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

RSA AND RABIN FUNCTIONS: CERTAIN PARTS ARE AS HARD AS THE WHOLE*

WERNER ALEXI†, BENNY CHOR‡, ODED GOLDREICH§ AND CLAUS P. SCHNORR†

Abstract. The RSA and Rabin encryption functions $E_N(\cdot)$ are respectively defined by raising $x \in Z_N$ to the power e (where e is relatively prime to $\varphi(N)$) and squaring modulo N (i.e., $E_N(x) = x^e \pmod N$, $E_N(x) = x^2 \pmod N$, respectively). We prove that for both functions, the following problems are computationally equivalent (each is probabilistic polynomial-time reducible to the other):

(1) Given $E_N(x)$, find x.

(2) Given $E_N(x)$, guess the least-significant bit of x with success probability $\frac{1}{2} + 1/\text{poly}(n)$ (where n is the length of the modulus N).

This equivalence implies that an adversary, given the RSA/Rabin ciphertext, cannot have a non-negligible advantage (over a random coin flip) in guessing the least-significant bit of the plaintext, unless he can invert RSA/factor N. The proof techniques also yield the simultaneous security of the log n least-significant bits. Our results improve the efficiency of pseudorandom number generation and probabilistic encryption schemes based on the intractability of factoring.

Key words. cryptography, concrete complexity, RSA encryption, factoring integers, partial information, predicate reductions

AMS(MOS) subject classifications. 11A51, 11K45, 11T71, 11Y05, 11X16, 11Z50, 68Q99, 94A60

1. Introduction.

One-way functions are the basis for modern cryptography [11] and have many applications to pseudorandomness and complexity theories [6], [29]. One-way functions are easy to evaluate, but hard to invert. Even though no proof of their existence is known (such proof would imply $P \neq NP$), it is widely believed that one-way functions do exist. In particular, if factoring large numbers (a classical open problem) is hard, then the simple function of squaring modulo a composite number is one-way [22].

Randomness and computational difficulty play dual roles. If a function f is one-way then, given $f(x)$, the argument x must be "random." It cannot be the case that every bit of the argument x is easily computable from $f(x)$. Therefore, some of these bits are unpredictable, at least in a weak sense. A natural question is whether these bits are strongly unpredictable. That is, are there specific bits of the argument x which cannot be guessed with the slightest advantage (over a random coin flip), given $f(x)$.

This question was first addressed by Blum and Micali [6]. They demonstrated such a strongly unpredictable bit for the discrete exponentiation function (which is believed to be one-way). This was done by reducing the problem of inverting the discrete exponentiation function to the problem of guessing a particular bit with any non-negligible advantage.

In this paper, we deal with two functions related to the factorization problem: Rabin's function (squaring modulo a composite number) [22] and the RSA (raising to a fixed exponent modulo a composite number) [23]. Both functions are believed to

* Received by the editors October 29, 1984; accepted for publication (in revised form) July 7, 1986.

† Fachbereich Mathematik, Universität Frankfurt, 600 Frankfurt/Main, West Germany.

‡ Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139. The work of this author was supported in part by the National Science Foundation under grant MCS-8006938 and by an IBM Graduate Fellowship.

§ Computer Science Department, Technion, Haifa 32000, Israel. This research was performed while the author was at the Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139. The author was supported in part by a Weizmann Postdoctoral Fellowship.

be one-way. We show that both functions have strongly unpredictable bits. In particular, inverting each of them is probabilistic polynomial-time reducible to guessing the least-significant bit of their argument with any non-negligible advantage.

Our results have various applications. They allow the construction of more efficient pseudorandom bit generators [6] than those previously known, based on the intractability assumption of factoring. They allow the construction of efficient probabilistic encryption schemes [15], which hide all partial information. Finally, our results imply that the RSA public-key perfectly hides all partial information about the $\log n$ least-significant bits of the plaintext (where n is the size of the RSA modulos).

Organization of the paper. In § 2 we formally define the question of security for RSA least-significant bit and cover previously known results. In § 3 we review the proof of the Ben-Or, Chor and Shamir result. This investigation is the basis for our work, which is described in § 4. Section 5 extends our proof to other RSA bits, and § 6, to bits in Rabin's scheme. In § 7 we discuss applications of our results for the construction of pseudorandom bit generators and probabilistic encryption schemes. Section 8 contains concluding remarks and two open problems.

2. Problem definition and previous results. We begin this section by presenting notations for two number theoretic terms which will be used throughout the paper.

DEFINITION 1. Let N be a natural number. Z_N will denote the ring of integers modulo N, where addition and multiplication are done modulo N.

It would be convenient to view the elements of Z_N as points on a circle (see Fig. 1).

Convention. Throughout the paper, $n = \lceil \log_2 N \rceil$ will denote the length of the modulus N. All algorithms discussed in this paper have inputs of length $O(n)$.

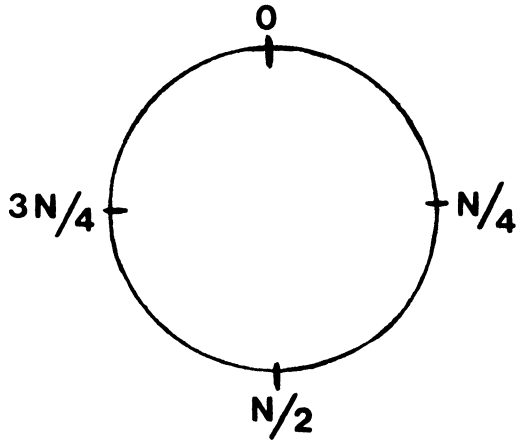

FIG. 1. *Cyclic representation of* Z_N.

DEFINITION 2. Let N be a natural number, and x an integer. $[x]_N$ will denote the remainder of x modulo N (notice that for all x, $0 \leqq [x]_N < N$).

The *RSA encryption function* is operating in the message space Z_N, where $N = pq$ is the product of two large primes (which are kept secret). The encryption of x is $E_N(x) = [x^e]_N$, where e is relatively prime to $\varphi(N) = (p-1)(q-1)$.

We now formally define the notion of bit security for the RSA.

DEFINITION 3. For $0 \leqq x < N$, $L_N(x)$ denotes the least-significant bit in the binary representation of x.

DEFINITION 4. Let $\varepsilon(\cdot)$ be a function from integers into the interval $[0, \frac{1}{2}]$. Let $\mathcal{O}_N$ be a probabilistic oracle which, given $E_N(x)$, outputs a guess, $\mathcal{O}_N(E_N(x))$, for $L_N(x)$ (this guess might depend on the internal coin tosses of $\mathcal{O}_N$). We say that $\mathcal{O}_N$ is an $\varepsilon(n)$-oracle for the least-significant bit (in short, $\varepsilon(n)$-oracle) if the probability that $\mathcal{O}_N$ is correct, given $E_N(x)$ as its input, is at least $\frac{1}{2}+\varepsilon(n)$. The probability space is that of all $x \in Z_N$ and all 0-1 sequences of internal coin tosses, with uniform distribution.

DEFINITION 5. We say that RSA least-significant bit is $\varepsilon(n)$-secure if there is a probabilistic polynomial time algorithm which on input N, e (relatively prime to $\varphi(N)$) and $x \in Z_N$, and access to an arbitrary $\varepsilon(n)$-oracle $\mathcal{O}_N$, outputs y such that $x = E_N(y) = y^e \pmod N$. (That is, the algorithm inverts E_N using any $\varepsilon(n)$-oracle $\mathcal{O}_N$.)

