The Cryptographic Security of the Sum of Bits

Richard Berger[*]
Howard J. Karloff[**]
David B. Shmoys[***]

University of California
Berkeley, California 94720

Abstract

We show that if there exists a deterministic oracle that can determine the sum of the bits in the binary representation of x when presented with the RSA encryption of x, then there exists a probabilistic algorithm using this oracle to recover x when presented with the RSA encryption of x. We present a similar result for Rabin encryption.

[*] Supported in part by DARPA grant N00039-C-0235-9-83 and a GTE Fellowship.

[**] Supported in part by the National Science Foundation by a graduate fellowship and under grant MCS-8105217.

[***] Supported in part by the National Science Foundation by a graduate fellowship and under grant MCS-7820054.

1. Introduction

While an encryption function may be computationally infeasible to invert, it may be easy to extract valuable partial information from the encrypted data. This problem was first discussed by Micali and Goldwasser[1]. They devise an encryption system for which all non-trivial, easily computed predicates of the cleartext are secure under the assumption that determining quadratic residuosity modulo composite numbers is hard. However, since this result is not true for the RSA or Rabin encryption, it is important to determine what partial information is secure for these encryptions. Ben-Or, Chor, and Shamir[2] discuss the least significant bit predicate and show that it is as secure for both the RSA and Rabin encryptions as the encryptions themselves. In our paper we consider the sum of bits predicate and show that it is secure in the same sense.

Blum, Blum, and Shub[3] have designed a pseudo-random number generator using the least significant bit of a Rabin encryption. We believe that our result might be used to form the basis of a pseudo-random number generator which will be faster asymptotically than any current algorithm. The number of bits in the binary representation of the sum of the bits of x is $lg\ lg\ x$, but is only 1 for the least significant bit of x; thus one might be able to build faster pseudo-random number generators.

Our main result is the following theorem:

Theorem: If there exists a deterministic oracle, **Sum Of Bits**:

Input: The encryption $E_N(x)$ where x is chosen from Z_N, and $E_N(x)$ is either the RSA or Rabin encryption of x.

Output: $\Sigma(x)$, the sum of bits in the binary representation of x,

then there exists a probabilistic algorithm A:

Input: $E_N(x)$, N

Output: x.

Furthermore, this algorithm runs in random polynomial time in the length of N.

In this paper we will present this algorithm and a proof of its correctness. Our paper makes use of the Ben-Or, Chor, Shamir result:

Let O_L be a deterministic oracle that on input $E_N(x)$ can guess the least significant bit of x, such that for a random x in Z_N the probability that O_L will err is at most $\frac{1}{4} - \epsilon$ (for some fixed $\epsilon > 0$). Then there is a random polynomial-time algorithm, using O_L, that breaks this $\begin{Bmatrix}\text{RSA}\\ \text{Rabin}\end{Bmatrix}$ encryption.

2. The Algorithm

The following algorithm computes a function that is *almost* the least significant bit; we call it $ALSB(x)$. More formally, we compute $ALSB{:}Z_N \rightarrow \{0,1\}$, a function such that $ALSB(x)$= least significant bit of x for $\geq \frac{3}{4} + \epsilon$ of the domain, where ϵ is a positive constant to be specified later.

Procedure *ALSB*
Input: N, $E_N(x)$ where x is in Z_N.
Output: $ALSB(x)$.
Algorithm:
Compute $E_N(x2^{-1} \bmod N) = E_N(x) * E_N(2^{-1} \bmod N) \bmod N$
Using oracle SOB compute $\Sigma(x)$ and $\Sigma(x2^{-1} \bmod N)$
If $\Sigma(x) \neq \Sigma(x2^{-1} \bmod N)$ then output 1
else output 0

3. Correctness of the Algorithm

The intuition behind the correctness of the algorithm is that for even x, $\Sigma(x)=\Sigma(x2^{-1} \bmod N)$, and therefore the algorithm is correct for these inputs. The hope is that for odd x the algorithm is correct frequently enough to guarantee the necessary probability overall. The remainder of this section is devoted to proving the following lemma.

Lemma 1: Let $N \neq 2^l-1$ be an odd integer. For at least $\frac{1}{2} + \epsilon$ of the odd $x \in Z_N$, $\Sigma(x) \neq \Sigma(x2^{-1} \bmod N)$.

Throughout this paper k will denote $2^{-1} \bmod N = \frac{N+1}{2}$. Furthermore, for any odd $x \in Z_N$, $r=r(x)$ will be an element of $Z_{\frac{N-1}{2}}$ such that $x=2r+1$. We now prove a few useful facts about *odd* x's.

