1987

Public key cryptography

Jennifer Seberry
University of Wollongong, jennie@uow.edu.au

Public key cryptography

Abstract

We are going to devote most of our attention in this talk to the RSA Public Key Cryptosystem because it not only remains unbroken but it has some other useful features for digital signatures and authentication. We will briefly mention some other methods which have been compromised to some degree, and one, McEliece's which has not, but which are still valid when both keys are kept secret and some have other features which may be useful.

Disciplines

Physical Sciences and Mathematics

Publication Details

Seberry, J, Public key cryptography, Secure Data Communications Workshop, Digest of Papers, IEEE, Melbourne, 1987, 1-17.

Paper at Melb
IEEE Workshop 31/7/87

PUBLIC KEY CRYPTOGRAPHY

Jennifer Seberry
University College
The University of New South Wales
Australian Defence Forces Academy
Canberra

INTRODUCTION

We are going to devote most of our attention in this talk to the RSA Public Key Cryptosystem because it not only remains unbroken but it has some other useful features for digital signatures and authentication.

We will briefly mention some other methods which have been compromised to some degree, and one, McEliece's which has not, but which are still valid when both keys are kept secret and some have other features which may be useful.

PUBLIC KEY SYSTEMS

A public key cryptosystem is a cryptographic system in which each encryption process is governed by not one but two keys. The two keys are inverses of each other, that is to say anything encrypted with one can be decrypted with the other and vice versa. The important additional property of a public key crptosystem is that given one of the keys, it is extremely difficult to find the other. This allows one of the keys to be made public while its inverse is kept secret, giving the systems their name. Public key cryptosystems have two very important properties.

Because it is not necessary to keep both of the keys secret, one can be made readily available, published in a phonebook for example. Anyone wanting to transmit a confidential message can encrypt it in the **public key** of the addressee with assurance that only the addressee will be able to read it.

Just as a message encrypted in a public key can be produced by anyone but can only be read by the holder of the corresponding secret key, a message encrypted in a secret key, a message encrypted in a secret key can be read by anyone, using the corresponding public key, but could only have been produced by the holder of the secret key. This gives it the fundamental property of a signature.

Use is made of *modular arithmetic*.

Mathematicians write the expression

$$a \equiv b (mod\ m)$$

(a is congruent to b modulo m) to denote the fact that the integer m divides exactly the difference of the integers a and b. For example,

$$32 \equiv -4 (mod\ 12).$$

Note that if the remainder on dividing a by m is b, then $a \equiv b (mod\ m)$. Hence,

$$5124491 \equiv 12172 (mod\ 21753).$$

In fact, the remainder on dividing a by m is the only number b which is congruent to a modulo m such that $0 \leq b < m$. One very important cosequence of the definition of congruence is that if $p(x)$ is any polynomial function of x with integer coefficients, then $p(a) \equiv p(b) (mod\ m)$ whenever $a \equiv b (mod\ m)$.

PUBLIC KEY DISTRIBUTION SYSTEM

A public key distribution system is a mechanism which allows two people who have never had any prior secure contact to establish a secure channel "out of thin air". Public key distribution systems do not provide any signature mechanism but, at present, some are faster and more compact than public key cryptosystems which makes them better for many applications.

The first practical public key distribution system makes use of the apparent difficulty of computing logarithms over a finite (Galois) field GF(q) with a prime number q of elements (the numbers $\{0,1,...,q-1\}$ under arithmetic mod q). Let

$$Y = \alpha^X mod\ q, \text{ for } 1 < X < q - 1,$$

where α is a fixed primitive element of GF(q) (that is the powers of α range over the nonzero elements $1,2,...,q-1$ of GF(q)), then X is referred to as the logarithm of Y to the base α, over GF(q):

$$X = \log_\alpha Y \text{ over GF}(q), \text{ for } 1 < Y < q - 1 .$$

Calculation of Y from X is easy, taking at most $2\log_2 q$ multiplications. For example

$$\alpha^{18} = (((\alpha^2)^2)^2 * \alpha^2.$$

Computing X from Y, on the other hand can be much more difficult and, using the best known algorithm, has a computional complexity similar to finding the factors of a number close to q . Each user generates an independent random number X_i chosen uniformly from the set of integers $1,2,...,q-1$. X_i is kept secret but

$$Y_i = \alpha^{X_i} mod\ q$$

is placed in a public file with the addressee's name and address. When users i and j wish to communicate privately they use

$$K_{ij} = \alpha^{X_i X_j} mod\ q$$

as their key. User i computes K_{ij} by obtaining Y_j from the public file and letting

$$K_{ij} = Y_j^{X_i} mod\ q = (\alpha^{X_j})^{X_i} mod\ q$$

User j obtains K_{ij} in similar fashion.

