11000110 corresponds to the element (1, 1, 0, 0, 0, 1, 1, 0) in $Z_2 \oplus Z_2 \oplus Z_2 \oplus Z_2 \oplus Z_2 \oplus Z_2 \oplus Z_2 \oplus Z_2$. Similarly, two binary strings $a_1a_2 \cdots a_n$ and $b_1b_2 \cdots b_n$ are added componentwise modulo 2 just as their corresponding elements in $Z_2 \oplus Z_2 \oplus \cdots \oplus Z_2$ are. For example,

$$11000111 + 01110110 = 10110001$$

and

$$10011100 + 10011100 = 00000000.$$

The fact that the sum of two binary sequences $a_1a_2 \cdots a_n + b_1b_2 \cdots b_n = 00 \cdots 0$ if and only if the sequences are identical is the basis for a data security system used to protect Internet transactions.

Suppose that you want to purchase a compact disc from **http://www.amazon.com**. Need you be concerned that a hacker will intercept your credit-card number during the transaction? As you might expect, your credit-card number is sent to Amazon in a way that protects the data. We explain one way to send credit-card numbers over the Web securely. When you place an order with Amazon, the company sends your computer a randomly generated string of 0's and 1's called a *key*. This key has the same length as the binary string corresponding to your credit-card number and the two strings are added (think of this process as "locking" the data). The resulting sum is then transmitted to Amazon. Amazon in turn adds the same key to the received string, which then produces the original string corresponding to your credit-card number (adding the key a second time "unlocks" the data).

To illustrate the idea, say you want to send an eight-digit binary string such as $s = 10101100$ to Amazon (actual credit-card numbers have very long strings) and Amazon sends your computer the key $k = 00111101$. Your computer returns the string $s + k = 10101100 + 00111101 = 10010001$ to Amazon, and Amazon adds k to this string to get $10010001 + 00111101 = 10101100$, which is the string representing your credit-card number. If someone intercepts the number $s + k = 10010001$ during transmission it is no value without knowing k.

The method is secure because the key sent by Amazon is randomly generated and used only one time. You can tell when you are using an encryption scheme on a Web transaction by looking to see if the Web address begins with "https" rather than the customary "http." You will also see a small padlock in the status bar at the bottom of the browser window.

Public Key Cryptography

Unlike auctions such as those on eBay, where each bid is known by everyone, a silent auction is one in which each bid is secret. Suppose that you wanted to use your Twitter account to run a silent auction.

How could a scheme be devised so that users could post their bids in such a way that the amounts are intelligible only to the account holder? In the mid-1970s, Ronald Rivest, Adi Shamir, and Leonard Adleman devised an ingenious method that permits each person who is to receive a secret message to tell publicly how to scramble messages sent to him or her. And even though the method used to scramble the message is known publicly, only the person for whom it is intended will be able to unscramble the message. The idea is based on the fact that there exist efficient methods for finding very large prime numbers (say about 100 digits long) and for multiplying large numbers, but no one knows an efficient algorithm for factoring large integers (say about 200 digits long). The person who is to receive the message chooses a pair of large primes p and q and chooses an integer e (called the *encryption exponent*) with $1 < e < m$, where $m = \text{lcm}\,(p - 1, q - 1)$, such that e is relatively prime to m (any such e will do). This person calculates $n = pq$ (n is called the *key*) and announces that a message M is to be sent to him or her publicly as $M^e \bmod n$. Although e, n, and M^e are available to everyone, only the person who knows how to factor n as pq will be able to decipher the message.

To present a simple example that nevertheless illustrates the principal features of the method, say we wish to send the messages "YES." We convert the message into a string of digits by replacing A by 01, B by 02, . . . , Z by 26, and a blank by 00. So, the message YES becomes 250519. To keep the numbers involved from becoming too unwieldy, we send the message in blocks of four digits and fill in with blanks when needed. Thus, the messages YES is represented by the two blocks 2505 and 1900. The person to whom the message is to be sent has picked two primes p and q, say $p = 37$ and $q = 73$, and a number e that has no prime divisors in common with $\text{lcm}\,(p - 1, q - 1) = 72$, say $e = 5$, and has published $n = 37 \cdot 73 = 2701$ and $e = 5$ in a public forum. We will send the "scrambled" numbers $(2505)^5 \bmod 2701$ and $(1900)^5 \bmod 2701$ rather than 2505 and 1900, and the receiver will unscramble them. We show the work involved for us and the receiver only for the block 2505. We determine $(2505)^5 \bmod 2701 = 2415$ by using a modular arithmetic calculator such as the one at **http://users.wpi.edu/~martin/mod.html**.†

†Provided that the numbers are not too large, the Google search engine at **http://www.google.com** will do modular arithmetic. For example, entering 2505^2 mod 2701 in the search box yields 602. Be careful, however: Entering 2505^5 mod 2701 does not return a value, because 2505^5 is too large. Instead, we can use Google to compute smaller powers such as 2505^2 mod 2701 and 2505^3 mod 2701 (which yields 852) and then enter $(852 \times 602) \bmod 2701$.

