MIT/LCS/TM-260

RSA/RABIN LEAST SIGNIFICANT BITS ARE $\frac{1}{2} + \frac{1}{\text{poly}(\log N)}$ SECURE

Benny Chor

Oded Goldreich

May 1984

RSA/Rabin least significant bits are $\frac{1}{2} + \frac{1}{poly(\log N)}$ secure

Benny Chor * Oded Goldreich **

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

May 10, 1984

Abstract —- We prove that RSA least significant bit is $\frac{1}{2} + \frac{1}{log^c N}$ secure, for any constant c (where N is the RSA modulus). This means that an adversary, given the ciphertext, cannot guess the least significant bit of the plaintext with probability better than $\frac{1}{2} + \frac{1}{log^c N}$, unless he can break RSA.

Our proof technique is strong enough to give, with slight modifications, the following related results:
(1) The $\log\log N$ least significant bits are simultaneously $\frac{1}{2} + \frac{1}{log^c N}$ secure.
(2) The above also holds for Rabin's encryption function.

Our results imply that Rabin/RSA encryption can be **directly** used for pseudo random bits generation, provided that factoring/inverting RSA is hard.

Keywords: Cryptography, Public Key Cryptosystems, Security of Partial Information, RSA Scheme, Factoring Integers, Pseudo-Random Bit Generators, Probabilistic Encryption, Predicate Reductions.

---

* Research supported in part by the National Science Foundation under Grant MCS-8006938.

** Research supported in part by a Weizmann Postdoctoral Fellowship.

1. Introduction

Given a secure public key cryptosystem [5], it is hard to recover the plaintext x from its encryption, $E(x)$. However, this does not necessarily mean that a cryptanalyst cannot gain some partial information about x without actually computing it. The ability to derive partial information can render a cryptosystem useless in specific applications (e.g. mental poker [17],[12],[10]). For example, even a moderate ability of guessing the least significant bit of the plaintext may be a threat to security.

In the current state of knowledge we are unable to prove even the existence of secure public key cryptosystems. However, under reasonable assumptions on the computational complexity of certain problems, secure public key cryptosystems do exist and can be explicitly constructed. One of the most fascinating questions regarding those systems is *"what partial information about the plaintext is hard to extract from the ciphertext?"*

This question was rigorously defined and studied, with respect to probabilistic encryption, by Goldwasser and Micali [10]. They constructed a public key cryptosystem which leaks no partial information. However, their system encrypts messages by expanding each plaintext bit into a ciphertext block, making it undesirable from a practical point of view.

The RSA [15] is the most widely known public key cryptosystem, and probably the first one which will be used in practice. It has been an open problem to demonstrate a predicate, $P(\cdot)$, such that having any advantage in guessing $P(x)$ given the encryption of x, is as hard as inverting RSA.

In this paper, we show that RSA least significant bit is $\frac{1}{2} + \epsilon$ secure for any polynomial fraction ϵ ($\epsilon^{-1} = O(log^c N)$), where N is the RSA modulus. With a small modification, our proof technique also allows us to show that $\log\log N$ of RSA bits are **simultaneously** $\frac{1}{2} + \frac{1}{\log^c N}$ secure. I.e., if RSA is indeed secure, then no heuristic which runs in polynomial time can get information about any function of these plaintext bits, given the ciphertext. Hence these bits provide instances of secure partial information for RSA.

Our results have important implications for generating sequences of cryptographically strong pseudo-random bits. RSA encryption E can be **directly** used for generating such sequences by starting from a random seed s and iterating E on it.

Slightly modifying our proof techniques, we also prove the same strong bit security for Rabin public key scheme [14]. This implies a fast and "direct" pseudo-random bits generator which is as hard to crack (distinguish its outputs from truly random strings) as factoring. Important consequences follow also w.r.t the probabilistic encryption scheme of Goldwasser & Micali [10] (see section 8.2).

Organization of the paper : In section 2 we formally define the question of security for RSA least significant bit and cover previously known results. In section 3 we sketch the proof of Ben-Or, Chor & Shamir result, and in section 4 - its improvement by Schnorr & Alexi. These two investigations are the basis for our work, which is described in section 5. Section 6 extends our proof to other RSA bits and section 7 - to bits in Rabin's scheme. Section 8 contains concluding remarks on the applications of our results for the **direct** construction of pseudo-random bit generators and probabilistic encryption schemes.

2. Problem definition and Previous Results

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 $E_N(x) = x^e \pmod N$, where e is relatively prime to $\varphi(N) = (p-1)(q-1)$. For $0 \leq x < N$, $L(x)$ denotes the least significant bit in the binary representation of x.

