Stronger Security Proofs for RSA and Rabin Bits

R. FISCHLIN and C.P. SCHNORR

Fachbereich Mathematik/Informatik
Universität Frankfurt
PSF 111932
60054 Frankfurt/Main, Germany

Abstract. The RSA and Rabin encryption function are respectively defined as $E_N(x) = x^e \bmod N$ and $E_N(x) = x^2 \bmod N$, where N is a product of two large random primes p, q and e is relatively prime to $\varphi(N)$. We present a much simpler and stronger proof of the result of ALEXI, CHOR, GOLDREICH and SCHNORR [ACGS88] that the following problems are equivalent by probabilistic polynomial time reductions: (1) given $E_N(x)$ find x; (2) given $E_N(x)$ predict the least-significant bit of x with success probability $\frac{1}{2} + \frac{1}{\text{poly}(n)}$, where N has n bits. The new proof consists of a more efficient algorithm for inverting the RSA/Rabin-function with the help of an oracle that predicts the least-significant bit of x. It yields provable security guarantees for RSA-message bits and for the RSA-random number generator for moduli N of practical size.

1 Introduction

Randomness is a fundamental computational resource and the efficient generation of provably secure pseudorandom bits is a basic problem. YAO [Y82] and BLUM, MICALI [BM84] have shown that perfect random number generators (RNG) exist under reasonable complexity assumptions. Some perfect RNG's are based on the RSA-function $E_N(x) = x^e \bmod N$ and the Rabin-function $E_N(x) = x^2 \bmod N$, where the integer N is a product of two large random primes p, q and e is relatively prime to $\varphi(N) = (p-1)(q-1)$. The corresponding RNG transforms a random seed $x_0 \in [1, N)$ into a bit string $b_1, \ldots, b_m$ of arbitrary polynomial length $m = n^{O(1)}$ according to the recursion $x_i := E_N(x_{i-1})$, $b_i := x_i \bmod 2$, where N has n bits. The security of these RNG's is related to a result of [ACGS88] that the RSA/Rabin-function can be inverted in polynomial time if one is given an oracle which predicts from given $E_N(x)$ the least-significant bit of x with success probability $\frac{1}{2} + \frac{1}{\text{poly}(n)}$. While the ACGS-result shows that the RSA/Rabin RNG is perfect in an asymptotic sense the practicality of this result has been questionable as the transformation of attacks against these RNG's into a full inversion of the RSA/Rabin-function (resp. the factorization of N) is rather slow.

The main contribution of this paper is a much simpler and stronger proof of the ACGS-result. The new proof gives a more efficient algorithm for the inversion of the RSA/Rabin-function if one is given an oracle that predicts the

W. Fumy (Ed.): Advances in Cryptology - EUROCRYPT '97, LNCS 1233, pp. 267-279, 1997.

least significant message bit. While the new method is primarily of theoretical interest, it yields a security guarantee for moduli N of practical size. We extend our results to the Rabin-function $E_N(x) = x^2 \bmod N$. The reduction from E_N-inversion, resp. factoring N, to prediction is particular efficient for the *absolute* Rabin-function $E_N^a(x) = |x^2 \bmod N|$, where $|y| = \min(y, N-y)$.

Notation. Let N be product of two large primes p, q. Let $\mathbb{Z}_N = \mathbb{Z}/N\mathbb{Z}$ be the ring of integers modulo N, and let $\mathbb{Z}_N^*$ denote the subgroup of invertible elements in $\mathbb{Z}_N$. We represent elements $x \in \mathbb{Z}_N$ by their least nonnegative residue in the interval $[0, N)$, i.e., $\mathbb{Z}_N = [0, N)$. We let $[ax]_N \in [0, N)$ denote the least nonnegative residue of $ax (\mathrm{mod} N)$. We use $[ax]_N$ for arithmetic expressions over $\mathbb{Z}$ while the arithmetic over $a, x \in \mathbb{Z}_N = [0, N)$ is done modulo N. Let n be the bit length of N, $2^{n-1} < N < 2^n$. For $x \in \mathbb{Z}$ we let $\ell(x) = x \bmod 2$ denote the *least-significant bit* of x. Let e be relatively prime to $\varphi(N) = (p-1)(q-1)$, $e \neq 1$. The RSA cryptosystem enciphers a message $x \in \mathbb{Z}_N$ into $E_N(x) = x^e \bmod N$. Let O_1 be an oracle running in expected time T which, given $E_N(x)$ and N, predicts the least-significant bit $\ell(x)$ of x with advantage ε: $\Pr_{x,w}[O_1(E_N(x)) = \ell(x)] \geq \frac{1}{2} + \varepsilon$, where the probability refers to random $x \in_R [0, N)$ and the internal coin tosses w of the oracle. We assume that the time T of the oracle also covers the n^2 steps for the evaluation of the function E_N. Throughout the paper we assume that ε^{-1} and n are powers of 2, $n \geq 2^9$. We let lg denote the logarithm function with base 2. For a finite set A let $b \in_R A$ denote a random element of A that is uniformly distributed. All time bounds count arithmetic steps using integers with $\lg(n\varepsilon^{-1})$ bits. We use integers of that size for counting the votes in majority decisions.

