er **Z** from the recurrence $y_0 = 0$ and
s exponential in log M, which is the
M.
on does not apply to the more general
$\leq 0\}$ generated by polynomial maps
Z, where

$$_n = S(\mathbf{x}_n),$$

odulus M are known. In this case the
ces the observed data $\{z_i : 0 \leq i \leq n\}$ is
time algorithm to solve it. The problem
y difficult.

H. Conway for suggesting that our
commutative rings with unit, to C. P.
Bezout's theorem, to J. P. Butler for
comments.

S

e generated by a linear congruence, Proc. 23rd
ns of Computer Science, 1982, pp. 153-159.
uced by pseudo-random number generators, Ph.D.
lif. Berkeley, CA, 1983.
-random number generators, preprint.
for the solution of linear diophantine equations,

IAS AND A. SHAMIR, *Reconstructing truncated*
this Journal, 17 (1988) pp. 262-280.
, Linear congruential generators do not produce
Conference on the Foundations of Computer

d excess intersection, in Enumerative Geometry
er, Boston, 1982.
, New York, 1969.
r-Verlag, New York, 1977.
c security of truncated linearly related variables,
Theory of Computing, 1985, pp. 356-362.
gorithms for computing the Hermite and Smith
l, 8 (1979), pp. 499-507.
g, Vol. 2, Seminumerical Algorithms, Addison-

MA, 1970.
nomials and $Q(x)$-binomial coefficients, preprint.
in the planes, Proc. Nat. Acad. Sci. U.S.A., 61

and pseudo-random numbers, Bull. Amer. Math.

bound, this Journal, 10 (1981), pp. 371-382.
ringer-Verlag, New York, 1977.
bra, vol. 2, Van Nostrand, Princeton, NJ, 1960.

SIAM J. COMPUT.
Vol. 17, No. 2, April 1988

© 1988 Society for Industrial and Applied Mathematics
010

THE DISCRETE LOGARITHM HIDES $O(\log n)$ BITS*

DOUGLAS L. LONG† AND AVI WIGDERSON‡

Abstract. The main result of this paper is that obtaining any information about the $O(\log |p|)$ "most significant" bits of x, given $g^x \pmod p$, even with a tiny advantage over guessing, is equivalent to computing discrete logarithms mod p.

Key words. discrete logarithm, one-way functions, secure transactions, bit security, pseudorandom bit generator, public-key cryptography

AMS(MOS) subject classifications. 68P25, 68Q15

1. Introduction. A function $f:[1, N] \to [1, N]$ is said to be one-way if computing $f(x)$ from x is easy, but computing x from $f(x)$ is hard. One-way functions are extremely attractive in cryptographic applications (see [5], [6], [16], [18], [19]). However, one should use them with care. The reason is that while x is hard to compute from $f(x)$, it may be possible to obtain *almost all* bits of x easily. It is clear, however, that some bits of x are hard to obtain.

In a key paper, Blum and Micali [5] introduced the notion of hiding a bit in a one-way function. A Boolean predicate $B:[1, N] \to \{0, 1\}$ is said to be hard for f if an oracle for $B(f(x))$ allows one to invert f easily. They prove that a certain Boolean predicate is hard for the discrete logarithm function, and show how to use that to construct good pseudorandom number generators.

Some natural questions arise in light of their work. Given a one-way function f:

1) What are its hard bits?
2) How many hard bits are there?
3) How many bits are hard simultaneously?

To explain what we mean by the last question, consider B, a hard Boolean predicate for f, and its complement, $\bar{B}$. Clearly, $\bar{B}$ is also hard for f. However, some information about the pair of bits $(B(f(x)), \bar{B}(f(x)))$ is available, namely the information that they are different. Therefore this two-bit predicate is not hard for f. We say that a k-bit predicate $B^k:[1, N] \to \{0, 1\}^k$ is hard for f if for *every* Boolean predicate $B:\{0, 1\}^k \to \{0, 1\}$, an oracle for $B(B^k(f(x)))$ will allow us to invert f easily. If such a B^k exists, we say that f hides k bits.

The fact that hiding only one bit suffices for many applications seems to reduce the motivation for the questions above. We would like to argue that this is not the case. From the practical point of view, the ability to hide k bits cuts the computation/communication per secure bit by a factor of k. One can flip k coins at a time or generate k bits at a time with a pseudorandom number generator without hurting security.

But what really motivates us is the theoretic point of view. Very little is known about one-way functions. In particular, a major open problem is whether they exist. We believe that discovering and studying the "hard core" (the collection of hard bits) of functions that are believed to be one-way may shed some light on this problem.

* Received by the editors September 17, 1985; accepted for publication (in revised form) June 24, 1986.

† Computer Science Department, Wellesley College, Wellesley, Massachusetts 02181. The work of this author was supported in part by Army Research Office grant DAAG29-80-K-0090 and a Garden State Fellowship from Princeton University, Princeton, New Jersey 08544.

‡ Mathematical Sciences Research Institute, Berkeley, California 94720. The work of this author was supported in part by an IBM Graduate Fellowship from Princeton University, Princeton, New Jersey 08544.

In this paper we focus our attention on the discrete logarithm function, which is widely believed to be one-way. Our main result states that the discrete log function hides $c \cdot \log |p|$ bits for any constant c, where p is the modulus and $|p|$ is the number of bits in the binary representation of p. This paper subsumes the results of [11] where we proved the same result using an additional assumption (see § 5 for details). Similar results for the case of the RSA and Rabin encryption functions were subsequently obtained by Alexi, Chor, Goldreich and Schnorr [3] and by Vazirani and Vazirani [17].

Results similar to those of [11] were obtained by Long [12], using similar techniques but a different set of bits. Recently, using a different technique (similar in spirit to that used in [17]), Peralta [13] showed that the discrete log function can be used for a pseudorandom bit generator which outputs $O(\log |p|)$ bits at time, a result which is equivalent to ours.

Section 2 states the main results of the paper. Section 3 contains some technical lemmas. In § 4 we extend the techniques of [5] to prove our main result for the case where the oracle is always correct. In § 5 we extend our results to the case where the oracle is more correct than incorrect. In § 6 we discuss an application of our work to pseudorandom bit generation. Section 7 concludes with an open problem.

2. Hiding bits. We start with some definitions. From this point on p will represent an n-bit prime, Z_p^* the group of units mod p, g a generator of Z_p^*, and k an integer such that $k \leqq n$.

DEFINITION 2.1. Suppose $z \in Z_p^*$. The **index of** z, index (z), is the unique $x \in Z_p^*$ such that $z \equiv g^x \pmod p$. x is also known as the **discrete logarithm of** z.