Let $\mathcal{O}_N$ be an oracle which, given $E_N(x)$, outputs a guess for $L(x)$ (this guess might depend on a random coin used by $\mathcal{O}_N$). Let $p(N)$ be a function from integers into the interval $[\frac{1}{2}, 1]$. We say that $\mathcal{O}_N$ is a $p(N)$-oracle if the probability that the oracle is correct, given $E_N(x)$ as its input, is $p(N)$ (the probability space is that of all $x \in Z_n$ with uniform distribution and -if $\mathcal{O}_N$ uses a random coin- also of all $0-1$ sequences of coin tosses with uniform distribution).

We say that RSA least significant bit is $p(N)$-secure if there is a probabilistic polynomial time algorithm which inverts E_N, using queries of any $p(N)$-oracle $\mathcal{O}_N$. Since an unbiased coin can be used as an $\frac{1}{2}$-oracle, the best possible security result can be $\frac{1}{2} + \epsilon$ security for any $\epsilon^{-1} = poly(logN)$ ($\frac{1}{2}$ security means RSA is breakable). These notions originate from Blum & Micali's work [4], where they have been stated w.r.t the discrete exponentiation function.

Goldwasser, Micali and Tong [11] showed that the least significant bit is as hard to compute as inverting the RSA. Furthermore, they showed that it is $(1 - \frac{1}{\log N})$-secure.

Ben-Or, Chor and Shamir [1] showed a $\frac{3}{4} + \epsilon$ security ($\epsilon^{-1} = poly(logN)$). They presented an algorithm which inverts the RSA by carrying out a gcd calculation on two multiples of the ciphertext and using any $(\frac{3}{4} + \epsilon)$-oracle. Sampling the oracle they amplified its $\frac{3}{4} + \epsilon$ advantage to "almost certainty", for a polynomial fraction of the massage space.

Vazirani and Vazirani [18] improved the result using a novel oracle-sampling technique. They proved that their modification is guaranteed to succeed when given access to any 0.732-oracle.

Goldreich [7] used a better combinatorial analysis to show that the Vazirani & Vazirani modification inverts even when given access to a 0.725-oracle. He also pointed out some limitations of the Vazirani & Vazirani and similar proof techniques.

Schnorr and Alexi [16] introduced a conceptual change in the way the oracle is used. This enabled them to greatly improve the result showing that the least significant bit is $(\frac{1}{2} + \epsilon)$-secure for any constant $\epsilon > 0$. Their result still leaves a gap towards the optimal $\frac{1}{2} + \frac{1}{poly(\log N)}$ security.

3. A Sketch of Ben-Or Chor & Shamir Algorithmic Procedure

The essence of the Inverting Algorithm:

Given an encrypted message, $E_N(x)$, the plaintext x is reconstructed by performing a *gcd* algorithm on two small multiples of it (small means in the interval $[\frac{-N\epsilon}{2}, \frac{N\epsilon}{2}] \pmod N$). A special binary variant is used for the *gcd* algorithm. To operate, this variant needs to know the parity of the absolute value of $O(log^2 N)$ small multiples of the plaintext. Thus, it is provided with a *subroutine* that determines the parity of these multiples.

Determining Parity using an Oracle which may err:

The *subroutine* determines the parity of a small multiple $d = kx$, of the plaintext ,x, by using an $p(N)$-oracle for RSA's l.s.b as follows. It picks a random r and asks the oracle for the least significant bit of both rx and $rx + d$, by feeding it in turn with $E_N(rx) = E_N(r)E_N(x)$ and $E_N((r+k)x) = E_N(r+k)E_N(x)$. The oracle's answers are processed according to the following observation. Since $d = kx$ is "small", with very high probability no wrap around 0 occurs when

d is added to rx. Then, the parity of $|d|$ is equal to 0 if the least significant bits of rx and $rx+d$ are identical; and equal to 1 otherwise. This is repeated many times; every repetition (instance) is called a d-measurement. Note that the outcome of a d-measurement is correct if the oracle was correct on both rx and $rx+d$ (the outcome is also correct if the oracle was wrong on both queries, but this fact is not used in [1]).

(Trivial) Measurement Analysis:

A d-measurement is correct with probability at least $1-2(1-p)=2p-1$.

(This suffices if $p=\frac{3}{4}+\epsilon$.)

