Don Coppersmith
IBM Research 32-256
P.O. Box 218
Yorktown Heights, N.Y. 10598
USA
Telephone (914) 945-2288
August 27, 1984

Cryptography

Seventeenth IBM/University of Newcastle upon Tyne International Seminar on the Teaching of Computing Science at University Level.

DESIGN OF CRYPTOGRAPHIC SYSTEMS

Historical systems:

The Enigma, a commercial enciphering machine manufactured in the 1920's, was adapted for use by the Germans in World War II. It contained three rotors, each of which had electrical leads going from 26 input positions to 26 output positions. These rotors would turn as each letter was enciphered, like an automobile odometer. At the back was a reflector, which paired up the 26 positions in an arbitrary manner. A cleartext letter would activate the input lead on the first rotor, thus making a path through all three rotors to the reflector, and taking a different path out, activating an output (ciphertext) letter. The initial positions of the rotors were secret, and changed each day. These initial positions formed the key. When the rest of the system was known to the cryptanalysts (by having obtained copies of the machine), only the key was secret. The task became that of deducing the key.

One weakness in the scheme was that at each position, the transformation from input to output character was an involution; i.e. if a cleartext "A" became a ciphertext "W", then at the same position a cleartext "W" became a ciphertext "A". Cryptanalysts were able to exploit this by group-theoretic techniques.

Another weakness was in the protocol for establishing session keys: under a master key, the session key encrypted and sent out twice in succession. Analysts were able to make use of coincidences in the double encryptions of session keys to make deductions about master keys in use.

Hodges' biography of Alan Turing gives a delightful account of the cracking of the Enigma.

The one-time pad is a truly unbreakable cipher. A long string of random data ("key") is prepared, and two copies are distributed, one to the sender, one to the receiver. The sender adds, character by character, his cleartext to the key, and transmits the resulting ciphertext. The receiver subtracts the same key from the ciphertext, character by character, and recovers the cleartext. If the key is truly random, the eavesdropper has absolutely no way to distinguish among the several possible messages; all information has been destroyed, from his point of view.

The chief disadvantage of this scheme is the key length. The key is as long as the enciphered message, and must be transmitted over secure channels. (This transmittal can be done beforehand, at the leisure of the communicants.) This is totally impractical for computer communication, due to the sheer bulk of data being communicated. It is

primarily useful for highly sensitive data, such as diplomatic communications and some intelligence data.

A good many schemes are fashioned after the one-time pad, but they are not one-time pads, and they do not enjoy the total security of this scheme. Generally a shorter key (a few hundred bits) is expanded to a long random-looking string, via some pseudo-random number generator. This string is then added to the cleartext. (An example is the multiple-loop Vernam cipher.) The expansion can be done at both sender and receiver, so that only the shorter key need be exchanged. The task of cryptanalysis is now that of analyzing the pseudo-random number generator.

These schemes enjoy the possibility of precomputation: the sender or receiver can generate the longer string at his leisure, dependent on storage, and then encryption/decryption is a simple matter of addition. A disadvantage is that a false sense of total security is imparted by the analogy to the one-time pad.

Modern systems.

An important system in use today is the DES or Data Encryption Standard, developed by IBM in the early 1970's. At heart is a block cipher, using a 56-bit key to transform a 64-bit cleartext into a 64-bit ciphertext. The idea of chaining (see below) is an important addition, which lends the DES strength against "brute-force" or "dictionary" attacks.

The workings of the DES will be sketched; further detail can be found in Meyer & Matyas, or Konheim. During each of 16 "rounds", the message is split into 32-bit halves. One computes a nonlinear function of the left half and the key, and adds it (mod 2) to the right half; then the left and right halves are reversed. The result of one round is that the new right half is the old left half, unscathed; the new left half is the sum of the old right half with this nonlinear function. A consequence of this scheme is that decoding is like encoding, except that the "rounds" are taken in reverse order. One has the "old left half" and the key available, so one can recompute the nonlinear function, subtract it from the "new left half" to obtain the "old right half"; thus one works backwards through each round.

