Appendix VII: Public Key Systems

Before about 1970 there were a few American nongovernment mathematicians who were interested in cryptology, some even to the extent of carrying out research in it over a period of years; [2], [145], and [155] are examples. However, public cryptology really came to life in 1976 with the publication of "New Directions in Cryptography" by Diffie and Hellman, [86], although that paper was clearly the result of earlier research on the part of the authors. (Their interest in cryptology had come to the attention of NSA somewhat earlier when Hellman objected strongly to the Data Encryption Standard (DES), [9].) One of the ideas in "New Directions" that caught the imagination of many mathematicians and computer scientists around the world was what the authors called "public key cryptography", an idea that had been put forth internally by James Ellis in 1970, [92], under the appellation "nonsecret encryption".

Encryption can be expressed abstractly as the transformation of a plain text message into a cipher message under the control of a key. In classical cryptographic systems, the transformation is usually fairly simple: for instance, the key may determine the starting point in a long sequence of "random" numbers which are to be added to the sequence of plain text values, the result being a sequence of cipher values. To decrypt, the same sequence of random values, starting at the same point, must be subtracted from the cipher sequence. The random sequence and the key (starting point) must therefore be available to both the sender and the receiver of the message, but not to anyone else. The transformation (addition) is simple and can easily be undone by subtraction.

In contrast, public key encryption uses some much more complicated transformation, one that is "easy" to perform with any specified key but can be undone only with a different key (this decryption key must obviously be related to the encryption key). The idea is that the transformation should be of such a nature that the relation between encryption and decryption keys is so complicated it requires an exorbitant amount of work to calculate one from the other unless one knows some secret ingredient of the relationship. The transformation algorithm itself can be made public as long as the secret ingredient is known only to the receiver of a message; to enable someone to send him a message, the recipient chooses an encryption key, calculates its related decryption key using the secret information that he has about the relationship, and openly sends the encryption key to the message sender. The sender (and possibly everyone else in the world) knows the transformation method and the encryption key and can therefore encrypt a message and send it to the receiver. However, only the receiver can decrypt it because no one else can feasibly determine the decryption key.

Not many transformations suitable for public key cryptosystems have been found so far. The best-known ones are the knapsack system, [163], the RSA system, [194], the exponentiation system, [77], and McEliece's system, [162]. Some variations of these as well as a few other public key cryptosystems are discussed in [55] and [180], with

sketches of attacks on most of them. Both the RSA and exponentiation systems were invented internally before their public appearances; C.C. Cocks, [73], devised what is essentially the RSA system, and Rick Proto, [188], had suggested the exponentiation scheme as an "irreversible" transformation, prior to Ellis' paper on nonsecret encryption. No public key cryptosystem has yet been used operationally,

[redacted]

A. Knapsacks, [163]

The knapsack public key system is based on the difficulty of solving the following problem: a large number (perhaps 100) of positive integers of various sizes are specified, and someone selects a subset of them, adds up the integers in the subset, and tells you that sum. Your problem is to determine which integers were selected for the subset.

This problem is hard in the sense that no one has succeeded in devising an algorithm guaranteed to solve any such problem (i.e., for any such set of specified integers) with an amount of work depending polynomially on the number of specified integers. There are certain classes of specified sets, though, for which it is easy to solve the problem. For example, suppose the integers a(1), a(2), ..., a(100) have the property that $a(1)<a(2)$, $a(1)+a(2)<a(3)$, ... , and so on up to $a(1)+a(2)+\ldots+a(99)<a(100)$. (Such a set is called superincreasing.) Then if the sum of a subset is S, it is easy to tell whether a(100) was in the subset because S must be less than a(100) if it was not in the subset and S must be greater than or equal to a(100) if it was in the subset. If a(100) is found to be in the subset, subtract it from S to get S'; otherwise, let S' equal S. Then it is easy to tell whether a(99) was in the subset by comparing it with S'. This process can be continued, and will quickly determine the subset whose sum is S.

