SECRET LINEAR CONGRUENTIAL GENERATORS ARE NOT CRYPTOGRAPHICALLY SECURE

JACQUES STERN

DÉPARTEMENT DE MATHÉMATIQUES
UNIVERSITÉ PARIS 7

ABSTRACT.--This paper discusses the predictability of the sequence given by outputing a constant proportion α of the leading bits of the numbers produced by a linear congruential generator. First, we make the assumption that the modulus of the generator is the only known parameter and we prove that, almost surely, a significant proportion of the bits can be predicted from the previous ones, once the generator has been used K times successively where K is $O(\sqrt{\log m})$. Next, we assume that all parameters of the generator are secret and we show how repeated observations of sequences of outputs of length K will probably allow an opponent to cryptanalyze the full sequence.

1. INTRODUCTION

A basic and very popular tool to generate pseudo-random sequences is provided by the linear congruential generator (LCG). The LCG works as follows: a modulus m is chosen as well as a multiplier a, relatively prime to m, and an increment b. Then, from a given seed x_1, one can generate the sequence (x_i) defined by

$$x_{i+1} = a.x_i+b \pmod m$$

Knuth (vol. 2) contains a thorough discussion of these generators : some requirements have to be met by a, m, b but it is not clear that these requirements make the sequence produced by the generator look really "random".

Because of the existence of many algorithms that run in random polynomial time, it is particularly important to produce from short seeds, long sequence of bits or of numbers that behave randomly. Problems of this kind have been greatly investigated in the past years, especially by Shamir (1980), Blum and Micaly (1982), Yao (1982). Therefore it seems important to discuss the predictability of the sequence of bits produced by a LCG.

In case all the bits of the successive x_i are announced, the sequence becomes exactly predictable even if the modulus, the multiplier and the increment are unknown. This result is due to Plumstead (1982). Following a suggestion of Knuth (1980), one can use a LCG by outputing the leading part of each of the x_i's, for example the leading half of the bits. The predictability of the resulting sequence has been investigated by Frieze, Kannan and Lagarias (1984). They show that, provided both the modulus m and the multiplier a are known, the sequence becomes completely predictable once the leading bits corresponding to the first few x_i's have been announced. Actually, their algorithm may fail on a set of exceptional multipliers but the proportion of integers (mod m) in this set is shown to be as small as $O(m^{-\frac{1}{5}})$.

In order to describe our results, we need some notation. We let n be the number of bits of the integer m. If we output a proportion α of the bits, we can write

$$x_i = 2^{\beta n} y_i + z_i \qquad 0 \leq z_i < 2^{\beta n}$$

where $\beta = 1-\alpha$ and where we assume implicity that βn is an integer. We consider the sequence σ consisting of the successive bits of $y_1, y_2 \ldots$ and we first ask the following question : given m, is it possible to predict a bit of the sequence σ from the previous ones, with a small proportion of mistakes ? We propose an algorithm A which solves this problem and we state the main properties of our algorithm for the case $\alpha = \frac{1}{2}$.

i) Almost always, the algorithm predicts the correct value of the next bit with a proportion of mistakes which is asymptotically bounded by $O((\log m)^{-\frac{1}{2}})$.

ii) The algorithm may fail exceptionnally. Failure depends on the (unknown) value of a and the proportion of multipliers a causing failure is asymptotically bounded by $O(m^{-\frac{1}{5}})$.

iii) Failure of the algorithm can be detected by observing a large proportion of errors persisting after the first $\sqrt{3 \log m}$ y_i's have been announced (i.e. the first $n\sqrt{3 \log m}$ bits of the sequence).

0272-5428/87/0000/0421$01.00 © 1987 IEEE

Those results should be compared to those of Frieze, Kannan and Lagarias (1984) : they discuss the case where a is known as well as m. Of course, because we make the hypothesis that a is kept secret it is not surprising that our algorithm requires more information (roughly $\sqrt{3 \log m}$ of the y_i's instead of 4).

