The Mathematics of Public-Key Cryptography

The search for privacy in an age of electronic communications has given rise to new methods of encryption. These methods are more practical than older ones and are mathematically more interesting

by Martin E. Hellman

The electronic communications systems that are proliferating throughout modern society offer speed, accuracy and ever diminishing cost. They also present serious problems of security. As the ordinary transactions conducted in person, on the telephone or by written correspondence have come increasingly to be conducted by new kinds of electronic systems the susceptibility of organizations and individuals to eavesdropping and forgery has grown dramatically. One way to prevent tampering with the new electronic systems and to protect the vast quantities of private information such as the credit records and medical histories now stored in computer data banks is to resort to cryptosystems: methods for encrypting, or transforming, information so that it is unintelligible and therefore useless to those who are not meant to have access to it.

Encryption is a special form of computation, and almost all modern cryptosystems depend on difficulty of computation for their security; they effect transformations of data so complicated that it is beyond the economic means of an eavesdropper to reverse the process. (Accounts of intelligence operations during World War II reveal that as recently as 35 years ago systems offering this type of security were not widely available. Since then the cost of computation has dropped by a factor of about a million, so that the equipment necessary for secure encryption is now reasonably priced.) Given unlimited computing power (an unrealistic assumption) such computationally secure systems could be broken, but in practice they appear to be unbreakable.

At present mathematicians lack the tools for proving systems to be computationally secure, and the history of cryptography demonstrates all too well that supposedly unbreakable systems often have hidden flaws. It is hoped that discoveries in complexity theory, a branch of mathematics that studies the difficulty (or cost) of computation, will eventually provide the tools needed to establish provably secure cryptosystems: computationally secure systems that can be guaranteed to be free of hidden flaws. In the meantime a group of mathematical problems characterized by a certain kind of computational intractability are serving as the basis of a new class of encryption procedures that are in many ways superior to current techniques. The proposed new systems, which were first put forward by Ralph Merkle, Whitfield Diffie and me at Stanford University, are termed public-key cryptosystems. To understand the significance of the term it is necessary to consider briefly how methods of encryption have developed historically.

A = 00000	B = 00001	C = 00010	D = 00011	E = 00100		
F = 00101	G = 00110	H = 00111	I = 01000	J = 01001		
K = 01010	L = 01011	M = 01100	N = 01101	O = 01110		
P = 01111	Q = 10000	R = 10001	S = 10010	T = 10011		
U = 10100	V = 10101	W = 10110	X = 10111	Y = 11000		
Z = 11001	= 11010	. = 11011	, = 11100	? = 11101		
; = 11110	: = 11111					
a = 00	b = 01	c = 02	d = 03	e = 04	f = 05	g = 06
h = 07	i = 08	j = 09	k = 10	l = 11	m = 12	n = 13
o = 14	p = 15	q = 16	r = 17	s = 18	t = 19	u = 20
v = 21	w = 22	x = 23	y = 24	z = 25	A = 26	B = 27
C = 28	D = 29	E = 30	F = 31	G = 32	H = 33	I = 34
J = 35	K = 36	L = 37	M = 38	N = 39	O = 40	P = 41
Q = 42	R = 43	S = 44	T = 45	U = 46	V = 47	W = 48
X = 49	Y = 50	Z = 51	0 = 52	1 = 53	2 = 54	3 = 55
4 = 56	5 = 57	6 = 58	7 = 59	8 = 60	9 = 61	= 62
. = 63	, = 64	; = 65	? = 66			

CRYPTOGRAPHIC SYSTEM is a mathematical system for encrypting, or transforming, information so that it is unintelligible and therefore useless to those who are not meant to have access to it. The encryption process generally begins with the conversion of the plaintext, or unenciphered message, into a string of numbers by means of a digital "alphabet" such as one of those shown here. In some cryptosystems it is more convenient to work with binary numbers, and so in the rather simple alphabet shown at the top five bits (binary digits) have been allocated to represent each letter, number or punctuation mark in the plaintext. Each bit can take two values (0 or 1), making a total of 2^5, or 32, characters in this alphabet. In other cryptosystems it is simpler to think in terms not of a binary (base-2) number system but a decimal (base-10) one. In alphabet shown at bottom two decimal digits have been allocated for each plaintext symbol, providing total of 10^2, or 100, characters. (Some of these may not be needed.)

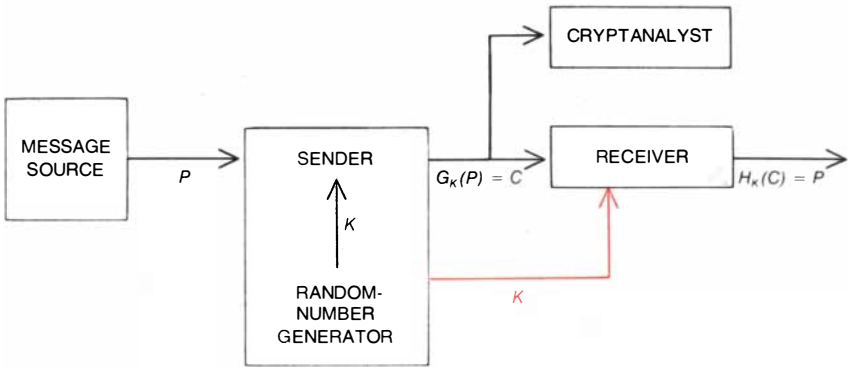
Any cryptographic technique, such as the substitution and transposition of symbols, that operates on a message without regard to its linguistic structure is called a cipher and is said to generate a ciphertext. (Codes, which I shall not discuss here, operate on larger linguistic units such as words or phrases.) More precisely, the basis of any cipher is an invertible function: an operation (performed by the sender of the message) that converts a plaintext, or unenci-

phered message, into a ciphertext and has an inverse operation (performed by the intended receiver of the message) that recovers the plaintext from the ciphertext.

Originally the security of ciphers depended on the secrecy of the entire encryption process, but eventually ciphers were developed for which the algorithm, or sequence of steps, of encryption could be revealed without compromising the security of a particular ciphertext. In such ciphers—the conventional cryptosystems of today—a set of specific parameters, called a key, is supplied along with the plaintext message as an input to the enciphering algorithm and along with the ciphertext message as an input to the deciphering algorithm. In other words, the specific transformations of the plaintext and the ciphertext depend on the key as well as on the enciphering and deciphering algorithms. In fact, the algorithms themselves can be made public, because the security of the ciphertext generated in such a system depends entirely on the secrecy of the key. In the new public-key cryptosystems not only the algorithms but also the key for implementing the enciphering algorithm can be revealed without compromising the security of the ciphertext.

To understand the advantages conferred by the public-key arrangement, consider a conventional cryptosystem employed for protecting information transmitted over an insecure communications channel such as radio. A system of this type can be viewed as a mathematical strongbox with a resettable combination lock. After the sender and the receiver agree on a sequence of numbers—the key—to serve as the combination of the lock, the sender places his message in the box, sets the combination and closes the lock. If the strongbox—the cryptosystem—is secure, no third party who intercepts the box while it is en route to the receiver will be able to get into it to read or alter the message. In other words, a conventional cryptosystem prevents eavesdroppers from extracting information from an insecure channel and prevents forgers from modifying information in the channel.

Until quite recently the principal users of cryptosystems were the military and diplomatic services of the world. The drawbacks of the conventional systems are particularly troubling, however, to the new commercial users of cryptography. To begin with, before any information can be enciphered and transmitted over an insecure channel the receiver and the sender must agree on a key. Since the security of the system depends exclusively on the secrecy of the key, the key must be transmitted by means of a secure channel such as a trusted courier, a system that is slow and costly. The distribution of keys is a particular problem in those instances when the individuals seeking privacy have had no prior communication or when privacy must be maintained over a large network, two situations that are often encountered in commercial dealings. Indeed, the cost and inconvenience of relying on couriers to distribute the amount of key information that is needed for any broad application of cryptography are virtually prohibitive.



IN A CONVENTIONAL CRYPTOSYSTEM someone who wants to send a private message is provided with an algorithm, or general enciphering procedure, *G* and obtains a key *K*, as is shown in this model of the flow of information in such a system. The key, which must be kept secret, is a set of parameters (typically a collection of large random numbers) for implementing the algorithm, which can be made public. In other words, the algorithm and key together specify the actual enciphering transformation G_K. The sender operates on the plaintext *P* with G_K to generate a ciphertext *C*, or $G_K(P)$. The ciphertext can then be transmitted over an insecure communications channel such as a radio channel. Another public algorithm effects the inverse operation G^{-1}, designated *H* for convenience. Knowing *H* and *K*, the receiver operates on the ciphertext *C* with H_K to recover the plaintext *P*, or $H_K(C)$. A cryptanalyst who knows *G* and *H* and intercepts *C* but does not know *K* will not be able to decipher the message. Hence security of such a system lies entirely in secrecy of its keys. As a result keys needed by users of system must be distributed by means of secure channel (*color*) such as trusted courier.

