On the cryptographic security of single RSA bits

by Michael Ben-Or[1], Benny Chor[1] and Adi Shamir[2]

[1]Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

[2]Applied Mathematics
The Weizmann Institute
Rehovot, Israel

Abstract—The ability to "hide" one bit in trapdoor functions has recently gained much interest in cryptography research, and is of great importance in many transactions protocols. In this paper we study the cryptographic security of RSA bits. In particular, we show that unless the cryptanalyst can completely break the RSA encryption, any heuristic he uses to determine the least significant bit of the cleartext must have an error probability greater than $\frac{1}{4} - \epsilon$. A similar result is shown for Rabin's encryption scheme.

1. Introduction

Clarifying the relationship between the cryptographic security and computational complexity of cryptosystems is one of the most important goals of current cryptographic research. The main difficulty seems to be the subtle interaction between information-theoretic and complexity-theoretic arguments, which is hard to quantify. Even if the decryption of cyphertexts is proved to be almost everywhere difficult, the cryptosystem may be useless in practice if the cryptanalyst can easily derive useful information about the cleartexts without actually computing them. This information can be in the form of particular cleartext bits, some functions of these bits, or just a non-uniform probability distribution on candidate cleartexts that can make exhaustive search more efficient. Even a single cleartext bit can be useful if the cryptanalyst knows that the cleartext is either a "yes" or a "no" answer to some question. As to a concrete example, consider the problem of factoring a number N which is the product of two large primes p and q. It is probably a difficult computational task for almost all such N, but if the least significant decimal digit of N is 3, it is easy to deduce that the least significant digits of p and q are 1 and 3, or 7 and 9. A small amount of information about the factors is thus easy to obtain, and the nature of the tradeoff between the information and complexity aspects of such problems is far from being understood.

One of the first attempts to analyze this question is described in Goldwasser and Micali [2]. In this innovative paper, they describe a new randomized public key cryptosystem in which each cleartext bit is encrypted separately. This independence enables them to prove that unless the problem of determining quadratic residuosity modulo a composite number is easy (a purely complexity-theoretic question), the cryptanalyst can not get any deterministic or statistical information about the cleartext bits. Their main observation is that any "magic box" that can guess

† Research supported by NSF grant MCS-8006938, and by a Weizmann Postdoctoral Fellowship to M. Ben-Or.

cleartext bits with $1/2 + \epsilon$ probability of success can be transformed into a quadratic residuosity "magic box" by analyzing the answers to a large number of independent random queries which are related to the same cleartext bit. However, the new cryptosystem they propose seems to be only of theoretical value: In spite of its high security, it is unlikely to be used in practice since it is extremely slow and it expands each cleartext bit into a ciphertext block which is hundreds of bits long.

In another recent work, Blum and Micali [1] showed that the discrete exponentiation 'hides' a certain bit of its argument in a very strong sense, and used this to generate sequences of pseudo random bits.

In this paper we analyze the cryptographic security of the RSA cryptosystem, which is one of the best known public key cryptosystems (see Rivest, Shamir and Adelman [7]). It transposes a block of cleartext bits into a block of ciphertext bits of the same length, and thus it lacks the independence property which is crucial in Goldwasser and Micali's proof technique. Recently, Goldwasser, Micali and Tong [3] showed that an oracle which determines the least significant bit of the cleartext from RSA cyphertexts can be used to break the RSA cryptosystem. Their result remains valid even if the oracle is allowed to make some errors, but the error probability allowed is small and tends to 0 as the message size tends to infinity.

Here, using novel arguments, we greatly improve this result. Let $\mathcal{O}$ be any one of the following oracles

($\mathcal{O}_I$) For any interval I of non-negligible length ($\epsilon < \frac{|I|}{N} < 1 - \epsilon$) determine for any cyphertext $E(x)$[1] whether $x \in I$.

($\mathcal{O}_k$) Given $E(x)$, determine the k-th binary bit of x.

($\mathcal{O}_L$) Given $E(x)$, guess the least significant bit of x with an error probability less than $\frac{1}{4} - \epsilon$.

[1] $E(x) = x^e \pmod N$, where N is the product of two large primes and $\gcd(e, \varphi(N)) = 1$.

We show that $\mathcal{O}$ can be transformed into a polynomial time algorithm for decrypting RSA ciphertexts. Thus the cryptanalyst cannot gain the partial information given by $\mathcal{O}$ without completely breaking the RSA cryptosystem. This seems to be the first time that such a strong result is proved about a practical cryptosystem, and it greatly enhances our belief in the cryptographic security of the system.

We apply similar arguments for the Rabin public-key scheme (see Rabin [6]) to get the same result concerning the oracle $\mathcal{O}_L$. This result is of special interest since Rabin's scheme is known to be equivalent to factoring.

The next section deals with the interval oracle $\mathcal{O}_I$, section 3 with $\mathcal{O}_L$, and section 4 with $\mathcal{O}_k$. Section 5 discusses $\mathcal{O}_L$ for Rabin's scheme, and section 6 contains some concluding remarks.