DEFINITION 6. We say that RSA least-significant bit is unpredictable if it is n^{-c}-secure for every constant $c > 0$.

2.1. Previous work. Goldwasser, Micali and Tong [17] were the first to investigate the security question of least-significant bit in RSA. They showed that the least-significant bit is as hard to determine as inverting the RSA. Furthermore, they showed that it is $(\frac{1}{2}-1/n)$-secure.

In a key paper, Ben-Or, Chor and Shamir [2] showed a $(\frac{1}{4}+1/\text{poly}(n))$-security result. They showed that inverting the RSA is polynomially reducible to determining the parity of messages taken from a certain small subset of the message space. To determine the parity of these messages, they performed many independent "measurements," each consisting of querying the oracle on a pair of related points. This sampling method amplified the $\frac{1}{4}+1/\text{poly}(n)$ overall advantage of the oracle to "almost certainty" in determining parity for the above-mentioned subset. On the negative side, the sampling of pairs of points doubles the error of the oracle and prevents the use of less reliable oracles.

Vazirani and Vazirani [27] showed that the "error doubling" phenomena could be overcome by introducing a new oracle-sampling technique. They proved that incorporating their technique in the Ben-Or, Chor and Shamir algorithm, yields a 0.232-security result. Goldreich [13] used a better combinatorial analysis to show that the Vazirani and Vazirani algorithm yields a 0.225 result. He also pointed out some limitations of the Vazirani and Vazirani proof techniques.

All these results leave a large gap towards the desired result of proving that the least-significant bit is unpredictable (i.e., $1/\text{poly}(n)$ secure).

3. A description of Ben-Or, Chor and Shamir reduction. In this section, we present the reduction used by Ben-Or, Chor and Shamir [2]. It consists of two major parts: An algorithm which inverts the RSA using a *parity subroutine*, and a method of implementing the parity subroutine by querying an oracle for the least-significant bit.

3.0. Definition of parity. Let x be an integer. We define

$$\text{abs}_N(x) \stackrel{\text{def}}{=} \begin{cases} [x]_N & \text{if } [x]_N < N/2, \\ N-[x]_N & \text{otherwise.} \end{cases}$$

Pictorially, abs (x) can be viewed as the distance from $[x]_N$ to 0 on the Z_N circle (see Fig. 2). Notice that $\text{abs}_N(x) = \text{abs}_N(-x)$.

The *parity* of x, $\text{par}_N(x)$, is defined as the least-significant bit of $\text{abs}_N(x)$. For example, $\text{par}_N(N-3) = 1$.

3.1. Inverting RSA using a parity subroutine. Given an encrypted message, $E_N(x)$, the plaintext x is reconstructed as follows. First, two integers $a, b \in Z_N$ are picked at

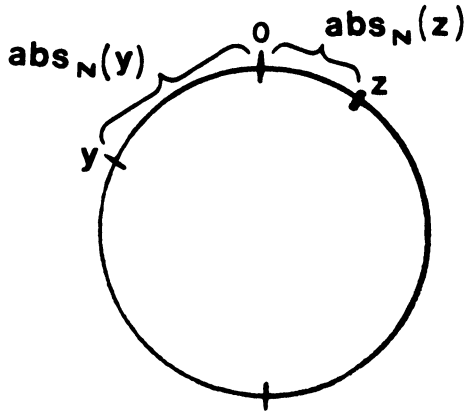

FIG. 2. *The* abs_N *function.*

random. A Brent-Kung gcd procedure [7] is applied to $[ax]_N$ and $[bx]_N$. This gcd procedure uses a parity subroutine PAR which we assume, at this point, to give correct answers. Even though neither $[ax]_N$ nor $[bx]_N$ are explicitly known, we can manipulate them via their encryption. In particular, we can compute the encryption of any linear combination $A[ax]_N + B[bx]_N$ when both A, B are known (since N and e are given, and E_N is a multiplicative function). When the gcd procedure terminates, we get a representation of $\gcd([ax]_N, [bx]_N) = [lx]_N$ in the form l and $E_N(lx)$. If $[ax]_N$ and $[bx]_N$ are relatively prime, then $[lx]_N = 1$. This fact can be detected since $E_N(1) = 1$. Therefore, $x \equiv l^{-1} \pmod N$ can be easily computed.

```
1.  procedure RSA INVERSION:
2.  INPUT ← E_N(x) (and N, e)
;   Step 1—Randomization
3.  Pick a, b ∈ Z_N at random.
;   Step 2—Brent–Kung GCD of [ax]_N, [bx]_N
    {z_1 = [ax]_N, z_2 = [bx]_N}
4.  α ← n,
5.  β ← n
6.  count ← 0
7.      repeat
8.          while PAR (b, E_N(x)) = 0 do
                Comment: PAR (b, E_N(x)) returns a guess for par_N (bx)
9.              b ← [b/2]_N
                {gcd [(ax]_N, [bx]_N) = gcd (z_1, z_2)}
10.             β ← β − 1
11.             count ← count + 1
12.             if count > 6n + 3 then go to line 3
13.         od
14.         if β ≦ α then swap (a, b), swap (α, β)
15.         if PAR ((a + b)/2, E_N(x)) = 0
                Comment: PAR ((a + b)/2, E_N(x)) returns a guess for
                  par_N ((ax + bx)/2)
16.             then b ← [(a + b)/2]_N
```

17. **else** $b \leftarrow [(a-b)/2]_N$
    $\{\gcd([ax]_N, [bx]_N) = \gcd(z_1, z_2)\}$
18. count $\leftarrow$ count $+1$
19. **if** count $> 6n+3$ **then go to line** 3
20. **until** $b = 0$
; **Step 3—Inverting**
21. **if** $E_N(ax) \neq \pm 1$ **then go to line** 3. $(E(ax) = E(a) \cdot E(x).)$
22. $x \leftarrow [\pm a^{-1}]_N$
23. **return** x.

We now consider a single run of Step 2. The assertions in the braces guarantee that if the parity subroutine does not err, then the gcd of the current $[ax]_N$, $[bx]_N$ is invariant. It is not hard to verify that the Brent–Kung gcd makes at most $6n+3$ calls to the parity subroutine.[1] Therefore, if the original pair $[ax]_N$, $[bx]_N$ is relatively prime and the parity subroutine answered correctly on all queries, the algorithm will retrieve x. By a famous theorem of Dirichlet [19, p. 324], the probability that two random integers in the range $[-K, K]$ are relatively prime converges to $6/\pi^2$ as K tends to ∞.

We assume so far that the parity subroutine always returns the correct answer. As a matter of fact, the test in line 21 of the code makes sure that the algorithm never errs, even if the answers of the parity subroutine are incorrect. The variable *count* guarantees that even if the parity subroutine occasionally errs, we will not make more than $6n+3$ parity calls in a single gcd iteration. The probability that x will be retrieved in any single gcd attempt is

$$\frac{6}{\pi^2} \cdot \Pr(\text{all answers of PAR in this gcd attempt are correct}).$$

Thus, to invert E_N in probabilistic polynomial-time it suffices to have a "reliable" parity subroutine. In fact, it suffices to have a parity subroutine which is almost always correct on every argument y with "small" $\text{abs}_N(y)$. This is the case since, if $\text{abs}_N(z_1)$ and $\text{abs}_N(z_2)$ are "small," then (unless the parity errs) all intermediate arguments to the parity subroutine are also "small." More formally, we use the following definition.