Actually, as will be seen in the sequel, our algorithm generally produces a polynomial $P(x)$ of degree $\sqrt{3 \log m}$ (approximately) such that

$$P(a) \equiv 0 \pmod m$$

Repeated use of algorithm A will produce a sequence of such polynomials P_j. We then propose an algorithm B which starts from a sequence of polynomials P_j of fixed degree such that

$$P_j(a) \equiv 0 \pmod m$$

and outputs a multiple $\hat{m}$ of m, which quickly decreases to m with high probability, when the length of the sequence increases.

Finally, we indicate how to recover a, once the correct value m of $\hat{m}$ has been reached. From the results of Frieze, Kannan and Lagarias, any further use of the given generator is highly insecure as the sequence of outputs can be predicted from its first four terms.

2. DESCRIPTION OF ALGORITHM A : PART 1.

In order to describe our algorithm, we need some more notation. We let v_i be the element of $\mathbb{Z}^3$ whose coordinates are

$$\begin{matrix} y_{i+1} - y_i \\ y_{i+2} - y_{i+1} \\ y_{i+3} - y_{i+2} \end{matrix}$$

As an intermediate step, we will use a variant of the well known algorithm of Lenstra, Lenstra and Lovâsz (1982), which finds a non trivial integer relation

$$\sum_{i=1}^{K} \lambda_i v_i = 0$$

between K elements $v_1,\dots,v_K$ of $\mathbb{Z}^3$. This variant can be found in a paper by Hastad, Helffich, Lagarias and Schnorr (1986). The resulting algorithm requires polynomial time and outputs an integer relation $\lambda = (\lambda_1,\dots,\lambda_K)$ whose length does not exceed too much the minimal possible length $\lambda_{\min}$ of such a relation. More precisely :

$$|\lambda| \leq 2^{\frac{K}{2}-1} \lambda_{\min}.$$

We now outline the first part of our algorithm. This part takes as imputs the first $K+3$ values

$$y_1, y_2, \dots, y_i, y_{K+3}$$

of the sequence y_i, where K is a constant and outputs a polynomial P with integer coefficients of degree at most K such that, presumably,

$$P(a) \equiv 0 \pmod m)$$

Reasonable choices for K will be described later.

Algorithm A (part 1)

```
1. Use the L.L.L. algorithm in order to find a short
   integer relation
       $\sum_{i=1}^{K} \lambda_i v_i = 0.$
2. Let $\ell := \max\{i : \lambda_i \neq 0\}$.
3. Output the polynomial $\sum_{i=1}^{\ell} \lambda_i x^{i-1}$.
```

Before we turn to the second part of the algorithm, we shall discuss the correctness of the first. We need a lemma.

Lemma 1.- There exists an integer relation

$$\sum_{i=1}^{K} \mu_i v_i = 0$$

such that $\max|\mu_i| \leq B$ where B is given by

$$\log B = \frac{3(\alpha n + \log K+1)}{K-3}.$$

Sketch of proof : Consider all possible linear combinations

$$\sum_{i=1}^{K} \nu_i v_i$$

with $0 \leq \nu_i < B$. By a counting argument using a bound on the coordinates, ckeck that two distinct linear combinations give the same result. ■

We now focus on the unknown sequence V_i defined by

$$V_i = \begin{pmatrix} x_{i+1} - x_i \\ x_{i+2} - x_{i+1} \\ x_{i+3} - x_{i+2} \end{pmatrix}$$

It is easily seen that all vectors V_i lie on the 3-dimensional lattice $L(a)$ generated by the vectors

$$\begin{pmatrix} 1 \\ a \\ a^2 \end{pmatrix} \quad \begin{pmatrix} 0 \\ m \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ m \end{pmatrix}$$

This lattice has determinant m^2, so that the average length of its short elements is around $m^{2/3}$. Thus, it is rather unlikely to find in $L(a)$ an element whose length is of order m^γ, for γ much smaller than $\frac{2}{3}$. In their paper, Frieze, Kannan and Lagarias have turned this remark into the following statement.

