RSA-bits are $0.5 + \varepsilon$ secure

C.P. Schnorr and W. Alexi
Fachbereich Mathematik
Universität Frankfurt

February 1984

Abstract We prove by some novel sampling techniques that the least significant bits of RSA-messages are $0.5 + \varepsilon$ - secure. Any oracle which correctly predicts the k-th least significant message bit for at least a $0.5 + \varepsilon$ - fraction of all messages can be used to decipher all RSA ciphertexts in random polynomial time (more precisely in time $(\log n)^{O(\varepsilon^{-2}+k)}$) .

T. Beth, N. Cot, and I. Ingemarsson (Eds.): Advances in Cryptology - EUROCRYPT '84, LNCS 209, pp. 113-126, 1985.

1. Introduction

The most interesting feature of modern cryptography is the interaction of arguments from complexity theory, information theory and number theory. As a result of this interaction the cryptographic security of simple pseudo random number generators has been based on reasonable number theoretic assumptions, see M. Blum, S. Micali (1982) and A. Yao (1982). The bit security of the RSA-scheme plays an important role in this context. If the problem of deciphering the RSA-ciphertexts can be reduced to the problem of getting partial information on single cleartext bits then an interesting situation arises. Either it is easy to decipher RSA-ciphertexts completely (in worst case without knowing the private key) or it is infeasible to get even partial information on single RSA-cleartext bits and in this latter case the RSA-encryption provides a simple cryptographically secure pseudo random number generator.

The bit security of the RSA-scheme was first studied by Goldwasser, Micali, Tong (1982). They showed that obtaining the least significant bit of an RSA-message is as hard as obtaining the entire message. Formally they proved that any oracle which queried on an RSA-ciphertext outputs the least significant bit of the corresponding message, can be used to decrypt RSA efficiently. Ben-Or, Chor and Shamir (1983) proved that the two least significant RSA bits are $3/4+\varepsilon$-secure, i.e. any oracle for these bits which is correct for an $3/4+\varepsilon$-fraction of the ciphertexts can be used to decrypt RSA efficiently. They also showed that certain other bits are $15/16+\varepsilon$-secure. The problem remained whether RSA-bits are $1/2+\varepsilon$-secure. This would imply that RSA-bits yield a cryptographically secure pseudo random number generator. With some novel sampling techniques U.V. and V.V. Vazirani (1983) proved that the least significant RSA bits are $0.732+\varepsilon$ secure.

In this paper we finally prove that the least significant RSA-bits are $0.5+\varepsilon$-secure. More formally any oracle which correctly predicts the k-th least significant RSA-bit for at least a $0.5+\varepsilon$-fraction of all messages can be used to decipher all RSA ciphertexts in time $(\log n)^{O(\varepsilon^{-2}+k)}$.

2. The binary gcd-method for deciphering the RSA-scheme

For an integer $n \geq 1$ let $\mathbb{Z}_n^* = \{x \bmod n \mid x \in \mathbb{Z}, \gcd(x,n) = 1\}$ be the multiplicative group of integers modulo n which are relatively prime to n. The elements of $\mathbb{Z}_n^*$ will be represented by the integers x with $0 < x < n$, $\gcd(x,n) = 1$. Let $l_i(x)$ be the i-th least significant bit of x, i.e. $x = \sum_{i \geq 1} l_i(x) 2^{i-1}$ with $0 < x < n$. Call $x \in \mathbb{Z}_n^*$ even if $l_1(x) = 0$. Note that 2x may be odd; this happens iff $n/2 < x < n$ with n odd. Throughout the paper n will be odd. Let $E: x \longrightarrow x^e$ be an RSA encryption function and let $\mathcal{O}$ be an oracle which given E(a) determines $l_1(a)$ the least significant bit of a.

Theorem 1 [Ben-Or, Chor, Shamir] There is a random polynomial time algorithm which queries the oracle $\mathcal{O}$ at most $O(\log_2 n)$ times and with probability $\geq 1/2$ deciphers E(x).

