Trapdoor Pseudo-random Number Generators, with Applications to Protocol Design.

*Umesh V. Vazirani**
Vijay V. Vazirani

University of California
Berkeley, CA 94720.

Abstract: We define the class of *trapdoor* pseudo-random number generators, and introduce a new technique for using these in cryptography. As an application for this technique, we present a provably secure protocol for *One-Bit Disclosures* i.e. for giving a one-bit message in exchange for receipt.

1. Introduction

The notion of a **deterministic random** number generator is certainly a contradiction. Yet the cryptographically secure pseudo-random number generators (Shamir [8], Blum and Micali [4], Yao [10], Blum, Blum and Shub [1]) have come close to achieving these two contradictory properties. More precisely:

> 1) Given a seed, there is an efficient **deterministic** algorithm for generating the pseudo-random sequence.
>
> 2) The pseudo-random sequence generated using a random seed is indistinguishable by polynomial time statistical tests from a truly **random** sequence.

The one-time pad [1] is, to our knowledge, the only application in which this powerful combination of properties has been used. In fact, so far pseudo-random number generator have been viewed primarily as **amplifiers of randomness** (i.e. given a *short* random seed as input, the generator outputs a *long* sequence that 'looks' random).

This research was supported by NSF-grant 82-04506.
* Supported also by IBM Graduate Fellowship.

In this paper, we define a special class of pseudo-random number generators, which we call **trapdoor generators.** Trapdoor generators are somewhat analogous to trapdoor functions: the knowledge of a secret key allows one to efficiently predict the pseudo-random sequence; however, without knowledge of the secret key, the sequence cannot be distinguished from a truly random sequence (we shall state this more precisely in Section 6). The $x^2 \bmod N$ generator of Blum, Blum and Shub [1] is an example of a trapdoor generator. We introduce a *new technique* to exploit the deterministic-random properties of trapdoor generators, and use it to design a provably secure protocol for **One-Bit Disclosures.** This technique appears to be fairly general and powerful, so we expect it to have more applications in cryptography.

In [4] Blum and Rabin describe a protocol for sending **certified mail.** Briefly, certified mail is mail in exchange for a receipt. Certified mail may be used for disclosures, e.g. when disclosing information about his latest invention, the inventor would at least like to get a receipt that he did so. Consider the following situation:

> Alice wants to find out from Bob, who is an expert siesmologist, whether there will be an earthquake in San Francisco in 1984. Bob wishes to send her this information in exchange for a receipt.

Notice that the crucial part of this message is a **single bit.** There are several ways in which a single bit can be sent using certified mail. e.g. as the exclusive or of the bits of a suitably chosen message. The problem with such schemes is that even though certified mail protocol may be secure, Alice may be able to extract the one-bit message without giving Bob a receipt. Instead, we propose using a One-Bit Disclosure protocol.

0272-5428/83/0000/0023$01.00 © 1983 IEEE

One-bit disclosure has a somewhat different flavour than certified mail. In certified mail, the recipient has no idea what the message is going to be; whether it is going to be useful or not. However, if the message is only one bit, then the recipient knows in advance how useful the message will be.

The idea behind our protocol is that Bob gives Alice information that makes her more and more certain about his bit. In exchange he accumulates receipts about this information. This may be thought of as sending a **fraction of a bit** in exchange for a fraction of the receipt. We first implement a protocol for one-bit disclosures in the presence of a trusted intermediary. We then use pseudo-random number generators to convert this into a secure protocol without the need for an intermediary.