DEFINITION 7. Let $\varepsilon(\cdot)$, $\delta(\cdot)$ be functions from integers into the interval $(0, \frac{1}{2})$. Let PAR be a parity subroutine, that on input d and $E_N(x)$ outputs a guess for $\text{par}_N(dx)$. We say that PAR is $(\varepsilon(n), \delta(n))$-reliable if for every $d, x \in Z_N$ with $\text{abs}_N(dx) < \varepsilon(n)N/2$,

$$\text{Prob}(\text{PAR}(d, E_N(x)) \neq \text{par}_N(dx)) < \delta(n).$$

From the above discussion we derive the following lemma.

LEMMA 1 (Ben-Or, Chor and Shamir [2]). *The RSA function E_N is invertible in $O(\varepsilon^{-2}(n) \cdot n)$ expected number of calls to a $(\varepsilon(n), 1/(12n+6))$-reliable parity subroutine.*

Proof. Let PAR be a $(\varepsilon(n), 1/(12n+6))$-reliable parity subroutine. It suffices to give a lower bound on the probability that all PAR calls in a single gcd attempt yield correct answers. We define the following events, as functions of the random variables a and b (assuming values in Z_N^*): Event $S_x(a, b)$ holds if both $abs_x(ax)$ and $abs_x(bx)$ are smaller than $\varepsilon(n)N/2$. Event $C_{i,x}(a, b)$ holds if the ith answer of PAR, on a run

[1] This follows from the fact that $\alpha + \beta$ decreases by 1 in every execution of line 10, and that throughout the execution of the gcd $|a| \leqq 2^{\alpha}$ and $|b| \leqq 2^{\beta}$. For further details see [7], [8].

of gcd $([ax]_N, [bx]_N)$, is correct. Then

$$\begin{aligned}
&\text{Prob}((\forall i)A_{i,x}(a,b)) \\
&\quad \geqq \text{Prob}(S_x(a,b)) \cdot \text{Prob}((\forall i)A_{i,x}(a,b) \mid S_x(a,b)) \\
&\quad = \varepsilon^2(n) \cdot \left(1 - \sum_{i=1}^{6n+3} \text{Prob}(\neg A_{i,x}(a,b) \mid S_x(a,b)(\forall j < i)A_{j,x}(a,b))\right) \\
&\quad \geqq \varepsilon^2(n) \cdot \left(1 - (6n+3) \cdot \frac{1}{12n+6}\right) \\
&\quad = \varepsilon^2(n)/2. \qquad \square
\end{aligned}$$

Remark. In Step (2) of the RSA inversion procedure, we used a Brent–Kung gcd, which is faster than the binary gcd originally used in [2].

3.2. Implementing the parity subroutine. Given d and $E_N(x)$ where $\text{abs}_N(dx) < \varepsilon(n)N/2$, the parity subroutine PAR determines (with overwhelming probability) $\text{par}_N(dx)$, by querying $\mathcal{O}_N$, an oracle for RSA least-significant bit, as follows. It picks a random r and asks the oracle for the least-significant bit of both $[rx]_N$ and $[rx+dx]_N$, by feeding the oracle in turn with $E_N(rx) = E_N(r)E_N(x)$ and $E_N((r+d)x) = E_N(r+d)E_N(x)$. The oracle's answers are processed according to the following observation. Since $\text{abs}_N(dx)$ is small, with very high probability ($\geqq 1 - \varepsilon(n)/2$) no wraparound[2] occurs when $[dx]_N$ is added to $[rx]_N$. If no wraparound occurs, the parity of $[dx]_N$ is equal to 0 if the least-significant bits of $[rx]_N$ and $[rx+dx]_N$ are identical; and equal to 1 otherwise. This sampling is repeated many times; every repetition (instance) is called a *dx-measurement.*

```
1.  procedure PAR:
    Comment: PAR has access to a least-significant bit oracle O_N
2.  INPUT ← d, E_N(x)
3.      count_0 ← 0
4.      count_1 ← 0
5.      for i ← 1 to m do
6.          pick r_i ∈ Z_N at random
7.          if O_N(E_N(r_i x)) = O_N(E_N(r_i x + dx))
8.          then count_0 ← count_0 + 1
9.          else count_1 ← count_1 + 1
10.     od
11.     if count_0 > count_1
12.     then return 0
13.     else return 1
```

3.3. Discussion. Analyzing the error probability of PAR on input d, $E_N(x)$ reduces to analyzing the error probability of a single dx-measurement. Suppose that the success probability of a single dx-measurement can be made at least $\frac{1}{2} + 1/\text{poly}(n)$. Then by performing sufficiently many independent dx-measurements, the majority gives the correct answer for $\text{par}_N(dx)$ with overwhelming probability.

There are two sources of error in the parity subroutine. One source is wraparound 0 in a dx-measurement, the other is oracle errors. Wraparounds are unlikely (i.e., they

[2] If $[dx]_N = abs_N(dx)$ then *wraparound* 0 means $[rx]_N + abs_N(dx) > N$. If $[dx]_N = N - abs_N(dx)$ then *wraparound* 0 means $[rx]_N - abs_N(dx) < 0$.

occur with probability $\leqq \varepsilon(n)/2$, since $\text{abs}_N(dx) \leqq \varepsilon(n)N/2$). Thus, the main source of errors in a dx-measurement is the errors of the oracle.

If no wraparound occurs, then the dx-measurement may be wrong only if the oracle errs on either end points $([rx]_N, [rx+dx]_N)$. Thus the error probability of a dx-measurement is no more than *twice* the error probability of the oracle. However, there are oracles for which the error probability of a dx-measurement is *twice* the oracle error. Overcoming the *error-doubling* phenomena requires a new parity subroutine, which constitutes the core of our improvement.

4. The main result. In this section we prove that RSA least-significant bit is unpredictable. Working in the Ben-Or, Chor and Shamir framework, we modify the parity subroutine. For this modification we introduce two new ideas. The first idea is to avoid the error-doubling phenomena which occurred in the dx-measurement as follows. Instead of querying the oracle for the least-significant bit of both end points, we query the oracle only for the least-significant bit of one end point. The least-significant bit of the other end point is known beforehand. The second idea is a method for generating many end points with known least-significant bits. These end points are generated in a way that guarantees them "random" enough to be used as a good sample of the oracle.

4.1. Generating $m = \text{poly}(n)$ "random" points with known least-significant bits. We present a method for generating $m = \text{poly}(n)$ random points, represented as multiples of x, with the following properties:

(1) Each point is uniformly distributed in Z_N;

(2) The points are pairwise independent;

(3) The least-significant bit of each point is known, with probability $\geqq 1 - \varepsilon(n)/4$.