The particular nonlinear function in use: each of eight "S-boxes" takes as input six bits of message, exclusive-ORed with six bits of key, performs a prescribed nonlinear function taking six bits to four bits, and sends those four bits to prescribed locations in a 32-bit word. From round to round, and from S-box to S-box, the locations of the key bits are changed in a known manner.

To all intents, the DES appears to be a random function taking 56+64 bits (key plus cleartext) into 64 bits (ciphertext), with the proviso that under the same key, two different cleartexts map to two different ciphertexts: $E_K(X) \neq E_K(Y)$ if $X \neq Y$. For a fixed key, it appears to be a random permutation on the space of 64-bit messages.

The DES is particularly well suited to hardware implementation. Software implementations are much faster than, say, the public key systems, but not as fast as hardware.

The DES is in used today in electronic funds transfer, bulk communication between main-frame computers in networks, and protection of data resident on main-frame computers.

DES is enhanced by the use of chaining. As a block cipher, the same key and the same cleartext will always produce the same ciphertext; this would allow "dictionary" attacks such as the Hellman attack (outlined below). To counteract this, one uses "chaining". The ciphertext is preceded by a random string of 64 bits (eight bytes) Y_0. Then instead of enciphering the cleartext as is, one adds to each block X_i of 64 bits of cleartext, the preceding block Y_{i-1} of 64 bits of ciphertext, and then enciphers. Thus

$$Y_i = E_K(X_i+Y_{i-1}).$$

The effect is that, although a particular cleartext string ("Dear Sir") may occur several times enciphered with the same key, the ciphertext will be different on each occasion.

PUBLIC-KEY ALGORITHMS

The concept of the public-key system is a relatively new idea; see Diffie and Hellman, "New Directions in Cryptography," IEEE Transactions on Information Theory, IT-22, pp.644-654, Nov. 1976. One objective is to avoid the logistical problem of distributing keys. In some schemes, the receiver concocts two keys, a secret deciphering key and a public enciphering key. He sends the public key to any senders. The sender

enciphers data and forwards it to the receiver. Only the receiver, in possession of the secret deciphering key, can read the messages; other senders, in possession of only the public enciphering key, cannot. Other variants exist.

One use for these systems is to distribute keys for a private-key system (such as DES). Public-key systems are traditionally slower than DES, but can be used without prior exchange of secret information. Having exchanged a key, the two parties can now revert to the faster DES (e.g.) for the bulk of their data.

The most popular, and most trusted, public-key system is the Rivest Shamir Adleman (RSA) scheme. Its difficulty is based on the difficulty of factoring large integers into their prime factors. (Actually, it is not yet known to be as hard as factoring, and the precise difficulty of factoring is not known. If one could factor, one could certainly solve the RSA scheme, and all known methods for solving RSA reduce to factoring.)

In this scheme, the owner (receiver) produces two large primes P and Q, which are kept secret. He publishes their product N=PQ. He also produces two integers D and E (the deciphering and enciphering keys), such that DE=1 modulo lcm(P-1,Q-1), and publishes E while keeping D secret.

A sender first encodes his message as a sequence X_i of integers modulo N. He then computes $Y_i=X_i^E$ modulo N (the ciphertext), and sends this to the receiver. The receiver computes $X'_i=Y_i^D$ modulo N; it is easy to show that $X'_i=X_i$.

This scheme can also be used for "signatures": the owner can apply D to a plaintext to produce a "signature" of D. Then anybody can apply E to the signature to recover the plaintext. With proper safeguards, the public can thus verify that only the owner could have produced the signature.

Another public-key scheme is the Diffie Hellman (discrete exponentiation) scheme. Its security is based on the difficulty of taking "logarithms" in finite fields. The scheme was originally proposed as a number-theoretic scheme (operating in GF(p), the integers modulo a large prime p), but has been adapted to operate in $GF(2^n)$ (the ring of polynomials over GF(2) reduced modulo a fixed irreducible polynomial of degree n) for ease of implementation in hardware. The present author showed that there are pitfalls in the $GF(2^n)$ version: "logarithms" are easier to compute in fields $GF(2^n)$ than in GF(p).