The idea of fractional bits has generated much interest recently. In [8], Luby, Micali and Rackoff describe a 'miraculous' protocol f›r exchanging one-bit secrets. Using several clever ideas they are able to ensure that the exchange is 'fair'. i.e. at every stage Alice has exactly as much information about Bob's bit as Bob has about Alice's. A suitable modification of their protocol can be used for one-bit disclosures. Our technique yields a simpler protocol for one-bit disclosures which requires fewer message exchanges. In another interesting paper, Tedrick [12] has a different way of viewing fractional bits, in the context of Blum's protocol for exchange of Secret Keys [2]. In Blum's protocol, Alice and Bob exchange their secret primes bit by bit along with proofs that neither is sending 'junk' bits. Tedrick views receiving each additional bit as cutting down the search space for the prime by half. He proposes that Alice could send a fraction of a bit by specifying for example that the next 3 bits of her prime are not 010. This information cuts down Bob's search space by only one eighth, thereby reducing his advantage, if he decides to stop before giving Alice the corresponding information about his secret prime.

2. One-Bit Disclosures with Trusted Intermediary.

Bob first reveals his one-bit message to the intermediary (who is trusted by both Bob and Alice). The intermediary flips a coin biased $\frac{1}{2} + \varepsilon$ towards Bob's bit (the bias is mutually decided on by Alice and Bob), and reports the flip to both Alice and Bob. In return, Alice gives Bob a receipt for this fractional information, and Bob checks the receipt. If Alice received m 1's and n 0's from Bob then she knows that Bob's bit is 1 with probability $1/(1 + k^{m-n})$, where $k = \frac{\frac{1}{2} - \varepsilon}{\frac{1}{2} + \varepsilon}$. We say that a receipt for these bits is **worth** $1/(1 + k^{m-n})$. Actually, Bob has receipt for all but possibly the last flip that Alice received. Therefore his receipt is worth at least $1/(1 + k^{m-n-sgn(m-n)})$,

$$\text{where } \operatorname{sgn}(x) = \begin{cases} 1 \text{ if } x>0. \\ -1 \text{ if } x<0. \\ 0 \text{ if } x=0. \end{cases}$$

Notice that without the trusted intermediary in this protocol, Bob may cheat, i.e. he may not report the actual flips of the coin, or he may use a coin with a different bias. In that case, Alice will misinterpret the information she receives from Bob, whereas Bob keeps getting Alice's receipts!

3. Removing the Intermediary.

In this section we show how the protocol of the previous section can be implemented without the intermediary, in an abstract setting. Suppose Alice could construct the following two coins:

1). **Without knowing** Bob's bit, Alice constructs a coin biased $\frac{1}{2} + \varepsilon$ towards either 0 or 1 (Alice does not know which, though she can make the two possibilities equiprobable). Moreover, if she knew which bit the coin was biased towards, she could determine Bob's bit.

2). A coin which is biased $\frac{1}{2} + \varepsilon$ towards the bit of Alice's choice (0 or 1). Moreover, Alice can **predict** the flips of this coin.

Alice sends Bob these coins in random order. These coins have the additional property that Bob cannot distinguish them. However, he can flip both coins. Now he simply reports the flips of the coins to Alice, and in return obtains a receipt for each flip. If Bob cheats, he will get caught with probability $\frac{1}{2}$, since Alice can predict the flips of the second coin. For convenience, we shall refer to the first coin as Alice's **info-coin,** and the second coin as her **control-coin.**

Though this protocol seems a figment of the imagination (or one of Alice's adventures in Wonderland), we show how to actually implement these coins using the deterministic and random properties of trapdoor generators. The **pseudo-random number technique** consists precisely in implementing these coins, thereby ensuring that the person flipping the $\frac{1}{2} + \varepsilon$ biased coin is forced to report the flips honestly.

4. The Coins and their Flips.

A 'coin' will be defined relative to a composite N, which is the product of two large primes each congruent to 3 (mod 4). Such composites (Blum Integers) have several useful cryptographic properties which were first pointed out by Blum [3]. The properties we shall use are:

1) -1 is a quadratic non-residue and has Jacobi Symbol[1] 1, mod N.
2) every quadratic residue has exactly one square root mod N which is itself a quadratic residue.

By a **coin** we mean any number x in Z_N^* having Jacobi Symbol 1. Given an $\varepsilon : 0 \le \varepsilon \le \frac{1}{2}$, and a pseudo-random number generator G, the **flips** of a coin are obtained as follows:
Step 1: If x is a non-residue, convert it into a residue by multiplying by -1 mod N.