Lemma 2.- The proportion of integers a, $0 < a < m-1$, such that $L(a)$ contains a non zero vector of length bounded by m^γ is an $O(m^{\frac{5\gamma-3}{2}+\varepsilon})$ for any $\varepsilon>0$.

Now if $\lambda = (\lambda_1,\dots,\lambda_k)$ is the integer relation given by the algorithm, we have :

$$|\lambda| \leq \lambda_{min} 2^{\frac{K}{2}-1}$$

and thus, if B is as above :

$$|\lambda| \leq B\sqrt{K}\, 2^{\frac{K}{2}-1}.$$

Comparing the unknown vector

$$U = \sum_{i=1}^{K} \lambda_i U_i$$

with the (zero) vector

$$\sum_{i=1}^{K} \lambda_i v_i$$

we get the following bound :

$$|U| \leq M\sqrt{3}$$

where $M = KB\, 2^{\frac{K}{2}-1}\, 2^{\beta n}$.

Since :

$$\log M = \log K + \frac{K}{2} - 1 + \beta n + \frac{3(\alpha n+\log K+1)}{K-3}$$

we get, taking K to be approximately $\sqrt{6\alpha \log m}$:

$$\log M = \beta \log m + \sqrt{6\alpha \log m} + O(\sqrt{\log m})$$

and therefore

$$M = O(m^{\beta+\delta}) \text{ for any } \delta>0.$$

The correctness of the algorithm relies on the fact that U is actually zero. In this case, we get

$$\sum_{i=1}^{K} \lambda_i (x_{i+1} - x_i)$$

so that

$$\sum_{i=1}^{K} \lambda_i a^{i-1} (x_2 - x_1).$$

Hence except in the special case where x_2-x_1 is not prime to m, we get

$$\sum \lambda_i\, a^{i-1} = 0 \pmod{m}.$$

Regardless of this property we have, for each j,

$$\sum_{i=1}^{K} \lambda_i (x_{i+j+1} - x_{i+j}) = 0 \pmod{m}.$$

If ℓ is the largest index i with $\lambda_i \neq 0$ we get

$$\lambda_\ell\, x_{\ell+j+1} = \sum_{i=1}^{\ell} \theta_i\, x_{i+1} \pmod{m}$$

where $\theta_i = \lambda_i - \lambda_{i-1}$ (where $\lambda_o=0$) and this equality will allow the cryptanalysis of the sequence of bits generated by the LCG.

Now since U belongs to $L(a)$ and $|U|$ is bounded by $O(m^{\beta+\delta})$, lemma 2 shows that taking $K = \sqrt{6\alpha \log m}$ will yield a successful algorithm in a proportion of cases exceeding $1-O(m^{\frac{5\beta-3}{2}+\frac{5}{2}\delta+\varepsilon})$ i.e. exceeding $1-O(m^{\frac{5\beta-3}{2}+\varepsilon'})$. This means that, asymptotically, the algorithm is almost always successful provided $\beta<\frac{3}{5}$ i.e. $\alpha>\frac{2}{5}$. If $\alpha=\frac{1}{2}$, then, choosing $\varepsilon'=\frac{1}{20}$ yields a proportion of failure bounded by $O(m^{-\frac{1}{5}})$.

Remark. In a forthcoming paper, Frieze, Hastad, Kannan, Lagarias and Shamir (FHKLS) establish a sharper version of Lemma 2, from which it follows that our algorithm is asymptotically almost always successful as soon as $\alpha > \frac{1}{3}$.

3. DESCRIPTION OF ALGORITHM A : PART 2

We now turn to the second part of our algorithm. We assume that the first part has been executed for a suitable value of K. We let $\lambda_1,\dots,\lambda_\ell$ denote the coefficients of the output polynomial and we try to predict the successive bits of some $y_{\ell+j+1}$ for $\ell+j > K+3$.