Blum and Micali [5] showed how to use the discrete logarithm to hide a single bit. They hide their bit by dividing the range $[1, p-1]$ into two equal-size intervals called the left and the right. A "0" bit is encoded by choosing x from the left and computing g^x. A "1" bit is encoded by choosing x from the right. They prove that even a small advantage in determining from g^x if x is in the left or the right can be used to solve the discrete logarithm problem. In fact, it is only necessary to be able to do this for x that are in the beginning of the left or beginning of the right. We can state this formally as the

t-Left-Right Problem. Given $g^x \in Z_p^*$, such that $x \in [1, (p-1)/t] \cup [(p-1)/2, (p-1)/2 + (p-1)/t]$, determine if $x \in [1, (p-1)/t]$ or $x \in [(p-1)/2, (p-1)/2 + (p-1)/t]$.

Blum and Micali's result can be stated as the following:

THEOREM 2.1 (Blum and Micali). *Suppose p is an n-bit prime and $\delta < \frac{1}{2}$. Suppose there exists an oracle which solves the t-left-right problem correctly with probability at least $1-\delta$. Then there exists a Las Vegas algorithm for the discrete logarithm which makes calls to this oracle and has expected running time $O(t \cdot n^3 (1-\delta)^{-n})$.*

We generalize Blum and Micali's method of hiding 1 bit to k bits by dividing the interval $[1, p-1]$ into 2^k consecutive, roughly equal, intervals $I_i^k = [L_i^k, U_i^k]$ for i from 0 to $2^k - 1$, where $L_i^k = \lfloor ((p-1)/2^k) i \rfloor + 1$, and $U_i^k = \lfloor ((p-1)/2^k)(i+1) \rfloor$. We will call this a partition at the kth level. For $x \in [1, p-1]$ let $I^k(x)$ be the interval which contains x, i.e., the integer i such that $x \in I_i^k$. Let $R^k = \lfloor (p-1)/2^k \rfloor$. The lower indices of I_i^k are computed modulo 2^k, i.e., I_{r+s}^k means $I_{r+s \pmod{2^k}}^k$.

In order to hide the k bits representing the integer i, $0 \leqq i < 2^k$, choose a random $x \in I_i^k$ and compute $g^x \pmod p$. An integer $z \in Z_p^*$ hides the bits which represent the interval of its index.

We claim that these bits are securely hidden. In order to justify this claim we must show that any advantage in determining not only the bits, but any partial information

about the bits, can be used to solve th
information about the bits means e
bits. We will call such a predicate a

DEFINITION 2.2. A **decision on**
If we consider d as a function from
extend d to all integers by defining

We claim any advantage in eval
the discrete log problem. First we f

DEFINITION 2.3. Suppose d is a
$(b_1, b_2, \cdots, b_k)$ such that $d(b_1, b_2,$

Without loss of generality we a

DEFINITION 2.4. Suppose d is a
is an α-**oracle** for d if $\theta_d(g^x) = d(I^k$
is known as a **perfect oracle** (i.e., an
has an ε-**advantage** in evaluating d

An oracle with an ε-advantage

We can now state the main r
evaluting any decision on k bits ca
state this formally as the following

THEOREM 2.2. *For any k, let d*
oracle for d. Then, using θ_d, it is possib
problem correctly with probability 1
conversion algorithm are polynomia
polynomial in n.)

COROLLARY 2.1. *Suppose $k = C$*
an ε-advantage oracle for d, then the
that runs in time polynomial in n an

Proof. Follows from Theorem

3. Properties of partitions. In t
outlining some of the properties of
on first reading.

The following lemma shows th

LEMMA 3.1. *For $0 \leqq i \leqq 2^k - 1$,*

Proof.

$$|I_i^k| = U_i^k - L_i^k +$$

Let $0 \leqq r_i < 1$ and $0 \leqq s_i < 1$ be the fra

$$|I_i^k| = \frac{p-1}{2^k} i + \frac{p-1}{2^k}$$

But $|s_i - r_i| < 1$ and $\lfloor (p-1)/2^k \rfloor \leqq ($
$\lfloor (p-1)/2^k \rfloor$ or $\lfloor (p-1)/2^k \rfloor + 1$. □

The following lemma shows tha
at the $(k-1)$st level.

LEMMA 3.2. $L_i^{k-1} = L_{2i}^k$, $U_i^{k-1} =$

Proof. Follows immediately fr

Throughout this paper, given
$w \leftarrow z g^{iR^k}$, $w \leftarrow \sqrt{z}$ (z is a quadratic re
and $w \leftarrow z g^{-1}$ (z is a nonresidue). T

discrete logarithm function, which is
t states that the discrete log function
is the modulus and $|p|$ is the number
ber subsumes the results of [11] where
sumption (see § 5 for details). Similar
ryption functions were subsequently
[3] and by Vazirani and Vazirani [17].
by Long [12], using similar techniques
ent technique (similar in spirit to that
crete log function can be used for a
og $|p|$) bits at time, a result which is

er. Section 3 contains some technical
o prove our main result for the case
end our results to the case where the
iscuss an application of our work to
es with an open problem.

s. From this point on p will represent
a generator of Z_p^*, and k an integer

of z, index (z), is the unique $x \in Z_p^*$
discrete logarithm of z.
e discrete logarithm to hide a single
$p-1]$ into two equal-size intervals
ed by choosing x from the left and
x from the right. They prove that
f x is in the left or the right can be
fact, it is only necessary to be able
ft or beginning of the right. We can

that $x \in [1, (p-1)/t] \cup [(p-1)/2,$
$)/t]$ or $x \in [(p-1)/2, (p-1)/2+$

following:
is an n-bit prime and $\delta < \frac{1}{2}$. *Suppose*
problem correctly with probability at
m for the discrete logarithm which
time $O(t \cdot n^3(1-\delta)^{-n})$.
iding 1 bit to k bits by dividing the
l, intervals $I_i^k = [L_i^k, U_i^k]$ for i from
$= \lfloor ((p-1)/2^k)(i+1) \rfloor$. We will call
$r^k(x)$ be the interval which contains
$-1)/2^k \rfloor$. The lower indices of I_i^k
$^k)$.
eger i, $0 \leqq i < 2^k$, choose a random
hides the bits which represent the

order to justify this claim we must
e bits, but any partial information

about the bits, can be used to solve the discrete logarithm problem. Determining partial information about the bits means evaluating some nontrivial Boolean predicate on k bits. We will call such a predicate a decision.

DEFINITION 2.2. A **decision on k bits** is a nonconstant function $d: \{0,1\}^k \to \{0,1\}$. If we consider d as a function from the integers $0, \cdots, 2^k - 1$ into $\{0,1\}$ then we can extend d to all integers by defining $d(i) = d(i \pmod{2^k})$.