4. A Sketch of Schnorr & Alexi Improvement: $\frac{1}{2}+\epsilon$ for any constant ϵ

Schnorr & Alexi [16] improvement is based on trying all possibilities for the least significant bit of $L=\theta(\log\log N)$ random, independent positions $w_i=r_ix$ and using these positions as "end points" in all measurements for the $O(log^2N)$ d's of the binary gcd algorithm. This way the oracle is queried only about one end-point of each measurement and the error is caused by single position queries rather than by pairs of positions. This enables the error probability per a single measurement to be approximately the oracle's error, rather than twice this magnitude as in Ben-Or, Chor & Shamir. Using the fact that the L positions are independent, Chernoff bound implies that the error probability in deciding the parity of d by the majority of d-measurements is $2^{-\Omega(L\epsilon^2)}<\frac{1}{\log^3 N}$ (here ϵ is a constant). This guarantees that the accumulated error probability in deciding the parity of all $O(\log^2 N)$ d's in the modified binary gcd algorithm is $<\frac{1}{2}$, small enough to put the algorithm in random polynomial time.

Note that the running time of Schnorr & Alexi's algorithm is exponential in L. On the other hand, the probabilistic analysis requires that $L=\Omega(\frac{\log\log N}{\epsilon^2})$. Thus, ϵ can not be replaced by any function which tends to 0 with $N\to\infty$.

5. Our Main Result

In this section we prove that RSA least significant bit is $\frac{1}{2}+\frac{1}{poly(\log N)}$ secure.

Let $\mathcal{O}_N$ be an oracle for RSA least significant bit whose error probability is $\frac{1}{2}-\epsilon$, where $\epsilon^{-1}\leq\log^c N$.

Instead of picking $\theta(\log\log N)$ random **independent** positions, we generate $L=\theta(\log^{2c+3}N)$ random positions which are only **pairwise independent**, such that we know (with very high probability) the least significant bit of each. As in Schnorr and Alexi's work, we query the oracle only about one end-point of each measurement and use the same "decision by majority" idea. Since the positions are not independent, Chernoff bound cannot be used in our case. However, since the points are pairwise independent, Chebyshev's inequality still holds. It gives an $O(\frac{1}{L\epsilon^2})$ upper bound on the error probability. With L being so large, this error is sufficiently small.

Generating L "random" positions knowing their least significant bits

We generate L positions by picking two random independent variables $y,z\in Z_N$ and trying all possibilities for their least significant bits and location in one of the intervals $[i\frac{N}{L^3},(i+1)\frac{N}{L^3})$, $0\leq i<L^3$. There are $(2L^3)^2$ possibilities altogether, and exactly one of them is correct. Let us now assume that we are dealing with the correct choice, i.e. both least significant bit and approximate magnitude of y,z are known. The positions we'll look at are $w_i=y+iz \pmod N$ for

$i = 1, 2, \ldots, L$. Notice that w_i is a random element in Z_N with uniform probability distribution. Since the location of both y and z are known up to $\frac{N}{L^3}$, the location of $w_i = y + iz$ is known up to $\frac{N}{L^3} + \frac{iN}{L^3} < \frac{2N}{L^2}$. The probability of w_i to be within an interval of length $\frac{2N}{L^2}$ containing 0 (mod N) is exactly $\frac{2}{L^2}$. If w_i is **not** in such interval, then its least significant bit is determined by i and the least significant bits of x and y. Therefore we get

$$Pr(\text{ least significant bit of } w_i \text{ is unknown}) \leq \frac{2}{L^2} .$$

Determining parity using the generated positions and the oracle

Let $d \in Z_N$ be any fixed "small" number (one of those generated by the *gcd* procedure). In order to determine the parity of $|d|$, we'll query the oracle about all points of the form $w_i + d$, XOR the answers with the (known) least significant bits of the corresponding w_i, and take the majority.[1] Using Chebyshev's inequality, we'll get a bound for the probability that the majority of the oracle's answers will be biased to the wrong direction.

Error analysis :

Suppose $d \in Z_n$ is any "small" number (in the interval $[\frac{-N\epsilon}{2}, \frac{N\epsilon}{2}]$). For a random $r \in Z_N$, the probability that a wrap around 0 (mod N) occurs when d is added to r is no greater than $\frac{\epsilon}{2}$. Hence if $|d|$ is even, the probability that $\mathcal{O}_N$, on input $E_N(r + d)$, gives the same answer as the (true) least significant bit of r is at least $\frac{1}{2} + \epsilon - \frac{\epsilon}{2} = \frac{1}{2} + \frac{\epsilon}{2}$. Similarly, if $|d|$ is odd, then with probability at least $\frac{1}{2} + \frac{\epsilon}{2}$, $\mathcal{O}_N$ answer to the least significant bit of $r + d$ is different than the least significant bit of r. By the above discussion, we get

$$Pr(w_i + d \text{ did not wrap around and } \mathcal{O}_N \text{ is correct on it}) \geq \frac{1}{2} + \frac{\epsilon}{2} , \text{ for every } i .$$