The requirement of key distribution is not the only drawback of the conventional cryptosystems currently in service. They also fail to meet fully the requirements of message authentication. Since a single key is shared between the sender and the receiver, there is nothing to prevent the receiver from sending himself messages that appear to come from the sender. Consider the difficulties such forgeries could cause in electronic mail or electronic banking systems. Conventional cryptosystems, then, cannot offer the same insurance against disputes over what message (if any) was sent that the exchange of signed documents can. The public-key systems, however, provide answers to both the problem of distributing keys and the problem of authentication.

In a public-key cryptosystem the sender and the receiver rather than agreeing on a single key each generate two distinct keys of their own: an enciphering key *E*, which serves to implement the system's enciphering algorithm, and a deciphering key *D*, which serves to implement the system's deciphering algorithm. The keys are related in the sense that they serve to implement inverse operations: operating on a plaintext message first with the transformation specified by *E* and then with the transformation specified by *D* reproduces the message, and in some (but not all) systems applying the transformations in the reverse order also reproduces the message. The trick is that it is computationally infeasible to derive *D* from *E*: the calculation would require a vast amount of computing time, perhaps thousands or even billions of years on the most powerful computer. Hence each user can publish his enciphering key in a public file such as a telephone book without compromising his deciphering key, which is kept secret. As in a conventional cryptosystem, the general procedures for enciphering and deciphering are public information. Therefore anyone who wants to transmit information to a particular person simply enciphers the information with that person's listed key *E* and sends the ciphertext over an insecure channel. Only the intended receiver, who knows the corresponding secret key *D*, will be able to decipher the transmitted message.

To return to the strongbox analogy, a public-key system provides a strongbox with a new kind of lock, which has two combinations: one to lock the box and one to unlock it. (The box does not lock automatically when it is closed.) The locking combinations of all such strongboxes are made public, so that anyone can lock information in a particular strongbox, but only the individual who owns the strongbox and has set the two combinations will be able to get the information out. With this kind of system there is obviously no need of a secure channel for the distribution of keys. Moreover, some of the public-key systems allow for the construction of a "digital signature" that prevents the forgery of messages by a receiver as well as

by a third party. In other words, these systems make it possible to dispense with the transporting of signed documents and to depend exclusively on the electronic transmission of information.

If an eavesdropper had unlimited computing resources, he could break a public-key system and recover a plaintext. The enciphering operation E is public and the number of possible plaintexts is immense but finite, and so E could be applied to each plaintext until the intercepted ciphertext was reproduced. Since such an attack requires an impossibly large amount of computing time, however, the public-key systems can still be computationally secure. There are also similar techniques for deriving the secret deciphering key D from the public enciphering key E, but once again the computational infeasibility of implementing those algorithms provides the systems with practical security. To put it another way, the systems are based on what are called trapdoor one-way functions. A one-way function is an easily computed function for which it is computationally infeasible to compute the inverse function. A trapdoor one-way function is an easily computed function for which it is computationally infeasible to compute the inverse function unless certain specific information that was employed in the design of the function is known. Hence like a trapdoor in the floor of a motion-picture haunted house, such functions are easy to go through in one direction, but unless one possesses the special trapdoor information (analogous in the haunted house to which brick to pull or which panel to push) the reverse process takes an impossibly long time.

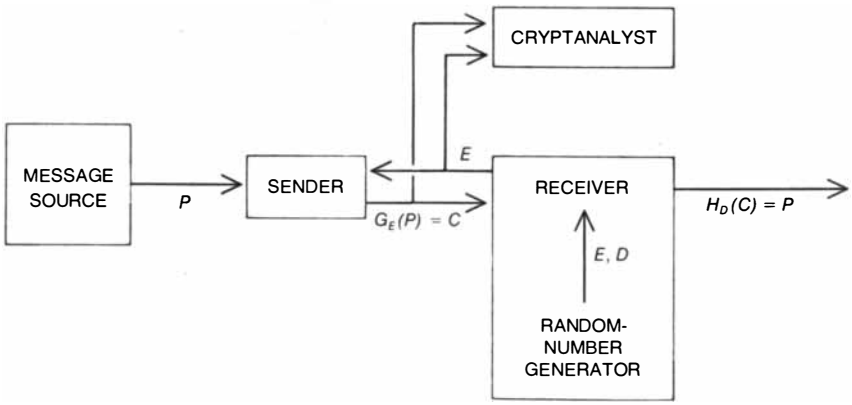
The search for trapdoor one-way functions on which to base public-key cryptosystems led naturally to the class of problems that complexity theory has identified as nondeterministic, polynomial-time problems, or NP problems. For the purposes of these cryptosystems the most important property of the NP problems is that at present all the algorithms that are known for finding general solutions to them call for rapidly increasing amounts of time, although a proposed solution can be quickly checked. In other words, as the size n of such a problem increases, the number of computational steps required to solve the problem increases in proportion to, say, an exponential function of n such as 2^n, whereas the number of steps required to check a possible solution increases in proportion to a polynomial function of n such as n^2. Exponential functions increase far more rapidly than polynomial ones, so that a method of solution that requires exponentially increasing amounts of computer time is impossible to implement for even moderate-size problems. For mathematicians concerned with cryptography the appeal of the NP problems resides in the fact that although it might take someone billions of years to find a solution to such a problem, once he found it he could convince the rest of the world of its validity in seconds. As a result these problems lend themselves readily to the construction of one-way functions. And for the NP problems on which public-key cryptosystems have been based it has been possible to build trapdoors into the functions as well.

I shall describe here two public-key cryptosystems based on NP problems: the trapdoor knapsack system, developed by Merkle and me, and the RSA system, developed by Ronald Rivest, Adi Shamir and Leonard Adleman at the Massachusetts Institute of Technology. The first of these cryptosystems is based on a well-known NP problem called the knapsack or subset sum problem: Given a knapsack of length C and a set of n rods all of the same diameter as the knapsack but of lengths $a_1, a_2, \ldots, a_n$, find a subset of the rods that completely fills the knapsack. To put it another way, given a set of numbers $a_1, \ldots, a_n$ and a sum C, determine which of the numbers add up to C.

The public-key cryptosystem based on this problem operates as follows. The sender begins by converting his message into a string of binary numbers. For example, five bits (binary digits) might be allocated for each letter, number or punctuation mark in the plaintext, providing an alphabet of 2^5, or 32, characters: A = 00000, B = 00001, C = 00010 and so on. Once the message is in binary form the sender consults a public directory of enciphering keys, which lists an ordered set of n numbers $a = (a_1, a_2, \ldots, a_n)$ for each user of the system. This set is called the user's trapdoor knapsack vector.

In mathematics an ordered set of n numbers is called an n-dimensional vector, and the "dot," or scalar, product of any two vectors of the same dimension is defined as follows: for vectors $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$ the dot product $a \cdot b$ equals $a_1b_1 + a_2b_2 + \ldots + a_nb_n$. This form of vector multiplication is the basic operation of the enciphering algorithm in the trapdoor knapsack system. To encipher the string of binary numbers that represents his message the sender first breaks the string into blocks of n bits, and for each block $x = (x_1, x_2, \ldots, x_n)$ he forms the dot product $C = a \cdot x$ of that block with the public enciphering vector a, that is, $C = a_1x_1 + a_2x_2 + \ldots + a_nx_n$.