We claim any advantage in evaluating any decision on k bits can be used to solve the discrete log problem. First we formalize what we mean by an advantage.

DEFINITION 2.3. Suppose d is a decision on k bits. Let f_d = the fraction of k-tuples $(b_1, b_2, \cdots, b_k)$ such that $d(b_1, b_2, \cdots, b_k) = 1$.

Without loss of generality we assume that $f_d \geqq \frac{1}{2}$.

DEFINITION 2.4. Suppose d is a decision on k bits. A function $\theta_d: [1, p-1] \to \{0,1\}$ is an **α-oracle** for d if $\theta_d(g^x) = d(I^k(x))$ for the fraction α of $x \in [1, p-1]$. A 1-oracle is known as a **perfect oracle** (i.e., an oracle that never makes a mistake). An α-oracle has an **ε-advantage** in evaluating d if $\alpha = f_d + \varepsilon$.

An oracle with an ε-advantage does ε better than an oracle that always guesses 1.

We can now state the main result of this paper. Any nontrivial advantage in evaluting any decision on k bits can be used to solve the discrete log problem. We state this formally as the following theorem and corollary.

THEOREM 2.2. *For any k, let d be a decision on k bits. Let θ_d be an ε-advantage oracle for d. Then, using θ_d, it is possible to construct an algorithm that solves the s-left-right problem correctly with probability $1-\delta$. The parameter s and the running time of the conversion algorithm are polynomial in ε^{-1}, δ^{-1}, n, and 2^k. (Note that ε^{-1} may be polynomial in n.)*

COROLLARY 2.1. *Suppose $k = O(\log n)$. Let d be a decision on k bits. If there exists an ε-advantage oracle for d, then there exists a Las Vegas algorithm for the discrete log that runs in time polynomial in n and ε^{-1}.*

Proof. Follows from Theorem 2.1 and Theorem 2.2. □

3. Properties of partitions. In this section we present a series of technical lemmas outlining some of the properties of partitions. The reader may wish to skip the proofs on first reading.

The following lemma shows that the intervals I_i^k are of roughly the same size.

LEMMA 3.1. *For $0 \leqq i \leqq 2^k - 1$, $|I_i^k| = R^k$ or $R^k + 1$.*

Proof.

$$|I_i^k| = U_i^k - L_i^k + 1 = \left\lfloor \frac{p-1}{2^k}(i+1) \right\rfloor - \left\lfloor \frac{p-1}{2^k} \right\rfloor.$$

Let $0 \leqq r_i < 1$ and $0 \leqq s_i < 1$ be the fractional parts of the expressions in the floors. Then

$$|I_i^k| = \frac{p-1}{2^k} i + \frac{p-1}{2^k} - r_i - \left(\frac{p-1}{2^k} i - s_i \right) = \frac{p-1}{2^k} - r_i + s_i.$$

But $|s_i - r_i| < 1$ and $\lfloor (p-1)/2^k \rfloor \leqq (p-1)/2^k < \lfloor (p-1)/2^k \rfloor + 1$ so $|I_i^k|$ must be either $\lfloor (p-1)/2^k \rfloor$ or $\lfloor (p-1)/2^k \rfloor + 1$. □

The following lemma shows that the partition at the kth level refines the partition at the $(k-1)$st level.

LEMMA 3.2. $L_i^{k-1} = L_{2i}^k$, $U_i^{k-1} = U_{2i+1}^k$. *Consequently* $I_i^{k-1} = I_{2i}^k \cup I_{2i+1}^k$.

Proof. Follows immediately from the definition of I_i^k. □

Throughout this paper, given an element $z \in Z_p^*$, we will apply the operations $w \leftarrow z g^{iR^k}$, $w \leftarrow \sqrt{z}$ (z is a quadratic residue and we pick the root with the smaller index), and $w \leftarrow z g^{-1}$ (z is a nonresidue). The effect of these operations on the index of z is

to shift it by a multiple of R^k, to divide it by two (when even), and to subtract one from it (when odd), respectively. We will call these operations *index operations* because they change the index in a specific way. The following lemmas show into which intervals the new index may fall.

LEMMA 3.3. *Let $x \in I_j^k$. Then for every $1 \leqq i \leqq 2^k - 1$*

(1) $x + iR^k \in I_{i+j}^k$, *or*

(2) $x + tR^k = L_{t+j}^k$ *for some $0 \leqq t < i$.*

Proof. By induction on i. If $i = 1$ then from Lemma 3.1 $x + R^k \in I_j^k$ if and only if $|I_j^k| = R^k + 1$ and $x = L_j^k$. If $i > 1$ then let $x' = x + (i-1)R^k$. If $x' \notin I_{i+j-1}^k$ then case (2) holds for x' and therefore for x. Otherwise $x' \in I_{i+j-1}^k$ so $x + iR^k \equiv x' + R^k \pmod{p-1}$ and the lemma follows by the same argument as for $i = 1$. □

LEMMA 3.4. *If $2x \in I_j^{k-1}$, then*

(1) $x \in I_j^k$ *or* $x \in I_{j+2^{k-1}}^k$, *or*

(2) $2x = L_j^{k-1}$ *or* $2x = U_j^{k-1}$.

Proof. Suppose $x \leqq (p-1)/2$. (The case $x > (p-1)/2$ is analogous.)

Suppose $2x > L_{2j}^k = L_j^{k-1}$. Then $1 + \lfloor ((p-1)/2^k)2j \rfloor < 2x$. So $((p-1)/2^k)2j < 2x$ and $\lfloor ((p-1)/2^k)j \rfloor \leqq ((p-1)/2^k)j < x$. Therefore $L_j^k = 1 + \lfloor ((p-1)/2^k)j \rfloor \leqq x$.

Now suppose $2x < U_{2j+1}^k = U_j^{k-1}$. Then $2x < \lfloor ((p-1)/2^k)((2j+1)+1) \rfloor \leqq ((p-1)/2^k)2(j+1)$. Thus $x \leqq \lfloor ((p-1)/2^k)(j+1) \rfloor = U_j^k$. □

LEMMA 3.5. *If $x \in I_j^k$ (and x is odd), then either*

(1) $(x-1) \in I_j^k$, *or*

(2) $x = L_j^k$.

Proof. Follows immediately from the definition. □

The index operations given above are well behaved (in the sense that they change the index of z in a specified way) except when $z = g^{L_j^k}$ or $z = g^{U_j^k}$. We call these points the *boundary points*. We can easily determine if z is a boundary point (and thereby compute index (z)) by maintaining a sorted list of these values. There are $O(2^k)$ values to maintain so sorting will cost $O(k2^k)$ time and each query $O(k)$ time. If z is not a boundary point then the index operations will result in the index of z falling in the intervals given in case (1) of Lemmas 3.3, 3.4 or 3.5. Henceforth, we will assume that whenever an index operation is performed on z, a test for case (2) of Lemmas 3.3, 3.4 or 3.5 is performed by checking to see if z is a boundary point.