Step 2: Obtain the unique square root of x (mod N) which is a quadratic residue.

Step 3: Feed this square root as a seed to G and obtain the pseudo-random sequence. (Notice that if the seed is random, this sequence 'looks like' the flips of a fair coin. This is formalized in section 6).

Step 4: Convert the pseudo-random sequence into the flips of a $\frac{1}{2}+\varepsilon$ biased coin, biased towards Heads. A bit-efficient method of doing so is described in section 8.

Step 5: Whenever the coin in Step 4 comes up Heads, output the correct quadratic residuosity of x, and whenever it comes up Tails, output the negation of the residuosity of x.

There is an efficient algorithm for flipping a coin, given the prime factors of N. In section 7 we shall formalize and prove that if x is randomly chosen from Z_N^*, and the prime factors of N are not known, then the flips of x 'look like' the flips of a real coin biased $\frac{1}{2}+\varepsilon$ towards the residuosity of x.

5. A Protocol Without Intermediary, that Almost Works.

In this section we shall present a protocol that uses no intermediary and illustrates the basic ideas of the pseudo-random number technique; however, we are unable to prove it secure. In Section 7 we present a more sophisticated version along with a proof.

The following conditions are required before starting the protocol:
Alice and Bob have already decided on an ε and a pseudo-random number generator, G. Bob has a composite N_B which is the product of two large equal length primes P_B and Q_B, each congruent to 3 (mod 4). Bob knows the prime factors of N_B, and Alice knows N_B, but not its prime factors. Bob's message is the quadratic residuosity of a random element $b \in Z_N^*$ having Jacobi Symbol 1. Alice knows b. Even so, under the Quadratic Residuosity Assumption[2] [6], Alice has no advantage in guessing Bob's bit.

[1]Let P be a prime. The Jacobi Symbol of $x \in Z_P^*$ with respect to P is defined as $(\frac{x}{P}) = x^{\frac{p-1}{2}}$ $(mod\ P)$. x is a quadratic residue mod P iff $(\frac{x}{P}) = 1$.
Let $N = P \times Q$, P, Q primes. The Jacobi symbol of $x \in Z_N^*$ with respect to N is defined as $(\frac{x}{N}) = (\frac{x}{P}) \times (\frac{x}{Q})$. The Jacobi Symbol can be computed in $O(|N|^3)$ time even without knowledge of the factorization of N.

[2]Let $N = P \times Q$, P, Q distinct odd primes. Exactly half the elements of Z_N^* having Jacobi Symbol 1 are quadratic residues mod N. Given N and $x \in Z_N^*$ with $(\frac{x}{n}) = 1$, the problem of deciding if x is a quadratic residue mod N is called the *quadratic residuosity problem*.
The Quadratic Residuosity Assumption [6] says that it is computationally infeasible to solve the quadratic residuosity problem. More precisely, let T be a probabilistic polynomial time procedure which on input N and x, each of length 2n, outputs a 0 or 1. Let δ be a real number between 0 and 1, and let t be a positive integer. Then for n sufficiently large, and all but δ fraction of numbers N of length n, $N = P \times Q$, P, Q, distinct, odd primes of length n, the probability that T is incorrect in guessing the quadratic residuosity of x mod N on input (N, x), with x picked uniformly from Z_N^* with $(\frac{x}{N})$ is greater than $1/n^t$.