Fact 1: $\Sigma(x)=\Sigma(r) + 1$.
Proof: Obvious.

Fact 2: $\Sigma(x2^{-1} \bmod N)=\Sigma(r+k)$.
Proof: Using straightforward manipulations we see that

$$x2^{-1} \bmod N = (2r+1)(\frac{N+1}{2}) \bmod N = rN + r + \frac{N+1}{2} \bmod N$$

$$=(r+k) \bmod N.$$

Since $x < N-1$, $r < \frac{N-2}{2}$ and $r+k < \frac{N-2}{2} + \frac{N+1}{2} < N$. Hence, $(r+k) \bmod N = r+k$.

Combining these two facts we obtain the following:

Fact 3: $\Sigma(x)=\Sigma(x2^{-1} \bmod N)$ if and only if $\Sigma(r)+1 = \Sigma(r+k)$.

If x and y are non-negative integers, let $C(x,y)$ denote the number of carries induced by the binary addition of x and y. Consider the following example:

$$\begin{array}{rl} & 1\ 111 \\ x = 17_{10} = & 10001_2 \\ y = 23_{10} = & 10111_2 \\ \hline x+y = 40_{10} = & 101000_2. \end{array}$$

Note that in the above example, $\Sigma(x) = 2$, $\Sigma(y) = 4$, $C(x,y) = 4$ and $\Sigma(x+y) = 2$, and thus $C(x,y) + \Sigma(x+y) = \Sigma(x) + \Sigma(y)$. In fact, this equality always holds.

Lemma 2: $C(x,y) + \Sigma(x + y) = \Sigma(x) + \Sigma(y)$.

The reader can verify the lemma by considering corresponding binary positions in x, y, $(x+y)$ and the carries into and out of that position. We use Lemma 2 to convert the original problem to a more convenient form. Combining Fact 3 and Lemma 2 we obtain the following corollary.

Corollary: $\Sigma(x)=\Sigma(x2^{-1} \bmod N)$ if and only if $\Sigma(k) = C(k,r)+1$.

We wish to bound the number of odd x for which $ALSB(x) \neq LSB(x)$. Since we are interested in

$$\{\text{odd } x \in Z_N \mid \Sigma(x)=\Sigma(x2^{-1} \bmod N)\},$$

we define

$$B(k) = \{r \mid \Sigma(k)=C(k,r)+1, \text{ where } 0 \leq r \leq \frac{N-3}{2} = k-2\}.$$

We wish to prove for all odd N, $N \neq 2^l - 1$, that $|B(k)| \leq (\frac{1}{2} - \epsilon)k$ for some fixed $\epsilon > 0$. Since $k=2^{l-1}$ if and only if $N=2^l-1$, it follows that $\Sigma(k)=1$ if and only if $N=2^l-1$. Therefore, if $N \neq 2^l - 1$ then $0 \notin B(k)$, since $C(k,0)=0$ and $\Sigma(k) \neq 1$. This enables us to restrict r to the range $1 \leq r \leq k-2$. We now consider the following superset of $B(k)$, $B^*(k)$:

$$B^*(k) = \{r \mid \Sigma(k) = C(k,r) + 1, \text{ where } 1 \leq r \leq k\}.$$

We are also interested in the fraction $b(k) = \frac{|B^*(k)|}{k}$. Since the fraction of odd inputs for which the algorithm answers incorrectly is at most $b(k)$, it is sufficient to show $b(k) \leq \frac{1}{2} - \epsilon$ for $k \neq 2^{l-1}$.

Claim: $b(k)=b(2k)$.

Proof: Note that $k \notin B^*(k)$; with this restriction it follows that $r \in B^*(k)$ if and

only if both $2r \in B^*(2k)$ and $2r+1 \in B^*(2k)$. This follows from considering the binary representations of r, $2r$, and $2r+1$. Since both $2r$ and $2r+1$ are $\in B^*(2k)$ or neither are, $b(k)=b(2k)$ follows.

Thus we are only required to prove that $b(k) \leq (\frac{1}{2} - \epsilon)$, for some fixed $\epsilon > 0$, for *odd* $k \geq 3$.

Lemma 3: Let r be an even integer in $B^*(k)$, for some odd $k \geq 3$. If $r \in B^*(k)$ then $r+1 \notin B^*(k)$.