Our results. Consider the problem to compute from $E_N(x)$ and N the message $x \in \mathbb{Z}_N$ with the help of the oracle O_1 but without knowing the factorization of N. The new method inverts E_N by iteratively tightening approximations uN of random multiples $[ax]_N$ with known multiplier a via binary division. The basic idea is that $[\frac{1}{2}ax]_N = \frac{1}{2}[ax]_N$ for even $[ax]_N$, $[\frac{1}{2}ax]_N = \frac{1}{2}([ax]_N + N)$ for odd $[ax]_N$. Thus we get from a rational approximation uN to $[ax]_N$ and the least-significant bit $\ell(ax)$ a tighter approximation to $[\frac{1}{2}ax]_N$:

$$[\tfrac{1}{2}ax]_N - \tfrac{1}{2}(u + \ell(ax))N = \tfrac{1}{2}([ax]_N - uN).$$

Without knowing x we get $\frac{1}{2}(u + \ell(ax))N$ from the multiplier a, the previous approximation uN and $E_N(x)$. This in turn yields $E_N(ax) = E_N(a)E_N(x)$ and a guess $O_1(E_N(ax))$ for $\ell(ax)$. Binary division without an oracle has already been used by GOLDWASSER, MICALI, TONG [GMT82]. The method of binary division is more efficient than the gcd-method in [BCS83], [ACGS88]. In order to decipher $E_N(x)$ it guesses the least-significant bits and approximate locations of two random multiples $[ax]_N, [bx]_N$ whereas the gcd-method requires four random multiples. Most importantly, the number of oracle calls becomes nearly minimal.

In section 2 we present our basic algorithm that inverts the RSA-function E_N in expected time $O(n^2\varepsilon^{-2}T + n^2\varepsilon^{-6})$, where T is the time and ε the advantage of oracle O_1. The expectation refers to the internal coin tosses of O_1 and of the inversion algorithm. This greatly improves the [ACGS88]-time bound $O(n^3\varepsilon^{-8}T)$ for oracle RSA-inversion. The new time bound differentiates the costs induced by the oracle calls and the *additional overhead*. The oracle calls induce $O(n^2\varepsilon^{-2}T)$

steps, we call the $O(n^2\varepsilon^{-6})$ other steps the additional overhead. We generalize our security result to the j-th least-significant message bit for arbitrary j. This generalization affects only the additional overhead of E_N-inversion, the number of oracle calls remains unchanged.

In section 3 we introduce the *subsample majority rule*, a trick that improves the efficiency of majority decisions. Suppose we are given pairwise independent 0,1-valued votes that each has an advantage ε in predicting the target bit $\ell(a_t x)$. A large sample size m is necessary in order to make the error probability $\frac{1}{m\varepsilon^2}$ of the majority decision sufficiently small. To reduce the computational costs of the large sample we only use a small random subsample of it. While the random subsample induces only a small additional error probability the time for the subsample majority decision reduces to the size of the small subsample. The large sample is only mentally used for the analysis, it does not enter into the computation. Using this trick we gain a factor $\frac{n}{\lg n}$ in the number of oracle calls and in the time for the inversion of E_N. The reduced number of oracle calls is optimal up to factor $O(\lg n)$.

In section 4 we process all possible locations for $[ax]_N, [bx]_N$ much faster than trying them separately. This reduces the additional overhead in the time for RSA-inversion to $O(n^2\varepsilon^{-4}\lg(n\varepsilon^{-1}))$.