2. The interval Oracle

Let $I \subseteq \mathbf{Z}_N$ be an interval with $\epsilon < \frac{|I|}{N} < 1 - \epsilon$ for some non-negligible $\epsilon > 0$, and let $\mathcal{O}_I$ be the oracle that on input $E(x)$ determines whether $x \in I$. In this section we prove that such an oracle can be used to invert the RSA encryption function in random polynomial time. We first prove this results for a special type of intervals.

Theorem 1. *Let $I = [0, t]$ with $\epsilon < t \leq N/2$. There is a random polynomial time algorithm using the oracle $\mathcal{O}_I$ that inverts the encryption function $E(x)$.*

Proof. Given $E(x)$ we want to find x. Having $E(x)$, we can compute $E(ix \pmod N)$ for all i, using the multiplicative structure of the RSA:

$$E(ix) = E(i)E(x) \pmod N.$$

When i ranges from 1 to N so does $ix \pmod N$, thus for a random i, $ix \in I$ with probability $\frac{|I|}{N} > \epsilon$. (Here, and throughout the paper we assume that x is relatively prime to N. If $\gcd(x, N) \neq 1$ then $\gcd(E(x), N) \neq 1$, so we can factor N and find the decryption key.)

We now find a random $b = ix \pmod N \in I$. This can be done by picking a random i, computing $E(ix)$, and querying the oracle $\mathcal{O}_I$ with the input $E(ix)$. This is repeated until an i with $b = ix \in I$ is found. In a similar way we find another random $c = jx \in I$. Note that the values of i and j are known, though x is unknown.

Generally, knowing $E(b), E(c)$ is not enough to compute $E(b \pm c)$. However, since $b = ix, c = jx$ and i, j are known, $E(b \pm c) = E(i \pm j)E(x)$. This can be used to determine whether $c < b$ or $b < c$ since

$$b < c \text{ iff } c - b \in I\,.$$

Using this ordering we can implement the Euclidean greatest common divisor algorithm to compute $\gcd(b, c)$. Suppose $b < c$. By performing a binary search on k, we determine the maximal k such that $kb < c$. This way we can divide $c = kb + r$, with $0 \leq r < b$, and $r = c - kb = (j - ki)x \in I$. Now substitute $c \leftarrow b$ and $b \leftarrow r$, and continue with Euclid's g.c.d. algorithm until we find $E(d)$ where $d = \gcd(b, c)$, and a representation $d = lx \pmod N$, where l is known.
If b and c were relatively prime then $d = 1$, so $E(d) = 1$. If this occurs then $x = l^{-1} \pmod N$. Since the probability that b and c will be relatively prime is $\approx \frac{6}{\pi^2}$, repeating this procedure will recover x in random polynomial time.∎

To prove the next theorem, about general intervals, we first introduce the method of "modified binary gcd" which will play a central role throughout this paper. To find the gcd of two integers, b and c, most standard algorithms require tests of the form 'is $b < c$?' . Unlike the special case of theorem 1, the other oracles will not enable us to perform such comparisons. We modify the original binary gcd (see [4, Vol. II, p. 321]) to avoid comparisons. The essence of our algorithm is that if $|b|$ and $|c|$ are both even, then $\gcd(b, c) = 2\gcd(\frac{b}{2}, \frac{c}{2})$, if $|b|$ is even and $|c|$ is odd, then $\gcd(b, c) = \gcd(\frac{b}{2}, c)$, while if both $|b|$ and $|c|$ are odd then $\gcd(b, c) = \gcd(\frac{b+c}{2}, \frac{b-c}{2})$. Furthermore, in the last case, one of $\frac{b+c}{2}, \frac{b-c}{2}$ is divisible again by 2. The following algorithm finds the gcd of two integers 'up to powers of 2'.

Modified binary gcd algorithm:

(Initialize) on input b, c, repeat $b \leftarrow \frac{b}{2}$ $c \leftarrow \frac{c}{2}$ until b, c are odd.

(Loop) $c \leftarrow \frac{b+c}{2}$, $b \leftarrow \frac{b-c}{2}$. Eliminate powers of 2 until left with odd numbers.

(Terminate) when either $b = 1, -1, 0$ or $c = 1, -1, 0$.

It is a simple matter to verify that after at most two passes over the loop, $\max(|a|, |b|)$ is reduced by at least a $\frac{3}{4}$ factor. Hence only polynomially many iterations are needed.