In one implementation of the Diffie Hellman scheme, a user C wishes to communicate a message K to user D. The users agree on a large prime p (which may be constant over the whole system or may be generated on the fly). User C generates a secret random number c relatively prime to p-1; he also generates c^{-1} such that $c^{-1}c = 1$ modulo p-1. User D similarly generates d. User C computes K^c and sends it to D. User D computes $(K^c)^d=K^{cd}$ and sends it to C. User C computes $(K^{cd})^{c^{-1}}=K^d$ and sends it to D. Finally User D computes $(K^d)^{d^{-1}}=K$. The transaction is complete. The eavesdropper has seen K^c, K^{cd}, and K^d, but not K. He can deduce K if he can take "logarithms": if, given two numbers X and Y in the field he can deduce an integer Z such that $X^Z=Y$ in the field, then

he can use this ability to find d' such that $(K^c)^{d'}=K^{cd}$, presume d'=d, and compute d^{-1} and K. All known attacks reduce to taking logarithms.

This scheme is used to exchange keys for private-key systems.

Merkle and Hellman produced a public-key system, the knapsack, whose difficulty was related to that of NP-complete problems.

The difficult problem: given a sequence A_i of m n-bit integers and a sum B, produce a set of coefficients ε_i, all 0 or 1, such that $\Sigma_i \varepsilon_i A_i=B$, or assert (truthfully) that none exists. This is in general a hard problem, since any NP-complete problem can be encoded as a special case of this.

The cryptographic system built on this problem: the legitimate owner takes an easy knapsack problem (say one in which each A'_i is larger than the sum of the previous ones) and transforms it into a difficult-looking problem A_i, by multiplying by a secret constant modulo another secret constant. He publishes A_i. Persons wishing to send him a message ε_i compute $B=\Sigma_i \varepsilon_i$ and send B. The legitimate owner can reverse his transformation, solve the easy knapsack, and recover the message. Outsiders presumably could not.

The hitch was that such knapsacks were easily broken (even though the general knapsack might not be). Shamir applied integer programming to the class of knapsacks constructed along these lines, and solved them. Later, Lagarias and Odlyzko showed that most low-density knapsacks were also easily solved, using the Lenstra Lenstra Lovasz lattice reduction techniques. ("Low density" means that the number of A_i's, m, is less than half the number of bits, n.) The lesson: although your

scheme is based on a "hard" problem, your scheme itself might still be easy to crack.

More recently, Ong, Schnorr, and Shamir have developed a family of fast signature schemes, based on number theoretic problems. Unlike RSA, these schemes are easy for both generating and checking signatures.

The owner publishes a polynomial P (in a few variables) and a large composite number N. To sign a message M, he constructs integers X,Y,...,Z such that P(X,Y,...,Z) = M modulo N. The way he constructs P gives him secret information which he can use to construct X,Y,...,Z given M; the outsider, without this secret information, presumably has a harder time of it. In the first scheme, $P(X,Y)=X^2+kY^2$, where $k=-1/u^2$ mod N and u is a secret random number. Then choosing a random number r, set $X=(r^2+M)/2r$, $Y=u(r^2-M)/2r$.

The first two versions of their system have been broken, by Pollard, but the hope remains that more complicated versions might survive.

Considerations for future systems.

Several concerns have been expressed about the present systems.

(1) Some contend that the key-length of DES renders exhaustive attacks possible. I don't believe that this is true yet, but in the next decade or two, as computing becomes cheaper, people might become less willing to rely on a 56-bit key. One solution is triple encryption:

repeating DES with three independent keys. This brings the apparent key-size to 2×56=112 (not 3 ×56=168), which should be sufficient in our lifetimes.

(2) The present schemes all are fairly slow in software, and (with the exception of DES) too expensive for hardware. A faster scheme (but not less secure) would facilitate application of cryptography to areas in which it is now too expensive.

(3) On the public-key side, the existing schemes (RSA and Diffie Hellman) both rely on number theory, on the difficulty of either factoring large integers or taking logarithms modulo large primes. Thus any breakthrough in number theory could disable both the existing schemes. New schemes are needed to protect against this possibility. These schemes might involve several ideas compounded, so that a breakthrough in one field does not necessarily antiquate the scheme immediately.

So there is plenty of room for improvement: key length, ease of implementation, and new ideas in public key schemes.