Define

$$\varsigma_i = \begin{cases} 0 & \text{if } w_i + d \text{ did not wrap around and } \mathcal{O}_N \text{ is correct on it and } w_i \text{ l.s.b. is known} \\ 1 & \text{otherwise} \end{cases}$$

Hence

$$\begin{aligned} Pr(\varsigma_i = 0) &\geq Pr(w_i + d \text{ did not wrap around and } \mathcal{O}_N \text{ is correct on it}) \\ &\quad - Pr(w_i \text{ least significant bit is unknown}) \\ &\geq \frac{1}{2} + \frac{\epsilon}{2} - \frac{2}{L^2} \\ &> \frac{1}{2} + \frac{\epsilon}{4} \quad (\text{for } L > \sqrt{\frac{8}{\epsilon}}) . \end{aligned}$$

Therefore, $Exp(\varsigma_i) = Pr(\varsigma_i = 1) < \frac{1}{2} - \frac{\epsilon}{4}$ and

$$Var(\varsigma_i) = Exp(\varsigma_i^2) - Exp^2(\varsigma_i) = Exp(\varsigma_i) - Exp^2(\varsigma_i) = Exp(\varsigma_i)(1 - Exp(\varsigma_i)) < \frac{1}{4} .$$

---

[1] Notice that this decision procedure is exactly the one employed in Ben-Or, Chor & Shamir. The crucial difference is that they had to use the oracle's answer to find w_i's least significant bit, while we know it beforehand (with overwhelming probability).

Since $Exp(\varsigma_i) < \frac{1}{2} - \frac{\epsilon}{4}$, we get

$$Pr\left(\frac{1}{L}\sum_{i=1}^{L}\varsigma_i \geq \frac{1}{2}\right) \leq Pr\left(\left|\frac{1}{L}\sum_{i=1}^{L}\varsigma_i - Exp(\varsigma_i)\right| \geq \frac{\epsilon}{4}\right).$$

We can apply Chebyshev's inequality (see Feller [6, p. 219]) and get,

$$Pr\left(\left|\frac{1}{L}\sum_{i=1}^{L}\varsigma_i - Exp(\varsigma_i)\right| \geq \frac{\epsilon}{4}\right) \leq \frac{Var(\frac{1}{L}\sum_{i=1}^{L}\varsigma_i)}{(\epsilon/4)^2}.$$

Since y and z are independent random variables and $y + iz$, $y + jz$ are linearly independent for $i \neq j$, then w_i and w_j are also independent random variables for any $i \neq j$. Therefore, for any $i \neq j$, ς_i and ς_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 $\overline{\varsigma_i} = \varsigma_i - Exp(\varsigma_i)$. By pairwise independence $Exp(\overline{\varsigma_i}\cdot\overline{\varsigma_j}) = Exp(\overline{\varsigma_i})\cdot Exp(\overline{\varsigma_j})$. Hence,

$$Var\left(\frac{1}{L}\sum_{i=1}^{L}\varsigma_i\right) = Exp\left(\left(\frac{1}{L}\sum_{i=1}^{L}\overline{\varsigma_i}\right)^2\right) = \frac{1}{L^2}\sum_{i=1}^{L}\sum_{j=1}^{L}Exp(\overline{\varsigma_i}\cdot\overline{\varsigma_j})$$

$$= \frac{1}{L^2}\left(\sum_{i=1}^{L}Exp(\overline{\varsigma_i}^2) + \sum_{1\leq i\neq j\leq L}Exp(\overline{\varsigma_i})Exp(\overline{\varsigma_j})\right) = \frac{1}{L^2}\cdot L\cdot Exp(\overline{\varsigma_1}^2) < \frac{1}{4L}.$$

Thus the probability that $\frac{1}{L}\sum_{i=1}^{L}\varsigma_i \geq \frac{1}{2}$ is smaller than $\frac{4}{L\epsilon^2}$. But $Pr\left(\frac{1}{L}\sum_{i=1}^{L}\varsigma_i \geq \frac{1}{2}\right)$ is exactly the error probability for a single d. We query at most $\log^2 N$ d's in the course of the *gcd* computation and thus the error probability (for one binary *gcd*) is bounded by

$$\log^2 N \cdot Pr(\text{ error for a single } d).$$

Taking $L = \log^{2c+3} N$, the overall error probability is bounded from above by

$$\log^2 N \cdot \frac{4}{L\epsilon^2} \leq \frac{4log^2N}{\log^{2c+3} N(\log^{2c} N)^{-1}} = \frac{4}{\log N}.$$

The running time of the inverting algorithm is

$$O(\epsilon^{-2}L^6\log^2 N) = O(\log^{14c+20} N) = O(\epsilon^{-14}\log^{20} N).$$

Hence we can recover the original message in random polynomial time, as desired. This implies **Theorem 1:** RSA least significant bit is $(\frac{1}{2} + \frac{1}{\log^c N})$-secure, for any constant c.