We generate m points $[r_i x]_N$ by picking two random independent elements $k, l \in Z_N$ with uniform distribution, and computing $[r_i x]_N = [(k+il)x]_N$ for $1 \leqq i \leqq m$ (see Fig. 3). Define the random variables $y, z \in Z_N$ by $y = [kx]_N$, $z = [lx]_N$. Each of the $[r_i x]_N$ is uniformly distributed in Z_N. We now show that (for $1 \leqq i \neq j \leqq m$) $[r_i x]_N$ and $[r_j x]_N$ are two independent random variables. For every $c_1, c_2 \in Z_N$, the equations $y + iz \equiv c_1 \pmod N$ and $y + jz \equiv c_2 \pmod N$ have a unique solution in terms of $y, z \in Z_N$. (This is the case since all of N's divisors are larger than m, and thus $i-j$ has a multiplicative inverse modulo N.) Thus, for every $c_1, c_2 \in Z_N$,

$$\Pr([r_i x]_N = c_1 \text{ and } [r_j x]_N = c_2) = \frac{1}{N^2} = \Pr([r_i x]_N = c_1) \cdot \Pr([r_j x]_N = c_2).$$

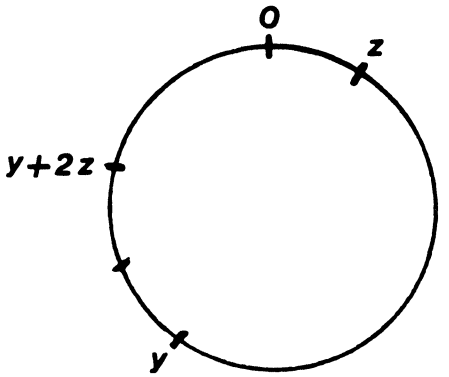

FIG. 3. *The points y, z and $[r_2 x]_N = y + 2z$.*

The least-significant bits of the $[r_i x]_N$'s are computed as follows: We try all possibilities for the least-significant bit of y, and for its location in one of the intervals $[j(\varepsilon(n)N/4), (j+1)(\varepsilon(n)N/4)]$, $0 \leqq j < 4\varepsilon^{-1}(n)$. We try all possibilities for the least-significant bit of z, and for its location in one of the intervals

$$\left[j\frac{\varepsilon(n)N}{4m}, (j+1)\frac{\varepsilon(n)N}{4m}\right], 0 \leqq j < 4m\varepsilon^{-1}(n).$$

There are $(2 \cdot 4\varepsilon^{-1}(n)) \cdot (2 \cdot 4m\varepsilon^{-1}(n)) = 2^6 m\varepsilon^{-2}(n)$ possibilities altogether, and exactly one of them is correct. We will refer to that possibility as the *right alternative for y and z.*

Let us now assume that we are dealing with the right alternative for y and z. Since the location of y and z are known up to $\varepsilon(n)N/4$ and $\varepsilon(n)N/4m$, respectively, the integer $w_i =_{\text{def}} y + iz$ is known up to $\varepsilon(n)N/2$ (remember $1 \leqq i \leqq m$). As $[w_i]_N$ is uniformly distributed in Z_N, the probability that the integer w_i falls in an interval of length $\varepsilon(n)N/2$ containing an integral multiple of N is exactly $\varepsilon(n)/2$. If w_i is not in such interval, then the integral quotient of w_i/N is determined by i and the approximate locations of y and z. This in turn, together with the least-significant bits of y and z, determines the least-significant bit of $[w_i]_N = [r_i x]_N$. In case the interval w_i belongs to contains an integral multiple of N, we make the (arbitrary) assumption that w_i is at the bigger part of the interval (out of the two parts determined by the integral multiple of N).

4.2. Using the generated $[r_i x]_N$ in the parity subroutine. The parity subroutine described in § 3.2 makes use of mutually independent random r_i's, and queries the oracle for the least-significant bits of both $[r_i x]$ and $[r_i x + dx]_N$. We modify it by using r_i's generated as in § 4.1, and querying the oracle only for the least-significant bit of $[r_i x + dx]_N$. In the sequel we will refer to the modified parity subroutine as to PAR*.

The generation of $[r_i x]_N$'s is performed once per each gcd invocation, as part of Step 1 (the randomization step) in the inversion procedure of § 3.1. The choice of k and l ($y = [kx]_N$, $z = [lx]_N$) is independent of the choice of a, b. We run $2^8 m\varepsilon^{-2}(n)$ copies of the gcd procedure in parallel, each with one of the possibilities for the approximate locations and least-significant bits of y and z. Each copy of the gcd procedure supplies its corresponding possibility for y and z as an auxiliary input to all calls of PAR* that it makes. Note that the run with the right alternative for y and z has the least-significant bit of $[r_i x]_N$ correct, for all $1 \leqq i \leqq m$, with very high probability.

4.3. Probability analysis of the modified parity subroutine. In this subsection we analyze the success probability of the modified parity subroutine PAR*. We show that given an $\varepsilon(n)$-oracle for RSA least-significant bit, $\mathcal{O}_N$, and setting $m = O(n \cdot \varepsilon^{-2}(n))$, makes the parity subroutine PAR* be $(\varepsilon(n), 1/(12n+6))$-reliable. The running time of the subroutine is polynomial in n and $\varepsilon^{-1}(n)$, and so is the expected running time of the entire RSA inversion procedure.

In analyzing the success probability of PAR* on input d, $E_N(x)$, we assume that $[dx]_N$ is small ($\text{abs}_N(dx) < \varepsilon(n)N/2$) and it is given, as auxiliary input, the right alternative for y and z. From this point on, probabilities are taken over all choices of y, z with uniform probability distribution (x and d are considered as fixed).

Recall that on input d, $E_N(x)$ the parity subroutine conducts m dx-measurements. Each measurement "supports" either $\text{par}_N(dx) = 0$ or $\text{par}_N(dx) = 1$. The subroutine returns the majority decision. For every $1 \leqq i \leqq m$, the ith individual dx-measurement consists of comparing the precomputed least-significant bit of $[r_i x]_N$ to the answer of

the oracle for the least-significant bit of $[r_i x + dx]_N$. Such measurement has three potential sources of error:

(1) The oracle errs on the least-significant bit of $[r_i x + dx]_N$;
(2) A wraparound 0 occurs when $[dx]_N$ is added to $[r_i x]_N$;
(3) The precomputed least-significant bit of $[r_i x]_N$ is wrong.

Note that $[r_i x + dx]_N$ is uniformly distributed in Z_N. Therefore, a type 1 error has probability $\frac{1}{2} - \varepsilon(n)$. Since $\text{abs}_N(dx) < \varepsilon(n)N/2$, a type 2 error has probability at most $\varepsilon(n)/2$. A type 3 error may occur only if $\text{abs}_N(r_i x)_N < \varepsilon(n)N/2$. A more careful look at the way the least-significant bit of $[r_i x]_N$ is determined in these "fuzzy" cases show that a type 3 error has probability at most $\varepsilon(n)/4$. (This follows from the assignment of correct least-significant bit values to points in the larger part of the interval.)

Thus, the overall error probability in a single dx-measurement is bounded above by $\frac{1}{2} - \varepsilon(n)/4$. Define the random variable

$$\zeta_i = \begin{cases} 1 & \text{if the } i\text{th } dx\text{-measurement is wrong,} \\ 0 & \text{if the } i\text{th } dx\text{-measurement is correct.} \end{cases}$$

Clearly, $\text{Exp}(\zeta_i) = \Pr(\zeta_i = 1) < \frac{1}{2} - \varepsilon(n)/4$ and $\text{Var}(\zeta_i) < \frac{1}{4}$. Since $\text{Exp}(\zeta_i) < \frac{1}{2} - \varepsilon(n)/4$, we get

$$\Pr\left(\frac{1}{m}\sum_{i=1}^{m} \zeta_i \geqq \frac{1}{2}\right) \leqq \Pr\left(\left|\frac{1}{m}\sum_{i=1}^{m} \zeta_i - \text{Exp}(\zeta_i)\right| \geqq \frac{\varepsilon(n)}{4}\right).$$

Applying Chebyshev's inequality (see [12, p. 219]) we get

$$\Pr\left(\left|\frac{1}{m}\sum_{i=1}^{m} \zeta_i - \text{Exp}(\zeta_i)\right| \geqq \frac{\varepsilon(n)}{4}\right) \leqq \frac{\text{Var}((1/m)\sum_{i=1}^{m} \zeta_i)}{(\varepsilon(n)/4)^2}.$$