The deciphering algorithm computes $b \in \mathbb{Z}_n^*$ such that $E(xb) = 1 \bmod n$ and $x := b^{-1} \bmod n$. A particular version of the binary gcd-algorithm computes b from randomly chosen elements b_1, b_2 with xb_1, $xb_2 < n/2$. The oracle is used for testing "$xb_1 < xb_2$?" and "xb_i even?". In fact $xb_1 < xb_2 \Leftrightarrow \mathcal{O}E(2x(b_1-b_2)) = 0$; xb_i even $\Leftrightarrow \mathcal{O}E(xb_i) = 0$.

An important observation is that oracle queries on large elements $2x(b_1-b_2)$ can be avoided. In section 3 we show that oracle queries on small elements can be answered correctly even if the oracle has error probability $0.5-\varepsilon$.

Algorithm 1 (the new deciphering algorithm)

1. pick random elements $b_1, b_2 \in \mathbb{Z}_n^*$ with $b_i x$ odd and $b_i x < 2^{-k} n$ for $i = 1,2$; $c_1 := c_2 := 2^{-k}$;

2. $i := \begin{cases} 1 & \text{if } c_1 \geq c_2 \\ 2 & \text{if } c_2 > c_1 \end{cases}$;

   $b_i := (b_1+b_2)/2 \bmod n$; $c_i := (c_1+c_2)/2$;

3. if $xb_i = 1 \bmod n$ then $[x := b_i^{-1} \bmod n$, stop] ;

   if $xb_1 = xb_2$ then stop (failure);

   if xb_i is odd then goto 2;

4. while xb_i even do $[b_i := b_i/2 \bmod n$, $c_i := c_i/2]$;

   goto 2;

The values c_i do not increase. An easy induction shows that $xb_i \leq c_i n$ throughout the computation. Since c_1, c_2 cannot become smaller than $1/n$, the oracle is queried at most $4 \log_2 n$ times in step 4. Since each pass of step 2 halves the difference (b_1-b_2) there are at most $\log_2 n$ consecutive passes of step 2. This proves

Lemma 2 In the new deciphering algorithm all oracle queries are on elements $bx < 2^{-k} n$. The oracle is queried at most $4 \log_2^2 n$ times.

3. The 0.5+ε-security of the least significant RSA-bit

Now consider the case that the oracle $\mathcal{O}$ makes errors. Let $\mathcal{O}_\varepsilon$ be an oracle which has an $1/2+\varepsilon$-advantage in predicting the least significant bit, i.e.

$\#\{x \in \mathbb{Z}_n^* : \mathcal{O}_\varepsilon[E(x)] = l_1(x)\} / \# \mathbb{Z}_n^* \geq 1/2 + \varepsilon$.

We will exploit the relation (let $\oplus$ be the exclusive or)

$$l_1(a+b) = l_1(a) \oplus l_1(b) \quad , \quad \text{for } a,b \in \mathbb{Z}_n^*$$

which holds provided that $a+b$ does not overlap n .

Suppose we like to decipher $E(x)$ and we already know some b with $xb < \varepsilon n/2$. Then we need to know $l_1(xb)$. We show that knowledge of $l_1(xr)$ for a random element $r \in \mathbb{Z}_n^*$ helps determing $l_1(xb)$:

Fact 3 Let $r \in \mathbb{Z}_n^*$ be a random element. Then for all $b\,x \in \mathbb{Z}_n^*$ with $bx < \varepsilon n/2$

$\text{prob}[\mathcal{O}_\varepsilon[E((r+b)x)] \oplus l_1(rx) = l_1(bx)] \geq (1+\varepsilon)/2$.