Thus

$$K_{ij} = Y_i^{X_j} mod\ q$$

and of course

$$\alpha^{X_j X_i} = \alpha^{X_i X_j} mod\ q.$$

Another user must compute K_{ij} from Y_i and Y_j, for example by computing

$$K_{ij}^{\log_\alpha Y_j} = \alpha^i mod\ q$$

before his/her own X_k can be used to establish a bogus key

$$K_{ik} = (\alpha^{X_i})^{X_k} mod\ q.$$

If logarithms over GF(q) are easily computed, the system can be broken, but at present neither a threateningly fast method of doing this computation nor a way to bypass the logarithm and compute K_{ij} from Y_i and Y_j without first obtaining either X_i or X_j is known. If q is a prime slightly less than w, all quantities are representable as w bit numbers. Exponentiation then takes at most $2\times w$ multiplications over GF(q), while computing the logarithm requires $q^{1/2} = 2^{w/2}$ operations, using the best currently known algorithm. The cryptanalytic effort therefore grows exponentially relative to encryption or decryption. If $w = 200$, at most 400 multiplications are required to compute Y_i from X_i, or K_{ij} from X_i and X_j, yet taking logarithms over GF(q) is thought to require 2^{100} or approximately 10^{30} operations. This system can be implemented efficiently with respect to both speed and storage, and a variation in which q is not prime was the basis for an experimental local secure network at the Mitre Corporation.

In the next sections we will see how modular arithmetic is used to convert the two problems, factorization and discrete logarithms, to public key cryptosystems. In fact, except for the scheme described by McEliece (see later) and Sloane, all known possible public key techniques are based on these two problems.

THE RSA PUBLIC KEY CRYPTOSYSTEM.

Shortly after the publication of Diffie and Hellman's seminal paper on public key cryptosystems, Rivest, Shamir and Adleman (for whom the RSA system is named) discovered a very elegant candidate for such a scheme. Their technique makes use of the following simple number theoretic result: if $R = pq$, where p, q are distinct primes, and $\phi(R)=(P-1)(Q-1)$, then

$$X^{\phi(R)} \equiv 1 (mod\ R),$$

for any X which is not divisible by either P or Q. The designer of an RSA cryptosystem selects at random two large (about 100 digits) primes P and Q and calculates R = PQ. The designer also selects at random a value $e(<R)$ such that the greatest common divisor of e and $\phi(R)$ (denoted here by $(e, \phi(R))$) is 1. The congruence

$$de \equiv 1 (mod\ \phi(R)),$$

is then solved for d such that $0 < d < R$. There is a simple procedure for doing this, based on the Euclidean algorithm, which requires $O(logR)$ operations. For this scheme the public encryption key is $K_1 = \{e,R\}$ and the secret decryption key is $K_2 = d$.

If some individual wishes to send a secure message M (such that gcd(M, R) = 1 and $M < R$) to the designer of this system,

$$C = E_{K_1}(M) \equiv M^e (mod\ R),$$

is sent where 0 < C < R. The designer calculates

$$D(C) \equiv C^d \equiv M^{ed} \equiv M^{1+k\phi(R)} \equiv (mod\ R).$$

Since M < R, it can now be uniquely determined. It might appear that the problem of calculating $M^e (mod\ R)$ for large e is very time consuming. In fact, there is a very simple and fast method for doing this which takes O $(log_2 e)$ steps to complete. Briefly, it is done by performing a sequence of squaring and multiplication by M operations as indicated by the binary representation of e (see [SEPI]).

Consider the following simple example. Here we put p = 11, q = 19, R = 209, e = 17. We find that $\phi(R)$=10.18 and

$$17d \equiv 1(mod\ 180)$$

for d = 53. If M = 5, then

$$C \equiv 5^{17} \equiv ((5^{2)^{2)^2}}.5 \equiv 80(mod\ 209) \text{ and } C = 80\ .$$

To decrypt C, we calculate

$$80^{53} \equiv ((((80^2.80)^{2)^2}.80)^{2)^2}.80 \equiv 5(mod\ 209).$$

The security of this scheme depends very much on the difficulty of factoring R. If a cryptanalyst can factor R, d can be readily calculated and thus all ciphertext can be decrypted. There are a large number of different factoring methods currently known (see Guy [GUY]), but the most powerful of these techniques (Dixon [DIXO] and Schroeppel [SCHR]) still require about

$$e^{\sqrt{\log N \log\log N}}$$

operations to factor N. Thus, a very fast computer (one multiprecise operation per 10^{-6} seconds) might require 3.8×10^9 years to factor a 200 digit number. It must be stressed here that many numbers which are very large can be factored relatively easily when their prime factors have certain special forms. As an extreme example of this, we mention that it is known that the truly immense number

$$2^{2^{4724}}+1$$

has $29.2^{4727}+1$ as a factor. Hence, great care must be taken by the designer of this type of cryptosystem when selecting primes P and Q. This problem has been discussed in Rivest [RIVE] and Williams and Schmid [WISC]. Although it is true that anyone who can factor R can decrypt messages sent under this scheme, it is not know whether the act of decrypting these messages is equivalent in difficulty to factoring R. Simmons and Norris [SINO] and Herlestam [HERL] have attacked this cryptosystem by using the fact that if some P can be found such that

$$C^P = (E_{K_1}(M))^P \equiv 1(mod\ R),$$

then M can be found without having to factor R. But in [WISC] it is shown that

the chance of finding such a value of P by a random search is very very small for

large, properly selected values of P and Q;

a fast method of finding P can almost certainly be converted into a fast method of factoring R.

Thus, it seems that, if the system designer has selected the value of R carefully, revealing its value and that of e gives too little information to a cryptanalyst in order for d to be deduced. For this reason it is felt that the RSA scheme is a valid public key cryptosystem.