The sum C is the information the sender transmits over the insecure channel, so that any eavesdropper is confronted with the task of recovering x from C and the numbers $a_1, \ldots, a_n$. In what follows it will be convenient to refer to the elements of a vector x as the x_i's (or to the elements of a vector a as the a_i's), where the values of i are taken to be the integers from 1 to n. Since each x_i is equal to either 0 or 1, one can see that the problem of recovering x from C is equivalent to solving the knapsack or subset sum problem for these values of



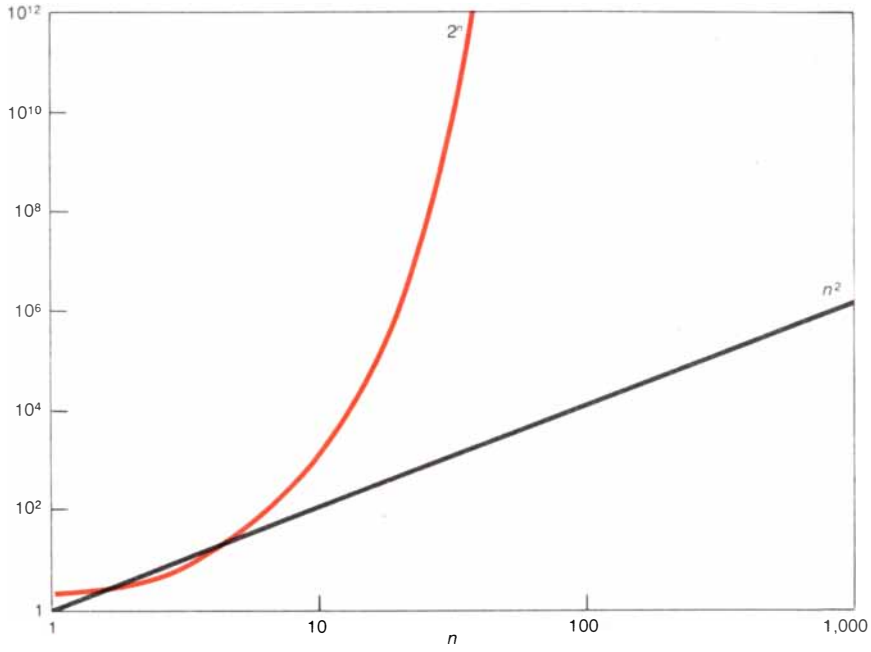
IN A PUBLIC-KEY CRYPTOSYSTEM there is no need of a secure channel for the distribution of keys. As is shown here, each receiver generates two distinct keys: a public key E for implementing the public enciphering procedure G and a secret key D for implementing the public deciphering procedure H, which is the inverse of G. The keys E and D are related in the sense that they serve to specify inverse transformations G_E and H_D, but given E it is computationally infeasible to derive D: computing D from E would require thousands or even billions of years on the largest computer imaginable. Hence the receiver may communicate his enciphering key E over an insecure channel, as is shown here, or even list it in a public directory without compromising his deciphering transformation. A person who wants to send the plaintext P to the receiver operates on it with the receiver's enciphering transformation G_E to generate a ciphertext C, or $G_E(P)$. This ciphertext is transmitted over an insecure channel, and the receiver operates on it with the deciphering transformation H_D to recover the plaintext P, or $H_D(C)$. As long as the deciphering key D is kept secret there is no way for an eavesdropper to decipher the transmitted message. The challenge in designing such a system is to find general procedures G and H for which pairs of inverse keys E and D are easily generated but for which it is computationally infeasible to compute D from E. A source of such pairs is a group of mathematical problems that are said to be in the class NP (*see illustration on opposite page*).

C and the a_i's. The receiver must solve the same knapsack problem, but to simplify the task he has additional information: his secret trapdoor parameters and deciphering key.

These steps should be made clear by a simple example [*see illustration on page 155*]. Consider a plaintext message in which the first word is HOW. In binary form the message begins 0011101-1101011011010. (This binary string, in which the last five-bit block represents the space between HOW and the next word in the message, is generated by the five-bit binary alphabet described above.) Now assume that the intended receiver's public enciphering key is $a = (2{,}292,\ 1{,}089,\ 211,\ 1{,}625,\ 1{,}283,\ 599,\ 759,\ 315,\ 2{,}597,\ 2{,}463)$, or $a_1 = 2{,}292$, $a_2 = 1{,}089$ and so on. Here n equals 10, and the first block of information, which consists of the first n bits in the binary plaintext, is $x = (0,\ 0,\ 1,\ 1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 0)$. It is enciphered, then, as $C = a_1x_1 + \ldots + a_nx_n$, or $C = (2{,}292 \times 0) + (1{,}089 \times 0) + (211 \times 1) + (1{,}625 \times 1) + (1{,}283 \times 1) + (599 \times 0) + (759 \times 1) + (315 \times 1) + (2{,}597 \times 1) + (2{,}463 \times 0)$. Therefore C equals 6,790, and to decipher the message it is necessary to determine which of the a_i's add up to 6,790. (If a_i is included in the sum, x_i is 1 and vice versa.)

None of the known methods for solving the knapsack problem is substantially less time-consuming than conducting an exhaustive search, that is, adding up all the 2^n possible subsets of the a_i's to see which subset yields C. In the example given above, where the number of elements n is equal to 10, this might be considered a workable approach. Someone intercepting C could try all the 2^{10}, or 1,024, possible combinations of the publicly listed a_i's and thereby recover the vector x. In this instance the number of elements in a is too small to provide real secrecy. The knapsack problem is an NP one, however, and therefore the computational difficulty of all known solution methods rapidly "blows up." When the number of elements n is, say, 1,000, the number of possible subsets $2^{1{,}000}$ is greater than the number of atoms in the known universe. Deciphering by checking $2^{1{,}000}$ different subsets is quite impossible, and so C effectively shields the secret information x. On the other hand, enciphering 1,000 bits of information in this system is quite efficient, requiring no more than 1,000 additions.

So far I have described what appears to be a one-way function: apparently no one, including the receiver, will be able to recover x. If the elements of vector a are chosen at random, this is exactly the type of system that results. Even in this simple example, however, a trapdoor has been built in. The vector a has been structured in such a way that with a small amount of additional information