Since (for $1 \leqq i \neq j \leqq m$) $[r_i x]_N$ and $[r_j x]_N$ are two independent random variables, ζ_i and ζ_j are also independent random variables with identical distribution. (Whenever the same function is applied to two independent random variables, the two results are independent random variables.) Let $\bar{\zeta}_i = \zeta_i - \text{Exp}(\zeta_i)$. By pairwise independence $\text{Exp}(\bar{\zeta}_i \cdot \bar{\zeta}_j) = \text{Exp}(\bar{\zeta}_i) \cdot \text{Exp}(\bar{\zeta}_j)$. Hence,

$$\begin{aligned} \text{Var}\left(\frac{1}{m}\sum_{i=1}^{m} \zeta_i\right) &= \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} \text{Exp}(\bar{\zeta}_i \cdot \bar{\zeta}_j) \\ &= \frac{1}{m^2}\left(\sum_{i=1}^{m} \text{Exp}(\bar{\zeta}_i^2) + \sum_{1 \leqq i \neq j \leqq m} \text{Exp}(\bar{\zeta}_i)\,\text{Exp}(\bar{\zeta}_j)\right) \\ &= \frac{1}{m^2} \cdot m \cdot \text{Exp}(\bar{\zeta}_1{}^2) \\ &< \frac{1}{4m}. \end{aligned}$$

Thus, $\Pr(1/m\sum_{i=1}^{m} \zeta_i \geqq \frac{1}{2}) < 4/m\varepsilon^2(n)$. The probability that $1/m\sum_{i=1}^{m} \zeta_i \geqq \frac{1}{2}$ is exactly the error probability of the parity subroutine PAR* on a single input $d, E_N(x)$. To summarize, we have proved

LEMMA 2. *Let* $d, x \in Z_N$ *and suppose that* $\text{abs}_N(dx) < \varepsilon(n)N/2$. Let $m \overset{\text{def}}{=} 64n \cdot \varepsilon^{-2}(n))$ *be the number of measurements done by* PAR*. *On input* $d, E_N(x)$, *the right alternative for* y *and* z, *and access to an* $\varepsilon(n)$*-oracle for* RSA *least-significant bit,* $\mathcal{O}_N$, *the probability that the parity subroutine* PAR* *outputs* $\text{par}_N(dx)$ *is at least* $1 - 1/(12n+6)$. *The probability space is that of all choices of* $y, z \in Z_N$ *and all internal coin tosses of* $\mathcal{O}_N$. *In other words, with the right alternative for* y *and* z *as an auxiliary input, the parity subroutine* PAR* *is* $(\varepsilon(n), 1/(12n+6))$*-reliable.*

4.4. Main theorem. Combining Lemmas 1 and 2, we get the following theorem.

THEOREM 1. *RSA least-significant bit is unpredictable.*

Proof. In Lemma 2, we have analyzed the success probability of PAR*, assuming that it is given the right alternative for y and z as an auxiliary input. When executing the RSA inversion procedure, only one of the copies of PAR* satisfies this condition, but this is good enough.

We conclude the proof by calculating the overall expected running time of the RSA inversion algorithm, given an $\varepsilon(n)$-oracle for the least-significant bit. We count elementary Z_N operations (addition, multiplication, division), RSA encryptions, and oracle calls at unit cost. The expected number of times that Step 1 of the RSA inversion procedure is repeated equals $O(\varepsilon^{-2}(n))$. For each execution of Step 1, $O(m\varepsilon^{-2}(n))$ copies of the gcd procedure are invoked. Each copy makes $O(n)$ calls to PAR*. The parity subroutine, in turn, makes $O(m)$ operations. Multiplying these terms and substituting $m = 64n\varepsilon^{-2}(n)$, the overall expected run time is

$$O(\varepsilon^{-4}(n) \cdot m^2 n) = O(\varepsilon^{-8}(n) \cdot n^3). \qquad \square$$

5. Extensions. By reductions due to Ben-Or, Chor and Shamir [2], we get

COROLLARY (to Theorem 1).

(a) *Let $I \subset [0, N-1]$ be an interval of length $N/2$. The I-*bit *of x is the characteristic function of I (i.e., 1 if $x \in I$ and 0 otherwise). This bit is unpredictable.*

(b) *The k*th *least-significant bit is $(1/4+(1/2^{n-k})+(1/2^k)+1/\text{poly}(n))$-secure. At least half of these bits are $(1/6+(1/2^{n-k})+(1/2^k)+1/\text{poly}(n))$-secure.*

5.1. Simultaneous security.

DEFINITION. We say that the j least-significant bits are *simultaneously secure* if inverting E_N is polynomial-time reducible to distinguishing, given $E_N(x)$, between the string of j least-significant bits of x and a randomly selected j-bit string.

We have defined the notion of simultaneous security in terms of an indistinguishability test. It is also possible to define simultaneous security in terms of an unpredictability test: Given $E_N(x)$ and the $j-1$ least-significant bits of x, the jth least-significant bit of x is still $1/\text{poly}(n)$ secure. Yao [29] has shown that the indistinguishability test is equivalent to the unpredictability test. (Although Yao's proof was given in a different setting, it still applies here.)

Our proof technique easily extends to show that $\log n$ least-significant bits pass the unpredictability test.

THEOREM 2. *Let $j \stackrel{\text{def}}{=} O(\log n)$.*

(a) *The j*th *least-significant bit in the binary expansion of the plaintext is $1/\text{poly}(n)$ secure.*

(b) *The j least-significant bits of the plaintext are simultaneously secure.*

Proof. (a) First note that when generating y and z, it is feasible to guess not only their 1st least-significant bit, but all j least-significant bits of y and z. The overhead for trying all possibilities is 2^{2j}, which is polynomial in n. Together with the locations of y and z, these bits will determine (with high probability) all j least-significant bits of each $[r_i x]_N$. Also, with probability about 2^{-2j}, the gcd of $[ax]_N$ and $[bx]_N$ is 2^{j-1} (instead of 1), which we will assume is the case. This way all $[dx]_N$'s in the gcd calculation will have zeros in all $j-1$ least-significant bits. Finally, we replace all references to the least-significant bit in the inverting algorithm, by references to the jth least-significant bit. This can be done since we now have access to an oracle for the jth least-significant bit. The reader may find it convenient to view this process as taking the gcd of $[ax/2^{j-1}]_N$ and $[bx/2^{j-1}]_N$. (This method of transforming certain inverting algorithms which use an oracle for the first least-significant bit into inverting

algorithms which use an oracle for the jth least-significant bit originates from Vazirani and Vazirani [27].)

(b) Going through the proof of part (a), notice that when querying the oracle about the jth least-significant bit of $[r_i x + dx]_N$ we can give it the $j-1$ previous bits of $[r_i x + dx]_N$. This is the case since, if no wraparound occurs, these $j-1$ bits are the same as the $j-1$ least-significant bits of $[r_i x]_N$, which we know. □

Remark. Vazirani and Vazirani [28] had previously shown that certain inverting algorithms which use an $\varepsilon(n)$-oracle for RSA least-significant bit, can be transformed into inverting algorithms which use an $\varepsilon(n)$-oracle for predicting x_j (given $x_{j-1}, \cdots, x_1$). It turns out that the inverting algorithm of § 4 falls into the above category [28]; this yields an alternative (but much harder) way of proving Theorem 2(b).