In summary, we point out that this system has several important and desirable properties.

It seems to be very secure (so far).

The key size is small.

It is also a signature scheme.

Unfortunately, it also possesses some disadvantages.

The processes of encryption and decryption are expensive. Approximately one second is needed per 6000 bits of information on a special purpose piece of hardware constructed at M.I.T. (L. Adleman, personal communication).

Determination of suitable keys is somewhat expensive.

Reblocking of the message or two different R values are required when the system is used for signatures.

We conclude this section by mentioning that some other cryptosystems have been developed which also use the difficulty of factoring as their means of providing security. One of these, the Lu-Lee (COMSAT) system [LULE], has been broken by Adleman and Rivest [ADRI] and others. Rabin [RABI], Khoo, Bird and Seberry [KHBI], and Williams [WILL] have presented public key cryptosystems for which it can be shown that decryption is equivalent in difficulty to factoring. In view of this it would seem that these systems are superior to the RSA system; however, because of the constructive nature of the proofs of their security, all of these schemes may be susceptible to a selected ciphertext attack [WILL]. This difficulty can be overcome by setting up the system very carefully, but the resulting schemes are somewhat cumbersome. It would be very desirable to have a nonconstructive proof of the equivalence of the problem of breaking the RSA system and the problem of factoring; at the moment, this seems very far from being achieved.

TRAPDOOR KNAPSACKS

Another public key system is called the "trapdoor knapsack system," a name imaginatively derived from the notion of attempting to choose just the right set of rods from those in a box so that when packed into a long thing knapsack, the rods would fit tightly and not rattle.

Trapdoor knapsacks have their roots in a field called combinatorial mathematics and depend on the fact that given a list of numbers it is easy to add up any specified subset, but given instead a list of numbers and a sum it is extremely difficult to discover a subset

which totals to exactly that sum. In order to do encryption in this system, the input block is treated as a specification of which numbers are to be selected from a list and added up; the output is their sum. The trapdoor knapsack system is based on Merkle's discovery that if the list of numbers is constructed correctly, certain details of that construction constitute a secret key which allows the constructor to take the sum and discover which members of the list were added.

Given a vector of integers $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ and an integer S, the knapsack problem is to find a subset of the α_i such that the sum of the elements of the subset is equal to S. Equivalently, given α and S find a binary n-vector $\mathbf{x}$ such that $\alpha \cdot \mathbf{x} = S$.

The knapsack problem is believed to be extremely difficult in general, belonging to a class of problems (the **NP** complete problems) that are thought not to be solvable in polynomial time on any deterministic computer.

Some cases of the knapsack problem are quite simple, however, and Merkle and Hellman's technique is to start with a simple one and convert it into a more complex form.

The vector α can be used to encipher a block by forming the dot product $S = \alpha \cdot \mathbf{x}$. Recovery of $\mathbf{x}$ from S involves solving a knapsack problem and was thus believed to be computationally infeasible if α and $\mathbf{x}$ are chosen randomly.

If the vector α is selected so that each element is larger than the sum of the preceding elements and each of the x_i is either 0 or 1, its knapsack problem is very simple. For example, if

$$\alpha^* = (171,197,459,1191,2410) \text{ and}$$

$$\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$$

$$S^* = 3798$$

then x_5 must equal 1. If it were 0 then even if $x_1 = x_2 = x_3 = x_4 = 1$, the dot product $\alpha \cdot \mathbf{x}$ would be too small. Then, knowing that $x_5 = 1$,

$$S^* - \alpha_5^* = 3798 - 2410 = 1388.$$

must be a sum of a subset of the first four elements of $\mathbf{x}$. Because $1388 > \alpha_4^* = 1191$, $\mathbf{x}$ must equal 1. Finally

$$S^* = \alpha_5^* - \alpha_4^* = 196 = \alpha_2^*$$

so $x_3 = 0$, $x_2 = 1$, and $x_1 = 0$.

This simple knapsack vector α^* cannot be used as a public enciphering key because anyone can easily recover $\mathbf{x}$ from S. The algorithim for generating public keys therefore generates a random simple knapsack vector α^* (with a hundred or more components) and keeps α^* secret. It also generates a random number m which is larger than $\Sigma\alpha^*$ and a random pair w, w^{-1} such that $ww^{-1} = 1 \ mod \ m$. It then generates the public knapsack vector or enciphering key β by multiplying each components of α^* by w mod m.

$$\beta = w \cdot \alpha^* \bmod m .$$

When another user wishes to send the message **x** to *A* he computes

$$S = \alpha^* \cdot \mathbf{x}$$

and sends this to *A*. *A* uses his secret information, w^{-1} and m to compute

$$\begin{aligned} S^* &= wS \bmod m \\ &= w^{-1}\Sigma\alpha_i x_i \bmod m \\ &= w^{-1}\Sigma(w\alpha_i^* \bmod m)x_i \bmod m \\ &= \Sigma(w^{-1}w\alpha_i^* \bmod m)x_i \bmod m \\ &= \alpha^* \cdot \mathbf{x} \end{aligned}$$

because $m > \Sigma\alpha^*$. For example, if the secret vector α^* is as above, then $w = 2550$ and $M = 8443$, results in the public vector,

$$\alpha = (5457,4213,5316,6013,7439),$$

which hides the structure present in α^*.