4. Oracles which are always correct. This section considers the case of a perfect oracle. In the next section we will consider what happens if the oracle is not perfect.

We will prove the following

THEOREM 4.1. *Let $k = O(\log n)$. If there exists a decision d on k bits, and a perfect oracle for d, θ_d, then it is possible to solve the 2^k-left-right problem. This can be done in time polynomial in n, including time for the boundary tests.*

We first note the following facts about decisions.

Decisions repeat themselves at intervals of 2^k. Some decisions repeat themselves at smaller intervals.

DEFINITION 4.1. Let d be a decision on k bits. The **period of** d is the smallest r such that $d(i) = d(i+r)$ for all i.

The period of a decision is always a power of two.

LEMMA 4.1. *Let d be a decision on k bits and let r be the period of d. Then $r = 2^l$ where $0 < l \leqq k$.*

Proof. Immediate from the definition of period. □

LEMMA 4.2. *Let d be a decision on k bits with period 2^l. Then there exists an $j < 2^{l-1}$ such that $d(j) \neq d(j + 2^{l-1})$. Moreover, such a j can be found in $O(2^k)$ time by linear search.*

Proof. Immediate from the def
nonconstant. □

Theorem 2.1 says that solving t
discrete log. In the following we wi
any decision on k bits to solve the t-
the partition of the set of intervals
whether i is even or odd. The metho
of quadratic residues in even (odd)
level $j+1$ (except at interval boundar
residues in I_0^1 (the left half) are con
residues in I_1^1 (the right-half) are co

A decision gives us informatio
this information to information abo
as shown by procedure REDUCE a

For $z \in [1, p-1]$ and $0 \leqq l \leqq k$ d

```
Procedure REDUCE (z, l)
  w ← z
  for i = 1 to k − l do
    if w is a nonresidue then w ←
    w ← √w (either root)
  endfor
  return (w)
```

LEMMA 4.3. *For any $z \in [1, p-1$*
ated in time polynomial in n.

Proof. Since a quadratic nonres
in the above procedure can be perfo

We proceed by showing how to
2^l, and REDUCE to solve the 2^k-lef
g^x, if $x \in I_0^k$ or $x \in I_{2^{k-1}}^k$. If the period
that $d(0) = d(2^{k-1})$ so an oracle for
is possible to use REDUCE to get a

LEMMA 4.4. *Let $z \in [1, p-1]$ a*
$y = \text{index}(w)$. *If $I^k(x) \equiv 0 \pmod{2^k}$*
then $I^k(y) \equiv 2^{l-1} \pmod{2^l}$.

Proof. Suppose $I^k(x) \equiv 0 \pmod{2}$

The case $l = k$ is clear. Ass
REDUCE $(z, l+1)$ and $y' = \text{index}$ (
(This operation does not change I^k(
(For example, $w = \text{REDUCE}(z, l)$ a
$I^k(y') \equiv 0 \pmod{2^{l+1}}$. Let $I^k(y') = 2$
$I_{j+2^{k-1}}^k$. In other words, $I^k(y) = I$
$0 \pmod{2^l}$. □

Now we are ready to prove The

Proof of Theorem 4.1. Suppose
d on k bits. Let the period of d be

We will construct an algorithm
that $x = \text{index}(z) \in I_0^k \cup I_{2^{k-1}}^k$. Let w
4.4 $x \in I_0^k$ if $I^k(y) \equiv 0 \pmod{2^l}$. Other

Proof. Immediate from the definition of period and the fact that a decision is nonconstant. □

Theorem 2.1 says that solving the t-left-right problem is as hard as inverting the discrete log. In the following we will show how to use the information provided by any decision on k bits to solve the t-left-right problem. For every j, $1 \leqq j \leqq k$, consider the partition of the set of intervals $\{I_i^j\}$ into two types, even and odd, depending on whether i is even or odd. The method is based on the following observation. The roots of quadratic residues in even (odd) intervals at level j fall in even (odd) intervals at level $j+1$ (except at interval boundaries). For example, note that the roots of quadratic residues in I_0^1 (the left half) are contained in I_0^2 and I_2^2, while the roots of quadratic residues in I_1^1 (the right-half) are contained in I_1^2 and I_3^2.

A decision gives us information about the intervals at some level. To translate this information to information about the first level we repeatedly take square roots as shown by procedure REDUCE and the following lemmas.

For $z \in [1, p-1]$ and $0 \leqq l \leqq k$ define the procedure REDUCE (z, l).

```
Procedure REDUCE (z, l)
  w ← z
  for i = 1 to k − l do
    if w is a nonresidue then w ← wg^{-1}
    w ← √w (either root)
  endfor
  return (w)
```

LEMMA 4.3. *For any $z \in [1, p-1]$ and $0 \leqq l \leqq k$, procedure* REDUCE *can be evaluated in time polynomial in n.*

Proof. Since a quadratic nonresidue, g, is known, the extraction of square roots in the above procedure can be performed in time $O(n^2)$ [1], [7]. □

We proceed by showing how to use an oracle for a decision on k bits, with period 2^l, and REDUCE to solve the 2^k-left-right problem. This amounts to deciding, given g^x, if $x \in I_0^k$ or $x \in I_{2^{k-1}}^k$. If the period, 2^l, is less than 2^k then it will always be the case that $d(0) = d(2^{k-1})$ so an oracle for such a decision is not of direct use. However, it is possible to use REDUCE to get around this problem.

LEMMA 4.4. *Let $z \in [1, p-1]$ and let $w =$* REDUCE (z, l). *Let $x =$* index (z) *and $y =$* index (w). *If $I^k(x) \equiv 0 \pmod{2^k}$ then $I^k(y) \equiv 0 \pmod{2^l}$. If $I^k(x) \equiv 2^{k-1} \pmod{2^k}$ then $I^k(y) \equiv 2^{l-1} \pmod{2^l}$.*

Proof. Suppose $I^k(x) \equiv 0 \pmod{2^k}$. (The other case is similar.)

The case $l = k$ is clear. Assume the lemma is true for $l+1$. Let $w' =$ REDUCE $(z, l+1)$ and $y' =$ index (w'). If y' is odd set $w' \leftarrow w' \cdot g^{-1}$ and $y' \leftarrow y' - 1$. (This operation does not change $I^k(y')$ by Lemma 3.5.) Let $w = \sqrt{w'}$ and $y =$ index (w). (For example, $w =$ REDUCE (z, l) and $y' \equiv 2y \pmod{p-1}$.) By the induction hypothesis $I^k(y') \equiv 0 \pmod{2^{l+1}}$. Let $I^k(y') = 2j$. Since $2y \in I_{2j}^k \subset I_j^{k-1}$, by Lemma 3.4 $y \in I_j^k$ or $I_{j+2^{k-1}}^k$. In other words, $I^k(y) = I^k(y')/2$ or $(I^k(y')/2) + 2^{k-1}$. Therefore $I^k(y) \equiv 0 \pmod{2^l}$. □

Now we are ready to prove Theorem 4.1.