Alice constructs a pair of coins (numbers) u and v, and sends them to Bob in random order:

1) She chooses a random number r_1 from $Z^*_{N_B}$, and picks her info-coin to be:

$$u = \begin{cases} r_1^2 \times b \ (mod\ N_B) \ with\ prob.\ \frac{1}{2}. \\ -r_1^2 \times b \ (mod\ N_B) \ with\ prob.\ \frac{1}{2}. \end{cases}$$

2) She chooses another random number r_2 from $Z^*_{N_B}$, and picks her control-coin to be:

$$v = \begin{cases} r_2^4 \ (mod\ N_B) \ with\ prob.\ \frac{1}{2}. \\ -r_2^4 \ (mod\ N_B) \ with\ prob.\ \frac{1}{2}. \end{cases}$$

Notice that u and v are independent and equi-distributed among numbers in $Z^*_{N_B}$ having Jacobi Symbol 1. Bob has no advantage in distinguishing between u and v since Alice sends them in random order. Alice knows the quadratic residuosity of v, and the seed for this coin (i.e. $r_2^2 \ (mod\ N_B)$). Therefore she can predict the flips that Bob must report on her control-coin. So, if Bob cheats he will get caught with probability ½.

Notice that residuosity of b

$$= \begin{cases} residuosity\ of\ u & \text{if } u = r_1^2 \times b \ (mod\ N_B). \\ -residuosity\ of\ u & \text{if } u = -r_1^2 \times b \ (mod\ N_B). \end{cases}$$

The flips of the info-coin will give Alice statistical information about the residuosity of u, from which she can obtain statistical information about Bob's bit.

The problem with this protocol is that Alice may be able to cheat. The cryptographic security of the pseudo-random number generator means that on a *randomly chosen* seed, the output of the generator cannot be distinguished from a truly random sequence by any polynomial time statistical test. The statistical test is not given any information about the seed. However, Alice *knows* the square of the seed for the control-coin (it is either u or -u). Even though the seed cannot be computed by Alice (because extracting square roots modulo a composite is as hard as factoring [9]), she has partial information about it which may give her some advantage in predicting the output of the pseudo-random number generator, and therefore obtain more information about Bob's bit.

6. Trapdoor Generators.

The pseudo-random number generator that enables us to design the provably secure protocol is the '$x^2\ mod\ N$-generator' [1]. Briefly, let N be the product of two large primes P and Q, each congruent to 3 (mod 4). Squaring (mod N) is a 1-1 onto function on the quadratic residues (mod N). Let x_0 be a random quadratic residue mod N. The $x^2\ mod\ N$-generator outputs $b_0 b_1 b_2 \cdots$ where $b_i = parity(x_i)$ and $x_{i+1} = x_i^2 \ (mod\ N)$. This sequence is said to be generated **forwards** on seed x_0. Similarly, the $x^2\ mod\ N$-generator can also output $b_{-1} b_{-2} b_{-3} \cdots$ by taking square roots, i.e. $b_i = parity(x_i)$ and x_{i-1} = the unique square root of x_i mod N, which is a quadratic residue. This sequence is said to be generated **backwards** on the seed x_0.

Blum, Blum and Shub point out that the $x^2\ mod\ N$-generator has several nice properties that promise many interesting applications. The properties that make this pseudo-random number generator especially useful for our purposes are:

1) Given x_0 and N, there is an efficient deterministic algorithm to generate the pseudo-random sequence forwards.

2) Given P and Q, there is an efficient algorithm for generating the pseudo-random sequence backwards, i.e. $b_{-1} b_{-2} b_{-3} \cdots$

3) However, without the knowledge of P and Q, the backwards generated sequence cannot be distinguished from a truly random sequence (by a polynomial-time statistical test).

We generalize properties 1, 2 and 3, and define a class of pseudo-random number generators which we shall call **trapdoor generators.** In [5], Blum and Micali give a set of sufficient conditions for constructing a pseudo-random number generator using a one-way function f. In particular, if f is required to be a trapdoor function, the resulting generator will be a trapdoor generator.

Below, we give a formal definition of trapdoor generators.

Definition: Let V (the set of *parameter values*) be a subset of the positive integers. Associated with each parameter $N \in V$ is an integer K_N called the *key of N*. For each $N \in V$, let $X_N \subset \{0,1\}^{|N|}$ be a set of *seeds*. The *seed domain*, $X = \{(N,x) \mid N \in V, \text{ and } x \in X_N\}$.