Theorem 2. *Let I be any interval with $\epsilon < \frac{|I|}{N} < 1 - \epsilon$, for some non negligible $\epsilon > 0$. There is a random polynomial time algorithm using the oracle $\mathcal{O}_I$ that inverts the encryption function $E(x)$.*

Proof. Without loss of generality, $|I| \leq \frac{N}{2}$, (otherwise replace I by its complement). Using $\mathcal{O}_I$, pick a random i such that $a_0 = ix \pmod N \in I$. Pick random j, k with $jx + a_0, kx + a_0 \in I$. Denote $b = jx, c = kx$. With probability $\frac{1}{8}$, a_0 is in the middle two quadrants of I, and $|b|, |c| < \frac{|I|}{4}$. We intend to run the modified binary gcd algorithm to find $\gcd(|b|, |c|)$ (expressed in terms of lx, as in the previous theorem). To do that, we use $\mathcal{O}_I$ to distinguish even $|b|$'s from odd once inside $[\frac{-|I|}{4}, \frac{|I|}{4}]$ in the following way:
If $b \in [\frac{-|I|}{4}, \frac{|I|}{4}]$ and $|b|$ is even, then $\frac{b}{2} \in [\frac{-|I|}{4}, \frac{|I|}{4}]$. On the other hand, if $|b|$ is odd, then $\frac{b}{2} \in [\frac{N}{2} - \frac{|I|}{4}, \frac{N}{2} + \frac{|I|}{4}]$. Therefore under the assumptions on a_0, b and c, and the fact that $|I| \leq \frac{N}{2}$, we have:

$$\text{If } |b| \text{ is even then } a_0 + \tfrac{b}{2} \in I.$$
$$\text{If } |b| \text{ is odd then } a_0 + \tfrac{b}{2} \notin I.$$

Note that since $\left|\frac{b+c}{2}\right|, \left|\frac{b-c}{2}\right| \leq \max(|b|, |c|)$, the new values for b, c will satisfy $a_0 + b, a_0 + c \in I$ as well as the previous bounds on their size. Therefore we can continue the iteration in the modified binary gcd algorithm. If the number of iterations performed exceeds the bound

$2\log_{4/3} N$, we abort and start with a new triple a, b, c. Every trial has probability $\frac{1}{8}\frac{6}{\pi^2}$ of yielding 1 (which enables inverting x), so the whole process is in random polynomial time. ∎

Corollary. *For every non trivial interval I, the cryptanalyst cannot determine whether $x \in I$ without completely breaking this RSA encryption.*

Definition: Let I, J be two disjoint intervals in $\mathbf{Z}_N$. Let O be a $0-1$ oracle (with input $E(x)$). We say that O distinguishes between I and J if for every $x \in I$, $y \in J$ $O(E(x)) \neq O(E(y))$. We say that I, J are opposite if $I = J + \frac{N}{2}$.

Lemma 3. Let I and J be opposite disjoint intervals of non-negligible length, and O an oracle distinguishing I from J. Then O can be used to break this RSA encryption in random polynomial time.

Proof. Use the same construction of theorem 2.∎

Corollary 4. *For almost all moduli N, the most significant bit of RSA cleartext cannot be determined from the ciphertext.*

Proof. If N is not very close to a power of 2, then the x's in $\mathbf{Z}_N$ with most significant bit 1 determine an interval of non negligible length. Our claim then follows from lemma 3. ∎

A similar argument shows that for any N, the other $\log\log N$ most significant cleartext bits cannot be determined unless the cryptanalyst can completely break the RSA encryption modulo N.

3. Least significant RSA bit

The main result in this section is the following theorem.

Theorem 5. *Let O_L be an oracle that, on input $E(x)$, can guess the least significant bit of x, such that for a random x the probability that O_L will err is at most $\frac{1}{4} - \epsilon$ (for some non-negligible $\epsilon > 0$). Then there is a random polynomial time algorithm, using O_L, that breaks this RSA encryption.*

Proof. Let $x \in \mathbf{Z}_N$, define

$$Half_N(x) = \begin{cases} \text{top} & \text{if } \frac{N}{2} < x < N \\ \text{bottom} & \text{otherwise} \end{cases}$$

Note that the least significant bit of x is 0 if and only if $Half_N(2^{-1}x) = \text{bottom}$. Hence any oracle O_L for the least significant bit can be transformed to an $Half_N$ oracle, O_H (and vice versa). Our goal is proving that any oracle O_H which, given the encryption of x, $E(x)$, can guess $Half_N(x)$ with $3/4 + \epsilon$ probability of success (for some non-negligible $\epsilon > 0$), can be used to break this RSA encryption in random polynomial time.

Define $x \equiv y$ if $Half_N(x) = Half_N(y)$. Notice that if x is very small then for almost all i, $i \equiv i + x$, while if $x \approx \frac{N}{2}$ then for almost all i, $i \not\equiv i + x$.

Fact 1: *Let $A(x) = 2|\frac{x}{N} - \frac{1}{2}|$. Then for random i, j*

$$A(x) = Pr(i \equiv i + x)$$
$$A(x) = Pr(jx \equiv (j+1)x)\,.$$

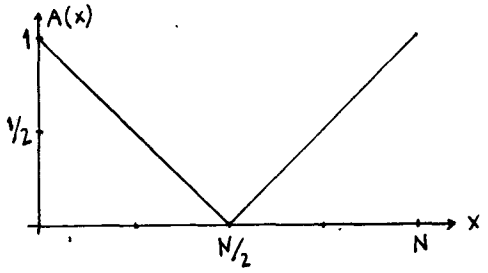

Figure 1: The function $A(x)$

This fact enables us to manipulate the $Half_N$ oracle into one which can distinguish x's which are very close to 0 from x's very close to $\frac{N}{2}$. This will suffice to perform a modified binary gcd algorithm which, in turn, can be used to invert $E(x)$.