6. Other RSA bits

Our proof technique easily extends to provide strong security results for several other RSA bits. In particular the following holds:

Theorem 2:

a) Let $I \subseteq [0, N]$ 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 $(\frac{1}{2} + \frac{1}{\log^c N})$-secure.

b) Let $k = O(\log\log N)$. The k-th bit in the binary expansion of the plaintext is is $\frac{1}{2} + \frac{1}{\log^c N}$ secure.

c) Let $k = O(\log\log N)$. The plaintext's k least significant bits are **simultaneously** secure. I.e., even if all least significant bits $x_{k-1}, \ldots, x_2, x_1$ are given together with $E_N(x)$, still x_k is $(\frac{1}{2} + \frac{1}{\log^c N})$-secure.[2]

d) All bits in the binary expansion of x (except maybe the $\log\log N$ most significant ones) are $(\frac{3}{4} + \frac{1}{\log^c N})$-secure. At least half of them are $(\frac{11}{16} + \frac{1}{\log^c N})$-secure.

Proof sketch :

(a) and *(d)* follow from Theorem 1, by reductions due to Ben-Or, Chor and Shamir [1].

b) First note that using our proof technique, it is possible to guess all k least significant bits of y and z. This determines all k least significant bits of each w_i.

Apply the *gcd* procedure to two small multiples of the plaintext, the greatest common divisor of which is 2^k. This way all d's in the *gcd* calculation will have zeros in all $k-1$ least significant bits. Replace all reference to the least significant bit, in the inverting algorithm (presented in section 5), by references to the k-th bit. Note that this time we have access to an oracle to the k-th bit.

This method of transforming certain inverting algorithms which use an oracle for the 1-st bit into inverting algorithms which use an oracle for the k-th bit originates from Vazirani and Vazirani [18].

c) Going through the proof of Theorem 2(b), notice that when querying the oracle about the k-bit of $w_i + d$ we can give it the $k-1$ previous bits of $w_i + d$. (The latter are equal to the $k-1$ least significant bits of w_i, which we know!)

Vazirani and Vazirani [19] had previously shown that, certain inverting algorithms which use a $p(N)$ oracle for RSA least significant bit, can be transformed into inverting algorithms which use a $p(N)$ oracle for predicting x_k (given $x_{k-1}, \ldots, x_1$). It turns out that the inverting algorithm of section 5 falls into the above category; this yields an alternative (but harder) way of proving Theorem 2(c).

7. Bits equivalent to factoring in Rabin's encryption function

7.1 Previous Results

The Rabin 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 $E_N(x) = x^2 \pmod N$. The ciphertext space is $Q_N = \{ y | \exists x \quad y \equiv x^2 \pmod N \}$. Rabin [14] has shown that extracting square roots ("inverting E_N") is polynomially equivalent to factoring.

[2]Equivalently, given $E_N(x)$ distinguishing between $x_k \cdots x_2 x_1$ and a randomly selected string of length k is as hard as inverting the RSA. This equivalence is due to Yao [20].

Note that the function E_N defined above is 4 to 1 rather than being 1 to 1 (as is the case in the RSA). Blum [2] has pointed out that if $p \equiv q \equiv 3 \pmod 4$ then E_N induces a permutation over Q_N. These N's will hereby be called *Blum integers*. Goldwasser, Micali and Tong [11] 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 $1 - \frac{1}{\log N}$.

Ben-Or,Chor and Shamir [1] considered the same predicate. Using a modification of their RSA techniques they showed $\frac{3}{4} + \epsilon$ security for this predicate. Their modification requires that N be a Blum integer and furthermore that there exists a small odd number l ($l = O(\log^c N)$) with $(\frac{l}{N}) = -1$. Its correctness proof makes use of non-elementary number theory.

7.2 Our Result

We transform our RSA security result into a similar result for the Rabin encryption function. Our transformation is simpler than the one used in [1], and its correctness proof is elementary. Furthermore, it holds for any Blum integer.