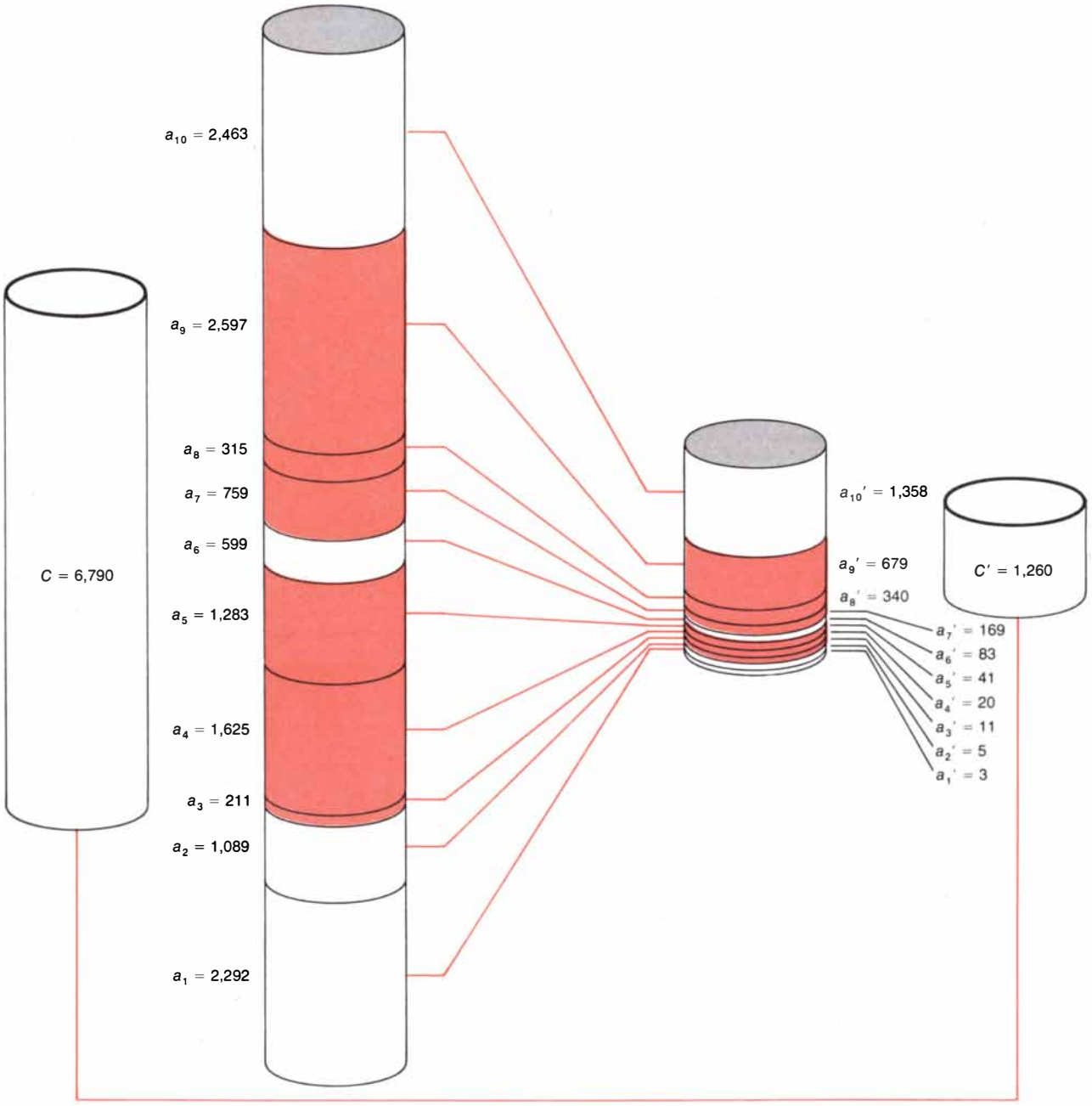


PROBLEMS IN THE CLASS NP (which stands for nondeterministic, polynomial time) are characterized by the fact that although it is easy to check a nondeterministic, or guessed, solution, it is hard to find a correct solution: As the size n of an NP problem increases, the number of computational steps and hence the time required to check a solution increase in proportion to a polynomial function of n such as n^2 (*black curve*), but all known methods of finding a solution increase in proportion to a more rapidly growing function of n, typically an exponential one such as 2^n (*colored curve*). Exponential functions increase far more rapidly, and when n is sufficiently large, NP problems become computationally infeasible. Hence they lend themselves readily to the design of one-way functions: easily computed functions whose inverses are infeasible to compute. In some cases such problems can be developed into trapdoor one-way functions: easily computed functions whose inverses are infeasible to compute unless certain facts employed in design of functions are known. Trapdoor one-way functions serve as basis of public-key cryptosystems: public key specifies easily computed function, which is infeasible to invert unless one knows secret key; that key specifies easily computed inverse function.

x can be derived from C much more rapidly than by an exhaustive search. As I have noted, not all NP problems lend themselves to the insertion of such a trapdoor. Here the trapdoor can be devised because there are certain vectors for which the knapsack problem is not difficult to solve. The receiver takes one of those special vectors a' and disguises it, publishing the resulting ordinary-looking vector a in the public file of enciphering keys. The trapdoor information enables him to move back and forth between a difficult knapsack problem involving a and the easy but equivalent knapsack problem involving a'.

To be more precise, in generating his public vector a the receiver begins by choosing a vector $a' = (a_1', \ldots, a_n')$ in which each element a_i' is larger than the sum of the preceding elements $a_1' + a_2' + \ldots + a_{i-1}'$. For example, if a' equals (3, 5, 11, 20, 41, 83, 169, 340, 679, 1,358), then a_2', which equals 5, is greater than a_1', which equals 3; a_3', which equals 11, is greater than $a_1' + a_2'$, which equals 3 + 5, or 8, and so on. Now, consider a ciphertext $C' = 1{,}260$ that was generated with this special vector a'. In other words, C' equals $a' \cdot x'$ for some binary vector $x' = (x_1', \ldots, x_n')$, that is, 1,260 equals $3x_1' + 5x_2' + 11x_3' + 20x_4' + 41x_5' + 83x_6' + 169x_7' + 340x_8' + 679x_9' + 1{,}358x_{10}'$.

Once again the problem of decipherment is equivalent to solving a knapsack problem, but in this instance because of the special property of the vector a' the solution x' is easily determined. To begin with, a_{10}', which equals 1,358, is greater than C', which equals 1,260, and so obviously cannot be part of the subset sum, that is, the rod is too long to fit into the knapsack. Hence x_{10}' must be 0. The next-largest element in the vector is a_9', or 679, which is less than C', or 1,260. As the special property of a' dictates, the sum of the eight remaining elements of a' must be less than 679, and so those elements alone cannot "fill" the knapsack of length 1,260. Therefore 679 must be part of the sum, and x_9' must be 1. Since x_9' equals 1 and x_{10}' equals 0, the equation $C' = a' \cdot x'$ can now be rewritten as $1{,}260 = 3x_1' + 5x_2' + 11x_3' + 20x_4' + 41x_5' + 83x_6' + 169x_7' + 340x_8' + 679 + 0$. Subtracting 679 from both sides of the equation reduces the problem to determining which of the elements $a_1', \ldots, a_8'$ add up to $1{,}260 - 679$, or 581 (the length of the still empty part of the knapsack). Since a_8'



KNAPSACK PROBLEM is an NP problem from which a trapdoor one-way function can be derived. The cylinder and set of rods shown at the left illustrate the classic knapsack problem: Given a knapsack, or cylinder, of length C and a set of n rods all of the same diameter as the knapsack but of lengths $a_1, a_2, \ldots, a_n$, find a subset of rods that fills the knapsack completely. This problem is in the class NP because the best method known for solving it is not much more efficient than trying all 2^n possible subset sums to see which one equals C, and yet a guessed solution can be checked with no more than n additions. Even in the small 10-rod examples shown here, finding a solution (*color*) by this method requires the testing of 2^{10}, or 1,024, different subsets, and when n is, say, 100, the task becomes impossible. An ordered set of numbers such as $a = (a_1, \ldots, a_n)$ or $x = (x_1, \ldots, x_n)$ is called a vector, and the "dot" product of two vectors $a \cdot x$ is defined as the sum $a_1x_1 + \ldots + a_nx_n$. Given a fixed vector a, a function of the variable vector x can be defined as the dot product of x with the vector a, that is, $f(x) = a \cdot x$. If the elements $x_1, \ldots, x_n$ of x are all equal to 0 or 1, then inverting this function, or determining which value of x gives a particular sum $C = a \cdot x$, is equivalent to solving the knapsack problem for C and the given values of $a_1, \ldots, a_n$. The function is one-way because the knapsack problem is in the class NP. Moreover, a trapdoor can be built into the function, because for certain vectors, or sets of rods, $a' = (a_1', \ldots, a_n')$ the knapsack problem is easy to solve. In these sets, such as the one shown in the problem at the right, each element is greater than the sum of the preceding elements. To determine which subset fills the knapsack begin with the last, or largest, element a_n'. In this case a_{10}' equals 1,358, which is greater than 1,260, the length of the cylinder C'. Hence a_{10}' is not in the subset (that is, in the sum $C' = a_1'x_1 + \ldots + a_{10}'x_{10}$, x_{10} equals 0). But a_9', which equals 679, is smaller than 1,260, and since the remaining elements in the set add up to a number even smaller than a_9', it must be in the subset (that is, x_9 equals 1). The problem is now reduced to filling the remainder of the cylinder, whose length is $C' - a_9'$, or 581, with a subset of the remaining rods $a_1', \ldots, a_8'$, and so on. Continuing in the same way, the problem can be solved (or the function based on it can be inverted) with no more than 10 comparisons and 10 subtractions. As colored lines indicate, there is a way to move back and forth between the easy and hard knapsack problems. Parameters for effecting that transformation are secret trapdoor information for trapdoor one-way function based on knapsack problem (*see illustration on page 155*).

equals 340, which is less than 581, it is included in the sum. Thus x_8' is 1. Continuing in this manner, it can be determined that the x' is the original message block $x = (0, 0, 1, 1, 1, 0, 1, 1, 1, 0)$.

Constructing an easy knapsack vector such as a' is not difficult, but how does the receiver get from a' to a and back again? To accomplish that feat he chooses two large random numbers w and m and generates the vector a according to the equation $a_i = a_i'w$ modulo m, for each i from 1 to n. The expression "modulo m" indicates that a_i should be taken to be the remainder left when $a_i'w$ is divided by m. For example, if w equals 764 and m equals 2,731, consider the element a_4' of vector a'. Since a_4' equals 20, $a_4'w$ equals 15,280. Dividing 2,731 into 15,280 gives 5 with a remainder of 1,625, so that a_4, or $a_4'w$ modulo m, equals 1,625.

Modular arithmetic plays a large part in public-key cryptosystems, because it turns smooth, or continuous and continually increasing or decreasing, functions into discontinuous ones, introducing a large factor of confusion into the calculation of their inverses: the values of x that correspond to particular values of $f(x)$. Consider the simple function $f(x) = 4x$. As x increases, the value of $f(x)$ increases in a very orderly way; for example, $f(3)$ is 12, $f(4)$ is 16, $f(5)$ is 20, $f(6)$ is 24 and so on. As a result if one is given a specific number $y = f(x)$, it is not difficult to determine x by a process of guesswork and elimination without ever actually solving the equation $y = 4x$. In other words, if a function is smooth, then no matter how hard solving explicitly for x is, it may still be possible to determine the value of x for a particular $f(x)$ through trial and error. For example, if $f(x)$ equals 20, one might guess that x equals 3. Then $f(x)$ would equal 12, which is too small, so that the correct value of x must be greater than 3. If, however, $x = 6$ were tried, $f(x)$ would equal 24, which is too large, so that x must be less than 6, and so on. Such smooth functions present problems in public-key systems, which depend on functions for shielding numbers.