The vector α^* is published by the user as a public key, while the parameters w^{-1} and m are kept secret as the private key. They can be used to decipher any message which has been enciphered with the user's public key, by computing

$$S^* = w^{-1}S \bmod m$$

and then solving the simple knapsack

$$S^* = \alpha^* \cdot \mathbf{x}.$$

This process can be iterated to produce a sequence of vectors with seemingly more difficult knapsack problems by using transformations (w_1, m_1), (w_2, m_2), etc. The overall transformation is not, in general, equivalent to any single (w,m) transformation. The trapdoor knapsack system requires special adaptation when used to produce signatures [SEPI]. Unlike the *RSA* system, knapsack systems are quite fast and speeds of one megabit appear easy to obtain. Unfortunately, the public keys are quite large, requiring approximately ten-thousand bits.

MERKLE-HELLMAN CRYPTOSYSTEM

The first asymmetric cryptosystem based on the Knapsack Problem was invented by Merkle and Hellman [MEHE78]. The Merkle-Hellman cryptosystem (*MH* system) allows n-bit messsages to be enciphered,

$$M=(m_1,m_2,...,m_n), \tag{1}$$

where $m_i \in \{0,1\}$ for $i = 1,...,n$, using the public key K

$$K=(k_1,k_2,\cdots,k_n). \tag{2}$$

where $k_i \in \mathbf{Z}_q = \{1,2 \ldots , q\text{-}1\}$ for $i = 1,...,n$, and the integer q is prime. Using the pair

(M, K), the sender A creates the cryptogram C according to the following formula :

$$C = \sum_{i=1}^{n} m_i k_i, \tag{3}$$

The enciphering process is extremely simple for it runs in linear time. To continue the description, we now consider the receiver B who always initiates the algorithm (the system). The receiver first chooses the initial condition which is a sequence of superincreasing integers

$$W = (w_1, w_2, \cdots, w_n), \tag{4}$$

where $w_i : i = 1,...,n,$ are an integers which satisfy an inequality in the form :

$$w_i > \sum_{j=1}^{i-1} w_j. \tag{5}$$

Note that the initial condition W defines the instance of the Easy Knapsack Problem which is solvable in linear time (see [SEPI]). Now the designer B transforms the instance of the Knapsack Problem. To do this, B first determines a suitable field $blodZ_q$ (q must be prime and a multiplier $r \in \mathbf{Z}_q$. Both q and r are usually chosen at random provided that

$$q > \sum_{i=1}^{n} w_i.$$

Next, he/she injects the vector W into the field $\mathbf{Z}_q$ according to the following congruence:

$$k_i = w_i r (mod\ q)\ ;\ i = 1, \cdots, n. \tag{6}$$

The vector $K=(k_1,...,k_n)$is sent to the sender A via an insecure channel while the triple (the initial condition W, the integer r, the modulus q) is kept secret by the receiver.

Assume now that the receiver A has obtained the cryptogram generated using (3). First of all, B transforms the cryptogram as follows :

$$C^* = Cr^{-1} (mod\ q). \tag{7}$$

From (3) and (5), we have

$$C^* = Cr^{-1} = \sum_{i=1}^{n} k_i m_i r^{-1} = \sum_{i=1}^{n} w_i m_i (mod\ q). \tag{8}$$

As w_i fulfills (5), the receiver easily finds components m_i of the message M.

Example. To illustrate the Merkle-Hellman system, assume that 5 bit messages are to be transmitted. The receiver initiates the algorithm by choosing the vector

$$W = (w_1, w_2, w_3, w_4, w_5) = (2,3,6,12,25).$$

Note that

$$w_2 > w_1,$$
$$w_3 > w_1 + w_2,$$

$$w_4 > w_1 + w_2 w + 3 \text{ , and}$$
$$w_5 > w_1 + w_2 + w_3 + w_4.$$

Next he/she chooses the pair (r,q) at random provided that q is prime and $q > w_1 + w_2 + w_3 + w_4 = 48$. Let $q = 53$ and $r = 46$. It is easy to check that $r^{-1} = 15 (\text{mod } 53)$. Subsequently, the receiver calculates the public key using (3.42), namely

$$k_1 = w_1 r (\text{mod } q) = 39 \text{ (mod } 53),$$
$$k_2 = w_2 r (\text{mod } q) = 32 \text{ (mod } 53),$$
$$k_3 = w_3 r (\text{mod } q) = 11 \text{ (mod } 53),$$
$$k_4 = w_4 r (\text{mod } q) = 23 \text{ (mod } 53),$$
$$k_5 = w_5 r (\text{mod } q) = 37 \text{ (mod } 53).$$

So, the public key

$$K = (k_1, k_2, k_3, k_4, k_{5)} = =(39,32,11,22,37)$$

is sent to the sender. Suppose now that the receiver has obtained the cryptogram C=119. To decrypt it, he/she first transforms it as follows:

$$C^* = Cr^{-1} = 119 \cdot 15 = 36 (\text{mod } 53),$$

and next solves the simple knapsack problem:

$$\text{as } C^* = 36 > w_5 = 25 \text{ we get } m_5 = 1,$$
$$\text{as } C^* - w_5 = 11 < w_4 \text{ we get } m_4 = 0,$$
$$\text{as } C^* - w_5 = 11 > w_3 = 6 \text{ we get } m_3 = 1,$$
$$\text{as } C^* - w_5 - w_3 = 5 > w_2 = 3 \text{ we get } m_2 = 1,$$

In other words the receiver has recreated the message M=(1,1,1,0,1).