Proof of Theorem 4.1. Suppose there exists a perfect oracle θ_d for some decision d on k bits. Let the period of d be 2^l. Choose a j such that $d(j) \neq d(j + 2^{l-1})$.

We will construct an algorithm for the 2^k-left-right problem. Suppose $z \in Z_p^*$ such that $x =$ index $(z) \in I_0^k \cup I_{2^{k-1}}^k$. Let $w =$ REDUCE (z, l) and $y =$ index (w). By Lemma 4.4 $x \in I_0^k$ if $I^k(y) \equiv 0 \pmod{2^l}$. Otherwise, $x \in I_{2^{k-1}}^k$. Let $w' = w \cdot g^{jR^k}$ and $y' =$ index (w').

Then $d(I^k(y')) = d(I^k(y+jR^k)) = d(I^k(y)+j)$. If $I^k(y) \equiv 0 (\mathrm{mod}\ 2^l)$ then $d(I^k(y')) = d(j)$. Otherwise $d(I^k(y')) = d(j+2^{l-1})$. Since $d(j) \neq d(j+2^{l-1})$ the query $\theta_d(w')$ will determine $I^k(y) (\mathrm{mod}\ 2^l)$ and thus $I^k(x) (\mathrm{mod}\ 2^k)$. □

5. Oracles which make mistakes. We now consider the case of a nonperfect oracle. For example, there exists an oracle that gives some partial information about the k bits, but it does not always give the correct answer.

Blum and Micali handled the case of a nonperfect oracle by using a random sampling technique which they called "Concentrating the Stochastic Advantage." They were able to use this technique to correctly determine the value of their predicate with high probability because they were able to sample over the entire range $[1, p-1]$. In order to extend these results to k bits using an oracle with an ε-advantage and the techniques developed in the previous section, it is necessary to be able to evaluate the value of the predicate correctly with high probability on a single interval I_i^k. Unfortunately, within an interval nothing is known about the proportion of x for which $\theta_d(g^x)$ gives the correct value. In [11], we assumed that the error in every interval was bounded. Here we make no such assumptions. It might be the case that θ_d gives the wrong answer for every x in some interval I_i^k. However, if this occurs there must be some other interval (or intervals) that have a higher percentage of correct answers. Many intervals can be sampled to obtain the necessary information.

As explained below, here we are faced with a continuous analogue of the discrete problem of § 4. In the previous section we used the Boolean values of a decision to extract information required to solve the left-right problem. We can interpret the values of a decision as a probability. For example, if $d(I^k(x)) = 1$ then the probability that $\theta_d(g^y) = 1$ for any $y \in I^k(x)$ is 1 since a perfect oracle always gives the same answer for all values within an interval. When the oracle makes mistakes the probability that the oracle answers 1 on a random value in an interval is no longer either 0 or 1, but could be any value in between. In the following we develop the techniques to extract the information needed to solve the left-right problem from this sequence of probabilities.

DEFINITION 5.1. For $0 \leqq i < 2^k$, let $q(i)$ = the fraction of $y \in I_i^k$ such that $\theta_d(g^y) = 1$.

If x is chosen randomly from I_i^k, then $q(i)$ is the probability that $\theta_d(g^x) = 1$. We will postpone discussion of how to measure the $q(i)$ and assume for the moment that given g^x we can measure $q(I^k(x))$.

We will also need the following definition.

DEFINITION 5.2. For $0 \leqq i < 2^k$, let $r(i)$ = the fraction of $y \in I_i^k$ such that $\theta_d(g^y) = d(i)$.

If x is chosen randomly from I_i^k, then $r(i)$ is the probability that $\theta_d(g^x) = d(i)$. Note that if $d(i) = 1$ then $r(i) = q(i)$ and otherwise $r(i) = 1 - q(i)$.

We are interested in certain families of subsequences of the $q(i)$. The indices of these subsequences are given below.

DEFINITION 5.3. Suppose $0 \leqq m \leqq k$ and $0 \leqq j < 2^m$. Let

$$Q_k(j, m) = (j, j+2^m, j+2 \cdot 2^m, \cdots, j+(2^{k-m}-1) \cdot 2^m).$$

The sequence $Q_k(0, 0)$ is denoted by Q_k. Also note that $i \in Q_k(j, m)$ if and only if $j \equiv i (\mathrm{mod}\ 2^m)$.

DEFINITION 5.4. The average value of the probabilities whose indices are given in each sequence $Q_k(j, m)$ is given by

$$EQ_k(j, m) = \frac{1}{|Q_k(j, m)|} \sum_{i=0}^{2^{k-m}-1} q(j+i \cdot 2^m).$$

The average $EQ_k(0, 0)$ is denoted b

In the discrete case a decision b
information from it. In the followin
analogue.

DEFINITION 5.5. Suppose $0 \leqq$
$|Q_k(j, m)| = 1$), then $Q_k(j, m)$ is γ-
only if $|EQ_k(j, m+1) - EQ_k(j+2^m,$
are each γ-constant.

The following lemma shows th
to the average.

LEMMA 5.1. *If $Q_k(j, l)$ is γ-cor*

$$|EQ_k$$

Proof. The proof is by backwa

If $l = k$ then $Q_k(j, k) = (j)$ and

Assume the lemma holds for l-
$1) - EQ_k(j+2^l, l+1)| \leqq \gamma$ and $Q_k(j$
the induction hypothesis, for all $i \in$

$$|EQ_k(j, l$$

and for all $i \in Q_k(j+2^l, l+1)$

$$|EQ_k(j+2^l,$$

But

$$EQ_k(j, l) = \frac{E}{}$$

Therefore, for all $i \in Q_k(j, l)$

$$|EQ_k(j, l) - q$$

In § 4 we were able to obtain
section the analogous situation wi
the following sense.

LEMMA 5.2. *If θ_d is an ε-adva*
constant.

Proof. Suppose Q_k is (ε/k)-c
$(\varepsilon/2)$. Thus $q(i) \leqq EQ_k + (\varepsilon/2)$ and
ber of k-tuples $(b_1, b_2, \cdots, b_k)$ su
then $r(i) = q(i)$ and if $d(i) = 0$ then
(i_1, i_2) such that $r(i_1) = q(i_1)$ an
$EQ_k + (\varepsilon/2) + 1 - EQ_k + (\varepsilon/2) = 1 +$
are not among these pairs, $r(i) \leqq 1$

The average $EQ_k(0,0)$ is denoted by EQ_k.