5.2. Multi-prime moduli with partial factorization. The results about bit security for the RSA function were described with respect to composite numbers N which are the product of two large primes. However, the same proofs hold for the case of multi-prime composite $N = p_1 p_2 \cdots p_k$, where the exponent e is relatively prime to $\varphi(N)$. In fact, exactly the same proofs hold also in the case that partial factorization of the modulus is given. Namely, given N, e and some of the p_i's, the following tasks are computationally equivalent:

(1) Given $E_N(x)$, find x;

(2) Given $E_N(x)$, guess the least-significant bit of x with success probability $\frac{1}{2} + 1/\text{poly}(n)$.

In this context, one may wonder whether RSA remains hard to invert given partial factorization of its modulus. Using the Chinese Remainder Theorem it is not hard to show that inverting E_M ($M = p_1 p_2$) is equivalent to inverting E_N ($N = p_1 p_2 p_3 \cdots p_k$ and both exponents in E_M and E_N are the same) when all primes but p_1, p_2 are known. For details see [8].

6. Bits equivalent to factoring in Rabin's encryption function. The Rabin encryption function is operating on the message space Z_N, where $N = pq$ is the product of two large primes (which are kept secret). The encryption of x is $E_N(x) = [x^2]_N$. The ciphertext space is $Q_N \overset{\text{def}}{=} \{y \mid \exists x\colon y \equiv x^2 \pmod N\}$. Rabin [22] has shown that extracting square roots ("inverting E_N") is polynomially equivalent to factoring.

6.1. Previous results. The function E_N defined above is 4 to 1 rather than being 1 to 1 (as is the case in the RSA). Blum [3] has pointed out the cryptographic importance of the fact that for $p \equiv q \equiv 3 \pmod 4$, E_N induces a permutation over Q_N. Composite numbers of this form will be called *Blum integers.*

Goldwasser, Micali and Tong [17] have presented a predicate the evaluation of which is as hard as factoring. Specifically, they showed that if $p \equiv 3 \pmod 4$ and $p \equiv q \pmod 8$ then factoring N is polynomially reducible to guessing their predicate with success probability $1 - (1/n)$.

Ben-Or, Chor and Shamir [2] considered the same predicate. Using a modification of their RSA techniques, they showed $\frac{1}{4} + 1/\text{poly}(n)$ security also for this predicate. Their modification requires that N be a Blum integer and furthermore that there exist a small odd number h ($h = \text{poly}(n)$) with $(h/N) = -1$. The correctness proof makes use of nonelementary number theory.

6.2. Our result. Using the techniques of § 4, we show that the least-significant bit in a variant of Rabin's encryption function is also unpredictable. Our proof uses only elementary number theory, and holds for all Blum integers.

Throughout this section we make use of the *Jacobi symbol.* Let us review the definition and some properties of the Jacobi symbol. Let p be an odd prime number, and h an integer relatively prime to p. The *Legendre symbol* (h/p) is defined to be 1 if h is a quadratic residue modulo p, and -1 otherwise. For $N = pq$, a product of two odd primes, and h relatively prime to N, the *Jacobi symbol* (h/N) is defined to be $(h/p) \cdot (h/q)$. Even though the definition of the Jacobi symbol uses the factorization of N, it can be easily computed even if N's factorization is not given. Other facts which are used in this section are: $(h \cdot h'/N) = (h/N) \cdot (h'/N)$, and for a Blum integer N, $(-1/N) = 1$. For further details, see [21, Chap. 3].

Let N be a Blum integer. Define

$$S_N \overset{\text{def}}{=} \left\{ x \,\middle|\, 0 \leqq x < \frac{N}{2} \right\},$$

$$M_N \overset{\text{def}}{=} \left\{ x \,\middle|\, 0 \leqq x < \frac{N}{2} \wedge \left(\frac{x}{N}\right) = 1 \right\}.$$

Redefine E_N for $x \in M_N$ as

$$E_N(x) = \begin{cases} [x^2]_N & \text{if } [x^2]_N < \dfrac{N}{2}, \\ [N - x^2]_N & \text{otherwise.} \end{cases}$$

This makes E_N a $1-1$ mapping from M_N onto itself. The intractability result of Rabin still holds. That is, factoring N is polynomially reducible to inverting E_N. Let $L_N(x)$ denote the least-significant bit of x.

The security of the least-significant bit of this function is now defined in a manner analogous to the RSA case. We would like to use the same techniques to demonstrate that the least-significant bit of the modified Rabin function is unpredictable. The difficulty is that the queries to the oracle may not be of the right form. Namely, we would like to feed the oracle with $E_N(r_i x + dx)$ and get the least-significant bit of $[r_i x + dx]_N$, but it might happen that $[r_i x + dx]_N \notin M_N$ and then the oracle's answer does not correspond to $[r_i x + dx]_N$ (but rather to the square root of $[(r_i x + dx)^2]_N$ which resides in M_N). This ($[r_i x + dx]_N \notin M_N$) may happen if either $[r_i x + dx]_N \notin S_N$ or $((r_i x + dx)/N) = -1$. Both cases are easy to detect with very high probability. When any of them occur, we discard this dx-measurement. We will show that we only discard about $\frac{3}{4}$ of the dx-measurements, and the remaining points constitute a large enough sample to retain the high reliability of the parity subroutine. A more elaborate exposition follows.

For technical reasons, we slightly change the definition of "small" here. In this section, h is small means $\text{abs}_N(h) < \varepsilon(n)N/8$ (instead of $\text{abs}_N(h) < \varepsilon(n)N/2$ as in § 4). This will restrict all $[dx]_N$'s in the gcd calculation to have $\text{abs}_N(dx) < \varepsilon(n)N/8$. Doing this, the probability that a wraparound of either 0 or $N/2$ occurs when $[dx]_N$ is added to $[r_i x]_N$ is no greater than $\varepsilon(n)/4$. Similarly, the partition of Z_N (for both y and z) is refined by a factor of 4.

Given the original encryption $E_N(x)$, pick $y = kx$ and $z = lx$, two random multiples of x. By exhausting all possibilities, the approximate magnitude in Z_N of y and z, and their least-significant bits are known. Let $[r_i x]_N = [y + iz]_N$ as before. If $[r_i x]_N$ is not in an $\varepsilon(n)N/8$ interval around either 0 or $N/2$, then we can determine whether $[r_i x]_N \in S_N$, and compute the least-significant bit of $[r_i x]_N$ as before. In the "fuzzy" cases, where $[r_i x]_N$ is in an $\varepsilon(n)N/8$ interval around either 0 or $N/2$, we determine its least-significant bit assuming that $[r_i x]_N \in S_N$.

It remains to determine the parity of $[dx]_N$ by comparing the known least-significant bit of $[r_i x]_N$ with the least-significant bit of $[r_i x + dx]_N$. We consider the following three cases:

(1) If $[r_i x]_N \notin S_N$ (according to y, z locations), then we ignore this dx-measurement;
(2) If $((r_i + d)/N) = -1$, then we ignore this dx-measurement;
(3) If $[r_i x]_N \in S_N$ (according to y, z locations) and $((r_i + d)/N) = 1$, then we feed the oracle $\mathcal{O}_N$ with $E_N(r_i x + dx)$, and take its answer as our guess for the least-significant bit of $[r_i x + dx]_N$.

In the analysis of this procedure, we assume that we are dealing with the right alternative for y and z. With high probability ($\geqq 1 - (\varepsilon(n)/8)$) we correctly determine whether $[r_i x]_N \in S_N$.

We first estimate the probability that we ignore the dx-measurement. Since the cardinality of M_N is $|Z_N^*|/4$, and our only error is in testing membership in S_N, we end up in Case (3) with probability $\geqq \frac{1}{4} - \varepsilon(n)/8$.