Fact 2: *Let $S_x = \{j | O_H \text{ guesses correctly on } jx\} \bigcap \{j | O_H \text{ guesses correctly on } (j+1)x\}$. Then $|S_x| \geq N(\frac{1}{2} + 2\epsilon)$.*

Proof: Recall that O_H guesses correctly on at least $N(\frac{3}{4} + \epsilon)$ elements, and use the principle of inclusion and exclusion.∎

$$\text{Denote} \qquad a(x) = \frac{|\{j | jx \equiv (j+1)x\} \bigcap S_x|}{N}\,.$$

Fact 3: *For every x, $|a(x) - A(x)| \leq \frac{1}{2} - 2\epsilon$.*

Let $s(x)$ be an approximation to $a(x)$, gained by sampling, i.e. by picking various independent j's and counting for how many of those O_N gives the same answer to $E(jx), E((j+1)x)$. By the weak law of large numbers, polynomially many (in $\epsilon^{-1}, \log N$) sampling points are enough to have $|s(x) - a(x)| < \epsilon$ with overwhelming probability (e.g. $Pr > 1 - \frac{1}{\log^3 N}$).

So, with overwhelming probability, for every x, we have

$$|s(x) - A(x)| \leq |s(x) - a(x)| + |a(x) - A(x)| \leq \frac{1}{2} - \epsilon .$$

Let $I_\epsilon = [-\frac{N\epsilon}{2}, \frac{N\epsilon}{2}]$ and let $J_\epsilon = \frac{N}{2} + I_\epsilon$. We get

$$\begin{array}{lllll} x \in I_\epsilon & \Rightarrow & A(x) \geq 1 - \frac{\epsilon}{2} & \Rightarrow & s(x) > \frac{1}{2} \\ x \in J_\epsilon & \Rightarrow & A(x) \leq \frac{\epsilon}{2} & \Rightarrow & s(x) < \frac{1}{2}. \end{array}$$

So by comparing the sample $s(x)$ to $\frac{1}{2}$ we have a very reliable evidence to either $x \notin I_\epsilon$ or $x \notin J_\epsilon$. The crucial point about the interval I_ϵ Is that if $x \in I_\epsilon$ then

$$|x| \text{ even} \quad \Rightarrow \quad \frac{x}{2} \in I_\epsilon, \quad |x| \text{ odd} \quad \Rightarrow \quad \frac{x}{2} \in J_\epsilon.$$

This makes the distinction between even and odd possible inside I_ϵ and this is all we need.

Description of inversion algorithm:
Given $E(x)$, pick random i, j and compute $E(ix), E(jx)$. Continue until i, j both satisfy $s(ix), s(jx) > \frac{1}{2}$. (otherwise $ix \notin I_\epsilon$ or $jx \notin I_\epsilon$).

(Initialize) Set $b \leftarrow ix$, $c \leftarrow jx$. While $s(2^{-1}b) > \frac{1}{2}$ do $b \leftarrow 2^{-1}b$ (and similarly for c).

(Loop) $b \leftarrow \frac{b+c}{2}$, $c \leftarrow \frac{b-c}{2}$ (Addition/subtraction is done as in the gcd algorithm of section 2). Eliminate factors of 2 from both b, c - if more than $\log_2 N$ "factors" are found, abort and return to the starting point.

(Termination) If loop is done more than $2\log_{4/3} N$ times - abort and return to starting point. If $b = 0$ or $c = 0$ return to the starting point. If either $b = 1, -1$ or $c = 1, -1$ use the gathered coefficients to represent $1 = l \cdot x$, where l is known. Recover $x = l^{-1} \pmod N$, and verify it by applying E to the resulting value.

Analysis: The last clause makes sure the algorithm never errs. To see that it is in random polynomial time, observe the following: With probability $\geq \epsilon^2$, the initial choices are in I_ϵ. Hence with probability $\geq \frac{6\epsilon^2}{\pi^2}$ they are also relatively prime. In this case, the algorithm proceeds with overwhelming probability of success without one single error. ∎

Remark 6: The results of this section remain true if we replace the O_N oracle by an oracle O_I with the **same error probability**, for any interval I of length $N/2$, since the relation $x \equiv_I y$, defined as $x \in I \leftrightarrow y \in I$, gives rise to the **same** function $A(x)$.

Corollary 7: *The second bit of the RSA has the same security as the first bit.*

Proof. Distinguish between two cases regarding the modulus N.

1. $N = 4k + 3$. In this case the second bit of x is 0 iff $x/4$ is in bottom half.

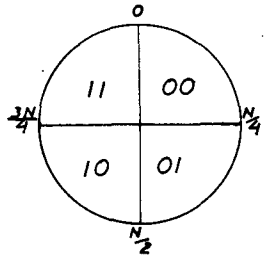

Figure 2: Quadrants for $\frac{x}{4}$ vs. two least significant bits of x, $N = 4k + 3$.