Algorithm A (second part)

1. For $i := 1$ to ℓ set $\theta_i := \lambda_i - \lambda_{i-1}$ (with $\lambda_o=0$).

2. Let t be the integer consisting of the first h bits of $y_{\ell+j+1}$ that have been announced so far. Let

$$r := \sum_{i=1}^{\ell} \theta_i (y_{i+j}.2^{\beta n}+2^{\beta n-1})$$

$k := \lceil(\lambda_\ell(t+1/2)2^{n-h}-r)/m\rfloor$ {this means the closest integer}

$y := \lfloor(k.m+r)/\lambda_\ell\rfloor$ {this means the integer part}.

3. Announce the $(h+1)$-th bit of y.

The correctness of this algorithm relies on the following fact.

Fact. Provided part 1 of algorithm A has been successful, one can find integers h_o, n_o such that :

i) whenever h is $\geq h_o$ the output y found in step 2 differs from the correct $x_{\ell+j+1}$ by at most 2^{n_o}.

ii) $n_o+h_o \leq \log M+2$ where M is a above.

From the fact, we can compute a bound for the average proportion of mistaken bits as

$$\frac{\log M - \beta n+5}{2}.$$

Using our previous estimates, we see that this is equivalent to $\sqrt{6\alpha \log m}/2$ when K is taken to be approximately $\sqrt{6\alpha \log m}$.

4. ALGORITHM B.

As was already noticed, the output of the first part of algorithm A is a polynomial P of degree $< K$ such that

$$P(a) \cdot (x_2-x_1) = 0 \pmod m.$$

If x_2-x_1 is prime to m, this yields

$$P(a) = 0 \pmod m.$$

Such a polynomial can be uniquely written as an integer linear combination of the polynomials.

$$q_i(x) = x^i-a^i \qquad 1 \leqslant i < K$$

and of the constant polynomial $q_0(x) = m$. Hence it belongs to the lattice $M_K(a,m)$ generated by those polynomials, whose determinant is m. Algorithm B starts from a sequence of polynomials.

$$P_1, P_2, \ldots, P_n$$

of the lattice $M_K(a,m)$ and computes the determinant $\hat{m}$ of the sublattice generated. $\hat{m}$ is a multiple of m and when n increases, $\hat{m}$ converges to m.

What we need is to convert a set of generators of a lattice into a basis. This is a linear algebra problem which was considered independently in the FHKLS paper where it is solved using Hermite normalform. Our algorithm is directly patterned after the algorithm of Lenstra, Lenstra and Lovasz (1982). We use the sequence (P_j^*) defined recursively by the following property : P_j^* is the component orthogonal to $P_1^*, \ldots, P_{j-1}^*$ of the first element $P_{i(j)}$ of the original sequence, which does not belong to the vector space spanned by $P_1^*, \ldots, P_{j-1}^*$. We let p be the largest value of j (i.e. the dimension of the lattice) and I be the set of indices $i(j)$, $1 \leqslant j \leqslant p$. Finally we define the coefficients μ_{ij}, $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant p$, by :

$$P_i = \sum_{j=1}^{p} \mu_{ij} P_j^*$$

<u>Algorithm B</u>.

```
1. Compute $\mu_{ij}$, $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant p$ and $|P_j^*|^2$.
2. {Test on termination}. If $P_i=0$ for any $i \leqslant n-p$,
   then, output :
        $\hat{m} = \prod_{j=1}^{p} |P_j^*|$
3. {Exchange step}. Let $i$ be the first index such
   that $i \in I$ and $i+1 \notin I$ ; let $\ell$ such that $i = i(\ell)$,
        $P_{i+1} := P_{i+1} - \lceil \mu_{i+1,\ell} \rfloor P_i$ ;
        exchange $P_{i+1}$ and $P_i$ ;
        update $|P_i^*|^2$ and $\mu_{ij}$, $\mu_{i+1,j}$ for $j = 1,\ldots,p$.
4. {reduction step}. For $j = \ell-1$ down to $1$ do :
        $P_{i+1} := P_i - \lceil \mu_{ij} \rfloor P_{i(j)}$ ;
        goto 2.
```

Algorithm B can be analyzed much in the same way as the L.L.L. algorithm. Progress in the reduction is related to the quantity

$$\prod_{i \notin I} 2^i \cdot \prod_{j=1}^{p} |P_j^*|$$

and the algorithm uses at most $O(K^3(\log R+n))$ arithmetical operations on $O(K.\log R)$ bits integers where R is a bound on the length of the given P_i's. We note that, as soon as p coïncides with the dimension K of the lattice $M_K(a,m)$, the output $\hat{m}$ is a multiple of m, bounded by R^K.