ATTACKS UPON CRYPTOGRAPHIC SYSTEMS

I present here some of the known methods of attack against various cryptographic systems. Some of the ideas have been used in several different contexts, and a study of these ideas in particular is necessary for a developing cryptographer.

Current attacks:

Current attacks include statistical and group theoretic attacks. I don't sufficiently understand the details of these to speak about them; however, I do know that a knowledge of statistics and group theory is a necessity.

If key-space is small enough, there is always brute force: just try all possible keys until you get a match. This was useful during World War II with the Enigma machine, for several reasons:

(1) the operators of the machine were using a weak protocol;

(2) the operators were unaware of the computing power that the cryptanalysts would bring to bear on the problem;

(3) the key-space actually in use was not sufficiently large (because of (2)).

In modern times, situations such as (2) are unlikely to occur; the cryptographers have a rough idea of the computing strength of the potential cryptanalysts, and are likely to devise keys long enough to foil such attacks. Still one must guard against the use of weak protocols; see below.

The ideas behind Rho method for factoring (due to Pollard) have been employed in two vastly different schemes; one for factoring integers, the other for a space-time trade-off for a "dictionary" attack against block ciphers.

To factor integers, one concocts a polynomial function such as $F(X)=X^2+3 \bmod N$. One then lets X_0 be a random integer, and repeatedly computes $X_{i+1}=F(X_i)$. If P is a factor of N, and if $X_i=X_j \bmod P$, then $X_{i+k}=X_{j+k} \bmod P$ for all positive k. Then $\gcd(N,x_{j+k}-X_{i+k})=P$, and we have factored N. (In practice, one lets $j+k=2^n$, and $2^n<i+k\leq 2^{n+1}$.) If one imagines the values of $X_i \bmod P$, they trace out a figure "rho": an initial tail, then around a cycle, then joining back on itself and continuing to go around the cycle. The expected time before completing the first cycle is $O(\sqrt{P})$. Thus one is hoping for a coincidence, $f(X_{i-1})=f(X_{j-1})$, to complete the cycle, to factor N.

In Hellman's brute-force attack against block ciphers, one is hoping to <u>avoid</u> coincidence. He attacks a block cipher with a message space of size N and a key-space of size $M<N$. The precomputation involves selection of plaintext M_0 which he can expect to find enciphered directly later, and encipherment of M_0 under essentially all possible keys K. The cleverness is in the way he can store the result of this precomputation in space $M^{2/3}$, and use it in time $M^{2/3}$ to deduce an individual key K, given $E_K(M_0)$.

For illustration, take $R=S=T=M^{1/3}$. Hellman constructs R functions f_r mapping message space to key-space. For each r, he selects S starting values X(r,s,0) in key space. Then for t ranging from 1 to T, he computes $X(r,s,t)$ as $f_r(E_{X(r,s,t-1)}(M_0))$. The pairs $(X(r,s,0),X(r,s,T))$ are sorted and stored.

To use the table (of size $RS=M^{2/3}$), we <u>suppose</u> that we have available $E_K(M_0)$ for the unknown key K. For each r, compute $Y(r,0)=f_r(E_K(M_0))$, and for each t, compute $Y(r,t)=f_r(E_{Y(r,t-1)}(M_0))$.

Compare Y(r,t) against X(r,s,T). If there is a match, recompute X(r,s,T-1-t) from X(r,s,0); this is a candidate for K.

(Notice that the assumption of the availability of ciphertext corresponding to chosen plaintext M_0 is defeated by chaining, so that Hellman's attack fails in the presence of chaining.)

The connection with the Rho method is in the possibility of coincidences. If, for a fixed r, we have X(r,s,t)=X(r,s',t'), then for all positive k, we will have X(r,s,t+k)=X(r,s',t'+k), and we will duplicate entries throughout this row of the tables. Thus R,S, and T must be chosen so as to make it unlikely that a given row X(r,s,*) will have a coincidence with ANY other entries in the same table X(r,s',t'). This leads to the restriction that $ST^2<M$. Without this restriction, more favorable time-space trade-offs could be effected.

The fact that the same ideas occur in the Pollard "Rho" method and the Hellman brute-force attack suggest that they are worth study by students of cryptography.