Let N be a Blum integer, $S_N = \{ x | 0 \leq x < \frac{N}{2} \}$ and $M_N = \{ x | 0 \leq x < \frac{N}{2} \quad \& \quad (\frac{x}{N}) = 1 \}$. Redefining E_N for $x \in M_N$ as

$$E_N(x) = \begin{cases} x^2 \pmod N & \text{if } x^2 \pmod N < \dfrac{N}{2} \\ -x^2 \pmod N & \text{otherwise} \end{cases}$$

makes E_N a $1-1$ mapping from M_N onto itself, without losing the intractability result of Rabin. I.e. factoring N is polynomially reducible to inverting E_N. Let $L(x)$ be the least significant bit of x.

The main idea in the reduction (as in the RSA case) is to pick L positions $w_i \in S_N$ which are uniformly distributed in S_N and pairwise independent, such that their least significant bits are known. We want to determine the parity of $|d|$ for small $d \in Z_N$ (not necessarily in M_N) by querying the oracle with inputs $E_N(w_i + d)$, as in the RSA case. However, if $w_i + d \notin M_N$, the oracle's answer does not correspond to $w_i + d$ (but rather to the square root of $(w_i + d)^2 \pmod N$ which resides in M_N).

We solve the above problem as follows: If $\left(\frac{w_i+d}{N}\right) = -1$, then we do not feed the oracle with $(w_i + d)^2$, but flip a coin instead. Once this is done, we can cut the error per a single d just as we did in the RSA case.

Restricting attention to residues in S_N:

Given $E_N(x)$, pick $y, z \in Z_N$ two random multiples of x, with known positions and least significant bits as before. Let $v_i = y + iz \pmod N$, $1 \leq i \leq L$. Define

$$w_i = \begin{cases} v_i & \text{if } v_i < \dfrac{N}{2} \\ N - v_i & \text{otherwise} \end{cases}$$

If w_i is not in a $\frac{2N}{L^2}$ interval around 0 or $\frac{N}{2}$, then the least significant bit of w_i is known. Therefore we get

$$Pr(\text{ least significant bit of } w_i \text{ is unknown}) \leq \frac{4}{L^2}.$$

Another modification is to restrict the inputs to the *gcd* computation to the interval $[\frac{-N\epsilon}{8}, \frac{N\epsilon}{8}]$ instead of restricting them to the interval $[\frac{-N\epsilon}{2}, \frac{N\epsilon}{2}]$ as in the RSA-inverting algorithm. This will restrict all numbers (i.e. all d's) in the *gcd* calculation to be in $[\frac{-N\epsilon}{8}, \frac{N\epsilon}{8}]$. Doing this, the probability that a wrap around **either** 0 (mod N) **or** $\frac{N}{2}$ (mod N) occurs when d is added to w_i is no greater than $\frac{\epsilon}{4}$.

Thus, we have restricted our attention to residues in S_N (the probability that the oracle is queried on a residue larger than $\frac{N}{2}$ is at most $\frac{4}{L^2} + \frac{\epsilon}{4}$). However, only half of these residues are in M_N. To deal with the other half we apply the following

Oracle Transformation:

Let $\mathcal{O}_N$ be a $(\frac{1}{2}+\epsilon)$-oracle for the least significant bit of E_N. I.e. on query $E_N(m)$ $(m \in M_N)$, $\mathcal{O}_N$ answers $L(m)$ with probability at least $\frac{1}{2} + \epsilon$, where $\epsilon = \frac{1}{\log^c N}$.

Given $E_N(x)$ $(x \in M_N)$ and access to the oracle $\mathcal{O}_N$, we define an oracle $\mathcal{O}_N^{(x)}$ as follows:
On query $k \in Z_N$,

$$\mathcal{O}_N^{(x)}(k) = \begin{cases} \mathcal{O}_N(E_N(kx)) & \text{if } \left(\frac{k}{N}\right) = 1 \\ \text{coin flip} & \text{otherwise} \end{cases}$$

Note that given $E_N(x)$ and access to the oracle $\mathcal{O}_N$, one can easily simulate the oracle $\mathcal{O}_N^{(x)}$. Also note that if $kx \in S_N$, then $\mathcal{O}_N^{(x)}(k)$ answers the least significant bit of kx with probability at least $(\frac{1}{2} + \frac{\epsilon}{2})$. I.e. $\mathcal{O}_N^{(x)}$ is a is a $(\frac{1}{2} + \frac{\epsilon}{2})$-oracle for the least significant bit of $kx \in S_N$.

Note that all the oracle queries the *gcd* computation initiates are of the form $w_i + d = kx$ when k is known to the *gcd* procedure.