Consider what happens, however, if modularity is added. When $f(x)$ equals, say, $4x$ modulo 7, as x increases, the value of $f(x)$ jumps around in a quite haphazard way. For example, $f(1)$ is 4, $f(2)$ is 1, $f(3)$ is 5, $f(4)$ is 2, $f(5)$ is 6 and $f(6)$ is 3. Even in such a simple case it is clear that this function that includes modularity provides far better protection for the values of x than the one that does not include it. In the case of the trapdoor knapsack system, applying modularity in the generation of the difficult knapsack vector a prevents the recovery of a' for anyone who does not know the secret transformation parameters w and m.

For anyone who does know w and m,

however, the conversion back into a' would not be difficult at all. In fact, with those parameters it is quite easy to convert the difficult knapsack problem involving the vector a and the transmitted ciphertext message C into an easy knapsack problem involving the vector a' and a new sum C' and then to solve (or decipher) for x. To begin with, it is a simple mathematical exercise to calculate the inverse of w modulo m, that is, the number w^{-1} that when multiplied by w modulo m gives 1. There is a fast procedure for finding inverses in modular arithmetic (based on Euclid's algorithm for finding the greatest common divisor of two numbers) that makes this calculation efficient, even in a realistic system in which w and m are on the order of 50 digits long. (Incidentally, for this purpose w and m must be chosen to be relatively prime; if they had a common factor, or divisor, there would be no multiplicative inverse of w modulo m.)

To decipher the message C, then, the receiver first calculates $C' = Cw^{-1}$ modulo m. To see what this operation accomplishes remember that C equals $a_1x_1 + \ldots + a_nx_n$. In modular arithmetic, as in ordinary arithmetic, it is permissible to multiply both sides of an equation by the same quantity so that Cw^{-1} modulo m equals $a_1x_1w^{-1} + \ldots + a_nx_nw^{-1}$, or $a_1w^{-1}x_1 + \ldots + a_nw^{-1}x_n$, modulo m. The vector a was generated from the vector a', however, by computing $a_i = a_i'w$ modulo m for each i. Hence a_iw^{-1} equals a_i' modulo m for each i, that is, a_1w^{-1} equals a_1' modulo m, a_2w^{-1} equals a_2' modulo m and so on. Substituting these last results into the preceding equation, one discovers that C', or Cw^{-1} modulo m, equals $a_1'x_1 + \ldots + a_n'x_n$, or $a' \cdot x$.

In other words, calculating Cw^{-1} modulo m is all that is needed to convert the problem of deciphering C into an easy knapsack problem. The receiver simply applies his secret vector a' to solve the knapsack problem for C' and recover x. For those who do not have the secret information w and m, however, there is no easily implemented method known of transforming C into C' (or translating the difficult vector a into the easy vector a') for efficient deciphering.

In the 10-element example I have been discussing it is easy to verify that the numbers w and m relating a and a' or C and C' are respectively 764 and 2,731, and that w^{-1} is 1,605. Notice that in this public-key system the trapdoor information w, m and a' is virtually synonymous with the secret deciphering key w^{-1}, m and a'. The same is not true of all public-key cryptosystems. (In practical cryptosystems based on the trapdoor knapsack scheme it may be desirable to introduce additional security by iterating the conversion process, so that the public and the private vectors differ by several transformations and several intermediate vectors.)

Since only the numbers w, or w^{-1}, and m and the vector a' must be kept secret, all users of the trapdoor knapsack system can employ the same public computer program for generating both their public key and their secret parameters. Utilizing a random-number generator to provide the program with a', w and m will serve to ensure that each user's pair of keys is distinct. Similarly, a public program could be made available that would encipher messages and, when it was supplied with the secret parameters, decipher messages. Therefore no mathematical ability is required to implement the trapdoor knapsack cryptosystem. Any useful public-key system must have this same characteristic.

The second public-key system I shall describe is based on an NP problem that has an even longer and more distinguished history of resisting solution than the knapsack problem: the problem of factoring a large number, or finding all the primes that divide it evenly. (A prime number is an integer that is divisible only by 1 and itself.) This problem has been studied since the time of the ancient Greeks, and although some progress has been made with it, factoring a 200-digit number would still take the most powerful modern computer about a billion years. To give a smaller example, consider the problem of factoring 29,083. Calculating by hand, it would take the better part of an hour to find the only two factors of this number: 127 and 229. It takes less than a minute, however, to verify that those factors are correct, suggesting that the problem of factoring is a good basis for the construction of a one-way function. Figuring out how to build a trapdoor into such a function presents more difficult obstacles, but they have been overcome by Rivest, Shamir and Adleman, the designers of the RSA system.

To generate a public enciphering key each user of the RSA public-key system (or rather a program run on his computer) chooses two large random prime numbers p and q. The product n of these two numbers and another random number E are placed in the public file as the user's enciphering key. To apply the key a sender first converts his message into a string of numbers, which he then breaks into blocks $P_1, P_2, \ldots$. In this instance it is not necessary to use binary numbers, but each plaintext number P_i must be between 0 and $n - 1$. (The enciphering and deciphering functions operate modulo n and so can distinguish between numbers in this range only.) Locating the user's public key (E, n) in the directory, the sender computes for each plaintext number P_i the ciphertext number $C_i = P_i^E$ modulo n. For example, if p equals 5, q equals 11 and E equals 3, then the user's enciphering key is (3, 55), and to encipher the plaintext information $P = 2$ a sender would compute $C = 2^3 = 8$ modulo 55. (Because the numbers in this example are so small modularity does not yet play a role.)

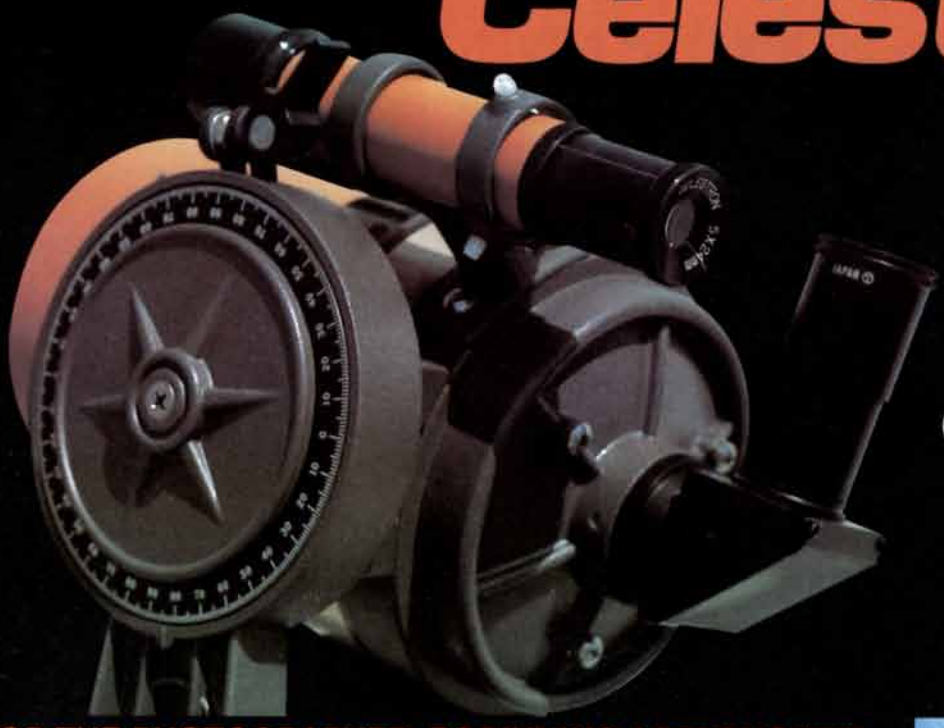
The RSA public-key cryptosystem is based on the fact that although finding large prime numbers is computationally easy, factoring the product of two such numbers is at present computationally infeasible. (It is important to understand that because there are computationally efficient primality tests, determining whether a number is prime is much easier than factoring a number of about the same size.) To decipher a ciphertext C_1, $C_2, \ldots$, the user employs n and a secret deciphering key D derived from the prime factors p and q of n.