Thus, the number 2415 is sent to the receiver. Now the receiver must take this number and convert it back to 2505. To do so, the receiver takes the two factors of 2701, $p = 37$ and $q = 73$, and calculates the least common multiple of $p - 1 = 36$ and $q - 1 = 72$, which is 72. (This is where the knowledge of p and q is necessary.) Next, the receiver must find $e^{-1} = d$ (called the *decryption exponent*) in $U(72)$—that is, solve the equation $5 \cdot d = 1 \bmod 72$. This number is 29. See **http://www.d.umn.edu/~jgallian/msproject06/chap8.html #chap8ex5** or use a Google search box to compute 5^k for each divisor k of $|U(72)| = \phi(9) \cdot \phi(8) = 24$ starting with 2 until we reach $5^k \bmod 72 = 1$. Doing so, we obtain $5^6 \bmod 72 = 1$, which implies that $5^5 \bmod 72 = 29$ is 5^{-1} in $U(72)$.

Then the receiver takes the number received, 2415, and calculates $(2415)^{29} \bmod 2701 = 2505$, the encoded number. Thus, the receiver correctly determines the code for "YE." On the other hand, without knowing how pq factors, one cannot find the modulus (in our case, 72) that is needed to determine the decryption exponent d.

The procedure just described is called the *RSA public key encryption scheme* in honor of the three people (Rivest, Shamir, and Adleman) who discovered the method. It is widely used in conjunction with web servers and browsers, e-mail programs, remote login sessions, and electronic financial transactions. The algorithm is summarized below.

Receiver

1. Pick very large primes p and q and compute $n = pq$.
2. Compute the least common multiple of $p - 1$ and $q - 1$; let us call it m.
3. Pick e relatively prime to m.
4. Find d such that $ed \bmod m = 1$.
5. Publicly announce n and e.

Sender

1. Convert the message to a string of digits.
2. Break up the message into uniform blocks of digits; call them M_1, $M_2,\ldots, M_k$.
3. Check to see that the greatest common divisor of each M_i and n is 1. If not, n can be factored and our code is broken. (In practice, the primes p and q are so large that they exceed all M_i, so this step may be omitted.)
4. Calculate and send $R_i = M_i^e \bmod n$.

Receiver

1. For each received message R_i, calculate $R_i^d \bmod n$.
2. Convert the string of digits back to a string of characters.

Why does this method work? Well, we know that $U(n) \approx U(p) \oplus U(q) \approx Z_{p-1} \oplus Z_{q-1}$. Thus, an element of the form x^m in $U(n)$ corresponds under an isomorphism to one of the form (mx_1, mx_2) in $Z_{p-1} \oplus Z_{q-1}$. Since m is the least common multiple of $p - 1$ and $q - 1$, we may write $m = s(p - 1)$ and $m = t(q - 1)$ for some integers s and t. Then $(mx_1, mx_2) = (s(p - 1)x_1, t(q - 1)x_2) = (0, 0)$ in $Z_{p-1} \oplus Z_{q-1}$, and it follows that $x^m = 1$ for all x in $U(n)$. So, because each message M_i is an element of $U(n)$ and e was chosen so that $ed = 1 + km$ for some k, we have, modulo n,

$$R_i^d = (M_i^e)^d = M_i^{ed} = M_i{}^{1+km} = M_i(M_i^m)^k = M_i 1^k = M_i.$$

In 2002, Ronald Rivest, Adi Shamir, and Leonard Adleman received the Association for Computing Machinery A. M. Turing Award, which is considered the "Nobel Prize of computing," for their contribution to public key cryptography.

An RSA calculator that does all the calculations is provided at **http://www.d.umn.edu/~jgallian/msproject06/chap8.html#chap8ex5**. A list of primes can be found by searching the Web for "list of primes."

Digital Signatures

With so many financial transactions now taking place electronically, the problem of authenticity is paramount. How is a stockbroker to know that an electronic message she receives that tells her to sell one stock and buy another actually came from her client? The technique used in public key cryptography allows for digital signatures as well. Let us say that person A wants to send a secret message to person B in such a way that only B can decode the message and B will know that only A could have sent it. Abstractly, let E_A and D_A denote the algorithms that A uses for encryption and decryption, respectively, and let E_B and D_B denote the algorithms that B uses for encryption and decryption, respectively. Here we assume that E_A and E_B are available to the public, whereas D_A is known only to A and D_B is known only to B, and that $D_B E_B$ and $E_A D_A$ applied to any message leaves the message unchanged. Then A sends a message M to B as $E_B(D_A(M))$ and B decodes the received message by applying the function $E_A D_B$ to it to obtain

$$(E_A D_B)(E_B(D_A(M)) = E_A(D_B E_B)(D_A(M)) = E_A(D_A(M)) = M.$$