Merkle and Hellman, [163], suggested that a public key cryptosystem could be based on a superincreasing set of positive integers by transforming it into another set as follows. First choose a modulus, m, that is larger than the sum of all integers in the set, and choose a multiplier x that is relatively prime to m; m, x, and the a's are secret cryptovariables. Then let b(i) be the least positive residue of x*a(i) modulo m, and use the b's as public cryptovariables. To send a message, convert it to binary form by any convenient coding, split it up into blocks of 100 bits each, and use each block in turn to select a subset of the b's according to whether successive bits are 0 or 1. Form the sums of the selected subsets and transmit them as cipher.

The recipient can decrypt (a block at a time) by forming y*S modulo m where S is a sum (cipher) and y is the inverse of x modulo m; only the recipient can determine y because no one else knows x and m. Since S = sum {b(i)*p(i)}, where p(i) is the i-th plain text bit of the block, it is true that S is congruent to sum {x*a(i)*p(i)} and therefore y*S is congruent to sum {a(i)*p(i)}. But then the problem is one of dealing with a superincreasing set and that is easy. A cryptana-

lyst, on the other hand, knows only the b's and is therefore faced with a facsimile of the general knapsack problem.

Hellman and Merkle also remarked that one could iterate the transform procedure by choosing another modulus m' larger than the sum of the b's and another multiplier x' relatively prime to m', and taking c(i) to be the least positive residue of x'*b(i). This could be repeated as often as desired. Note, however, that the public cryptovariables (the "knapsack numbers") will get larger and larger, on the average, and therefore the cipher sums will also get larger and larger. In fact, even a single transformation will result in an increase of the average number of cipher bits over the number of plain text bits.

B. RSA, [194]

The so-called RSA public key cryptosystem is based on the difficulty of factoring integers, a process that is known empirically to be laborious but which has not been proved to have any specified degree of difficulty. C.C. Cocks of GCHQ in 1973, [73], suggested a slightly less general version of the RSA system, which did not appear until four years later.

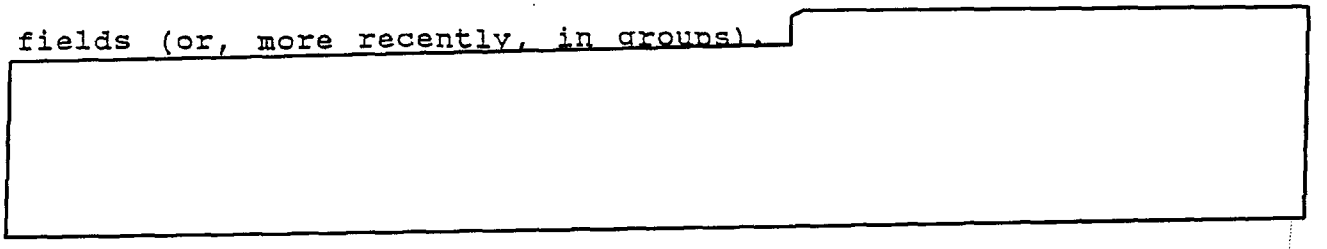
In the RSA system, two large prime numbers p and q (of perhaps 100 digits each) are selected by the recipient to be the secret cryptovariables. The recipient also chooses a number e relatively prime to both p-1 and q-1. The numbers m=p*q and e are the public cryptovariables. To encipher a message, the sender converts it by any convenient means into a sequence of positive integers each less than m. He then raises each of them to the e-th power modulo m and transmits the sequence of powers as cipher. Since the recipient knows p and q, he is able to determine the unique number d for which d*e is congruent to 1 modulo (p-1)*(q-1); he then raises each cipher number to the d-th power modulo m to obtain the sequence of plain text values. If a cryptanalyst were able to factor the publicly available modulus m, then he also could decrypt the message; the assumption is that factoring is too hard.

