8. G. Ruggiu, "Cryptology and complexity theories," *Advances in Cryptology, Proceedings of Eurocrypt 84,* Springer-Verlag, 1985, 3–9.

2 RSA

In looking for a trapdoor function f to use for a public key cryptosystem, one wants to use an idea which is fairly simple conceptually and lends itself to easy implementation. On the other hand, one wants to have very strong empirical evidence — based on a long history of attempts to find algorithms for f^{-1} — that decryption cannot feasibly be accomplished without knowledge of the secret deciphering key. For this reason it is natural to look at an ancient problem of number theory: the problem of finding the complete factorization of a large composite integer whose prime factors are not known in advance. The success of the so-called "RSA" cryptosystem (from the last names of the inventors Rivest, Shamir, and Adleman), which is one of the oldest (16 years old) and most popular public key cryptosystems, is based on the tremendous difficulty of factoring.

We now describe how RSA works. Each user first chooses two extremely large prime numbers p and q (say, of about 100 decimal digits each), and sets $n = pq$. Knowing the factorization of n, it is easy to compute $\varphi(n) = (p-1)(q-1) = n+1-p-q$. Next, the user randomly chooses an integer e between 1 and $\varphi(n)$ which is prime to $\varphi(n)$.

Remark. Whenever we say "random" we mean that the number was chosen with the help of a random-number generator (or "pseudo-random" number generator), i.e., a computer program that generates a sequence of digits in a way that no one could duplicate or predict, and which is likely to have all of the statistical properties of a truly random sequence. A lot has been written concerning efficient and secure ways to generate random numbers, but we shall not concern ourselves with this question here. In the RSA cryptosystem we need a random number generator not only to choose e, but also to choose the large primes p and q (so that no one could guess our choices by looking at tables of special types of primes, for example, Mersenne primes or factors of $b^k \pm 1$ for small b and relatively small k). What does a "randomly generated" prime number mean? Well, first generate a large random integer m. If m is even, replace m by $m+1$. Then apply suitable *primality tests* to see if the odd number m is prime (primality tests will be examined systematically in the next chapter). If m is not prime, try $m+2$, then $m+4$, and so on, until you reach the first prime number $\geq m$, which is what you take as your "random" prime. According to the Prime Number Theorem (for the statement see Exercise 13 of § I.1), the frequency of primes among the numbers near m is about $1/log(m)$, so you can expect to test $O(log\, m)$ numbers for primality before reaching the first prime $\geq m$.