2. $N = 4k + 1$. In this case the second bit of x is 0 iff $x/4$ has small 'absolute value' (i.e. $-\frac{N}{4} < \frac{x}{4} < \frac{N}{4}$).

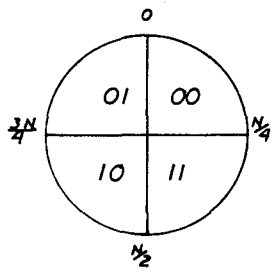

Figure 3: Quadrants for $\frac{x}{4}$ vs. two least significant bits of x, $N = 4k + 1$.

So in both cases , determining second bit is equivalent to determining an interval of length $N/2$, and hence cannot be done with error probability smaller than $\frac{1}{4} - \epsilon$. ∎

4. Other RSA bits

In this section we show that not only the least significant RSA bit (and second bit), but almost all other bits are secure. The proof of this fact is based on the following reduction:

Let O_k be an oracle which, given $E(x)$, determines the k-th bit of x. Represent $N = r2^k \pm m$ with $|m| < 2^{k-1}$. Define

$$O_I(E(x)) = O_k(E(r^{-1}x)) .$$

We first show that O_I (almost always) distinguishes between two opposite intervals of non-negligible length, thus, by lemma 3, settling the deterministic question. Then we show that O_I can be viewed as an oracle for an appropriate interval I of length $\frac{N}{2}$, and the error introduced by this reduction is almost always substantially smaller than $\frac{1}{4}$. This result, combined with remark 6, will prove the stronger result that almost all the bits cannot be guessed without a noticeable error.

We give here a detailed description of the case $m > 0$. The case $m < 0$ can be treated similarly. For $N = r2^k + m$, $0 < m < 2^{k-1}$. Denote

$$G_l = \left\{ x \left| \frac{lN}{r} \le x < \frac{(l+1)N}{r} \right. \right\}$$

for $l = 0, \ldots, r-1$ (G_l's are a partition of $[0, N-1]$ to disjoint intervals).

Lemma 8: *If $x \in G_l$ then the k-th bit of x is 1 iff $rx \pmod N \in [r2^{k-1} - lm, r2^k - lm)$.*

Proof. Follows from the fact that for every $0 \le l \le r-1$, $[l2^k + 2^{k-1}, (l+1)2^k] \subseteq G_l$. ∎

Figure 4 describes the range of $x \to rx \pmod N$ as a function of the G_l's. Each G_l gets sent to a sequence of points, spaced in distances of r apart. Denote

$$I_1 = [r2^{k-1}, r2^k - (r-1)m)$$

$$J_1 = (-m, r2^{k-1} - (r-1)m)$$

then $|I_1| = r(2^{k-1} - m) + m$ and $I_1 + \frac{N}{2} \subseteq J_1$.

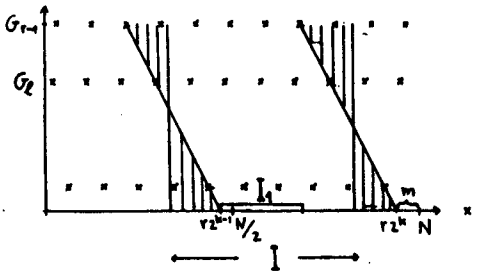

Figure 4: The function $x \to rx$

Lemma 9. *O_I distinguishes I_1 from J_1.*

Proof. By lemma 8, if $x \in I_1$ then the k-th bit of $r^{-1}x$ is 1, so $O_I(E(x)) = O_k(E(r^{-1}x)) = 1$. Similarly, if $y \in J_1$ then $O_I(E(y)) = 0$. ∎

Theorem 10. (a).*For almost all N, an oracle O_k for the k-th bit, $1 \le k < \log N$, can be used to break the RSA in random polynomial time.*
(b). *For every N, for almost all k, $1 \le k < \log N$, the oracle O_k can be used to break the RSA in random polynomial time*

Proof. Represent $N = r2^k + m$. Let $0 < \delta < 1$ such that $m < (1-\delta)2^{k-1}$, then $|I_1| = r(2^{k-1} - m) + m > r2^{k-1} \cdot \delta + m > N\frac{\delta}{2}$. So for every fixed δ, if m satisfies this bound, then O_I separates opposite intervals of non-negligible length. To complete the proof, it suffices to notice that for a given δ, the following holds: For every N almost all k's are such that $m < 2^{k-1}(1-\delta)$, and for almost all N's every k has this property. ∎

The result of theorem 10 settles the question of k-th bit immunity against attacks with no errors. In order to deal with O_k which do err, we extend I_1 to an interval I of length exactly $\frac{N}{2}$, by defining

$$I = [r2^{k-1} - \frac{rm}{2} + \frac{m}{4}, r2^k - \frac{rm}{2} + \frac{3m}{4}] .$$

Interpreting O_I's answer as membership in this interval I, we show that O_I does not err too frequently.

Lemma 11. *Let $N = r2^k + m$, where $|m| < 2^{k-1}$. The total number of errors, e, done by O_I, satisfies $e \le \frac{|mr|}{2} + r$.*

The proof of this lemma is based on bounding the number of points that can fall into the shaded area in figure 4. This number is proportional to the size of this area (the $\frac{|mr|}{2}$ term) plus at most r boundary points.