One drawback to the RSA system is that it requires one to find prime numbers large enough to make factoring infeasible. It is thought that each prime should have in the neighborhood of 100 digits. It might seem to a nonmathematician that finding such primes would itself involve a factoring process, but that is not the case. Testing an integer for primality is much easier than actually finding its factors, and the appearance of RSA in fact stimulated some new ideas which have improved such testing. Public researchers also discovered weaknesses associated with using primes of a certain form in the RSA system, so that finding acceptable ones is somewhat more complicated than merely finding large ones. Of course, performing modular arithmetic with such large numbers demands either multiple precision computer routines or specially designed chips.

C. Exponentiation, [77]

The public key cryptosystem based on exponentiation relies for security on the difficulty of finding "logarithms" of elements in finite

fields (or, more recently, in groups).



Logarithms in finite fields are defined differently from ordinary logarithms. The nonzero elements of a finite field of order p^n may be represented as powers of a primitive element of the field. There is more than one primitive element available, but once one has been selected (called the generator), the logarithm of an element relative to the generator is defined to be the least power to which the generator must be raised to equal the element. The logarithm is unique modulo $(p^n)-1$.

In the exponentiation system, some large finite field $GF(p^n)$ is selected to be the public cryptovariable. To encrypt a message, the sender converts it by any convenient method to a sequence of elements of the finite field. To encipher each value y of the message, he then chooses some exponent e which has an inverse d modulo p^n-1, and transmits to the recipient y^e. The recipient also chooses some exponent g which has an inverse f modulo p^n-1, and transmits back to the sender the field element $(y^e)^g$. The sender forms $((y^e)^g)^d$ which equals y^g and sends that to the recipient who can then calculate $(y^g)^f=y$. Since (e,d) and (g,f) are known only to the sender and recipient, respectively, no one else is supposed to be able to decipher the message y. If a cryptanalyst were able to solve the discrete logarithm problem, however, he could recover e and g, determine d and f, and so decrypt the message.

Note that when p=2, the field elements can be represented as polynomials with coefficients in the field of integers modulo 2, reduced modulo some irreducible polynomial f(x). Assuming that f is primitive, any nonzero polynomial of the field may then be represented as some power of x, and exponentiation can be performed by an appropriate type of linear shift register with feedback polynomial f. This is the connection with the distance problem.

A drawback of the exponentiation public key cryptosystem is that it requires three transmissions (two from sender to receiver, one from receiver to sender) in order to communicate information. Because of this, it seems to be suitable only for short and infrequent messages. In fact, when it first appeared (in [86]) the authors (Diffie and Hellman) proposed it merely for use in establishing the cryptovariables to be used for a conventional cryptographic system.

The idea of using the group of points on an elliptic curve over a finite field, and exponentiating in that group, is receiving some attention at the present time, [165]. Not much is known about it as yet, although elliptic curves are being intensively studied in connection with factoring algorithms as well.

D. McEliece's System, [162]

In this public key cryptosystem, the public key is a large k by n matrix G', where k<n, formed in the following way. First, the recipient chooses a k by n generator matrix G for some Goppa code ([137] contains a definition of Goppa codes and further references for them) with the ability to correct t errors. Then he chooses a random nonsingular k by k matrix S (one with not too many 0 entries) and a random n by n permutation matrix P. The matrices G, S, and P are the secret cryptovariables, and G' is the matrix product S*G*P.

To encipher a message, the sender converts it to a sequence of bits by any convenient means, breaks it up into k-bit segments, and enciphers each segment by considering it as a vector and multiplying it by G', then garbling t randomly chosen bits of the product. That is, the n-bit cipher sequence c is m*G'+e where m is the plain text segment and e is the "error vector".

The recipient deciphers the message by first multiplying c by the inverse of P to get the vector $c*P^{-1}=m*S*G+e*P^{-1}$. He then uses G to "correct" the "errors" the sender deliberately introduced, the result being m*S. Finally, he multiplies this by S^{-1} (i.e., S inverse) to obtain the message m. A cryptanalyst, not knowing S, P, or G (the code), is supposedly unable to decipher the message without expending a prohibitive amount of work.