In order to discuss how quickly $\hat{m}$ converges to m, we make the assumption that the P_i's are random elements of $M_K(a,m)$ of length $<R$ where R is large w.r.t. λ, the supremum of the successive minima of the lattice. We make the following observations :

1. If p is $<K$, then the probability that adding a new polynomial P_{n+1} does not increase p is extremely small. Using volume computations it can be estimated as $O(\frac{\lambda}{R})$ provided K is small w.r.t. B (say $K = O(\log R)$). Hence almost surely $p=K$ as soon as $n \geqslant K$.

2. If $p=K$ and $\hat{m}>m$, the probability that adding a new P_{n+1} does not decrease $\hat{m}$ is bounded by $\frac{m}{\hat{m}}$. From this, we get, setting $\rho = \log(\frac{R^K}{m})$, that the probability that $\hat{m}$ is still $>m$ after adding $\rho \log \rho$ polynomials can be given the (crude) bound

$$\frac{(\log \rho)^\rho}{\Gamma(\rho)}$$

which is dramatically small.

Our method, in order to cryptanalyze the sequence (y_i), is to use repeatedly algorithm A (part 1) and to apply algorithm B to the resulting family of polynomials. Keeping our previous choice of K, we get the following :

$$K \sim \sqrt{6\alpha \log m}$$
$$\log R \sim \sqrt{6\alpha \log m}$$
$$\rho \sim (6\alpha-1)\log m.$$

We can also consider that $\log \lambda$ is of order $(\log m)^{\frac{1}{K}}$ because $M_K(a,m)$ has determinant m. Finally, the estimates that have been used in the analysis of algorithm B are met and the algorithm should succeed with a number of polynomials bounded by $(6\alpha-1)\log m \log\log m$.

Still, we cannot prove formally the correctness of our algorithm because two hypotheses remain questionable.

Firstly, the random behaviour of the

sequence of output polynomials cannot be established mathematically. Nevertheless, we can give a (heuristic) argument explaining why distinct polynomials are obtained. Remember that the output of algorithm A provides an actual relation

$$\sum_{i=1}^{K} \lambda_i (x_{i+1} - x_i)$$

(as opposed to a mod m relation). If the same polynomial appeared all the way, then (x_i) would satisfy some linear recurrence relation. But the asymptotic behaviour of such sequences is quite regular : except in very special cases, they tend over to zero or infini

Se we have to discuss the hypothesis asserting the output polynomial P satisfies

$$P(a) = 0 \pmod{m}.$$

This uses the fact $d=1$, where

$$d = \gcd(m, x_2 - x_1).$$

Before we turn to the case where d is $\neq 1$, let us briefly outline an algorithm that will give the value of a provided the correct value of m has been reached through algorithm B. If this happens, algorithm B actually outputs a basis of $M_K(a,m)$ as :

$$P_i \quad n-p+1 \leqslant i \leqslant n$$

It is possible to design an algorithm that uses only polynomially many basic transformations of the type:

$$P_i := P_i - \theta\, P_j \pmod{m} \quad \theta \in \mathbb{Z}$$

and ends up with a basis Q_i, $1 \leqslant i \leqslant K$, such that

$$Q_i = x^{K-i} - \theta_i \qquad 1 \leqslant i \leqslant K-1.$$
$$Q_K = m$$

Once this is done, we have

$$a^j = \theta_{K-j} \pmod{m},$$

and thus we know the value of a.