We now estimate the error probability given that we are in Case (3). It is easier to estimate the error given that $[r_i x]_N \in S_N$ and $((r_i + d)/N) = 1$ (i.e., given that we should have been in Case (3)). These two conditional probabilities are very close, as we will see below. Let $\mathcal{A}$ denote the event that we produce an incorrect value for the ith dx-measurement, $\mathcal{B}$ denotes the event that we are in Case (3), and $\mathcal{C}$ denotes the event $[r_i x]_N \in S_N$ and $((r_i + d)/N) = 1$.

$$\begin{aligned}\Pr(\mathcal{A} \mid \mathcal{B}) &= \Pr(\mathcal{A} \cap \mathcal{C} \mid \mathcal{B}) + \Pr(\mathcal{A} \cap \bar{\mathcal{C}} \mid \mathcal{B}) \\ &\leqq \Pr(\mathcal{A} \mid \mathcal{C}) \cdot \frac{\Pr(\mathcal{C})}{\Pr(\mathcal{B})} + \Pr(\bar{\mathcal{C}} \mid \mathcal{B}).\end{aligned}$$

The reader can easily verify that

$$\Pr(\bar{\mathcal{C}} \mid \mathcal{B}) < \frac{\varepsilon(n)}{4},$$

$$\Pr(\mathcal{C}) \leqq \frac{1}{4} + \frac{\varepsilon(n)}{8},$$

$$\Pr(\mathcal{B}) \geqq \frac{1}{4} - \frac{\varepsilon(n)}{8}.$$

The error we would have made when $[r_i x]_N \in S_N$ and $((r_i + d)/N) = 1$ stems from two sources.[3] The oracle's error and the possibility that $[r_i x + dx]_N \notin S_N$ (although $[r_i x]_N \in S_N$). The first conditional probability is bounded above by $\frac{1}{2} - \varepsilon(n)$ and the second by $\varepsilon(n)/8$. Thus, $\Pr(\mathcal{A} \mid \mathcal{C}) < \frac{1}{2} - \varepsilon(n) + \varepsilon(n)/8$. Using the calculations above, we get

$$\begin{aligned}\Pr(\mathcal{A} \mid \mathcal{B}) &\leqq \left(\frac{1}{2} - \frac{7\varepsilon(n)}{8}\right) \cdot \frac{\frac{1}{4} + \varepsilon(n)/8}{\frac{1}{4} - \varepsilon(n)/8} + \frac{\varepsilon(n)}{4} \\ &< \frac{1}{2} - \frac{\varepsilon(n)}{8}.\end{aligned}$$

Define the random variable

$$\zeta_i = \begin{cases} \frac{1}{2} & \text{if the } i\text{th } dx\text{-measurement is ignored,} \\ 1 & \text{if the } i\text{th } dx\text{-measurement is wrong,} \\ 0 & \text{if the } i\text{th } dx\text{-measurement is correct.} \end{cases}$$

[3] In case $[r_i x]_N \in S_n$ we have determined correctly the least-significant bit of $[r_i x]_N \in S_N$. This follows by the manner in which we determine the least-significant bit of $[r_i x]_N$.

The reader can easily verify that $\text{Exp}(\zeta_i) = \Pr(\zeta_i = 1) < \frac{1}{2} - \varepsilon(n)/64$ and $\text{Var}(\zeta_i) < \frac{1}{4}$. The rest of the analysis is similar to the analysis presented in § 4 (as also here the probability that $(1/m)\sum_{i=1}^{m} \zeta_i \geqq \frac{1}{2}$ is exactly the error probability of the parity subroutine). This implies

THEOREM 3. *The least-significant bit for the modified Rabin encryption function is unpredictable. (That is, inverting $E_N(\cdot)$ is probabilistic polynomial-time reducible to the following: Given $E_N(x)$ (for $x \in M_N$), guess the least-significant bit of x with success probability $\frac{1}{2} + 1/\text{poly}(n)$.)*

COROLLARY (to Theorem 3). *Factoring a Blum integer, N, is polynomially reducible to guessing $L_N(x)$ with success probability $\frac{1}{2} + 1/\text{poly}(n)$ when given $E_N(x)$, for $x \in M_N$.*

The proofs from the previous section about simultaneous security of the $\log n$ least significant bits hold here just as well. The extension of the result to multi-prime moduli is possible, but much harder. For details see [10].

7. Applications. In this section we present applications of our result to the construction of pseudorandom bit generators and probabilistic encryption schemes.

7.1. Construction of pseudorandom bit generators. A pseudorandom bit generator is a device that "expands randomness." Given a truly random bit string s (the seed), the generator expands the seed into a longer pseudorandom sequence. The question of "how random" this pseudorandom sequence is depends on the definition of randomness we use. A strong requirement is that the expanded sequence will pass all polynomial time statistical tests. Namely, given a pseudorandom and a truly random sequence of equal length, no probabilistic polynomial time algorithm can tell which is which with success probability greater than $\frac{1}{2}$ (this definition was proposed by Yao [29], who also showed it is equivalent to another natural definition—unpredictability [6]).

Blum and Micali [6] presented a general scheme for constructing such *strong pseudorandom* generators. Let $g: M \to M$ be a $1-1$ one-way function, and $B(x)$ be an unpredictable predicate for g. Starting with a random $s \in M$, the sequence obtained by iterating g and outputting the bit $b_i = B(g^i(s))$ for each iteration is strongly pseudorandom. Using their unpredictability result for the "$half_p$ bit" in discrete exponentiation modulo a prime p, Blum and Micali gave a concrete implementation of the scheme, based on the intractability assumption of computing discrete logarithm. More generally, if $B_1(x), \cdots, B_k(x)$ are simultaneously secure bits for g, then the sequence obtained by iterating g, and outputting the string $\langle b_{i,1} \cdots b_{i,k} \rangle = \langle B_1(g^i(s)) \cdots B_k(g^i(s)) \rangle$ for each iteration, is strongly pseudorandom. Long and Wigderson [20] have shown that the discrete exponentiation function has $\log \log p$ simultaneous secure bits.[4] Their result implies a pseudorandom bit generator which produces $\log \log p$ bits per each iteration of the discrete exponentiation.

Using our results, we get an efficient implementation of strong pseudorandom generators, based on the intractability assumption of factoring. The modified Rabin function E_N is iteratively applied to the random seed $s \in M_N$. In the ith iteration, the generator outputs the $\log n$ least-significant bits of $E_N^i(s) = \pm s^{2^i} \bmod N$. Thus it outputs $\log n$ pseudorandom bits at the cost of one squaring and one subtraction modulo N, and is substantially faster than the discrete exponentiation generator. Previous strong pseudorandom generators based on factoring ([17], [2], [27]) required the use of the exclusive-or construction of Yao [29] and were less efficient.

Another efficient pseudorandom generator was previously constructed by Blum, Blum and Shub [4]. Their generator output one pseudorandom bit per one modular

[4] Kaliski [18] has recently simplified and generalized the argument.

multiplication. Blum, Blum and Shub proved that their generator is a strong pseudorandom generator if the problem of deciding quadratic residucity modulo a composite number is intractable. Vazirani and Vazirani [28] have pointed out that, using our techniques, the Blum, Blum and Shub generator is strong also with respect to the problem of factoring Blum integers.