Another technique used in several cryptographic attacks is the Morrison-Brillhart factoring technique. This is used in factoring (RSA), in discrete logarithms in GF(p) (Diffie-Hellman), and in discrete logarithms in $GF(2^n)$ (adapted Diffie-Hellman). I will describe the technique as used in factoring.

To factor N, we wish to find integers X and Y such that $X^2=Y^2 \bmod N$, while $X \neq Y \bmod N$ and $X \neq -Y \bmod N$. Then gcd(X+Y,N) will yield a nontrivial factor of N.

We begin by finding integers A_i such that $A_i^2 \bmod N$ is "smooth", i.e. expressible as the product of small primes. ("Small primes" are smaller than, say,

$$e^{c(\log N \log\log N)^{1/2}}$$

for a given small constant c.) (One way to do so, the "quadratic sieve", is to search through integers A_i near $\sqrt{N}$, applying a "sieve" to find those A_i for which a given small prime q divides A_i^2-N, and looking for A_i which are hit by many "sieve" values q.) For each such A_i, express

$$A_i^2 = \Pi_j q_j^{(e_{ij})} \bmod N.$$

Now, by Gaussian elimination, find a collection I of subscripts i such that

$$(\Sigma_{i\ \varepsilon I} e_{ij}) = 0 \bmod 2 \text{ for all } j$$

Then

$$(\Pi_{i\ \varepsilon I} A_i^2) = \Pi_j q_j^{(\ \Sigma_{i\ \varepsilon I} e_{ij})} \bmod N,$$

$$(\Pi_{i\ \varepsilon I} A_i)^2 = (\Pi_j q_j^{(\ \Sigma_{i\ \varepsilon I} e_{ij}/2)})^2 \bmod N,$$

$$X^2=Y^2 \bmod N,$$

and there is no reason to think that X=Y or X=-Y mod N.

This technique, with modifications, forms the basis for all the modern integer factorization routines, for Adleman's algorithm for the discrete logarithm mod p, and for Coppersmith's algorithm for the discrete logarithm in $GF(2^n)$; in the latter instance it works much better because a quirk of fields of small prime characteristic enables us to produce small quantities which we require to be smooth, and small

quantities are more likely to be smooth than large quantities. Having found three applications, it too is recommended for study by cryptographic students.

The Lenstra Lenstra Lovasz Basis reduction algorithm is a recent advance which is finding applications to several cryptanalytic algorithms.

Their algorithm begins with an integer lattice (i.e. a subset of Z^n closed under addition), and with a basis for this lattice, and finds a relatively short basis for the lattice (each basis vector is relatively short). It has been used to factor univariate polynomials over Z, factor univariate polynomials over Z/p, and, in a cryptographic application by Lagarias and Odlyzko, solve low-density knapsack problems.

If one can test whether one's hypothesized "key" is "close" to the correct key (in some sense), then one can apply hill-climbing techniques. Suppose some easily computed "objective function" is correlated to the distance of the hypothesized key from the correct key. One makes an initial guess at the key (perhaps a random guess), and repeatedly tries making small changes, seeing what effect they have on the objective function, and keeping those which have a favorable effect.

This is unlikely to work against well-designed systems. It should be considered, however, in the design of systems.

One should know all the tricks which have worked in the past. But most important is ingenuity, to discover tricks which will work in the future, and to apply existing techniques in new areas. This is much harder to teach than mere technique, but it is essential.

A strong cryptographic scheme can still be attacked if its protocols are weak: if there is a weakness in the way that the system is applied. To give a few examples:

The RSA (Rivest Shamir Adleman) scheme is multiplicative, in that the encryption of message X, times the encryption of message Y, is the encryption of message XY. If it is employed as a signature scheme by a "notary public", the possibility exists that I could ask the "notary public" to sign two innocuous-looking documents X and Y, and produce his signature to the damaging document XY.

With the similar Rabin scheme (the signature of the message X is a Y such that $Y^2=X \bmod N$, where N is a composite whose factors are secret), I could ask a notary public to sign a document X (where I have secretly computed $X=Z^2$), and comparing the signature Y with Z, I might deduce the factorization of N: gcd(N,Z-Y) can be a nontrivial factor of N. Here one signature was enough to break the whole scheme.