In the discrete case a decision had to be nonconstant in order to be able to extract information from it. In the following recursive definition we formalize the continuous analogue.

DEFINITION 5.5. Suppose $0 \leqq \gamma \leqq 1$, $0 \leqq j < 2^k$, and $0 \leqq m \leqq k$. For $m = k$ (i.e., $|Q_k(j,m)| = 1$), then $Q_k(j,m)$ is γ-**constant**. For $m < k$, $Q_k(j,m)$ is γ-**constant** if and only if $|EQ_k(j,m+1) - EQ_k(j+2^m, m+1)| \leqq \gamma$ and $Q_k(j,m+1)$ and $Q_k(j+2^m, m+1)$ are each γ-constant.

The following lemma shows that in a γ-constant subsequence all values are close to the average.

LEMMA 5.1. *If $Q_k(j,l)$ is γ-constant then for all $i \in Q_k(j,l)$*

$$|EQ_k(j,l) - q(i)| \leqq \frac{k-l}{2}\gamma.$$

Proof. The proof is by backwards induction on l.

If $l = k$ then $Q_k(j,k) = (j)$ and so $EQ_k(j,k) = q(j)$.

Assume the lemma holds for $l+1$. Suppose $Q_k(j,l)$ is γ-constant. Then $|EQ_k(j,l+1) - EQ_k(j+2^l, l+1)| \leqq \gamma$ and $Q_k(j,l+1)$ and $Q_k(j+2^l, l+1)$ are both γ-constant. By the induction hypothesis, for all $i \in Q_k(j,l+1)$

$$|EQ_k(j,l+1) - q(i)| \leqq \frac{k-(l+1)}{2}\gamma$$

and for all $i \in Q_k(j+2^l, l+1)$

$$|EQ_k(j+2^l, l+1) - q(i)| \leqq \frac{k-(l+1)}{2}\gamma.$$

But

$$EQ_k(j,l) = \frac{EQ_k(j,l+1) + EQ_k(j+2^l, l+1)}{2}.$$

Therefore, for all $i \in Q_k(j,l)$

$$|EQ_k(j,l) - q(i)| \leqq \frac{\gamma}{2} + \frac{k-(l+1)}{2}\gamma = \frac{k-l}{2}\gamma. \qquad \square$$

In § 4 we were able to obtain information from a nonconstant decision. In this section the analogous situation will be that the sequence Q_k must be nonconstant in the following sense.

LEMMA 5.2. *If θ_d is an ε-advantage oracle for a decision d then Q_k is not (ε/k)-constant.*

Proof. Suppose Q_k is (ε/k)-constant. By Lemma 5.1, for $0 \leqq i < 2^k$, $|q(i) - EQ_k| \leqq (\varepsilon/2)$. Thus $q(i) \leqq EQ_k + (\varepsilon/2)$ and $1 - q(i) \leqq 1 - EQ_k + (\varepsilon/2)$. Let $t = 2^k \cdot f_d$ = the number of k-tuples $(b_1, b_2, \cdots, b_k)$ such that $d(b_1, b_2, \cdots, b_k) = 1$. Recall that if $d(i) = 1$ then $r(i) = q(i)$ and if $d(i) = 0$ then $r(i) = 1 - q(i)$. Hence, there are $2^k - t$ disjoint pairs (i_1, i_2) such that $r(i_1) = q(i_1)$ and $r(i_2) = 1 - q(i_2)$. For these pairs $r(i_1) + r(i_2) \leqq EQ_k + (\varepsilon/2) + 1 - EQ_k + (\varepsilon/2) = 1 + \varepsilon$. For each of the other $2 \cdot t - 2^k$ numbers i which are not among these pairs, $r(i) \leqq 1$. Since the oracle gives the correct answer for the

fraction $(1/2^k)\sum r(i)$ we have

$$f_d+\varepsilon=\frac{1}{2^k}\sum_{i=0}^{2^k-1} r(i)\leqq\frac{1}{2^k}[(2^k-t)(1+\varepsilon)+2\cdot t-2^k]$$

$$=\frac{1}{2^k}[t-t\varepsilon+2^k\varepsilon]<\frac{1}{2^k}[t+2^k\varepsilon]=\frac{t}{2^k}+\varepsilon=f_d+\varepsilon$$

which is a contradiction. Therefore, Q_k is not (ε/k)-constant. □

Lemma 4.2 showed that a nonconstant decision must have a pair of distinguishable values, i.e., two values that differ by a power of two where the decision is different. The following corollary is the analogue of Lemma 4.2. It shows that there exists two indices that differ by a power of two and distinguish between two subsequences of the sequence Q_k.

COROLLARY 5.1. *If θ_d is an ε-advantage oracle for a decision d then there exists numbers j and l such that $0<l\leqq k$, $j<2^{l-1}$ and*

$$|EQ_k(j,l)-EQ_k(j+2^{l-1},l)|>\frac{\varepsilon}{k}.$$

Proof. Follows immediately from Lemma 5.2 and Definition 5.5. □

We still must answer the questions: How do we measure $q(i)$ given i and how do we measure $q(I^k(x))$ given g^x? The answer to these questions is that it is not possible to measure these values exactly. However we can make high precision estimates of the values by using the technique of "Concentrating the Stochastic Advantage" developed by Blum and Micali [5].

This technique is based on the weak law of large numbers. We summarize it here as the

SAMPLING LEMMA. *Suppose $0\leqq\phi<\frac{1}{2}$ and $0\leqq\delta<\frac{1}{2}$. Let* trials $(\phi,\delta)=9/16\delta\phi^2$ *and $M=$* trials (ϕ,δ). *Let Y_i be the outcome of an independent random trial that has outcome 1 with probability α and 0 otherwise. Suppose M trials are made and the results are recorded as follows. With probability $1-1/M$ let $X_i=Y_i$ and with probability $1/M$ let the value of X_i be set by an adversary. This latter case is known as a defective trial. Let $S_M=\sum_{i=1}^{M}X_i$. Then with probability at least $1-\delta$*

$$\left|\frac{S_M+1}{M}-\alpha\right|\leqq\phi.$$

Note that if none of the trials are defective then S_M/M is an even better estimate of α.

We can use the Sampling Lemma to measure $q(i)$.