Proof "$\mathcal{O}_\varepsilon[E((b+r)x)] \oplus l_1(rx) = l_1(bx)$" holds if $\mathcal{O}_\varepsilon[E((b+r)x)] = l_1((b+r)x)$ and if $bx + rx$ does not overlap n . Since $bx < \varepsilon n/2$, overlap over n occurs with probability $\leq \varepsilon/2$. Moreover

$\text{prob}[\mathcal{O}_\varepsilon[E((b+r)x)] = l_1((b+r)x)] \geq 1/2 + \varepsilon$

Q.E.D.

By a majority decision we can determine $l_1(bx)$ for $bx < \varepsilon n$ with arbitrary high security provided that we know $l_1(r_i x)$ for sufficiently many independent elements $r_i \in \mathbb{Z}_n^*$:

Lemma 4 Let $r_1,\ldots,r_t \in \mathbb{Z}_n^*$, t odd, be independent random elements. Then for all $x, b \in \mathbb{Z}_n^*$ with $xb < \varepsilon n/2$, $\varepsilon \le 1/4$ the event "$l_1(xb) \neq \lceil \frac{1}{t} \Sigma_{i=1}^{t} \mathcal{O}_\varepsilon[E(x(b+r_i))] \oplus l_1(xr_i) \rfloor$" has probability $\le 2\exp(-t\varepsilon^2/3)$ ($\lceil \alpha \rfloor$ denotes the nearest integer to α).

Proof We use the following version of the law of large numbers, see e.g. Renyi VII, §4, theorem 1:

Let $X_1,\ldots,X_t$ be independent random variables with mean value m and variance d, $|X_i - m| \le K$. Then

$$\text{prob}\left[\left|\frac{1}{t}\Sigma_{i=1}^{t} X_i - m\right| \ge \mu d/\sqrt{t}\right] \le 2\exp\left[\frac{-\mu^2}{2(1+\frac{\mu K}{2d\sqrt{t}})^2}\right]$$

for all $\mu \le d\sqrt{t}/K$.

We apply the theorem to

$$X_i = \mathcal{O}_\varepsilon[E(x(b+r_i))] \oplus l_1(xr_i) \oplus l_1(xb)$$

Clearly $X_i = 1$ iff $\mathcal{O}_\varepsilon[E(x(b+r_i))] \oplus l_1(xr_i) \neq l_1(xb)$.

We know from Fact 3 that $m \le (1-\varepsilon)/2$. $\varepsilon \le 1/4$ implies $K = 5/8$, $2 \le 1/d \le (2\frac{5}{8}\cdot\frac{3}{8})^{-1} \le 2.14$.

If "$l_1(xb) \neq \lceil \frac{1}{t}\Sigma_{i=1}^{t} \mathcal{O}_\varepsilon[E(x(b+r_i))] \oplus l_1(xr_i) \rfloor$" then $\left|\frac{1}{t}\Sigma_{i=1}^{t} X_i - m\right| \ge \varepsilon/2$.

By the law of large numbers with $\mu = \sqrt{t}\,\varepsilon/(2d)$ the latter event has probability $\le$

$$2\exp\left[\frac{-t\varepsilon^2/(4d^2)}{2(1+\frac{\varepsilon K}{(2d)^2})^2}\right] \le 2\exp[-t\varepsilon^2/3].$$

Q.E.D.

Corollary 5 Let $bx \in \mathbb{Z}_n^*$, $bx < \varepsilon n/2$, $\varepsilon \leq 1/4$ and let $r_1,\ldots,r_t \in \mathbb{Z}_n^*$ be independent random elements, $t \geq 3\varepsilon^{-2}\log(2s)$. Then $\text{prob}[l_1(bx) \neq \lceil \frac{1}{t} \Sigma_{i=1}^{t} O_\varepsilon[E((r_i+b)x)] \oplus l_1(rx) \rfloor \leq 1/s$.

Proof By Lemma 4 the event in question has probability $\leq 2\exp[-t\varepsilon^2/3]$. Thus it is sufficient to choose $t \geq 3\varepsilon^{-2}\log$ $t \geq 3\varepsilon^{-2}\log(2s)$. Q.E.D.