To exchange keys in the Enigma, the practice was to encrypt the session key under a master key, twice in succession. Cryptanalysts were able to make use of coincidences in the subsequent encryption to make deductions about the master key.

The PIN (personal identification number) of an electronic credit card, loses its cryptographic value if the owner writes it on the back of his card.

Suppose we use a record-chaining version of DES for message authentication: the message to be authenticated is $(X_1,X_2,\ldots,X_n)$, and we are able to "sign" a shorter message such as (K,Y_n). Then a potential message authentication scheme: set $Y_0=0$, $Y_i=E_K(X_i+Y_{i-1})$, and "sign" K,Y_n. The trouble: I can concoct any message $X'_1,\ldots,X'_{n-1}$, and as long as I know K and Y_n, I can easily figure out the $X'_n=D_K(Y_n)-Y'_{n-1}$ that will give the same "signature".

Future attacks:

Who knows? It depends on the ingenuity of the cryptanalyst.

APPLICATIONS OF CRYPTOGRAPHIC SYSTEMS

The most obvious application for cryptography is privacy, or protection of data against unauthorized disclosure.

Data can also be authenticated, either privately or publicly. Privately, one can insert redundancy into one's data before encrypting, and check that that redundancy exists after decrypting. If the redundancy is sufficiently nonlinear and time-dependent, one can be assured that the message has not been tampered with. (However, linear redundancy such as error-correcting codes, along with linear cryptosystems such as a one-time pad, do not afford this security.) Publicly, with a public-key scheme, one can arrange that anybody can verify that a message came from the alleged source.

These two properties point to applications in the military; in diplomatic traffic; in electronic funds transfer. Proposals have been made for encryption on identity papers, particularly to aid authentication and prevent forgeries. Sensitive data on computer files is a natural place to employ cryptography.

Protection of commercial products: software, online stock services, pay-TV. These demand a range of schemes: the pay-TV must involve fast decryption, but can afford to be a weak scheme; software protection can be more slowly decrypted but must be somewhat stronger (depending on the value of the software). As cryptography matures, this kind of application will find more use.

More exotic new applications include "provable" random number generators (Goldwasser et al: a random number generator such that, if it fails to appear "random" for your particular application, then your application provides a way to break a particular cryptosystem).

DISCUSSION

Lecture 1

Peter Brownn raised a question: what will happen if the input (clear text) happens to be a massive sequence of blanks, when the output (cipher text) is derived as the function of the input and the previous output. The answer from Dr. Coppersmith was that the output will be cycling.

The issue of key length in the DES encryption chip sparked off some discussion. Professor Turn asked why IBM's Lucifer encryption algorithm had employed a 128 bit key. Dr. Coppersmith believed that the long key had been justified on the grounds of security. Mr. Davies thought that Lucifer had been used in the system applied by IBM to Lloyds bank for automated cash dispensing. Dr. Coppersmith observed that the retention of a longer key for the DES could have averted the widespread criticism that a 64 bit key was open to an (albeit expensive) exhaustive search. Professor Turn noted that the criticism was directed at National Bureau of Standards and not at IBM.

Miss Barraclough asked how keys were exchanged at the first instance, to which Dr. Coppersmith replied that session keys were usually exchanged using a master key, and that master keys were established by means of a secure channel (e.g. by meeting, via a courier or by using a one-time pad).

Professor Llewellyn asked what were the relative effects of accidental errors in transmission on the different encryption schemes. Dr. Coppersmith noted that both accidental and deliberate corruptions of cipher text had to be considered. Some protection against some classes of error could be provided by using redundancy for error correction purposes. Professor Churchhouse noted that for some methods changing a single bit could destroy the information content of all subsequent bits.

Professor Tanenbaum wondered if it could be possible to attack any scheme using prime numbers by considering fairly standard techniques used to generate primes. Dr. Coppersmith pondered over all the difficulties this could entail, and Professor Churchhouse pointed out that generating a prime number is very difficult - instead integers can be generated and tested for primality.