We now come back to the situation where d is $\neq 1$. In this case algorithm B will probably output $\tilde{m} = \frac{m}{d}$ in place of and from $\tilde{m}$, we can extract the value of $\tilde{a} = a \bmod m$. We claim that this will be just as good. Actually, it can be checked that the method of Frieze, Kannan and Lagarias (1984), applied with $\tilde{m}$, $\tilde{a}$ in place of m, a will almost always disclose the hidden bits, provided d is not too large, say $d \leqslant m^{1-\frac{5}{3}\beta-\varepsilon}$ i.e., for $\beta = \alpha = \frac{1}{2}$, $d \leqslant m^{\frac{1}{6}-\varepsilon}$. But this inequality is quite likely to hold : the expected value of $\log d$ is $O(\log\log m)$ and i can be shown that the probability that d exceeds m^{δ} is asymptotically small.

5. DISCUSSION AND POSSIBLE EXTENSIONS OF THE ALGORITHM

We close the paper by various remarks.

1) First of all, we claim that the range of applicability of the method is quite satisfactory. For example, with $n = 100$, the estimated proportion of mistaken bits for algorithm A is 0.28 and the number of consecutive observations needed is around 20. It should be said that the bound on $|\lambda|$ obtained through lemma 1 together with estimates on the output of the.L.L.L. algorithm is very crude. Thus, our algorithms behave more nicely. Furthermore, the L.L.L. algorithm usually provides more than one short integer relation so that algorithm B will actually require only a few extra observations.

2) In the simple case where m is a prime number, it is known that $d=1$ and in order to disclose a it is also possible to solve the equation $P(a) = 0 \pmod{m}$.

3) The case where the trailing bits of the successive x_i's are annouced in place of the leading bits can be attacked by a similar technique, at least if m is odd.

4) Our algorithm is a special case of a family of algorithm corresponding to working with k-dimensional vectors. Thus, if only a small proportion α of the leading bits is announced and more generally, if algorithm A fails, it is still possible to try the four dimensional version of this algorithm by replacing v_i by

$$\begin{pmatrix} y_{i+1} - y_i \\ y_{i+2} - y_{i+1} \\ y_{i+3} - y_{i+2} \\ y_{i+4} - y_{i+3} \end{pmatrix}$$

In the analysis of this algorithm $L(a)$ is replaced a 4-D lattice with determinant m^3. The FHKLS paper provides the asymptotic estimates needed to analyze all these algorithms, provided m is squarefree (or at least not highly composite).

5) The FHKLS paper also includes an idea which can be used in order to adapt our techniques to the case where m is prime and a window of successive bits of the x_i's are annouced. Because some loss of information occurs the k-dimensional version of our algorithms will presumably be needed, with $k>3$.

6) Acknowledgement.

Shortly after the present article was accepted, I received a preliminary version of the FHKLS paper. I am grateful to J. Hastad for making this reference available to me and for various nice remarks and suggestions.

REFERENCES

[1] M. Blum and S. Micali, How to generate cryptographically strong sequences of pseudo-random bits ? Proceedings of the 23rd IEEE Symposium on the Foundations of Computer Science (1982).

[2] A.M. Frieze, R. Kannan and J.C. Lagarias, Linear congruential generators do not produce random sequences. Proceedings of the 25th IEEE Symposium on the Foundations of Computer Science (1984), 480-484.

[3] A.M. Frieze, J. Hastad, R. Kannan, J.C. Lagarias and A. Shamir, Reconstructing truncated integer variables satisfying linear congruences, SIAM J. Computing (to appear).

[4] J. Hastad, B. Helfrich, J.C. Lagarias and C.P. Schnorr, Polynomial-time algorithms for finding integer relations among real numbers, Proceedings of the 3rd STACS (1986).

[5] D.E. Knuth, Seminumerical Algorithms. The Art of Computer Programming, Vol. 2. Addison-Wesley (1969).

[6] D.E. Knuth, Deciphering a linear congruential encryption, Technical report n° 024800, Stanford University (1980).

[7] A.K. Lenstra, H.W. Lenstra and L. Lovàsz, Factoring polynomials with rational coefficients Math. Ann. 21 (1982), 515-534.

[8] J. Plumstead, Inferring a sequance generated by a linear congruence, Proceedings of the 23rd IEEE Symposium on the Foundations of Computer Science (1982), 153-159.