LEMMA 5.3. *Suppose $\phi>0$ and $\psi>0$. Given i, $q(i)$ can be measured in time polynomial in ϕ^{-1} and ψ^{-1} with error at most ϕ with probability at least $1-\psi$.*

Proof. Let $z=g^{L_i^k}$ and $M=$ trials (ϕ,ψ). Choose M random values $r_1, r_2, \cdots, r_M$ with uniform probability such that $0\leqq r_i\leqq R^k$. Let $z_i=z\cdot g^{r_i}$. The indices of the z_i will be uniformly distributed in I_i^k. Let $X_i=\theta_d(z_i)$. The probability that $X_i=1$ is $q(i)$. Let $\bar{q}(i)=(1/M)\sum_{i=1}^{M}X_i$. Then by the Sampling Lemma

$$|\bar{q}(i)-q(i)|\leqq\phi$$

with probability at least $1-\psi$. □

Unfortunately, if we are given g^x and wish to compute $q(I^k(x))$ the above sampling technique fails. However, under some circumstances the above technique will work.

LEMMA 5.4. *Suppose $\phi>0$ and $\psi>0$. Suppose $M=$* trials (ϕ,ψ). *Given g^x such that $L_i^k\leqq x\leqq L_i^k+(R^k/M)$, $q(I^k(x))$ can be measured with error at most ϕ with probability at least $1-\psi$.*

Proof. For any r_i such that 0
$1)/M)R^k<r_i<R^k$ then $x+r_i$ may not
defective. This occurs with probability
then by the Sampling Lemma

$$|\tilde{q}(i)$$

with probability at least $1-\psi$. □

We can use the above sampling tec
$EQ_k(j+2^{k-l},l)$ with high probability.

LEMMA 5.5. *Suppose j and l are as*
let $M=$ trials $(\phi,\delta/2^{k-l})$. *Let $m=I^k$*
$L_m^k+(R^k/M)$. Then given g^r it is possib
with probability at least $1-\delta$.

Proof. For each interval m, $m+2^l$,
to estimate the corresponding prob
$\sum_{i=0}^{2^{k-l-1}}\tilde{q}(m+i\cdot 2^l)$. Let $E=(1/2^{k-l})\sum_i^2$

$$\overline{EQ}_k(j,l)=\overline{|Q_k}$$

and $\overline{EQ}_k(j+2^{l-1},l)$ which is defined sim
5.3. The probability that

$$|\overline{EQ}_k(j,l)-$$

is at least $(1-(\delta/2^{k-l}))^{2^{k-l}}>1-\delta$. Simi

$$|\tilde{E}-$$

is at least $1-\delta$. If $\phi=(1/2^{k-l})(\varepsilon/8k$
$|\tilde{E}-\overline{EQ}_k(j,l)|\leqq\varepsilon/4k$ or $|\tilde{E}-\overline{EQ}_k(j+$
these will hold with probability at least

We are now ready to prove Theor

Proof of Theorem 2.2. Let $s=$ trial
an oracle for the s-left-right problem. T
Theorem 4.1 except that Lemma 5.5 i
$2^{l-1},l)$. The number of samples requir
δ^{-1}, ε^{-1}, and 2^k. The number of values
are polynomial in n so the time bound

6. Pseudorandom bit generation. V
formalized using the notation of [5] an

It is natural at this point to extend
to output k bits per step. If the current
the k bits hidden by it, namely $I^k(x)$
ensured if for every i, given $b_1b_2\cdots b_i$
as computing discrete logarithms.

This definition of "hard bits" seem
concerned with computing any Boolea
between the two definitions was establi
result holds for a wide class of generat

THEOREM 6.1 (Vazirani and Vazira
is secure if and only if for every $I\subset\{1,2,$
the exclusive or of b_i, $i\in I$, is equivalent

Proof. For any r_i such that $0 \leqq r_i \leqq ((M-1)/M)R^k$, $x + r_i \in I_i^k$. If $((M-1)/M)R^k < r_i < R^k$ then $x + r_i$ may not be in I_i^k. In this latter case the trial may be defective. This occurs with probability $1/M$. If we define $\tilde{q}(i) = (1/M)[1 + \sum_{i=1}^{M} X_i]$ then by the Sampling Lemma

$$|\tilde{q}(i) - q(i)| \leqq \phi$$

with probability at least $1 - \psi$. □

We can use the above sampling techniques to distinguish between $EQ_k(j, l)$ and $EQ_k(j + 2^{k-l}, l)$ with high probability.

LEMMA 5.5. *Suppose j and l are as in Corollary 5.1. Suppose $\delta > 0$ and $\phi > 0$ and let $M = \text{trials}(\phi, \delta/2^{k-l})$. Let $m = I^k(y)$. Suppose $m \equiv j \pmod{2^{l-1}}$ and $L_i^k \leqq y \leqq L_m^k + (R^k/M)$. Then given g^y it is possible to distinguish $EQ_k(j, l)$ from $EQ_k(j + 2^{l-1}, l)$ with probability at least $1 - \delta$.*

Proof. For each interval m, $m + 2^l$, $m + 2 \cdot 2^l$, $\cdots$, $m + (2^{k-l} - 1)2^l$ use Lemma 5.4 to estimate the corresponding probability $q(m + i \cdot 2^l)$ and let $\tilde{E} = (1/2^{k-l}) \times \sum_{i=0}^{2^{k-l}-1} \tilde{q}(m + i \cdot 2^l)$. Let $E = (1/2^{k-l}) \sum_{i=0}^{2^{k-l}-1} q(m + i \cdot 2^l)$. We must compare $\tilde{E}$ with

$$\overline{EQ}_k(j, l) = \frac{1}{|Q_k(j, l)|} \sum_{i=0}^{2^{k-l}-1} \bar{q}(j + i \cdot 2^l)$$

and $\overline{EQ}_k(j + 2^{l-1}, l)$ which is defined similarly. These values are estimated using Lemma 5.3. The probability that

$$|\overline{EQ}_k(j, l) - EQ_k(j, l)| \leqq 2^{k-l}\phi$$

is at least $(1 - (\delta/2^{k-l}))^{2^{k-l}} > 1 - \delta$. Similarly the probability that

$$|\tilde{E} - E| \leqq 2^{k-l}\phi$$

is at least $1 - \delta$. If $\phi = (1/2^{k-l})(\varepsilon/8k)$ then with probability at least $1 - \delta$ either $|\tilde{E} - \overline{EQ}_k(j, l)| \leqq \varepsilon/4k$ or $|\tilde{E} - \overline{EQ}_k(j + 2^{l-1}, l)| \leqq \varepsilon/4k$. By Corollary 5.1 only one of these will hold with probability at least $1 - \delta$. □

We are now ready to prove Theorem 2.2.