Note that

$$Pr(\mathcal{O}_N^{(x)} \text{ gives the least significant bit of } w_i + d) \geq$$
$$Pr(w_i + d \in S_N \text{ and } \mathcal{O}_N^{(x)} \text{ is correct about it}) - Pr(w_i + d \notin S_N) \geq$$
$$(\tfrac{1}{2} + \tfrac{\epsilon}{2}) - (\tfrac{4}{L^2} + \tfrac{\epsilon}{4}) = \tfrac{1}{2} + \tfrac{\epsilon}{4} - \tfrac{4}{L^2}\,.$$

The rest of the analysis is similar to the analysis presented in section 5. This implies

Theorem 3: The least significant bit for the modified Rabin encryption function is $(\frac{1}{2} + \frac{1}{\log^c N})$-secure, for any constant c.

Corollary: Factoring a Blum integer, N, is polynomially reducible to guessing $L(x)$ with success probability $\frac{1}{2} + \frac{1}{\log^c N}$ when given $E_N(x)$, for $x \in M_N$.

The proofs from the previous section about simultaneous security of $\log \log N$ least significant bits and of bit intervals (for intervals of length $\frac{N}{4}$ out of the $\frac{N}{2}$ long interval containing M_N) hold here just as well, thus all these bits are also $\frac{1}{2} + \frac{1}{\log^c N}$ secure.

8. Applications

8.1 Direct Construction of Pseudo-Random Bit Generators

A pseudo-random bits generator is a device which "expands randomness". Given a truly random bit sequence s (the seed), it expands it to a longer pseudo-random sequence. The question of "how random" this pseudo-random sequence is depends on what exact definition of randomness we are after. A strong requirement is that the expended sequence will pass all polynomial time statistical tests, namely given a pseudo-random and a truly random sequences of equal length, no

probabilistic polynomial time algorithm can tell which is which with better than $50-50$ success (this definition was proposed by Yao [20], who also showed it is equivalent to some other natural definitions like unpredictability).

Blum and Micali were the first to construct such strong pseudo-random generators. Their construction combines two results:
a) If $g: M \to M$ is a $1-1$ one way function, and $B_N(x)$ is $\frac{1}{2}+\epsilon$ secure bit for g (where $\epsilon =$ any polynomial fraction), then starting with a random $s \in M$, the sequence obtained by iterating g and outputting $b_i = B(g^i(s))$ for each iteration is pseudo random (in the sense that each of its bits can not be predicted better than 50-50, from the previous ones).
b) Demonstrating that a specific bit is $\frac{1}{2}+\epsilon$ secure for the discrete exponentiation function.

We say that a generator is *direct w.r.t* the (underlying) one way function g if it produces at least one bit per one iteration of g. We say that a generator is *strong w.r.t* an (assumed) intractable problem, P, if distinguishing its output from truly random sequences is as hard as solving P. Notice that both the Blum & Micali generator and the Long & Wigderson generator[3]([13]) are direct w.r.t discrete exponentiation and strong w.r.t discrete log.

Another direct generator was constructed by Blum, Blum and Shub [3]. Their generator is direct w.r.t squaring modulo a composite number and strong w.r.t deciding quadratic residuocity.

Yao [20] made some generalizations to the Blum & Micali result. He showed that having a $1-1$ one way function f is enough and it is not necessary to have a specific secure bit. The main idea is that if f is one way then some bits must be secure (even though not necessarily $\frac{1}{2}+\epsilon$ secure). Picking a polynomial number of random seeds $s_1, s_2, \ldots, s_k$, we get one strongly pseudo-random bit b_i by computing

$$\bigoplus f^i(s_1) \oplus f^i(s_2) \ldots \oplus f^i(s_k)\,.$$

($\oplus$ denotes bitwise XOR, and $\bigoplus$ is the one bit XOR of the result.)

Yao XORing trick works for any $1-1$ one way function, f, **but the generators achieved that way are not direct** w.r.t f - to produce one bit, many applications of f are needed. For further details on Yao's XORing trick and its proof consult Goldwasser [8].

All **previously known results** about the cryptographic security of Rabin/RSA scheme (including Schnorr & Alexi result) **do not suffice for constructing generators** which are **strong w.r.t factoring/inverting the RSA** and **direct w.r.t Rabin/RSA encryption function.**

With $\frac{1}{2}+\frac{1}{poly \log N}$ security, we can finally get generators which are direct w.r.t Rabin/RSA encryption function and strong w.r.t factoring/inverting RSA. Each of the bits whose $\frac{1}{2}+\frac{1}{poly \log N}$ security is proven can be used as the "hard bit" the generator outputs. As a matter of fact, with the stronger result that all $\log\log N$ least significant bits are simultaneously $\frac{1}{2}+\frac{1}{poly \log N}$ secure, we can get $\log\log N$ random bits per one application of the encryption function. Since the encryption in Rabin scheme is just one squaring and one subtraction, we get a very fast generator, whose security is equivalent to factoring a Blum integer[4].