Definition: A *two-way infinite sequence* s is a map from the Integers to $\{0,1\}$, i.e. $s: \mathbf{Z} \to \{0,1\}$. Let S be the set of two-way infinite sequences. For any $k \in \mathbf{Z}$, and two-way infinite sequences s and s', s' is said to be a k *shift* of s if for every $n \in \mathbf{Z}$, $s'(n) = s(n+k)$.

Definition: A *trapdoor generator* G is a map from the set of seeds to the set of two-way infinite sequences, i.e. $G: X \to S$, satisfying the following four properties. Properties 1, 2, and 3 correspond to the three properties of the $x^2 \; mod \; N$ generator stated above. (let $(N,x) \in X$, and let $G(N,x) = s$):

0) For every $k \in \mathbf{Z}$, there exists a seed $(N,y) \in X$ such that $G(N,y)$ is a k shift of $G(N,x)$. Moreover, there is a polynomial time procedure which on input (N,x), and k in *unary*, outputs (N,y).

1) There is a polynomial time procedure which on input (N,x) and a positive integer k in *unary*, outputs the first k bits of $G(N,x)$, i.e. $s(0), s(1), \cdots s(k-1)$. This process is called generating the pseudo-random sequence **forwards** on seed (N,x).

2) There is a polynomial time procedure which on input (N,x), an integer k in *unary*, and the key K_N, outputs $s(-1), s(-2), \cdots s(-k)$. This process is called generating the pseudo-random sequence **backwards** on seed (N,x).
Comment: Because of properties 0, 1 and 2, the k bit backwards generated sequenle on seed (N,y) is simpply the reverse of the k bit forwards generated sequence on seed (N,x).

3) For every non-negative integer t and positive fraction ε no probabilistic polynomial time procedure on input (N,x) of length n and the sequence of n^t bits of $G(N,x)$ generated backwards has even an ε advantage in predicting the next bit of the sequence.

It follows from Yao's Theorem that no probabilistic polynomial time statistical test can distinguish between sequences produced by a trapdoor generator backwards and truly random sequences *even* if *the seed is supplied* to the statistical test. The following definition adequately captures the notion of a probabilistic polynomial time statistical test, for our purposes.

Definition (Yao): A *probabilistic poly-time statistical test*, T, is a probabilistic poly-time algorithm which on input a $(N,x) \in X$, and a sequence outputs a 0 or a 1.

1) Let $\alpha_{n,m}$ denote the average value of T's output over all inputs (N, x, S), where |N| = n, x is a seed, and S is the m-bit pseudo-random sequence produced by the trapdoor generator G going *backwards* on seed x.
2) Let $\beta_{n,m}$ denote the average value of T's output over all inputs (N, x, S), where |N| = n, x is a seed, and S is *any* m-bit sequence.

We say that the generator G *passes test T* if for every positive integer t, and polynomial p, $|\alpha_{n,p(n)} - \beta_{n,p(n)}| < 1/n^t$ for all sufficiently large n.

Theorem 1 (Blum, Blum, Shub): Modulo the Quadratic Residuosity Assumption, the $x^2 \; mod \; N$ generator passes all probabilistic polynomial time statistical tests.

7. The Protocol.

The conditions required before starting the protocol are the same as in Section 5, except that Alice and Bob have agreed on using the $x^2 \; mod \; N$ generator run **backwards.** In the final paper we shall prove that any trapdoor generator can be used to design a provably secure protocol for one-bit disclosure.