A key observation is that once we have guessed $l_1(r_i x)$ for sufficiently many random elements $r_i \in \mathbb{Z}_n^*$ then we can determine with sufficiently high security $l_1(bx)$ for any $bx < \varepsilon n/2$.

Theorem 6 There is a random $(\log n)^{O(\varepsilon^{-2})}$-time algorithm using oracle O_ε that inverts the encryption function $E(x)$.

Proof In order to decipher $E(x)$ do the following.

1. pick random elements $r_1,\ldots,r_t \in \mathbb{Z}_n^*$, (t will be determined below).
2. guess $b_1,b_2 \in \mathbb{Z}_n^*$ with $b_i x \leq \varepsilon n/2$ for $i = 1,2$.
3. guess $l_1(r_i x)$ for $i = 1,\ldots,t$.
4. simulate the binary gcd deciphering method but stop after at most $4\log_2^2 n$ oracle queries. For each oracle query compute $l_1(xb) := \lceil \frac{1}{t} \Sigma_{i=1}^{t} O_\varepsilon[E((r_i+b)x)] \oplus l_1(r_i x) \rfloor$.

The algorithm succeeds if all the query answers are correct

and if the binary gcd method succeeds with initial values b_1, b_2. By Lemma 2 the oracle is queried at most $4 \log_2^2 n$ times. By Corollary 5 for $t := \lceil 3 \varepsilon^{-2} \log(2s) \rceil$ each query answer has error probability $\leq 1/s$. Choose $s := 8 \log_2^2 n$, then with probability $\geq 1/2$ all query answers are correct. It is important for our argument that $r_1, \ldots, r_t$ are independent of all intermediate elements bx occuring in the binary gcd method with initial values b_1, b_2. Guessing b_1, b_2 and $l_1(r_i x)$ for $i = 1, \ldots, t$ can be done within $2^t \varepsilon^{-2} = (\log n)^{O(\varepsilon^{-1})}$ trials. Therefore E(x) can be deciphered in $(\log n)^{O(\varepsilon^{-2})}$ steps.

Q.E.D.

4. The security of other RSA-bits

Consider $l_i(x)$ the i-th least significant bit of $x \in \mathbb{Z}_n^*$, i.e. $x = \Sigma_{i \geq 1} l_i(x)\, 2^{i-1}$ for x with $0 < x < n$. We note that for small elements $a \in \mathbb{Z}_n^*$ we can express $l_1(a)$ by $l_k(2^{k-1}a)$. In fact

(*) $\quad l_1(a) = l_k(2^{k-1}a)$ for all $a < 2^{-k+1}n$.

Let $\mathcal{O}_{L,k}$ be any oracle which for given $E(a)$ determines $l_k(a)$. By (*) we can implement the binary gcd deciphering method using $\mathcal{O}_{L,k}$ provided we guess two initial values b_i with $b_i x < 2^{-k}n$. This proves

Theorem 6 For every k there is a random $(2^k \log n)^{O(1)}$-time algorithm using the oracle $\mathcal{O}_{L,k}$ which inverts the encryption function $E(x)$.

Now suppose that $\mathcal{O}_{L,k}$ makes errors. Let $\mathcal{O}_{\varepsilon,k}$ be any oracle that has ε-advantage in predicting the k-th least significant bit $l_k(x)$ of x, more formally:

$$\#\{x \in \mathbb{Z}_n^* : \mathcal{O}_{\varepsilon,k}[E(x)] = l_k(x)\}/\#\mathbb{Z}_n^* \geq 1/2 + \varepsilon .$$

We implement the binary gcd deciphering method with oracle $\mathcal{O}_{\varepsilon,k}$ as follows:

1. pick random elements $r_1, \ldots, r_t \in \mathbb{Z}_n^*$, $t = \lceil 3(4 + \log\log n)\varepsilon^{-2} \rceil$.
2. guess $l_k(r_i x)$ for $i = 1, \ldots, t$.
3. guess elements b_1, b_2 such that $x b_i \leq \varepsilon 2^{-k} n$ for $i = 1, 2$.
4. simulate the binary gcd deciphering method as follows. For each query on $xb < \varepsilon 2^{-k}n$ put

$l_1(xb) := \lceil \frac{1}{t} \Sigma_{i=1}^{t} \mathcal{O}_{\varepsilon,k}[E(x(2^{k-1}b + r_i))] \oplus l_k(xr_i) \rfloor$
and stop after at most $4 \log_2 n$ queries.

We have $l_1(xb) = l_k(x2^{k-1}b) = l_k(x(2^{k-1}b+r_i)) \oplus l_k(xr_i)$ provided that $x2^{k-1}b + xr_i$ does not overlap n. Since $bx < \varepsilon 2^{-k}n$ overlap over n occurs with probability $\leq \varepsilon/2$. Therefore

$$l_1(xb) = \mathcal{O}_{\varepsilon,k}[E(x(2^{k-1}b + r_i))] \oplus l_k(xr_i)$$

occurs with probability $\geq (1 + \varepsilon)/2$. By the law of large numbers each oracle query has error probability $\leq 2 \exp[-t\varepsilon^2/3] \leq 1/(27 \log n)$ for $\varepsilon \leq 1/4$. Hence with probability $\geq 1/2$ all query answers are correct.

The algorithm succeeds if all query answers are correct and if $\gcd(xb_1, xb_2) = 1$. Guessing such elements b_1, b_2 with $xb_i \leq \varepsilon 2^{-k}n$ and guessing $l_k(xr_i)$ for $i = 1,\ldots,t$ can be done by $\varepsilon^{-2} 2^{2k} 2^t = (\log n)^{O(\varepsilon^{-2})} 2^{2k}$ trials. This proves

Theorem 7 For every k there is a random $(\log n)^{O(\varepsilon^{-2}+k)}$-time algorithm using oracle $\mathcal{O}_{\varepsilon,k}$ which inverts the encryption function E(x).

5. Efficient deciphering with random oracles

The previous time bound $(\log n)^{O(\varepsilon^{-2}+k)}$ can be considerably improved if the oracle $\mathcal{O}_{\varepsilon,k}$ has no particular structure. We will prove that almost all oracles $\mathcal{O}_{\varepsilon,k}$ can be used to invert the encryption function $E(x)$ in random time $[\varepsilon^{-1} \log n\, 2^k]^{O(1)}$.

For fixed k we define a probability distribution on the set of all 0-1 valued oracles $\mathcal{O}$. Oracle $\mathcal{O}$ has probability weight $(0.5+\varepsilon)^s (0.5-\varepsilon)^{\varphi(n)-s}$ with $s = \#\{y \in \mathbb{Z}_n^* \mid \mathcal{O}[E(y)] = l_k(y)\}$, $\varphi(n) = \#\mathbb{Z}_n^*$. Let $\mathcal{OR}_{\varepsilon,k}$ be a random oracle with respect to this distribution, i.e. $\mathcal{OR}_{\varepsilon,k}[E(y)]$, for $y \in \mathbb{Z}_n^*$, are 0,1-valued, independent random elements with $\text{prob}[\mathcal{OR}_{\varepsilon,k}[E(y)] = l_k(y)] = 0.5+\varepsilon$. We implement the binary gcd deciphering method with oracle $\mathcal{OR}_{\varepsilon,k}$ as follows.

1. for $t := \lceil 3(4+\log\log n)\varepsilon^{-2} \rceil$ guess a random element r with $rx < n/(2t)$ and $rx = 0 \bmod 2^k$.
2. guess elements b_1, b_2 such that $b_i x \le 2^{-k} n$ for $i = 1,2$.
3. simulate the binary gcd deciphering method; for each query on $xb < 2^{-k}n$ put
   $l_1(xb) := \lceil \frac{1}{t} \Sigma_{i=1}^{t} \mathcal{OR}_{\varepsilon,k}[E(x(2^{k-1}b + ir))] \rfloor$
   and stop after at most $4 \log_2 n$ queries.