Shamir and Zippel [SHA80b] showed that, if the modulus q is compromised, the multiplier r can be readily calculated from the public key K. The obvious remedy is multiple applications of the initial condition in several different fields, i.e., W is injected into $Z_{q_1}, Z_{q_2}, \cdots$ using multipliers $r_1, r_2, \cdots$, and primes q_i satisfying the inequality $q_1 < q_2 <$.... The resulting cryptosystem is called the multiply iterated knapsack system.

GRAHAM-SHAMIR CRYPTOSYSTEM

Graham and Shamir [LEMP79] independently discovered a variant of the Merkle-Hellman system to obscure the superincreasing property of initial conditions. A Graham Shamir initial condition vector $W=(w_1, \cdots, w_n)$ has the property that each w_j has the following binary representation:

$$W_j = R_j I_j S_j,$$

where R_j and S_j are long random bit strings, and I is a bit string of length n such that j–th high-order bit is 1 and the remaining $n-1$ bits are 0's. Each random bit string S_j has $\log_2 n$ zero's in its high-order bit positions so that summing them does not cause them to overflow into the area of the I_j's. Thus, $D = W \times M$ has the binary representation

$$D=(R,M,S),$$

where $R = \sum_{j=1}^{n} R_j m_j$ and $S = \sum_{j=1}^{n} S_j m_j$. Now the vector of bit strings $((I_n, S_n), \cdots, (I_1, S_1))$ is a easy knapsack vector. The R_j's are added to obscure the superincreasing property. These knapsacks are even easier to solve than Merkle-Hellman trapdoor knapsacks because M can be extracted from the binary representation of D.

Example. Let $n=5$ when W is given by

$$\begin{aligned}
(R_1\ I_1\ S_{1)} &= (011010\ 10000\ 000101) = w_1\\
(R_2\ I_2\ S_{2)} &= (001001\ 01000\ 000011) = w_2\\
(R_3\ I_3\ S_{3)} &= (010010\ 00100\ 000100) = w_3\\
(R_4\ I_4\ S_{4)} &= (011000\ 00010\ 000111) = w_4\\
(R_5\ I_5\ S_{5)} &= (000110\ 00001\ 000001) = w_5.
\end{aligned}$$

Let the message be $M=(0,1,0,0,1)$. Then

$$\begin{aligned}
D &= W \times M = w_2 + w_5\\
&= (R_2 + R_5;\ I_2 + I_5;\ S_2 + S_{5)} = (001111\ 01001\ 000100).
\end{aligned}$$

The initial condition W is converted to a hard knapsack vector K as in Merkle-Hellman scheme, by picking q and r and computing $K = rW\ (mod\ q)$. Similarly, a message M is enciphered as in the Merkle-Hellman system, whence $C = \sum_{i=1} k_i m_i$. At the receiver's end, C is deciphered by computing $D=Cr^{-1}(mod\ q)$ and extracting from D the bits representing M.

Shamir and Graham believed this variant was safer, faster and simpler to implement than original scheme proposed by Merkle and Hellman. It has been broken by A. Shamir.

SECURITY OF THE MERKLE-HELLMAN SYSTEM

Merkle and Hellman originally suggested using knapsacks of approximate size n=100. However, Schroeppel and Shamir [SCHR79] developed an algorithm to solve knapsacks of this size. By trading time and space their method can solve the knapsack problem in time $T = O(2^{n/2})$ and space $O(2^{n/4})$. For $n = 100$, $T = 2^{50} \approx 10^{15}$. Thus a single processor can find a solution in 11,574 days. But for n=200, assuming 8.64×10^{10} instructions per day, the algorithm is computationally infeasible.

The Merkle-Hellman system has two drawbacks which arise from its construction. They are the high cryptogram redundancy and the huge public key length. For n=200, every key component is 400-bit sequence, so the public key

Once Merkle and Hellman had announced their cryptosystem, many scientists tried to break it. Merkle promised a prize of 1000 dollars to the first person to successfully crack his system. There are basically two attacks on the Merkle-Hellman system. The first relies upon finding an efficient algorithm to solve knapsacks defined in the form required by the system. While there is no efficient algorithm to solve the general knapsack as it belongs to the class NPC, the knapsacks in question are only a small subset of all knapsacks.

The second attack is based on the knowledge of the public key only. The public key creates the hard knapsack. A number of articles addressed the following question: is there an polynomial time algorithm to calculate easy knapsacks (initial conditions) knowing the hard ones (public keys). Subsequently an algorithm was invented by Shamir ([SHAM82]. Shamir's algorithm works for the basic Merkle-Hellman system only and not for all hard knapsacks vectors. At the same time Adleman [ADLE82, ADLE83] examined the iterated Merkle-Hellman system and showed that even this system is insecure. Some comments on his attack can be found in [BRI83c]. Next, Brickell [BRI83b] and Lagarias et al [LAG83b] proved that any cryptosystem based on low density knapsacks (the Merkle-Hellman system is one such knapsack) is breakable in polynomial time. Subsequently, Lagarias [LAG82, LAG83a] examined applications of simultaneous diophantine approximation problems (see [CASS65]) to design a polynomial time algorithm for breaking knapsack cryptosystems.