Step 1: Bob bitwise flips[3] to Alice four random numbers, r, r', s, s', such that $0 \le r, r', s, s' < N_B$,

Step 2: Alice constructs one pair of coins (numbers) from r and r', and another pair from s and s'. The procedure for the construction of the two pairs is identical. We describe the construction of the first pair below:

b) Alice's control-coin is:

$$u = \begin{cases} r^{2^k} (mod \; N_B) \; with \; prob. \; \frac{1}{2}. \\ -r^{2^k} (mod \; N_B) \; with \; prob. \; \frac{1}{2}. \end{cases}$$

Using $r^2 (mod \; N_B)$ as seed to the $x^2 mod \; N_B$-generator, Alice generates a (k-1) bit pseudo-

[3]A protocol for flipping a bit is proposed in [11]. Briefly, if Bob flips a bit a to Alice then
1. a is equally likely to be 0 or 1.
2. Alice knows a, but Bob has no idea what a is.
3. Alice can prove to Bob that he flipped a to her (she cannot do so for $\bar{a}$).

random sequence **forwards** (k is chosen large enough by Alice and Bob, so that at the end of the protocol Alice knows Bob's bit with high probability. Since Alice already knows the pseudo-random sequence (see comment in Section 6), and the residuosity of u, she can predict Bob's response on this coin.

a) Alice's info-coin is:

$$v = \begin{cases} b \times r'^2 (mod\ N_B)\ with\ prob.\ ½. \\ -b \times r'^2 (mod\ N_B)\ with\ prob.\ ½. \end{cases}$$

Alice now sends Bob u and v in random order.

Step 3: Bob picks one of the two pairs of coins at random and asks Alice to prove that she constructed them according to Step 2. This involves proving to Bob two of the random numbers that he flipped to her.

Step 4: Bob flips the other pair of coins (say x_1 and x_2), running the $x^2\ mod\ N_B$ generator **backwards**.

Step 5: Repeat Steps 5a and 5b $O(1/\varepsilon^2)$ times:

a) Bob now reveals to Alice one flip of the x_1 and x_2 coins.

b) Alice checks Bob's response on her control-coin. The other response gives her fractional information about Bob's bit. She gives him a signed receipt for both flips (because she does not want Bob to figure out which coin is which). A bit-secure signature scheme for doing so is described in [6]. The convention will be that a receipt is invalid if Alice can demonstrate (by the procedure of Step 3) that the flip sequence was generated using her control-coin. So Bob's only valid receipt is that on Alice's info-coin.

In Theorems 2 and 3 we shall prove that if either Alice or Bob deviated from protocol in Steps 1-5a the other would detect this. In Theorem 3 we shall establish that Alice cannot get any more information from Bob's response on her info-coin, than she would have if he had used a 'real coin'.

Theorem 2: If Alice follows protocol and Bob cheats in Steps 1-5a then with probability at least ½ Alice would detect this in Step 5b.

Proof: The numbers sent by Alice in Step 2 are independent and equidistributed among numbers in $Z^*_{N_B}$ having Jacobi Symbol 1. These numbers are sent in random order, so Bob has no information about which number correspond to which coin. If Bob tries to cheat by giving Alice an incorrect bit in Step 5a, then with probability at least ½ he would do so on a control-coin, and Alice would detect this.

This probability of cheating can be reduced to $1-\frac{1}{j}$ if (j - 1) control-coins an d one info-coin were used.

Theorem 3: If Bob follows protocol and Alice cheats in Steps 1-5a then with probability at least ½ Bob would detect this in Step 3.

Proof: Alice can cheat only in Step 2 when she constructs the coins. However, since Bob randomly picks out one of the two pairs of coins for her to prove in Step 3, Alice will get caught with probability at least ½.

This probability of cheating can be reduced to $1-\frac{1}{j}$ if j sets of coins are constructed and Bob asks Alice to prove (j - 1) of them in Step 3.

We would like to prove that Alice cannot know Bob's bit with any greater certainity than that given by the Bayesian Probability. We shall prove a stronger statement: Alice cannot correctly guess Bob's bit on a non-negligibly greater fraction of the pseudo-random sequences than the truly random sequences (this guessing probability may be unrelated to the Bayesian Probability).