If $rx < n/(2t)$ and $rx = 0 \bmod 2^k$ then $l_k(irx) = 0$ and $irx = 0 \bmod 2^k$ for $i \le t$. If $xb < 2^{-k}n$ then $l_k(xb) = l_1(x2^{k-1}b)$.

Hence for all $xb < 2^{-k} n$:

$\text{prob}[\mathcal{OR}_{\varepsilon,k}[E(x(2^{k-1}b + ir))] = l_1(xb)] = 0.5 + \varepsilon$.

By the law of large numbers each query answer has error probability $\leq 2 \exp(-\varepsilon^2 t/3) \leq 1/(27 \log n)$ (This means that the fraction of oracles $\mathcal{O}$ which in step 3 give a wrong value $l_1(xb)$ for a particular query, is at most $2\exp(-\varepsilon^2 t/3)$; this fraction is exponentially small for large t). With probability $\geq 1/2$ all $4 \log_2 n$ query answers with $\mathcal{OR}_{\varepsilon,k}$ are correct. Hence the algorithm using oracle $\mathcal{OR}_{\varepsilon,k}$ succeeds with probability $\geq 1/2$ if b_1, b_2, r have been guessed as specified.

Guessing r and b_1, b_2 can be done (with probability $\geq 1/2$) with $2^k 2t 2^{2k} = O(2^{3k} \varepsilon^{-2} \log \log n)$ trials. Thus $E(x)$ can be deciphered (with probability $\geq 1/2$) in $O(2^{3k} \varepsilon^{-2} (\log n)^4)$ bit operations. $((\log n)^3$ bounds the number of bit operations for evaluating $E(y)$). Thus we have proved

Theorem 8 For every k there is a random $[\varepsilon^{-1} \log n\, 2^k]^{O(1)}$-time algorithm using oracle $\mathcal{OR}_{\varepsilon,k}$ which inverts the encryption function $E(x)$.

It clearly follows that the time bound of theorem 8 holds for all but a negligible fraction of oracles $\mathcal{O}_{\varepsilon,k}$ which have an ε - advantage in predicting l_k . It is an open problem whether the time bound $[\varepsilon^{-1} \log n\, 2^k]^{O(1)}$ can be obtained for all oracles $\mathcal{O}_{\varepsilon,k}$.

References:

M. Ben-Or, B. Chor, A. Shamir, On the Cryptographic Security of Single RSA Bits, Proc. STOC 1983, 421-430

L. Blum, M. Blum, M. Snub, A Simple Secure Pseudo-Random Number Generator. Crypto 1982

M. Blum & S. Micali, How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits, Proc. FOCS 1982, 112-117.

S. Goldwasser, S. Micali, P. Tong, Why and How to Establish a Private Code on a Public Network, Proc. FOCS 1982, 134-144.

M. Rabin, Digital Signatures and Public Key Functions as Intractable as Factorization, MIT/LCS/TR-212, Technical Report, MIT, 1979.

A. Renyi, Wahrscheinlichkeitsrechnung
VEB Deutscher Verlag der Wissenschaften Berlin 1966.

R. Rivest, A. Shamir & L. Adelman, A Method of Obtaining Digital Signatures and Public Key Cryptosystems, CACM, February 1978.

A. Shamir, On the generation of Cryptographically Strong Pseudo-Random Sequences, 1981 ICALP.

A. Yao, Theory and Applications of Trapdoor Functions, proc. FOCS 1982, 80-91

U. V. and V. V. Vazirani, RSA bits are $.732 + \varepsilon$ secure.
TR U. Berkeley and Harvard University 1983.