7.2. Construction of probabilistic public-key encryption schemes. A probabilistic encryption scheme is said to leak no partial information if the following holds: *Whatever is efficiently computable about the plaintext given the ciphertext, is also efficiently computable without the ciphertext* [15]. Goldwasser and Micali presented a general scheme for constructing public-key probabilistic encryption schemes which leak no partial information, using a "secure trap-door predicate" [15]. A *secure trap-door predicate* is a predicate that is easy to evaluate given some "trap-door" information, but infeasible to guess with the slightest advantage without the "trap-door" information. Goldwasser and Micali also gave a concrete implementation of their scheme, under the intractability assumption of deciding quadratic residucity modulo a composite number. A drawback of their implementation is that it expands each plaintext bit into a ciphertext block (of length equal to that of the composite modulus).

Using our results, we get an implementation of a probabilistic public-key encryption scheme that leaks no partial information, based on the intractability assumption of factorization. This implementation is more efficient that the one in [14], which is also based on factoring. However, our implementation still suffers from a large bandwidth expansion.

Recently, Blum and Goldwasser [5] used our result to introduce a new implementation of probabilistic encryption, equivalent to factoring, in which the plaintext is only expanded by a *constant factor.* Blum and Goldwasser's scheme is approximately as efficient as the RSA while provably leaking no partial information, provided that factoring is intractable.

8. Concluding remarks and open problems. Standard sampling techniques draw mutually independent elements from a large space. We employed a strategy of getting elements with limited mutual independence (only pairwise independence). This strategy allows more control on properties of the chosen elements.

Trading off statistical independence for control turned out to be fruitful in our context. We believe that such trade-off may be useful in other contexts as well.

We conclude by presenting two open problems:

(1) In § 5 (6), we have shown simultaneous security results of $O(\log n)$ bits in RSA (Rabin) encryption function. Extending the result beyond $O(\log n)$ bits is of major theoretical and practical importance. In particular, if more bits are shown to be simultaneously secure, then the efficiency of the resulting pseudorandom generator will be greatly improved.

(2) Another interesting question is that of investigating the bit security of the internal RSA bits—are they also $1/\text{poly}(n)$ secure?

Acknowledgments. We would like to thank Michael Ben-Or, Shafi Goldwasser, Silvio Micali, Ron Rivest and Adi Shamir for very helpful discussions and useful ideas.

REFERENCES

[1] 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 Symp. on Foundations of Computer Science, 1984, pp. 449-457.

[2] M. BEN-OR, B. CHOR AND A. SHAMIR, *On the Cryptographic Security of Single* RSA *Bits*, Proc. 15th ACM Symp. on Theory of Computation, April 1983, pp. 421–430.
[3] M. BLUM, *Coin Flipping by Telephone*, IEEE Spring COMCON, 1982.
[4] L. BLUM, M. BLUM AND M. SHUB, *A simple unpredictable pseudo-random number generator*, this Journal, 15 (1986), pp. 364–383.
[5] M. BLUM AND S. GOLDWASSER, *An efficient probabilistic encryption scheme which hides all partial information*, in Advances in Cryptology: Proceedings of Crypto84, G. R. Blakely and D. Chaum, eds., Springer-Verlag, Berlin, New York, 1985, pp. 288–299.
[6] M. BLUM AND S. MICALI, *How to generate cryptographically strong sequences of pseudo-random bits*, this Journal, 13 (1984), pp. 850–864.
[7] R. P. BRENT AND H. T. KUNG, *Systolic VLSI arrays for linear time gcd computation*, VLSI 83, IFIP, F. Anceau and E. J. Aas, eds., Elsevier Science Publishers, 1983, pp. 145–154.
[8] B. CHOR, *Two Issues in Public Key Cryptography*, MIT Press, Cambridge, MA, 1986.
[9] B. CHOR AND O. GOLDREICH, RSA/*Rabin least significant bits are* $\frac{1}{2}+1/\text{poly}(\log N)$ *secure*, in Advances in Cryptology: Proceedings of CRYPTO84, G. R. Blakely and D. Chaum, eds., Springer-Verlag, Berlin, New York, 1985, pp. 303–313. (Also available as Technical Memo TM-260, Laboratory for Computer Science, Massachusetts Inst. of Technology, May 1984.)
[10] B. CHOR, O. GOLDREICH AND S. GOLDWASSER, *The bit security of modular squaring given a partial factorization of the modulus*, in *Advances in Cryptology—Proceedings of CRYPTO*85, H. C. Williams, ed., Springer-Verlag, Berlin, 1986, pp. 448–457.
[11] W. DIFFIE AND M. E. HELLMAN, *New Directions in Cryptography*, IEEE Trans. Inform. Theory, Vol. IT-22 (1976), pp. 644–654.
[12] W. FELLER, *An Introduction to Probability Theory and its Applications*, Vol. I, John Wiley, New York, 1962.
[13] O. GOLDREICH, *On the number of close-and-equal pairs of bits in a string* (*with implications on the security of* RSA's *l.s.b.*), in Advances in Cryptology: Proceedings of EuroCrypt84, T. Beth et al., eds., Springer-Verlag, Berlin, New York, 1985, pp. 127–141. (Also available as Technical Memo TM-256, Laboratory for Computer Science, MIT, March 1984.)
[14] S. GOLDWASSER, *Probabilistic encryption*: *Theory and applications*, Ph.D. thesis, Univ. of California, Berkeley, 1984.
[15] S. GOLDWASSER AND S. MICALI, *Probabilistic encryption*, J. Comp. System Sci., 28 (1984), pp. 270–299.
[17] S. GOLDWASSER, S. MICALI AND P. TONG, *Why and How to Establish a Private Code on a Public Network*, Proc. 23rd IEEE Symp. on Foundations of Computer Science, November 1982, pp. 134–144.
[18] B. S. KALISKI, *A pseudo-random bit generator based on elliptic logarithms*, M.Sc. thesis, MIT, Cambridge, MA, 1987.
[19] D. E. KNUTH, *The Art of Computer Programming, Vol.* 2, *Seminumerical Algorithms*, 2nd edition, Addison-Wesley, Reading, MA, 1981.
[20] D. L. LONG AND A. WIGDERSON, *How discrete is discrete log?*, Proc. 15th ACM Symp. on Theory of Computation, April 1983, pp. 413–420. (A better version is available from the authors.)
[21] I. NIVEN AND H. S. ZUCKERMAN, *An Introduction to the Theory of Numbers*, John Wiley, New York, 1980.
[22] M. O. RABIN, *Digital signatures and public key functions as intractable as factorization*, Technical Memo TM-212, Laboratory for Computer Science, Massachusetts Inst. of Technology, 1979.
[23] R. L. RIVEST, A. SHAMIR AND L. ADLEMAN, *A method for obtaining digital signature and public key cryptosystems*, Comm. ACM, 21 (1978), pp. 120–126.
[24] C. P. SCHNORR AND W. ALEXI, RSA *bits are* $0.5+\varepsilon$ *secure*, in Advances in Cryptology: Proceedings of EuroCrypt84, T. Beth et al., eds., Springer-Verlag, Berlin, New York, 1985, pp. 113–126.
[25] A. SHAMIR, *On the generation of cryptographically strong pseudo-random number sequences*, ACM Trans. Comput. Systems, 1 (1983), pp. 38–44.
[27] U. V. VAZIRANI AND V. V. VAZIRANI, RSA *bits are* $.732+\varepsilon$ *secure*, in Advances in Cryptology: Proceedings of CRYPTO83, D. Chaum, ed., Plenum Press, New York, 1984, pp. 369–375.
[28] ———, *Efficient and secure pseudo-random number generation*, Proc. 25th IEEE Symp. on Foundations of Computer Science, 1984, pp. 458–463.
[29] A. C. YAO, *Theory and applications of trapdoor functions*, Proc. 23rd IEEE Symp. on Foundations of Computer Science, 1982, pp. 80–91.