Definition: A *bit-predictor*, M, is a probabilistic polynomial time algorithm which on input (N, x, b, r, S), where N is a Blum Integer, x, b, r are elements of Z^*_N having Jacobi Symbol 1, satisfying $x = \pm r^2 \times b$ (mod N), and S is a binary sequence, outputs a 0 or a 1 (its guess for the quadratic residuosity of b mod N).

1) Consider input tuples (N, x, b, r, S) where N is picked uniformly from n-bit Blum Integers, b and r are uniformly and independently picked from elements of Z^*_N having Jacobi symbol 1,

$$x = \begin{cases} r^2 \times b\ (mod\ N)\ with\ prob\ ½. \\ -r^2 \times b\ (mod\ N)\ with\ prob\ ½.' \end{cases}$$

and S is the unique m-bit sequence of flips of coin x. Let $\alpha^*_{n,m}$ denote the probability that M correctly outputs the residuosity of b on an input tuple picked from this distribution.

2) Consider input tuples (N, x, b, r, S), where N is picked uniformly from n-bit Blum Integers, b and r are uniformly and independently picked from elements of Z^*_N having Jacobi symbol 1,

$$x = \begin{cases} r^2 \times b\ (mod\ N)\ with\ prob\ ½. \\ -r^2 \times b\ (mod\ N)\ with\ prob\ ½.' \end{cases}$$

and S is picked from the set of m-bit binary sequences with probability equal to the probability that S is generated by the flips of a coin biased $\frac{1}{2}+\varepsilon$ towards the quadratic residuosity of b mod N. Let $\beta^{*}_{n,m}$ denote the probability that M correctly outputs the residuosity of b on an input tuple picked from this distribution.

We shall say that the protocol is ***fair relative to bit−predictor M*** if for every positive integer t, and polynomial p, $|\alpha^{*}_{n,p(n)}-\beta^{*}_{n,p(n)}| < 1/n^{t}$ for all sufficiently large n.

Theorem 4: Modulo the Quadratic Residuosity Assumption, the protocol is fair relative to all probabilistic polynomial time bit-predictors.

Proof: Suppose to the contrary that there is a bit-predictor, M, relative to which the protocol is not fair. We shall use M to design a probabilistic polynomial time statistical test T which the x^2 *mod N* generator fails to pass, thus contradicting Theorem 1. The statistical test T shall do the following: on input (N, x, S) where where the length of N is n, and the length of S is $2\times p(n)$, it picks a random number r in Z^{*}_{N} having Jacobi Symbol 1. It will give as input to M the tuple (N, y, b, r, S'), where

$$y = \begin{cases} r^2\ (mod\ N)\ with\ prob\ \frac{1}{2}. \\ -r^2\ (mod\ N)\ with\ prob\ \frac{1}{2}.' \end{cases}$$

r is randomly chosen from Z^{*}_{N}, and $b=y\times r^{-2}\ (mod\ N)$. Notice that T knows the residuosity of b. S' is the sequence of length p(n) biased $\frac{1}{2}+\varepsilon$ towards the residuosity of b, constructed from S (which is of length $2\times p(n)$) as described in Section 8. If M correctly outputs the residuosity of b then T outputs 1, otherwise it outputs 0.

Clearly, $\alpha_{n,2p(n)}$ for T is equal to $\alpha^{*}_{n,p(n)}$ for M and $\beta_{n,2p(n)}$ for T is equal to $\beta^{*}_{n,p(n)}$ for M. Since the protocol is not fair relative to bit-predictor M, there exist an integer t, and a polynomial p: for infinitely many values of n: $|\alpha^{*}_{n,p(n)}-\beta^{*}_{n,p(n)}| > 1/n^{t}$. But this implies that for infinitely many values of n: $|\alpha^{*}_{n,2p(n)}-\beta^{*}_{n,2p(n)}| > 1/n^{t}$. Thus the x^2 *mod N* generator fails to pass test T, contradicting Theorem 1.