To understand how the deciphering key is derived it is necessary to consider the number $(p - 1)(q - 1)$: a well-known object in number theory called Euler's totient function. This function, which is written $\phi(n)$, is defined as the number of integers between 1 and n that have no common factor with n. It is not hard to see that if n equals pq, then $\phi(n)$ equals $(p - 1)(q - 1)$. The number $\phi(n)$ is introduced here because in functions, such as the one used for enciphering in the RSA system, that are calculated modulo n, arithmetic in the exponent is carried out not modulo n but modulo $\phi(n)$. An example may make this idea clearer. Consider the expression 2^{11}

P	0	1	2	3	4	5	6	7	8	9	10	...
$C = P^3$	0	1	8	27	64	125	216	343	512	729	1000	...
$C' = P^3$ modulo 11	0	1	8	5	9	4	7	2	6	3	10	...

MODULAR ARITHMETIC is employed in many cryptosystems to further disguise information already transformed by an enciphering function. As is shown here, the value of an integer a modulo another integer b is defined as the remainder left when a is divided by b. For example, 27 modulo 11 equals 5, because 11 goes into 27 twice with 5 left over. The usefulness of this operation is shown for a simple enciphering function $C = P^3$. As P increases, the continuous way P^3 increases makes it possible to invert the function, or determine what value of P corresponds to a particular value of C, even though there is no simple formula for expressing P as the cube root of C. More precisely, a value of P that gives too small a value of C is itself too small, whereas value of P that gives too large value of C is itself too large. When modularity is added, however, so that C' equals P^3 modulo 11, values of function are thrown into disarray. As P increases, C' changes in a quite discontinuous way, effectively shielding P.

Celestron 90



MAKSUTOV-CASSEGRAIN TELESCOPE

Dealer inquiries invited

FOR THE PHOTOGRAPHER: BEGINNING ASTRONOMER: FOR THE HOME WITH A VIEW: FOR THE NATURALIST:

The Celestron 90 is the most compact and lowest-priced member of the Celestron family of telescopes. And, judging from the overwhelming response to the C90, it may well become the most popular Celestron telescope.

The C90 is unusually small and lightweight for its focal length (1000mm). This — coupled with its unusually-low price and high quality — makes the C90 Astro Telescope extremely desirable as a telephoto lens or spotting scope too. Accordingly, we've designed a complete product line around the C90 optical system.

The heart of the C90 series is the 90mm aperture, f/11, Maksutov-Cassegrain optical system, which is identical on all three C90 versions. As a result, the C90 is a telescope that's also a telephoto or a telephoto that's also a telescope. By adding the appropriate optional accessories, it's possible to use any C90 version in another mode.

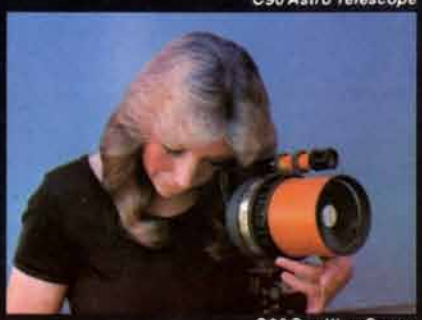
The C90 Astro Telescope is supplied with advanced features usually found only on larger, more expensive telescopes — features like star locating setting circles, an automatic tracking system and slow-motion controls on both axes — that make using the telescope easy and convenient. The standard ocular and Barlow lens provide magnifications of 55×, 140× and 200×.

As a daytime telescope, the C90 is remarkably powerful. With it, you'll see bees buzzing in flowers 20 feet away, recognize the faces of friends ½ mile away, read auto license plates 2 miles away, or identify airliners many miles away.

At night, there are countless wonders in the night sky visible through the C90. The Moon, our nearest celestial neighbor, will display a multitude of craters, rugged mountains, rills and relatively smooth plains (maria).

C90 Telephoto Lens

C90 Astro Telescope



C90 Spotting Scope

Free 4-page brochure. Or Send $2.00 for 32-page full color catalog on how to select and use a Celestron telescope.

Celestron

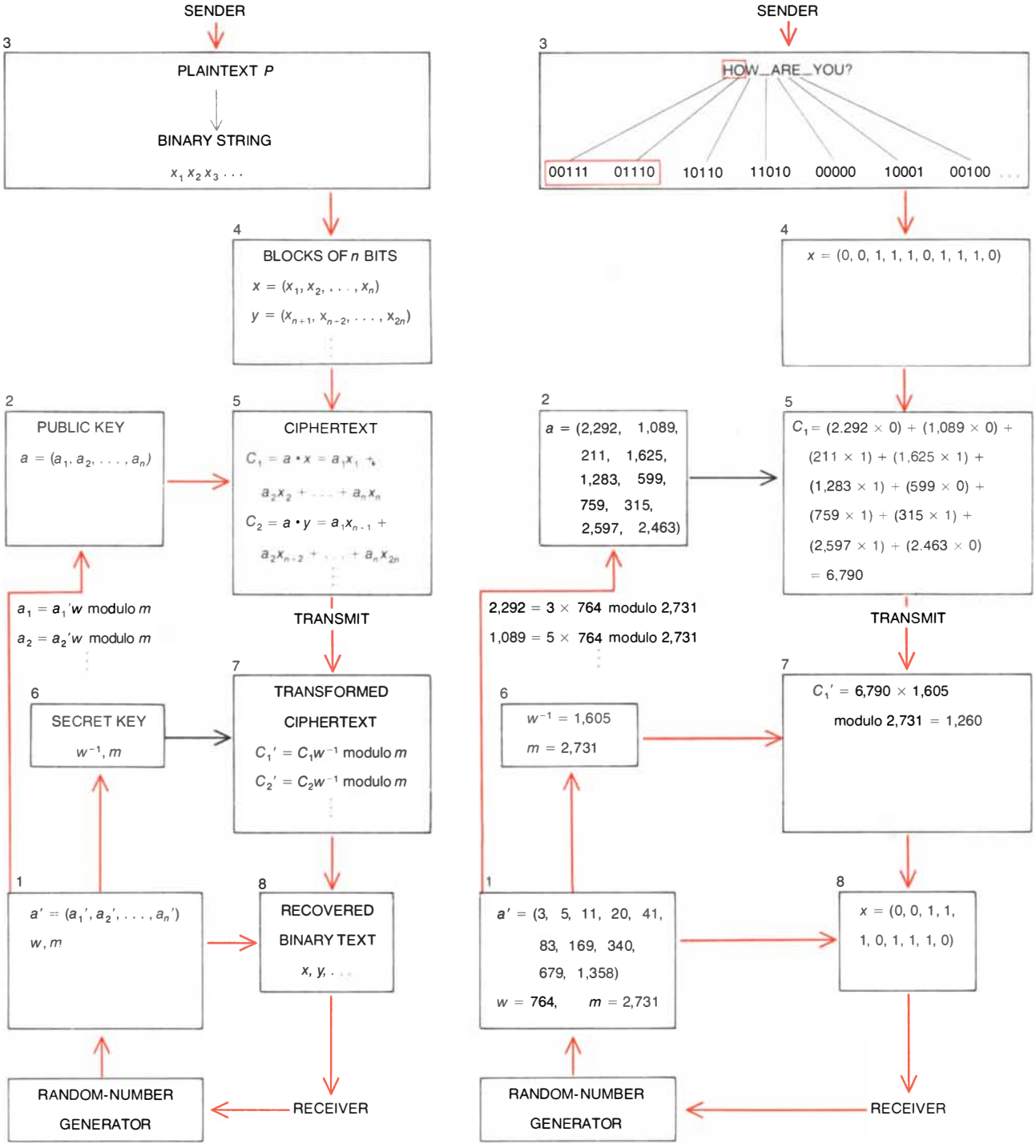
2835 Columbia Street ● Box 3578-2C ● Torrance, CA 90503 ● Phone (213) 328-9560

modulo 10. Since 2^{11} is equal to 2,048 and dividing 10 into 2,048 leaves a remainder of 8, the expression is equal to 8. Note that calculating by first reducing the exponent modulo 10 does not give the correct answer, since 11 modulo 10 equals 1, and 2^1 equals 2. On the other hand, 10 equals 2×5, and so $\phi(10)$ equals $(2 - 1)(5 - 1)$, or 4. Since 11 modulo 4 equals 3, calculating 2^{11} modulo 10 by first reducing the exponent modulo 4 gives the correct answer: 2^3, or 8.

Now, the properties of $\phi(n)$ guarantee that there is always a multiplicative inverse D of E modulo $\phi(n)$, that is, ED modulo $(p - 1)(q - 1)$ is equal to 1. In fact, there is always a fast, computationally easy method for deriving D. (It is not hard to see that in the example discussed above, where p equals 5, q equals 11 and E equals 3, $(p - 1)(q - 1)$ equals 40 and D equals 27, because 3×27 is one more than 2×40.) This inverse D is the secret deciphering key for the RSA system. To decipher a ciphertext the receiver computes C_i^D modulo n for each ciphertext number C_i. C_i equals P_i^E modulo n, so that C_i^D modulo n equals $(P_i^E)^D$, or P_i^{ED}, modulo n. Because arithmetic in the exponent is performed modulo $\phi(n)$ and ED modulo $\phi(n)$ equals 1, P_i^{ED} modulo n equals P_i^1, or P_i. In other words, raising the ciphertext to the Dth power and reducing modulo n recovers the plaintext.