The Merkle-Hellman system was finally shown to be insecure by Brickell (BRI84a, BRI85] who invented a polynomial time algorithm which allowed the easy knapsack vectors to be recreated from the hard knapsack vectors. Brickell's algorithm was based upon the recently published algorithm for factoring polynomials with rational coefficients, due to Lenstra, Lenstra, and Lovasz [LEN82, LEN83]. Needless to say that Brickell won Merkle's prize of 1000 dollars. Readers interested in details of breaking the Merkle-Hellman system are referred to two papers of Brickell et al [BRI83a, BRI83d] or the book by OConnor and Seberry [OCSE87].

CRYPTOSYSTEM BASED ON IDEMPOTENT ELEMENTS (PIEPRZYK)

There are many modifications of the Merkle-Hellman system. We consider one of them described in [PIEP85]. In this modification called *IE* system, the simple knapsack is defined differently using idempotent elements. As before the system encrypts n-bit messages. The initial condition, however, consists of n different primes $p_1, \cdots, p_n$. If we accept that $N = p_1, \ldots p_n$, then the set $\mathbf{Z}_N = \{1, \ldots, N\text{-}1\}$, along with addition and multiplication modulo N defines a suitable arithmetic. The Chinese Remainder Theorem says that any integer $a \in \mathbf{Z}_N$ can be uniquely presented in the form of the vector:

$$[a_1, \cdots, a_n] = [a \,(mod\, p_1), \cdots, a \,(mod\, p_n)].$$

Now there are n elementary idempotent elements of the form:

$$\begin{aligned} e_1 &= [1(mod\, p_1),\ 0(mod\, p_2),\ldots,\ 0(mod\, p_n)] = [1,0,\ldots,0] \\ e_2 &= [0(mod\, p_1),\ 1(mod\, p_2),\ldots,\ 0(mod\, p_n)] = [0,1,\ldots,0] \qquad (9) \\ &\vdots \\ e_n &= [0(mod\, p_1),\ 0(mod\, p_2),\ldots,\ 1(mod\, p_n 0] = [0,0,\ldots,1]. \end{aligned}$$

The elements create an algebraic space and they can be used as basis vectors. To hide the vectors and simultaneously create the public key. The idempotent elements are transformed using the random integer r and the modulus q as before, namely

$$k_i = re_i (mod\, q); \quad i = 1,\ldots,n, \qquad (10)$$

while $q > \sum_{i=1}^{n} e_i$ and q is prime. Using the public key $K=(k_1,...,k_n)$, the sender creates the cryptogram C for the message $M=(m_1, \cdots, m_n)$ according to the following formula:

$$C = | \sum_{i=1}^{u} k_{j_i} m_{j_i} - \sum_{i=u+1}^{n} k_{j_i} m_{j_i} |; \tag{11}$$

where both subsets of binary elementary messages $M^+ = m_{j_1}, \cdots, m_{j_u}$ and $M^- = m_{j_{u+1}}, \cdots, m_{j_n}$ are selected arbitrarily provided that $M^+ \cup M^- = 0$ contains all the binary elements m_i, $i = 1,...,n$, and $M^+ \cap M^- = \varnothing$.

In turn, the receiver, having the cryptogram C, transforms it using the inverse r^{-1} as follows:

$$C^* = Cr^{-}1(mod\ q). \tag{12}$$

Again, he/she calculates $C'' = N - C^*$. As only one element of the pair (C^*, C") conveys the message, the receiver presents them as vectors of the form:

$$C^* = [C^* mod\ p_1, \cdots, C^* mod\ p_n]$$

$$C'' = [\ C'' mod\ p_1, \cdots, C'' mod\ p_n] \tag{13}$$

and selects the vector all of whose components are either $-1(mod\ p_i)$ or $1(mod\ p_i)$ for i=1, ... ,n. Now if the cryptogram is generated by (11) and there is a vector, say C'', all of whose components are from the set {0,1,-1}, then the second one, C", must contain at least one component (C" mod p_j) different from 0,1,-1 (mod p_j). To illustrate the enciphering and deciphering processes in *IE* system, consider the following example.

Example. Assume that the communicating parties have agreed to transmit 5-bit messages and the receiver has already selected the initial condition $(p_1,p_2,p_3,p_4,p_5)=(2,3,5,711)$. Now $N=p_1 \cdot p_2 \cdot p_3 \cdot p_4 \cdot p_5 = 2310$ and

$$e_1 = [1\ mod\ p_1,\ 0\ mod\ p_2,\ 0\ mod\ p_3,\ 0\ mod\ p_4,\ 0\ mod\ p_5] = 1155(mod\ 2310),$$

$$e_2 = [0\ mod\ p_1,\ 1\ mod\ p_2,\ 0\ mod\ p_3,\ 0\ mod\ p_4,\ 0\ mod\ p_5] = 1545(mod\ 2310),$$

$$e_3 = [0\ mod\ p_1,\ 0\ mod\ p_2,\ 1\ mod\ p_3,\ 0\ mod\ p_4,\ 0\ mod\ p_5] = 1389(mod\ 2310),$$

$$e_4 = [0\ mod\ p_1,\ 0\ mod\ p_2,\ 0\ mod\ p_3,\ 1\ mod\ p_4,\ 0\ mod\ p_5] = 330(mod\ 2310),$$

$$e_5 = [0\ mod\ p_1,\ 0\ mod\ p_2,\ 0\ mod\ p_3,\ 0\ mod\ p_4,\ 1\ mod\ p_5] = 210(mod\ 2310).$$