Proof: Assume $r \in B^*(k)$, so that $\Sigma(k) - C(r,k) = 1$. We will show that $C(r,k) \neq C(r+1,k)$, which proves the claim. Since k is odd and r is even there is no carry out of the least significant bit when adding r and k. However, when adding $r+1$ and k there is a carry out of the least significant bit. In fact this carry propagates up to the least significant bit position such that r and k are 0 in that position. To the left of this position the carries are identical and to the right of this position there is a strictly greater number of carries when adding $r+1$ and k than when adding r and k.

Using the fact that $1 \notin B^*(k)$, we can conclude that for $k \neq 2^{l-1}$, $b(k) \leq \frac{1}{2}$. Now we will construct a constant ϵ such that $b(k) \leq \frac{1}{2} - \epsilon$. We do a case analysis on the form of the binary representation of k.

We first consider two very general cases; after this the remaining scattered cases will be dealt with in a *ad hoc* fashion. Our notation for describing these cases is borrowed from formal language theory.

Suppose that $k=1\{0,1\}^l 10\{0,1\}^m 01$ for some $l \geq 1$ and $m \geq 0$. Consider the proof that $b(k) \leq \frac{1}{2}$. Since $1 \notin B^*(k)$, we pair off the numbers [2,3], [4,5] up to $[k-1,k]$. (Recall that k is odd.) By Lemma 3, we know that for every pair at most one of each pair is in $B^*(k)$, that is, *bad*. In order to prove that $b(k) \leq \frac{1}{2} - \epsilon$, we need to show that for a fixed fraction of the pairs, neither number in the pair is bad. Suppose that $[r,r+1]$ is a pair where one of r and $r+1$ is bad. Further assume that r is of the form, $0x0^3y00$ where $x \in \{0,1\}^{l-1}$ and $y \in \{0,1\}^m$. Consider

the pair $[r',r'+1]$ where $r'=0x110y00$. It is not hard to verify that

$$C(r,k)<C(r,k)+1=C(r+1,k)<C(r',k)=C(r'+1,k)-1<C(r'+1,k).$$

These inequalities guarantee that if either of r or $r+1$ is bad, then neither of r' and $r'+1$ is bad. One can show that this procedure is in fact a 1-1 mapping between pairs of the appropriate forms. As a result, this mapping proves that for 1/32 of the pairs, neither number is bad. This proves that an additional 1/64 of the numbers are not in $B^*(k)$; thus $b(k)\leq\frac{1}{2}-\frac{1}{64}$ for k of the form given above.

For k of the form $1\{0,1\}^l10\{0,1\}^m11$ with $l\geq1$ and $m\geq0$, the proof is nearly identical. The only difference is that we consider $[r,r+1]$ pairs with r of the form $1x000y10$, where $x\in\{0,1\}^{l-1}$ and $y\in\{0,1\}^m$. In this case, the $[r,r+1]$ pairs get mapped to $[r',r'+1]$ pairs, with r' of the form $1x110y10$.

It remains to consider the values of k that are not covered by the previous two cases. We can show that the remaining possibilities can be summarized by the following six cases:

1. $k=1^l$;
2. $k=1^l01$;
3. $k=10^l1^m$;
4. $k=10^l1^m01$;
5. $k=110^l1^m$;
6. $k=110^l1^m01$;

where $l,m\geq1$ for each of these cases. It is tedious but not difficult to show that for each of these cases $b(k)\leq\frac{1}{2}-\frac{1}{64}$.

4. Conclusions

In this paper we have proved that determining the sum of bits of the cleartext, given only the RSA (Rabin) encryption, is as difficult as decrypting. It is significant to note that we have actually proved something slightly stronger. The oracle need not always be correct; in particular the oracle may lie on a fixed fraction δ of the inputs, where $\delta<\frac{1}{256}$. Recently, Vazirani and Vazirani[4] have claimed an improvement in the Ben-Or, Chor, Shamir bound of $\frac{1}{4}-\epsilon$. This result allows us to use an oracle that lies a greater percentage of the time.

5. References

1. S. Goldwasser and S. Micali, "Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information," *14th STOC,* 1982, 365-377.
2. M. Ben-Or, B. Chor, and A. Shamir, "On the Cryptographic Security of Single RSA bits," *15th STOC,* 1983, pp.421-430.
3. L. Blum, M. Blum, and M. Shub, "A Simple Secure Pseudo-Random Number Generator," in: D. Chaum, R.L. Rivest, A.T. Sherman (eds.), *Advances in Cryptology.* Plenum Press, New York, 1983, pp. 61-78.
4. U. Vazirani and V. Vazirani, *private communication.*