As a consequence, we conclude that if

$$|m| < \frac{2r}{2r+1}(1-\delta)2^{k-1} - 2$$

then the error probability due to the reduction $\mathcal{O}_k \rightarrow \mathcal{O}_I$ is bounded by $\frac{1-\delta}{4}$. If $k \leq \log N - \log\log N - 1$, then $\frac{2r}{2r+1} \geq 1 - \frac{1}{2\log N}$. Therefore if we restrict ourselves to all except few 'upper' bits we know that if $|m| < 2^{k-1}(1-\delta) - 2$, then the error probability due to the reduction is bounded by $\frac{1-\delta}{4} + \frac{1}{2\log N}$. This term can be rather close to $\frac{1}{4}$ for some k's, but if this happens it will guarantee that the error size for consecutive bits is small. For example, if $m_k \geq 2^{k-1}(1 - \frac{1}{4})$, then $m_{k-1} < 2^{k-2}(1-\frac{1}{2})$. Hence the error probability introduced by our reduction cannot exceed $\frac{3}{16}$ for both $\mathcal{O}_{k-1}$ and $\mathcal{O}_k$.

Theorem 12. *For every moduli N and every k, $3 \leq k < \log N - \log\log N - 1$, either $\mathcal{O}_k$ or $\mathcal{O}_{k-1}$ with error probability not exceeding $1/16 - \epsilon$, can be used to break the RSA in random polynomial time.*

The results of this section show that almost all single RSA bits are secure. This does not necessarily imply that the cryptanalyst cannot determine, given $E(x)$, whether the third bit of x is the same as the seventh bit or not. We can extend our results, using the novel techniques introduced by Long and Widgerson [5], to show that the cryptanalyst cannot determine without error any non-constant Boolean function of a fixed number of least significant bits.

Theorem 13. Let $B(\cdots)$ be a non-constant Boolean function of k_0 Boolean variables. An oracle $\mathcal{O}_B$ which, on input $E(x)$, determines $B(x_{k_0}, \ldots, x_1)$ (x_i's are the k_0 least significant bits of x), can be used to break the RSA in random polynomial time.

Proof. As shown in [5], B has a period of length 2^k ($k \leq k_0$). The transformation $x \rightarrow 2^{-k}x$ gives 2^k intervals (of length $\frac{N}{2^k}$ each) where B is constant. Furthermore, there are at least two **opposite** intervals I, J ($I = J + \frac{N}{2}$), such that B gets different values on I, J. So by lemma 3, $\mathcal{O}_B$ can be used to invert this RSA encryption in random polynomial time. ∎

5. The Rabin Scheme

Let $N = p \cdot q$ where p, q are primes, $p \equiv q \equiv 3 \pmod 4$ and $p \not\equiv q \pmod 3$. Under these conditions one can define an invertible one-to-one encryption function E from

$$M = \{x | x \leq \frac{N}{2} \quad \& \quad (\frac{x}{N}) = 1\}$$

to the set

$$Q = \{y | \exists x \quad y = x^2 \pmod N\}$$

by $E(x) = x^2$. Rabin [6] has shown that inverting this function is polynomially equivalent to factoring N. The special conditions on p, q guarantee that $(\frac{2}{N}) = (\frac{-1}{N}) = 1$ and $(\frac{3}{N}) = -1$. Define

$$Half_{N/2}(x) = \begin{cases} \text{top} & \text{if } N/4 < x < N/2 \\ \text{bottom} & \text{otherwise} \end{cases}$$

As in section 3, an oracle for $Half_{N/2}$ is equivalent to an oracle for the least significant bit of x.

Theorem 14. *Let $\mathcal{O}_L$ be an oracle that on input $E(x)$ can guess the least significant bit of x, such that for a random x the probability that $\mathcal{O}_L$ will err is at most $\frac{1}{4} - \epsilon$ (for some non-negligible $\epsilon > 0$). Then there is a random polynomial time algorithm, using $\mathcal{O}_L$, that factors N.*

Proof. The algorithm uses the same reduction idea of theorem 3, however some modification are required to cope with the fact that the message space is not closed under addition/subtraction. We encounter two main difficulties:

(1) In performing our binary gcd algorithm, even if $b = ix$, $c = jx$ are both in M, $b \pm c$ might be outside M.

To overcome this, notice that $d \notin M \Leftrightarrow 3d \in M$, and multiplication by 3 moves the intervals of length $\frac{\epsilon}{3}$ around 0 and $\frac{N}{2}$ to intervals of length ϵ around 0 and $\frac{N}{2}$ respectively. Thus to test ix for $i \notin M$ we can test $3ix$.

Remark: Since $(\frac{2}{N}) = 1$, division by 2 causes no problems.

(2) When computing $s(x)$ by sampling pairs $jx, (j+1)x$, having $j \in M$ does not guarantee that $j+1 \in M$.