If the receiver now selects the modulus $q = 4637 > \sum_{i=1}^{5} e_i = 4621$ (q is prime) and picks r = 3475 at random ($r^{-1} = 3372$), then components of the public key are:

$$k_1 = e_1 r(mod\ q) = 1155.3475 = 2620(mod\ 4637),$$
$$k_2 = e_2 r(mod\ q) = 1540.3475 = 402(mod\ 4637),$$
$$k_3 = e_3 r(mod\ q) = 1386.3475 = 3144(mod\ 4637),$$

$$k_4 = e_4 r (mod\ q) = 330.3475 = 1411 (mod\ 4637),$$
$$k_5 = e_5 r (mod\ q) = 210.3475 = 1741 (mod\ 4637).$$

In other words the public key is

$$K=(2620,402,3144,1411,1741).$$

Here, both the initial condition and the pair (r, q) are kept secret. Using K, the sender can encipher his/her message M=(1,0,1,1,1). First, he/she constructs two subsets M^+ and M^-. Let them be M^+={ m_1, m_2, m_4 } and M^- = { m_3, m_5 }. Next he/she computes the cryptogram

$$C = |(k_1 + k_4) - (k_3 + k_5)| = |4031 - 4885| = 854.$$

Finally, the cryptogram is forwarded to the receiver. In turn, the receiver, knowing the inverse element r^{-1}, transforms the cryptogram

$$C^* = Cr^{-1} (mod\ q) = 854.3372 = 111 (mod\ 4637).$$

Now, one element of the pair (C^* = 111, C" = N - C' = 1199) conveys the message. To find this element, the receiver converts the pair into vectors

$$C^* = [111(mod\ 2),\ 111(mod\ 3),\ 111(mod\ 5,)\ 111(mod\ 7),\ 111(mod\ 11)] = [1,0,1,6,1] = [1,0,1,-1,1],$$

$$C" = [1199(mod\ 2),\ 1199(mod\ 3),\ 1199(mod\ 5),\ 1199(mod\ 7),\ 1199(mod\ 11) = [1,2,4,2,0].$$

The first vector indicates the message m = [1,0,1,1,1].

OTHER PROPOSALS FOR PUBLIC KEY CRYPTOSYSTEMS.

Some other cryptosystems have also been developed which are based on the knapsack problem but we cannot develop them futher here.

Leung and Vacon [LEVA] have presented a cryptosystem, designed around the knapsack problem, which seemed to be very secure. Unfortunately, when this system was being used, it was necessary to transmit 100 times more ciphertext than corresponding message text. We also mention that Shamir [SHAM] has developed a signature scheme around the knapsack problem. This scheme, while requiring a large key, is still very simple and fast; however, it has been broken as a public syatem and was not designed to be used as a cryptosystem.

McELIECE'S ALGEBRAIC CODES CRYPTOSYSTEM

McEliece suggested in 1978 [MCEL78] that error correcting codes are excellent candidates for providing public-key cryptosystems. His work has not received the prominence, or detailed study it deserves, because error correcting codes are effective by virtue of their redundancy, which leads to data expansion, which has not usually been considered desirable in cryptography.

The author believes that when security is required on noisy channels, such as satellite communications, mobile radios or car telephones, error correction incorporated with security is the wisest course to take. It may be that encryption should be applied first and then error correction via, say convolutional codes is most appropriate. Nevertheless, the combined area of encryption and error correction, is valuable to study for both digital and analogue systems.

McEliece based his cryptosystem on the **Goppa codes**, a superset of the **BCH** or the **Hamming** polynomial codes, because they are easy to implement in hardware and a fast decoding algorithm exists for the general Goppa cods while no such fast decoding algorithm exists for a general linear code.

Corresponding to each irreducible polynomial,

$$p(x) = x^t + p^{t-1}x^{t-1} + \cdots + p_1 x + 1, \quad p_i \in \{0,1\}.$$

of degree t over GF(2^m), there exists a binary irreducible Goppa code of length $n=2^m$, dimension $k > n-tm$, capable of correcting any pattern of t or fewer errors.

Patterson has given a fast algorithm, with running time, $O(nt)$, for decoding these codes (see[MCEL77, Problem 8.18]).

The cryptosystem designer now chooses a desirable value of n and t and then randomly picks an irreducible polynomial of degree t over GF(2^m). The probability that a randomly selected polynomial of degree t is irreducible is about $1/t$ and Berlekamp [BERL68, Chapter 8] describes a fast algorithm for testing irreducibility so this is a reasonable step. Next the system designer produces a $k \times n$ generator matrix **G** for the code, which could be in canonical form, that is

$$\mathbf{G} = [I_k \quad F_{k \times (n-k)}].$$

The usual error correction method would now multiply a message vector $\mathbf{a} = (a_1, a_2, \ldots, a_k)$ onto **G** to form the codeword $\mathbf{b} = (b_1, b_2, \cdots b_n)$ which is transmitted via a channel which usually corrupts the codeword to **b'** which must be then corrected and then the message recovered.