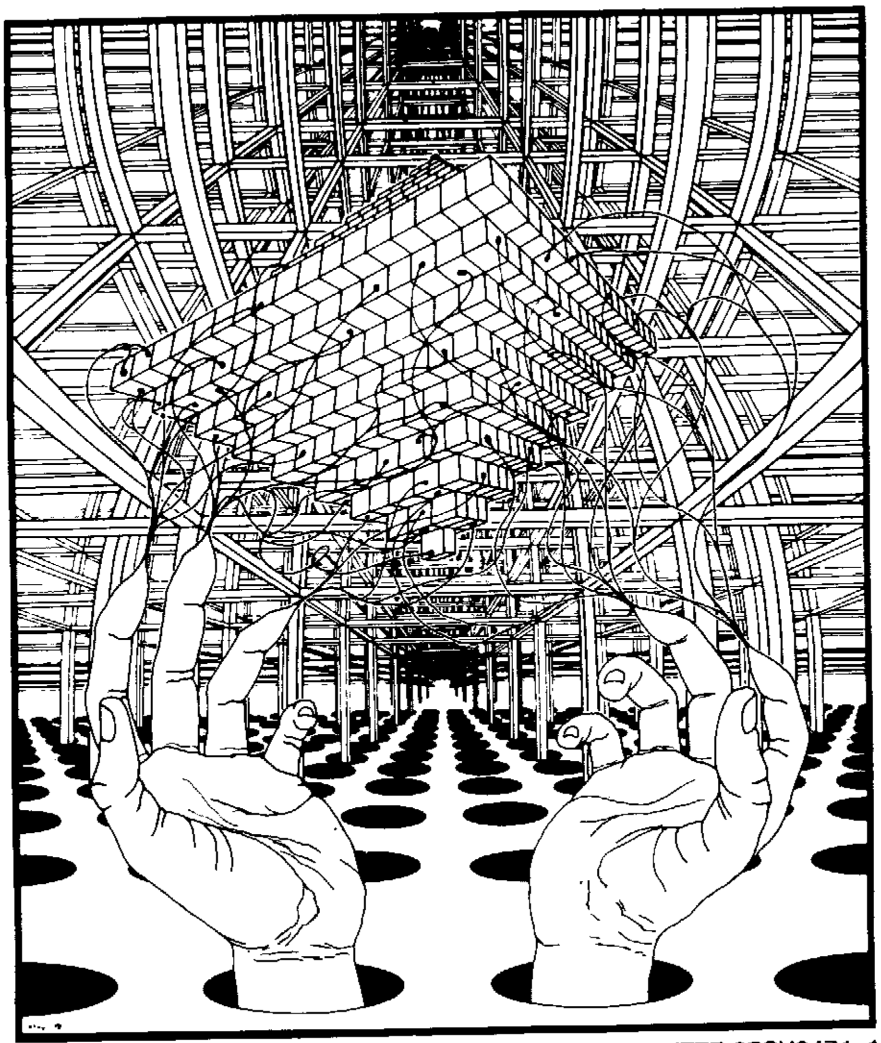
[9] A. Shamir, On the generation of cryptographically strong pseudo-random sequences, Seventh ICALP (1980).

[10] A. Yao, Theory and applications of trapdoor functions, Proceedings of the 23rd IEEE Symposium on the Foundations of Computer Science (1982), 80-91.

28th Annual

Symposium on Foundations of Computer Science

(Formerly called the Annual Symposium on Switching and Automata Theory)



OCTOBER 12-14, 1987 IEEE 87CH2471-1

sponsored by
the Computer Society of the IEEE Technical Committee on
Mathematical Foundations of Computing

THE COMPUTER SOCIETY OF THE IEEE
THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.
COMPUTER SOCIETY PRESS

ISSN 0272-5428
Computer Society Order Number 807
Library of Congress Number 80-646634
IEEE Catalog Number 87CH2471-1
ISBN 0-8186-0807-2
SAN 264-620X