Hence in the RSA cryptosystem modularity plays a dual role, not only blocking the recovery of the secret deciphering key D from the public enciphering key (E, n) but also, by its presence in the enciphering algorithm, preventing a direct recovery of the plaintext from the ciphertext. The difficulty of computing D from the public information (E, n) depends on the difficulty of factoring n, or of deriving p and q from n. Once again the example I have given is too small to provide real secrecy, but since factoring large numbers is a very difficult problem, the difficulty of breaking the cipher blows up rapidly as n increases. When p and q are chosen so that n is about 200 digits long, it appears to be computationally infeasible for anyone but the intended receiver to decipher the message.

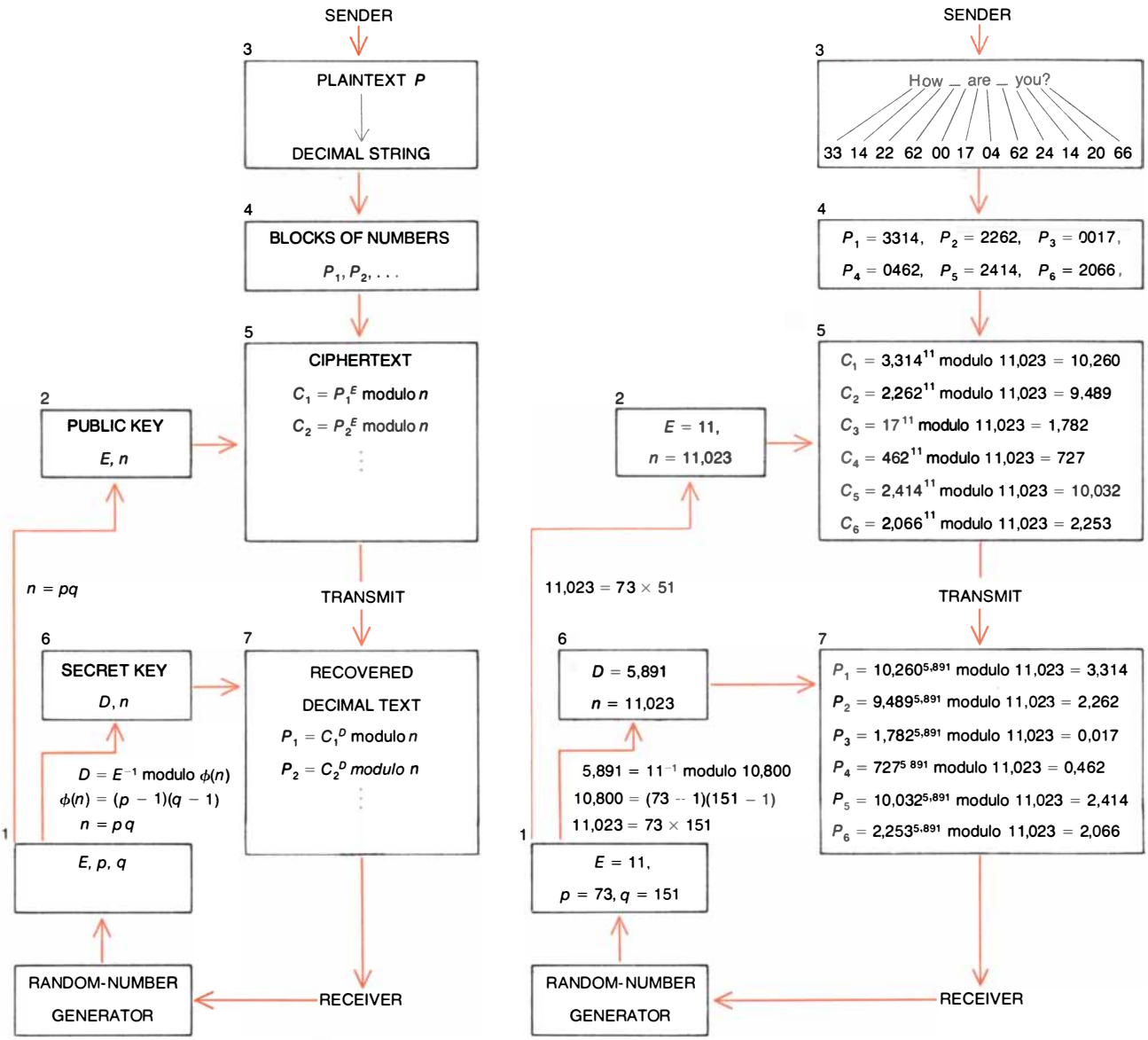
Just as the deciphering procedure (without the trapdoor information) must be computationally infeasible, the public enciphering procedure and secret deciphering procedure must be computationally efficient. At first the implementation of the RSA system appears to present some practical problems in this area. Consider the simple example in which the plaintext number $P = 2$ was transformed into the ciphertext number $C = 8$. To apply the deciphering algorithm $P = C^D$ modulo n it is necessary to calculate 8^{27} modulo 55 (which does indeed equal 2). Multiplying 8 by itself 27 times is, however, a cumbersome process involving large numbers and a great



FLOW OF INFORMATION in the trapdoor knapsack cryptosystem is shown at the left. The corresponding transformations of the first block of plaintext HO are shown at the right. In this public-key system based on the knapsack problem each receiver (by means of a random-number generator) selects a secret vector $a' = (a_1', \ldots, a_n')$ with the property that each element is greater than the sum of the preceding elements and also selects two large random numbers w and m with no common factors (*1*). The numbers w and m are the trapdoor parameters for converting the secret "easy" vector a' into a "difficult" public vector $a = (a_1, \ldots, a_n)$ by means of the equations $a_1 = a_1'w$ modulo m, $a_2 = a_2'w$ modulo m and so on. The difficult vector a is transmitted to the sender over an insecure channel or is listed in a public directory as the receiver's enciphering key (*2*). To encipher a plaintext P a sender begins by converting it into a binary string according to, say, the five-bit binary alphabet given at the top of the illustration on page 146 (*3*). The sender then looks up the receiver's public key and breaks the string into blocks of n binary digits (*4*). For example, in the system shown at the right (in which the numbers are far too small to provide real secrecy) there are 10 elements in the public vector, and so the binary string is divided into blocks of 10 bits each. Every block is enciphered by forming its dot product with a (*5*), that is, if the first block is $x = (x_1, \ldots, x_n)$, the first ciphertext number C_1 equals $a \cdot x$, or $a_1x_1 + \ldots + a_nx_n$, and so on. The ciphertext numbers are transmitted to the receiver over an insecure channel. The receiver recovers, say, x by calculating first w^{-1} (*6*) and then $C_1' = C_1w^{-1}$ modulo m (*7*). (The number w^{-1} is the inverse of w modulo m, the number that when multiplied by w gives 1 modulo m.) Remember that a_1 equals $a_1'w$ modulo m, a_2 equals $a_2'w$ modulo m and so on, and therefore a_1' equals a_1w^{-1} modulo m, a_2' equals a_2w^{-1} modulo m and so on. Hence C_1', or C_1w^{-1} modulo m, which equals $a_1x_1w^{-1} + \ldots + a_nx_nw^{-1}$, or $a_1w^{-1}x_1 + \ldots + a_nw^{-1}x_n$, modulo m, equals $a_1'x_1 + \ldots + a_n'x_n$. In other words, C_1' equals $a' \cdot x$, and the difficult knapsack problem of recovering x from C_1 and a has been converted back into the easy problem of recovering x from C_1' and a'. Only receiver, who possesses trapdoor information w and m and knows secret vector a', can effect transformation and recover x (*8*).

many computational steps. In more realistic RSA systems, where D would be a 200-digit number, this procedure would be impossible to carry out, even on a very powerful computer.