8. Constructing a $\frac{1}{2}+\varepsilon$ Biased Coin.

We wish to convert the flips of an unbiased coin into the flips of a coin biased $\frac{1}{2}+\varepsilon$ towards 0. Let the binary expansion of $\frac{1}{2}+\varepsilon$ be $x = 0\,.\,x_1x_2x_3\cdots$. We shall describe below the procedure for outputting one flip of the biased coin.

```
procedure:
    do k = 1 to infinity,
        Let $f_k$ = next flip of the unbiased coin.
        if $f_k = x_k$ then continue.
        else if $f_k < x_k$ then output 0 and HALT.
        else if $f_k > x_k$ then output 1 and HALT.
    end
end.
```

Theorem 5: The above procedure outputs a bit biased $\frac{1}{2}+\varepsilon$ towards 0. The expected number of flips of the unbiased coin needed to output one flip of the biased coin is 2.

Proof: Consider an infinite sequence of flips of the unbiased coin: $f_1, f_2, f_3, \cdots$ and interpret it as the binary number $f = 0\,.\,f_1f_2f_3\cdots$ in the interval [0, 1].

Define $\text{bit}(f) = \begin{cases} 0 \text{ if } f \in [0,x). \\ 1 \text{ if } f \in (x,1]. \end{cases}$

Since $f \in [0,x)$ with probability $\frac{1}{2}+\varepsilon$ and $f \in (x,1]$ with probability $\frac{1}{2}-\varepsilon$, bit(f) is biased $\frac{1}{2}+\varepsilon$ towards 0. Moreover, to determine bit(f) it suffices to find the least k: $f_k \neq x_k$, since bit(f) = 0 if $f_k < x_k$ and 1 if $f_k > x_k$. From elementary probability theory, the expected value of k is 2. So on the average we need only two flips of the unbiased coin for each flip of the biased coin.

Shannon Coding Theory [11] gives a method of obtaining more than one flip of the biased coin for each flip of the unbiased coin on the average, provided the biased flips are output block at a time instead of a bit at a time.

Acknowledgements We are extremely indebted to: Manuel Blum for supervising this research with his unique sense of judgement; Richard Karp for showing us the elegant and efficient procedure of Section 8; Avi Wigderson, whose knack for weeding out the inessential led us to a clean definition of trapdoor generators; Claus Schnorr for some very enlightening discussions on number-theoretic cryptography; Mike Luby for sharing his knowledge of random walks; Clark Thompson for introducing us to the notion of 'fractional bits' in the context of Communication Complexity in VLSI and to Tom Tedrick for several useful discussions on fractional bits.

References

1). L. Blum, M. Blum and M. Shub, "A Simple Secure Pseudo-Random Number Generator," to appear in SIAM Journal of Computing.

2). M. Blum, "How to Exchange (Secret) Keys," ACM Transactions on Computer Systems (1983). (1982).

3). M. Blum, "Coin Flipping by Telephone," in Proc. of IEEE Spring COMPCON (1982). 4). M. Blum and M. Rabin, "Mail Certification by Randomization," to appear.

5). M. Blum and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits," 1982 FOCS.

6). S. Goldwasser and S. Micali, "Probabilistic Encryption and How to Play Mental Poker Keeping Secret all Partial Information," 1982 STOC.

7). S. Goldwasser, S. Micali and A. Yao, "Strong Signature Schemes," 1983 STOC.

8). M. Luby, S. Micali and C. Rackoff, "The MiRackoLus Exchange of a Secret Bit," 1983 FOCS.

9). M. O. Rabin, "Digital Signatures and Public-key Functions as Intractable as Factorization," MIT/LCS/TR-212 Tech. memo, MIT, 1979.

10). A. Shamir, "On the Generation of Cryptographically Strong Pseudo-Random Sequences," 1981 ICALP.

11). C. E. Shannon, "A Mathematical Theory of Communication," Bell Systems Journal, 27 (1948).

12). T. Tedrick, "How to Exchange Half a Bit," CRYPTO '83.

13). A. Yao, "Theory and Applications of Trapdoor Functions," 1982 FOCS.