If a were multiplied onto **G** in the canonical form b would be

$$\mathbf{b} = (a_1, a_2, \cdots, a_k, f_1(a_i), \cdots, f_{n-k}(a_i))$$

and if there was no corruption the message is trivially recovered as the first k bits of **b**.

Thus McEliece "scrambles" **G** by selecting a random dense $k \times k$ nonsingular matrix S, and a random $n \times n$ permutation matrix P. He then computes

$$\mathbf{G'} = SGP'$$

which generates a linear code with the same rate and minimum distance as the code generated by **G**. **G'** is called the public generator matrix.

Sloane [SLOA79] has written an excellent article describing how the random matrices S and P can be obtained.

Thus the algorithm can be described as

> Encryption : Divide the data to be encrypted into k-bit blocks. If **u** is such a block, transmit $\mathbf{x} = \mathbf{uG'} + \mathbf{z}$ where **G'** is the public generator matrix and **z** is a locally generated random vector of length n and weight t.
>
> Decryption: On receipt of **x** the receiver computes $\mathbf{x'} \times P^{-1}$ where P^{-1} is the inverse of the permutation matrix P. **x'** will then be a codeword of the Goppa code previously chosen.

The decoding algorithm is then used to find $\mathbf{u} = \mathbf{u'}S^{-1}$

SECURITY OF McELEICE'S CRYPTOSYSTEM.

The encryption and description algorithms can be implemented quite simply. We need to determine the security of the system. If an opponent knows **G'** and intercepts **x** can he/she recover **u**. There are two possible attacks:

> to try to recover **G** from **G'** and so be able to use the decoding algorithm;
> to attempt to recover **u** from **x** without knowing **G**.

The first attack appears hopeless if n and t are large enough because there are so many possibilities for **G**, not to mention the possibilities for S and P.

The second attack seems more promising but the basic problem to be solved is that of decoding a more or less arbitrary (n, k) linear code in the presence of up to t errors.

Berlekamp, McEliece and van Tilborg [BEMT78] have proved that the general coding problem for linear codes if **NP-complete**, so one can certainly expect that if the code parameters are large enough, that this attack too will be infeasible.

Example. If n=1024=2 sup 10 and t=50 there are about 10^{149} possible Goppa polynomials and vast number of choices for S an P. The dimension of the code will be about 524. Hence, a brute-force approach to decoding based on comparing **x** to each codeword has a work factor of about $s^{524}=10^{158}$; and a brute-force approach based on coset leaders has a work factor of about $2^{500}=10^{151}$.

A more promising attack is to select k of the coordinates randomly and hope none are in error and then calculate **u**. The probability of no error, however, is about

$$(1-t/n)^k,$$

and the amount of work involved in solving k simultaneous equations in k unknowns is about k^3. Hence before finding **u** using this attack one expects a work factor of

$$k^3(1-t/n)^{-k}.$$

For n=1024,k=524,t=50 this is about $10^{19}\approx 2^{65}$.

Remark. This algorithm would have a communication rate of about 10^6 bits/sec so would have quite viable implication speeds. On the other hand this cryptosystem is not suitable for producing "signatures" as the algorithm is truly asymmetric and not one to one. Other authors e.g. Kak [KAK83], have discussed joint encryption and error-correction coding suggested further avenues for research.

In this section and the last we have seen that many possible public key cryptosystems have been developed. Even though all of them have certain disadvantages, if not secure for secret public key use, several of them may, with limited usage, be used for the very important task of exchanging the secret keys needed by certified conventional cryptosystems. The exchanging of these keys, a very important proble which has not, in general been satisfactority solved, need not take place very often but must be done in an environment of extreme security.

CONCLUSION

In spite of the number of proposed public key cryptosystems, it must be stressed here that we are still a long way from demonstrating that any of them is provably secure. Since the problem of factoring is so old, it might be felt that a scheme which is as difficult to break as it is difficult to factor a certain large number is, in a sense, certified in its security. However, it could be that in the future someone might develop a method that can be used

to factor numbers of a certain (but now unknown) type. In a directory of public keys for cryptosystems based on the factoring problem there could be several schemes whose security would be compromised by this discovery. At the moment we simply have no way of knowing whether this could occur.

Also, it should not be forgotten that there is the possibility that no such thing as a provably secure public key cryptosystem exists. This would certainly seem to be the case if it were ever proved that $\mathbf{P} = \cap\mathbf{Co\text{–}NP}$ [SEPI].

The simple elegance and beauty of several of these recent public key encryption schemes should not be allowed to lull us into a feeling of complacency.

REFERENCES

I apologize to all those readers and authors who should have been referenced here but due to the quirks of our computer could not be produced, although online, before this paper went to press. All papers are properly referenced in

[SEPI]
Jennifer Seberry and Joseph Pieprzyk, *Cryptography: An Introduction to Computer Security*, Prentice-Hall, Sydney, New-York, 1987.