Proof of Theorem 2.2. Let $s = \text{trials}((1/2^{k-l})(\varepsilon/8k), \delta/2^{k-l})$. We will construct an oracle for the s-left-right problem. The construction is the same as in the proof of Theorem 4.1 except that Lemma 5.5 is used to distinguish $EQ_k(j, l)$ from $EQ_k(j + 2^{l-1}, l)$. The number of samples required to estimate $\bar{q}(i)$ and $\tilde{q}(i)$ is polynomial in δ^{-1}, ε^{-1}, and 2^k. The number of values estimated is $O(2^k)$ and all other calculations are polynomial in n so the time bound follows. □

6. Pseudorandom bit generation. We keep this section informal. It can be easily formalized using the notation of [5] and [17].

It is natural at this point to extend the Blum-Micali pseudorandom bit generator to output k bits per step. If the current value in the generator is g^x, then it will output the k bits hidden by it, namely $I^k(x) = b_1 b_2 \cdots b_k$. The security of this generator is ensured if for every i, given $b_1 b_2 \cdots b_i$, guessing b_{i+1} with an ε-advantage is as hard as computing discrete logarithms.

This definition of "hard bits" seems to differ from the one we give in § 2. We are concerned with computing any Boolean function of the b_i's. However, a connection between the two definitions was established in [17], at least when $k = O(\log n)$. Their result holds for a wide class of generators, but we state it here only for the one above.

THEOREM 6.1 (Vazirani and Vazirani). *Assume $k = O(\log n)$. The generator above is secure if and only if for every $I \subset \{1, 2, \cdots, k\}$, obtaining any partial information about the exclusive or of b_i, $i \in I$, is equivalent to computing the discrete log.*

As Theorem 2.2 states that any decision on k bits, including all these exclusive or's, are hard, it implies the security of this generator.

7. Beyond $O(\log n)$ bits. The problem of showing that the discrete log (or any other function) hides asymptotically more than $O(\log n)$ bits seems to us a fundamental question. In our proof of Theorem 2.2 the factor 2^k comes into the complexity analysis in three different places, and we have no way of circumventing even one of them. A similar situation occurs in [3], [13] and [17]. We strongly believe that any improvement will involve totally new techniques.

REFERENCES

[1] L. ADLEMAN, K. MANDERS AND G. MILLER, *On taking roots in finite fields*, Proc. 18th IEEE Symposium on the Foundations of Computer Science, 1977, pp. 175–178.

[2] L. ADLEMAN, *A subexponential algorithm for the discrete logarithm problem with applications to cryptography*, Proc. 20th IEEE Symposium on the Foundations of Computer Science, 1979, pp. 55–60.

[3] W. ALEXI, B. CHOR, O. GOLDREICH AND C. P. SCHNORR, RSA/*Rabin bits are* $\frac{1}{2}+1/\text{poly}(\log n)$ *secure*, Proc. 25th IEEE Symposium on the Foundations of Computer Science, 1984, pp. 449–457.

[4] E. BERLEKAMP, *Factoring polynomials over large finite fields*, Math. Comput., 24, (1970), pp. 713–735.

[5] M. BLUM AND S. MICALI, *How to generate cryptographically strong sequences of pseudorandom bits*, Proc. 23rd IEEE Symposium on the Foundations of Computer Science, 1982, pp. 112–117.

[6] W. DIFFE AND M. HELLMANN, *New directions in cryptography*, IEEE Trans. Inform. Theory IT-22, 6 (1976), pp. 644–654.

[7] J. FINN, *Probabilistic methods in number-theoretic algorithms and digital signature schemes*, Ph.D. dissertation, Princeton Univ., Princeton, NJ, June 1982.

[8] S. GOLDWASSER AND S. MICALI, *Probabilistic encryption and how to play mental poker keeping secret all partial information*, Proc. 14th ACM Symposium on the Theory of Computing, 1982, pp. 365–377.

[9] K. IRELAND AND M. ROSEN, *Elements of Number Theory*, Bogden and Quigley, New York, 1972.

[10] R. LIPTON, *How to cheat at mental poker*, 1979, unpublished manuscript.

[11] D. LONG AND A. WIGDERSON, *How discrete is the discrete log?*, Proc. 15th ACM Symposium on the Theory of Computing, 1983, pp. 413–420.

[12] D. LONG, *The security of bits in the discrete logarithm*, Ph.D. dissertation, Princeton Univ., Princeton, NJ, October 1983.

[13] R. PERALTA, *Simultaneous security of bits in the discrete log*, 1984, unpublished manuscript.

[14] S. POHLIG AND M. HELLMAN, *An improved algorithm for computing logarithms over GF(p) and its cryptographic significance*, IEEE Trans. Inform. Theory IT-24, 1 (1978), pp. 106–110.

[15] M. RABIN, *Probabilistic algorithms in finite fields*, this Journal, 9 (1980), pp. 273–280.

[16] A. SHAMIR, R. RIVEST AND L. ADLEMAN, *Mental Poker*, MIT Technical Report, February 1979.

[17] U. VAZIRANI AND V. VAZIRANI, *Efficient and secure pseudorandom number generation*, Proc. 25th Symposium on the Foundations of Computer Science, 1984, pp. 458–463.

[18] A. YAO, *Theory and applications of trapdoor functions*, Proc. 23rd Symposium on the Foundations of Computer Science, 1982, pp. 80–91.

[19] ——— *Protocols for secure computations*, Proc. 23rd Symposium on the Foundations of Computer Science, 1982, pp. 160–164.

SIAM J. COMPUT.
Vol. 17, No. 2, April 1988

HOW TO CONSTRUCT
FROM PSEU

MICHAEL LU

Abstract. We show how to efficiently c
a pseudorandom function generator. Goldr
tions," Proc. 25th Annual Symposium on Fo
the notion of a pseudorandom function ge
function generator from a pseudorandom b
Data Encryption Standard for our constructi
bit generator can be used to construct a b
plaintext attack, which is one of the strong

Key words. cryptography, pseudorando

AMS(MOS) subject classifications. 68F

1. Introduction. The main resu
ing a pseudorandom invertible per
generator. The question of whethe
first posed by Goldreich et al. [GC
pseudorandom bit generator can b
cryptosystem which is provably se
of the strongest possible attacks kn
will be other applications of pse
cryptographic protocols. Before de
questions which motivated our wo
questions.

The field of cryptography has c
years ago, cryptography was an a
based on clever ad hoc ideas. The
cleverness of the designer; a crypto
many cases the cryptographer only
design rules were recognized as play
the apparent increase in security ob
the folklore, although there were
foundations of cryptography were
framework for talking about a cryp
no formal way to either confirm or

More recently, researchers hav
and security and today cryptograph
Many cryptosystems whose security
out of this research. However, the s
is still based on the old folklore.
formalize and to analyze some of t

* Received by the editors November 22,

† Department of Computer Science, U
this author was supported by Natural Scien
and University of Toronto grant 3-370-126-

‡ Department of Computer Science, U
this author was supported by Natural Scien