---

[3]Long & Wigderson's generator produces $\log\log p$ bits per each iteration of the discrete exponentiation (mod p) function. This is due to their proof that this function has $\log\log p$ simultaneously hard bits.

[4]A composite number $N = pq$, such that p and q are both primes congruent to 3 modulo 4

8.2 Direct Construction of Probabilistic Encryption Schemes

Observation, similar to the ones of section 8.1, apply to the probabilistic encryption scheme suggested by Goldwasser and Micali [10]. Using our result we introduce the first **direct** probabilistic encryption equivalent to factoring/inverting RSA. However, this implementation still has the bandwidth expansion drawback; the plaintext is expended by a factor of $\theta(\frac{\log N}{\log \log N})$.

Recently, Goldwasser [9] 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**. Goldwasser's scheme is approximately as efficient as the RSA while provably leaking no partial information, provided that factoring is intractable.

Acknowledgments

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

Oded Goldreich would like to thank Dassi Levi for her unique existence.

References

[1] Ben-Or,M., Chor,B., and Shamir,A., "On the Cryptogrsphic Security of Single RSA Bits", *15th ACM Symp. on Theory of Computation*, April 1983, pp. 421-430.

[2] Blum, M., "Coin Flipping by Telephone", *IEEE Spring COMCON*, 1982.

[3] Blum,L., Blum,M., and Shub,M., "Comparison of Two Pseudo- Random Number Generators", *Advances in Cryptology: Proceedings of Crypto82*, Chaum,D., et al. eds., Plenum Press, 1983, pp. 61-79.

[4] Blum,M., and Micali,S., "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", to appear in the *SIAM Jour. on Computing.*

[5] Diffie,W., and Hellman,M.E., "New Directions in Cryptography", *IEEE Trans. on Inform. Theory*, Vol. IT-22, No. 6, November 1976, pp. 644-654.

[6] Feller,W., *An Introduction to Probability Theory and its Applications*, John Wiley & Sons Inc., Vol. I, (1962).

[7] Goldreich,O., "On the Number of Close-and-Equal Pairs of Bits in a String (with Implications on the Security of RSA's L.s.b.)", MIT/LCS/TM-256, March 1984.

[8] Goldwasser,S., "Factoring Based Encryption: Using the Exclusive-Or Function as a Security Amplifier", presented at EuroCrypt84, Paris, April 1984.

[9] Goldwasser,S., "An Efficient Probabilistic PKCS as Secure as Factoring", in preparation.

[10] Goldwasser,S., and Micali,S., "Probabilistic Encryption & How to Play Mental Poker Keeping Secret all Partial Information", *Proc. of the 14th ACM Symp. on Theory of Computation*, 1982, pp. 365-377. To appear in the *JCSS special issue from the 14th STOC.*

[11] Goldwasser,S., Micali,S., and Tong,P., "Why and How to Establish a Private Code on a Public Network", *Proc. of the 23rd IEEE Symp. on Foundation of Computer Science*, November 1982, pp. 134-144.

[12] Lipton,R., "How to Cheat at Mental Poker", Proceeding of the AMS short course on Cryptology, January 1981.

[13] Long,D.L., and Wigderson,A., "How Discreet is Discrete Log ?", *15th ACM Symp. on Theory of Computation*, April 1983, pp. 413-420. A better version is in preparation.

[14] Rabin,M.O., "Digital Signatures and Public Key Functions as Intractable as Factorization", MIT/LCS/TR-212, 1979.

[15] Rivest,R.L., Shamir,A., and Adleman,L., "A Method for Obtaining Digital Signature and Public Key Cryptosystems", *Comm. of the ACM* ,Vol.21, February 1978, pp. 120-126.

[16] Schnorr,C.P. and Alexi, W., "RSA bits are $0.5 + \epsilon$ secure", presented at EuroCrypt84, Paris, April 1984.

[17] Shamir,A., Rivest,R.L., Adleman,L., "Mental Poker", MIT/LCS/TM-125, February 1979.

[18] Vazirani,U.V., and Vazirani,V.V., "RSA Bits are .732 Secure", to appear in the proceedings of *Crypto83.*

[19] Vazirani,U.V., and Vazirani,V.V., "Efficient and Secure Pseudo-Random Number Generation", preprint, April 1984.

[20] Yao,A.C., "Theory and Applications of Trapdoor Functions", *Proc. of the 23rd IEEE Symp. on Foundation of Computer Science*, 1982, pp. 80-91.