We could sample only pairs for which both $j \in M$ and $j+1 \in M$, but this would force us to allow the error probability of $\mathcal{O}_{\mathcal{H}}$ to be only $\frac{1}{8} - \epsilon$. To prove our claim we must spread our sampling over the entire message space. Denote $B = 64\epsilon^{-3}$, $\eta = (\frac{\epsilon}{8})^3$, and let $I = [-\frac{\eta N}{2B}, \frac{\eta N}{2B}]$ and $J = I + \frac{N}{2}$. Multiplication by any odd k, $|k| < B$, maps the intervals I and J into the intervals of length $N\eta$ around 0 and $N/2$ respectively. Thus we can try to distinguish between the opposite intervals I and J by comparing jx and $(j+k)x$, for any small odd k such that both j and $j+k$ are in M. Given $E(x)$ we do the following test: Pick a random odd k, $|k| < B$, and a random $j \in M$ such that $(\frac{j+k}{N}) = 1$ and apply the oracle $\mathcal{O}_{\mathcal{H}}$ to $E(jx)$ and $E((j+k)x)$.

We claim that if $x \in I$ then the probability that the oracle will give the same answer is greater than $\frac{1}{2} + \frac{\epsilon}{2}$, while if $x \in J$ the answers will be different with probability greater than $\frac{1}{2} + \frac{\epsilon}{2}$. Proving this claim will finish the proof of the theorem, since then we know that repeating this test (polynomially many times) will enable us to distinguish between the intervals I and J (with error probability $< 1/\log^3 N$), and by lemma 3, this will enable us to invert $E(x)$.

Let $x \in I$ (the case $x \in J$ is treated similarly), and let k be an odd integer, $|k| < B$. Define the following 0-1 random variables on M:

$$X_k(m) = 1 \text{ iff } (\frac{x(m+k)}{N}) = 1.$$

$$Z_k(m) = 1 \text{ iff } (\frac{x(m-k)}{N}) = 1.$$

$$Y(m) = 1 \text{ iff } \mathcal{O}_{\mathcal{H}} \text{ is correct on } E(xm).$$

$$Y_k(m) = 1 \text{ iff } \mathcal{O}_{\mathcal{H}} \text{ is correct on } E(x(m+k)).$$

Lemma 15. *Let $N = pq$, where $p > q > 2B$, then for all k, l, $0 < |k|, |l| < B$,*
(a) *The expectation of X_k, $Exp(X_k) = \frac{1}{2} \pm \eta + O(q^{-1})$.*
(b) $Exp(Z_k) = \frac{1}{2} \pm \eta + O(q^{-1})$.
(c) $Exp(X_k X_l) = \frac{1}{4} \pm \eta + O(q^{-1})$ *for $k \neq l$, and similarly for $Exp(Z_k Z_l)$.*
(d) $Exp(Y) > \frac{3}{4} + \epsilon$.
(e) $Exp(Y_k | X_k) = Exp(Y | Z_k)$.
(f)

$$|\frac{1}{B} \sum_{\substack{|k| \leq B \\ k \text{ odd}}} Exp(Y|X_k) - Exp(Y)| < \frac{3}{4}\epsilon$$

and similarly for Z_k.

Proof. (a) If $x \in M$ then $(\frac{x(m+k)}{N}) = (\frac{m+k}{N})$, so let R be the number of m, $0 \leq m < N$, such that $(\frac{m}{N}) = (\frac{m+k}{N}) = 1$ and let

$$Q = \frac{1}{4} \sum_{m=0}^{N-1} (1 + (\frac{m}{N}))(1 + (\frac{m+k}{N}))$$

then $|Q - R| < \frac{2}{4}(p+q) < p$. Now

$$Q = \frac{1}{4}(N + 2\sum_m (\frac{m}{N}) + \sum_m (\frac{m}{N})(\frac{m+k}{N}))$$

Denote $f(m) = m(m+k)$. Since $\sum_m (\frac{m}{N}) = 0$ we have

$$\begin{aligned} |Q - \frac{N}{4}| &\leq \frac{1}{4} |\sum_m (\frac{f(m)}{N})| \\ &\leq \frac{1}{4} |\sum_{u=0}^{p-1} \sum_{v=0}^{q-1} (\frac{f(uq+vp)}{N})| \\ &\leq \frac{1}{4} |\sum_u \sum_v (\frac{f(uq)}{p})(\frac{f(vp)}{q})| \\ &\leq \frac{1}{4} |(\sum_u (\frac{f(uq)}{p}))(\sum_v (\frac{f(vp)}{q}))| \\ &\leq \frac{1}{4}. \end{aligned}$$

The last inequality follows because the sum of a character over a finite field on a non square degree

2 polynomial is ± 1, (see [8]). Thus $|\frac{Q}{N} - \frac{1}{2}| < \frac{1}{q}$. Since the value of the Jacobi symbol modulo N is symmetric around $\frac{N}{2}$, and the error introduced by adding kx is less than η, this proves our claim.

(b) By the same argument as in (a).

