of factoring n. The obvious way of computing d requires knowing $\phi(n)$. We show later that this is equivalent to knowing p and q. Is there another way? We will show that if Eve can find d, then she can probably factor n. Therefore, it is unlikely that Eve finds d.

Since Eve knows $c \equiv m^e \pmod{n}$, why doesn't she simply take the eth root of c? This works well if we are not working mod n but is very difficult in our case. For example, if you know that $m^3 \equiv 3 \pmod{85}$, you cannot calculate the cube root of 3, namely $1.2599\ldots$, on your calculator and then reduce mod 85. Of course, a case-by-case search would eventually yield $m = 7$, but this method is not feasible for large n.

How does Bob choose p and q? They should be chosen at random, independently of each other. How large depends on the level of security needed, but it seems that they should have at least 100 digits. For reasons that we discuss later, it is perhaps best if they are of slightly different lengths. When we discuss primality testing, we'll see that finding such primes can be done fairly quickly. A few other tests should be done on p and q to make sure they are not bad. For example, if $p - 1$ has only small prime factors, then n is easy to factor by the $p - 1$ method (see Section 6.4), so p should be rejected and replaced with another prime.

Why does Bob require $\gcd(e, (p-1)(q-1)) = 1$? Recall (see Section 3.3) that $de \equiv 1 \pmod{(p-1)(q-1)}$ has a solution d if and only if $\gcd(e, (p-1)(q-1)) = 1$. Therefore, this condition is needed in order for d to exist. The extended Euclidean algorithm can be used to compute d quickly. Since $p - 1$ is even, $e = 2$ cannot be used; one might be tempted to use $e = 3$. However, there are dangers in using small values of e (see Section 6.2 and Computer Problem 14), so something larger is usually recommended. For example, one could let e be a moderately large prime. Then there is no difficulty ensuring that $\gcd(e, (p-1)(q-1)) = 1$.

In the encryption process, Alice calculates $m^e \pmod{n}$. Recall that this can be done fairly quickly and without large memory, for example, by successive squaring. This is definitely an advantage of modular arithmetic: If Alice tried to calculate m^e first, then reduce mod n, it is possible that recording m^e would overflow her computer's memory. Similarly, the decryption process of calculating $c^d \pmod{n}$ can be done efficiently. Therefore, all the operations needed for encryption and decryption can be done quickly

(i.e., in time a power of $\log n$). The security is provided by the assumption that n cannot be factored.

We made two claims. We justify them here. Recall that the point of these two claims was that finding $\phi(n)$ or finding the decryption exponent d is essentially as hard as factoring n. Therefore, if factoring is hard, then there should be no fast, clever way of finding d.

Claim 1: Suppose $n = pq$ is the product of two distinct primes. If we know n and $\phi(n)$, then we can quickly find p and q.

Note that

$$n - \phi(n) + 1 = pq - (p-1)(q-1) + 1 = p + q.$$

Therefore, we know pq and $p+q$. The roots of the polynomial

$$X^2 - (n - \phi(n) + 1)X + n = X^2 - (p+q)X + pq = (X-p)(X-q)$$

are p and q, but they can also be calculated by the quadratic formula:

$$p, q = \frac{(n - \phi(n) + 1) \pm \sqrt{(n - \phi(n) + 1)^2 - 4n}}{2}.$$

This yields p and q.

For example, suppose $n = 221$ and we know that $\phi(n) = 192$. Consider the quadratic equation

$$X^2 - 30X + 221.$$

The roots are

$$p, q = \frac{30 \pm \sqrt{30^2 - 4 \cdot 221}}{2} = 13, \quad 17.$$

Claim 2: If we know d and e, then we can probably factor n.

In the discussion of factorization methods in Section 6.4, we show that if we have a universal exponent $b > 0$ such that $a^b \equiv 1 \pmod{n}$ for all a with $\gcd(a, n) = 1$, then we can probably factor n. Since $de - 1$ is a multiple of $\phi(n)$, say $de - 1 = k\phi(n)$, we have

$$a^{de-1} \equiv (a^{\phi(n)})^k \equiv 1 \pmod{n}$$

whenever $\gcd(a, n) = 1$. The method for universal exponents can now be applied.

One way the RSA algorithm can be used is when there are several banks, for example, that want to be able to send financial data to each other. If there are several thousand banks, then it is impractical for each pair of banks to have a key for secret communication. A better way is the following. Each bank chooses integers n and e as before. These are then published in a public

Year	Number of digits
1964	20
1974	45
1984	71
1994	129
1999	155

Table 6.1: Factorization Records

6.5 The RSA Challenge

When the RSA algorithm was first made public in 1977, the authors made the following challenge.

Let the RSA modulus be

$$n = 11438162575788886766923577997614661201021829672124236256256184293570693524573389783059712356395870505898907514759929002687954354 1$$

and let $e = 9007$ *be the encryption exponent. The ciphertext is*

$$c = 96869613754622061477140922254355882905759991124574319874695120930816298225145708356931476622883989628013391990551829945157815154.$$

Find the message.

The only known way of finding the plaintext is to factor n. In 1977, it was estimated that the then-current factorization methods would take 4×10^{16} years to do this, so the authors felt safe in offering \$100 to anyone who could decipher the message before April 1, 1982. However, techniques have improved, and in 1994, Atkins, Graff, Lenstra, and Leyland succeeded in factoring n.

They used 524339 "small" primes, namely those less than 16333610, plus they allowed factorizations to include up to two "large" primes between 16333610 and 2^{30}. The idea of allowing large primes is the following: If one large prime q appears in two different relations, these can be multiplied to produce a relation with q squared. Multiplying by $q^{-2} \pmod{n}$ yields a relation involving only small primes. In the same way, if there are several