Fortunately there is a much faster method for calculating functions of this kind. First the binary expansion of the exponent (the expression of the exponent as a sum of powers of 2) is utilized to break up the function into a product of smaller factors; for example, 27 equals $1 + 2 + 8 + 16$, and so 8^{27} equals $8 \times 8^2 \times 8^8 \times 8^{16}$. Now, by calculating the smaller factors first and then taking their product, the number of operations required can be limited. For example, 8^2 modulo 55 can be evaluated with only one modular multiplication (an ordinary multiplication followed by an ordinary division), since 8×8, or 64, modulo 55 equals 9. Then 8^4, which is equal to $8^2 \times 8^2$, can be evaluated with an additional modular operation, $9 \times 9 = 81 = 26$ modulo 55, and so on. (Substituting the value of 8^2 modulo 55, namely 9, into the larger factors prevents the size of the numbers involved in the computation from blowing up.) Hence only seven modular multiplications are needed to calculate 8^{27}: four to evaluate 8^2, 8^4, 8^8 and 8^{16} and three more to multiply 8 times 8^2 times 8^8



RSA PUBLIC-KEY CRYPTOSYSTEM is based on the problem of factoring a large number, or finding all the prime numbers that divide it evenly. (A prime number is an integer that is divisible only by 1 and itself.) As is shown at the left, each receiver in the RSA system generates two large random prime numbers p and q, which serve as his secret trapdoor parameters, and a large random number E (1). There are computationally efficient tests for identifying primes such as p and q, but when these numbers are sufficiently large, it is computationally infeasible to derive them from their product pq, or n. The receiver lists E and n as his public deciphering key (2). To encipher a plaintext P the sender first converts it into a string of numbers, using the decimal alphabet shown at the bottom of the illustration on page 146 (3), and then breaks the string into blocks of equal length $P_1, P_2, \ldots$, so that each number P_i in this series is less than n (4). Each block is converted into a ciphertext number by raising it to the Eth power and then reducing modulo n, that is, C_1 equals P_1^E modulo n, C_2 equals P_2^E modulo n and so on (5). The ciphertext numbers $C_1, C_2, \ldots$ are transmitted over an insecure channel. Arithmetic in the exponent of a function that is calculated modulo n must be carried out modulo $\phi(n)$, where $\phi(n)$ equals $(p-1)(q-1)$, and so the receiver utilizes p and q to determine $\phi(n) = (p-1)(q-1)$ and then $D = E^{-1}$ modulo $\phi(n)$, which serves as his secret deciphering key (6). To convert $C_1, C_2, \ldots$ back into plaintext numbers the receiver raises each one to the Dth power and reduces it modulo n (7). Because C_i^D modulo n equals $(P_i^E)^D$, or P_i^{ED}, modulo n and ED modulo $\phi(n)$ equals 1, P_i^{ED} equals P_i modulo n. In other words, this operation inverts the enciphering transformation, recovering the plaintext number blocks $P_1, P_2, \ldots$. In the example shown at right the values of E, p and q are too small to provide real security.

times 8^{16}. Even when D is a 200-digit number, this method results in a deciphering procedure that is quite efficient, requiring at most 1,330 modular multiplications rather than the 10^{200} operations necessary in the straightforward approach.

The RSA system is a public-key cryptosystem that allows the direct generation of a digital signature: a number that can be appended to a ciphertext message to solve the problems of authentication mentioned above. To be of real service such a signature must be easy for the sender to generate and for the receiver to check but must be computationally infeasible for a third party or the receiver himself to generate. Of the various methods for generating digital signatures the simplest involves exploiting the inverse relation of the public enciphering and secret deciphering keys by reversing their roles. For example, in the RSA system the sender can utilize his own secret deciphering key D as a signing key, to compute the signature $S_i = P_i^D$ modulo n for each P_i in the series of plaintext numbers P_1, $P_2, \ldots$ that represent a message to be transmitted. (Remember that each P_i is chosen to be between 0 and $n - 1$.) Once the signatures $S_1, S_2, \ldots$ have been generated the sender enciphers each signed message block (P_i, S_i), using the receiver's public enciphering key. This second operation has nothing to do with signature generation; it simply ensures the privacy of the communication.

The receiver uses his own secret key to recover the signed message block (P_i, S_i), and then he looks up the sender's public key (E, n) and computes S_i^E modulo n for each i. D and E effect inverse operations regardless of the order of their application, and so since S_i equals P_i^D modulo n, S_i^E modulo n should be equal to P_i. If that is the case, the receiver can be sure that the message comes from the apparent sender and that it has not been tampered with. Since the digital signature depends on both the sender and the message sent, it offers a level of security different from that of a written signature, which is the same for all messages. With the digital signature neither the receiver nor a third party can alter the message without destroying the validity of the signature. (When the message to be sent is long, rather than signing each submessage separately it may be desirable to compress the message and calculate a single signature S; that compression can be effected in such a way that S still depends on the entire message P.)

The traditional difficulties of solving the knapsack and factoring problems can be taken as an encouraging sign that the public-key cryptosystems based on these problems are in practical terms secure. A past history of intractability cannot, however, be considered a proof that a system is secure. It is always possible, if unlikely, that at some time in the future computationally efficient general methods for solving these problems will be found. An even greater hazard is that a method will be discovered for breaking one of the cryptosystems without solving the corresponding general problem. For example, it is possible that although solving most regular knapsack problems is computationally infeasible, there is an easy way to solve the much smaller set of trapdoor knapsack problems. Similarly, it may be possible to recover the plaintext enciphered by the RSA technique without finding the factors of n. (Michael O. Rabin of the Hebrew University of Jerusalem has recently shown, however, that in the case where the enciphering exponent E is 2, the security of the RSA system is not simply dependent on the difficulty of factoring n but is actually equivalent to it. This finding constitutes an important first step toward the goal of developing provably secure systems.)

Cryptography has not yet advanced to the stage where it can prove the computational security of even a conventional system or a one-way function. Hence it is not surprising that there is no way to establish the security of the public-key systems, which are based on the more complex trapdoor one-way functions. It is hoped, however, that over the next decade or two complexity theory will advance to the point where such proofs can be formulated. Some progress has been made in this direction through the study of a special subset of the NP problems.

Remember that the NP problems are ideal candidates for one-way functions because finding a solution to them is computationally difficult but checking a proposed solution is computationally easy. Some of these problems such as the knapsack problem (but not factoring) belong to the subset of the NP problems that is called NP-complete. The NP-complete problems have the added property that if any one of them had an easily implemented method for finding general solutions, then all the NP problems would. Now, all cryptanalytic problems—problems of breaking cryptosystems—are in the class NP, since it is always easy to check the validity of a proposed key. Therefore if any NP-complete problem can be solved rapidly, it follows that all cryptographic systems can be broken easily. Roughly speaking, then, if the security of a cryptosystem could be shown to be equivalent in difficulty to an NP-complete problem, it would be as secure as any cryptographic system can be.

One flaw in this type of evaluation is that complexity theory deals with the "worst case" computational difficulty of solving a problem, whereas cryptography is concerned with the average or typical difficulty of solving a problem. For example, in current complexity theory a problem whose solution requires $10^{1,000}$ operations 1 percent of the time but only 100 operations 99 percent of the time is considered to be difficult. Obviously a cryptosystem that can be broken 99 percent of the time is worthless. Workers in complexity theory are aware of this shortcoming and are currently developing more suitable measures of computational difficulty.

Although factoring is not an NP-complete problem, it has through the years largely resisted the attack of some of the best mathematical minds. That is why Rabin's proof, which establishes an equivalence between the difficulty of factoring and breaking an RSA scheme, is an extremely important result. Until such time as the security of proposed cryptosystems can be formally evaluated, however, it is a worthwhile (and intellectually challenging) exercise to try to break them.

In electronic communications systems, as in any new technology, there is a potential for misuse. For example, the danger of foreign or domestic intelligence organizations spying on American citizens who rely on these systems is a real one. It has recently been revealed that U.S. microwave telephone traffic is being monitored in at least one foreign embassy in Washington. In the late 1960's the U.S. Government's "Operation Shamrock" intercepted international Telex communications to and from "targeted" individuals, including antiwar activists. If such excesses are to be limited, both legal and technical safeguards are needed.

There is always a trade-off between the rights of citizens to privacy and the desire of government intelligence agencies to limit the availability of secure cryptosystems. A conflict in this area has recently arisen concerning the Federal Data Encryption Standard, a conventional cryptosystem issued by the National Bureau of Standards for nonmilitary encryption purposes. The National Security Agency convinced the International Business Machines Corporation, the company that designed the standard, to reduce the key size to 56 bits. Although there is controversy surrounding the issue, I believe the reduction in key size was meant to weaken the standard so that if it were ever employed by a foreign organization, it could be broken by the National Security Agency. Issues similar to this one will certainly arise as the new public-key systems become commercial realities. It is to be hoped that these issues will be decided by an open discussion of the relative needs of the intelligence community and the citizenry rather than, as appears to have been the case, by the unilateral decision of the intelligence community that its needs take precedence.