(c) Again let R be the number of m such that $(\frac{m}{N}) = (\frac{m+k}{N}) = (\frac{m+l}{N}) = 1$ and let

$$Q = \frac{1}{8}\sum_{m}^{N-1}(1+(\frac{m}{N}))(1+(\frac{m+k}{N}))(1+(\frac{m+l}{N}))$$

then $|Q - R| < \frac{3}{8}(p+q) < p$. Denote $f(m) = m(m+k)(m+l)$ then by the same argument as in (a)

$$|Q - \frac{N}{8}| \leq \frac{1}{8}(3 + |\sum_{m=0}^{N-1}(\frac{f(m)}{N})|)$$
$$\leq \frac{1}{8}(3 + |(\sum_{u=0}^{p-1}(\frac{f(uq)}{p}))(\sum_{v=0}^{q-1}(\frac{f(vp)}{q}))|)$$

By the deep results of A. Weil [8] we have

$$|Q - \frac{N}{8}| \leq \frac{1}{8}(3 + (2p^{-\frac{1}{2}}) \cdot (2q^{-\frac{1}{2}})) < p$$

which proves our claim.

(c) By the inclusion exclusion principle.

(d) Because the error probability of O_X is less than $\frac{1}{4} - \epsilon$.

(e) Immediate from the definitions.

(f) Let $S = \frac{1}{B}\sum_k X_k$ then by (a) $Exp(S) = \frac{1}{2} \pm \eta$, and by (b) the X_k are pairwise independent so we can bound the variance of S, $V(S) \leq 2^{-7}\epsilon^3$. Thus by Chebyshev's inequality

$$P(|S - \frac{1}{2}| > \frac{\epsilon}{4}) < \frac{\epsilon}{8}.$$

Now

$$\frac{1}{B}\sum_k Exp(Y \mid X_k) = \frac{1}{B}\sum_k 2Exp(YX_k)$$
$$= 2Exp(YS)$$

but

$$Exp(YS) < (\frac{1}{2} + \frac{\epsilon}{4})Exp(Y) + \frac{\epsilon}{8} \leq \frac{1}{2}Exp(Y) + \frac{3}{8}\epsilon$$

and

$$Exp(YS) > (\frac{1}{2} - \frac{\epsilon}{4})Exp(Y) - \frac{\epsilon}{8} \leq \frac{1}{2}Exp(Y) - \frac{3}{8}\epsilon$$

Thus $|2Exp(YS) - Exp(Y)| < \frac{3}{4}\epsilon$.

Thus if $x \in I$ the probability that the oracle will be correct on both queries is given by

$$\frac{1}{B}\sum_k Exp(YY_k|X_k)$$
$$\geq \frac{1}{B}\sum_k (Exp(Y|X_k) + Exp(Y_k|X_k) - 1)$$
$$\geq \frac{1}{B}\sum_k Exp(Y|X_k) + \frac{1}{B}\sum_k Exp(Y|Z_k) - 1$$
$$\geq 2E(Y) - \frac{3}{2}\epsilon - 1 \geq \frac{1}{2} + \frac{\epsilon}{2}$$

and this proves our claim. ∎

Let us demonstrate our results by discussing the question of how many messages are needed in order to hide one bit, using Rabin's encryption, such that an adversary cannot have more than 1 percent advantage in guessing this bit (unless he can factor N). Using the wiretap techniques (see Yao [9]), 6 messages ($\log N$ bits long) are enough. Previous bounds (Goldwasser, Micali and Tong [3]) would require at least $3 \log N$ messages.

6. Discussion

The strong bit security of the RSA and Rabin schemes is a double edged sword. On one side, it strongly enhances our belief in the cryptographic security of these systems. On the other hand, it means that the (potentially easier) task of revealing one bit is sufficient to crack the system altogether. This risk is emphasized in the presence of cryptographic protocols, which might leak one bit through communication. These mutual connections between complexity, information and protocols will undoubtedly play a central role in future cryptography research.

Acknowledgment

The second author would like to thank Mike Sipser for many helpful discussions.

References

[1] Blum, M. and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo Random Bits", *Proceedings of the 23rd FOCS*, 1982.

[2] Goldwasser, S. and S. Micali, "Probabilistic Encryption & How to Play Mental Poker, Keeping Secret All Partial Information", *Proceedings of the 14th STOC*, 1982.

[3] Goldwasser, S., S. Micali and P. Tong, "Why and How to establish a Private Code On a Public Network" (extended abstract), *Proceedings of the 23rd FOCS*, 1982.

[4] Knuth, D., *The Art of Computer Programming*, Addison-Wesley, 1969.

[5] Long, D. and A. Widgerson, "How Discreet is the Discrete Log ?", these proceedings.

[6] Rabin, M., "Digitalized Signatures and Public key Functions as Intractable as Factorization", MIT/LCS/TR-212, Technical Report, MIT, 1979.

[7] Rivest, R., A. Shamir and L. Adelman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *CACM*, February 1978.

[8] Schmidt, W.A., *Equations Over Finite Fields, An Elementary Approach*, Springer-Verlag, Lecture Notes in Math. 536, 1976.

[9] Yao, A., "Theory and Applications of Trapdoor Functions," *Proceedings of the 23rd FOCS*, 1982.