Professor Randell asserted that history shows that breaches of ciphers are most often due to mistakes in their operation, a point reinforced by Mr. Lindley who thought that mistakes are inevitable given the personnel involved in transmitting enciphered material. Dr. Coppersmith concurred, and suggested that the greater part of the work in dispensing protocols for using ciphers is in trying to make them idiot proof. Professor Churchhouse advised that the most direct attack against any technique lies in theft of the method, of the keys, even the clear text, by bribery, blackmail, burglary, etc.

Agreeing, Dr. Coppersmith mentioned that key security was the subject of work on the "key management problem". Professor Randell returned to the past to observe that at any time in history the currently most sophisticated ciphers were considered to be secure, based on the current theory and understanding, but that this reasoning has invariably been overturned by subsequent developments.

Mr. Davies closed the discussion on a technical note that Diffie and Hellman's paper had not actually supplied the algorithm, as credited to them by the speaker, though the essential insight had been provided by them.

Lecture 2

Professor Turn asked if parallel processing will help the factorising algorithm work out faster. Dr. Coppersmith answered positively and observed how the CRAY, a vector based system, can be exploited.

Lecture 3

Professor Randell, referring to the choice of public-key versus private-key, asked whether public-key systems were likely always to be slower than equally secure private-key systems (given some suitable measure of security)? Dr. Coppersmith replied that this could be a historical accident; it was not always necessarily so.

Dr. Freeman asked what the speaker meant by such terms as "truly random sequence" or a "fair coin". In the chip testing example, we have in principle one very long but finite string of inputs that will exhaustively exercise all possible faults; either we use an exhaustive test, in which case no randomness is necessary, or we can only be using a small fraction of possible inputs, which could be generated by a suitable algorithm. This application doesn't necessarily need randomness; the speaker had not emphasised the sequential aspects - and sequential complexity is quite a different kettle of fish from combinatorial complexity.

Professor Denning intervened to clarify the discussion, by putting two distinct questions: firstly, it is really necessary to use random number generation to test chips, and secondly how do we effectively compare different random number generators?

Dr. Coppersmith replied that it was indeed not necessary to use such methods, but that people DO use them. To the second question, the answer was to apply polynomial-time tests.

Dr. Freeman asserted that the only satisfactory measure of randomness known to him was Kolmogorov's, based on the complexity of a minimal Turing machine capable of generating the sequence. If the argument on the board worked, that would seem to knock a great hole

in Kolmogorov's results. In reply, Dr. Coppersmith pointed out that Kolmogorov's is not an effective test: one should include statistical test, an effective randomness test and consider the application aspects, so polynomial tests are preferred.

A brief discussion on the security of PAY-TV signals followed, with some disagreement on the complexity of the problem. Professor Randell noted that the "messages" in this situation were at most valuable for a few minutes; the speaker had pointed out that breaking cyphers required both ingenuity and time, and he assumed that it was not yet possible to buy ingenuity, in the form of black boxes for completely automatic decryption. Quite a lot of PAY-TV has no encryption, and it was pointed out that some early systems merely added noise to the signal. The economic case for encryption was doubtful, and Dr. Hitchcock said that what most companies in this field worried about was pirates manufacturing cheaper de-scramblers to replace those provided, and hence by-passing their revenue collection! Dr. Coppersmith classified the people against whom the security should be provided into "casual hackers" (easy to handle), and "pirates" who are prepared to invest time and money in breaking the system (hence demanding severe measures).

Dr. Lipner observed that the problem of software piracy was one of the toughest to deal with, as one had to end up at some stage with "clear-text" in one's possession for execution on a processor. There had been a proposal from INTEL for a "crypto-micro-processor", based on RSA/DES encryption, a single-chip processor and a key management scheme, but this was not absolutely or extremely secure.

Professor Randell suggested that the solution might be to go back to selling programs and their processors "bundled" together again! Almost any technology providing great advantages was accompanied by equally great disadvantages.

Dr. Hartley questioned whether the only form of protection worth considering was encryption, and referred to copyright and other forms of legal protection. Mr. Voysey argued that it might be totally wrong to prevent distribution in these ways, and that we should treat software as we do books, with a change of view from that of selling industrial devices to literary production; while another speaker pointed out that we expect a manufacturer to provide a manual for software products, giving a detailed description of its behaviour, from which an equivalent product might be re-developed.