In section 5 we give conclusions for the security of RSA-message bits and of the RSA-random number generator for moduli N of practical size. These conclusions are preliminary as the additional overhead can be further reduced.

In section 6 we extend the oracle inversion algorithm to the Rabin-function E_N and we derive a security guarantee for the $x^2 \bmod N$ generator under the assumption that factoring is hard. The oracle inversion of the absolute Rabin-function is as fast as that of the RSA-function . For the centered Rabin-function the inversion runs in time $O(n\varepsilon^{-4}\lg(n\varepsilon^{-1})T)$. The latter improves the previous time bound $O(n^3\varepsilon^{-11}T)$ due to [VV84] in connection with [ACGS88].

2 RSA-inversion by binary division

We introduce a novel method for inverting the RSA-function without knowing the factorization of N if one is given an oracle O_1 that predicts the least-significant message bit with non-negligible advantage ε. The algorithm RSA-inversion is a simple version of the new method, that will be made more efficient by subsequent modifications. In order to invert $E_N(x)$ it picks two random multipliers a, b and guesses the least-significant bits and the approximate locations for the message multiples $[ax]_N, [bx]_N$. For $a_t := a\,2^{-t} \bmod N$ it iteratively constructs rational approximations $u_t N$ so that $|[a_t x]_N - u_t N| \leq \frac{\varepsilon N}{4 \cdot 2^t}$ for $t = 1, ..., n$. To this end it uses the method of binary division explained in the introduction. From the approximation $u_n N$ to $[a_n x]_N$ we get the message $x = a_n^{-1} \lfloor u_n N + \frac{1}{2} \rfloor \bmod N$. The main work is to determine the bits $\ell(a_t x)$ by majority decision using the oracle O_1.

The majority decision for $\ell(a_t x)$ uses multipliers $a_t + i a_{t-1} + b$ that are pairwise independent. Recall that the arithmetic on a, b, x, a_t is done modulo N.

The algorithm determines an integer $w_{t,i}$ that most likely satisfies the equation $(a_t + ia_{t-1} + b)x = [a_t x]_N + i[a_{t-1}x]_N + [bx]_N - w_{t,i}N$, in which case we call $w_{t,i}$ *correct*. The i-th measurement guesses $\ell(a_t x)$ by evaluating ℓ for both sides of the latter equation. We guess ℓ for the left hand side via the oracle O_1 and we use that the right hand side is linear in $\ell(a_t x)$. The majority decision performs $m = \min\{2^t, 2n\}\varepsilon^{-2}$ measurements, where m and the set A_m of integers i is chosen as to optimize the trade-off between error probability and efficiency of the majority decision. The use of pairwise independent multipliers for majority decision is a crucial contribution of [ACGS88].

RSA-inversion

1. INPUT $E_N(x)$, N
   $t := 0$ (t is the *stage*), pick random integers $a, b \in_R \mathbb{Z}_N^* \subset [0, N)$,
   guess rational integers $u \in \frac{\varepsilon^3}{4}[0, 4\varepsilon^{-3})$, $v \in \frac{\varepsilon}{4}[0, 4\varepsilon^{-1})$ satisfying
   $$|[ax]_N - uN| \leq \tfrac{\varepsilon^3}{8}N, \quad |[bx]_N - vN| \leq \tfrac{\varepsilon}{8}N.$$
   Guess the least-significant bits $\ell(ax)$, $\ell(bx)$, $a_0 := a$, $u_0 := u$.
2. WHILE $t < n$ DO
   $t := t + 1$, $a_t := \frac{1}{2}a_{t-1} \bmod N$, $u_t := \frac{1}{2}(u_{t-1} + \ell(a_{t-1}x))$,
   $m := \min\{2^t, 2n\}\varepsilon^{-2}$.
   $A_m := \{i \mid |1 + 2i| \leq m\}$, $w_{t,i} := \lfloor u_t + i\,u_{t-1} + v \rfloor$ for all $i \in A_m$.
   Majority decision
   $$z := \#\left\{ i \in A_m \;\middle|\; \begin{array}{l} O_1(E_N((a_t + ia_{t-1} + b)x)) = \\ i\ell(a_{t-1}x) + \ell(bx) - w_{t,i}N \bmod 2 \end{array} \right\}$$
   $\ell(a_t x) := [\,0$ if $z \geq \frac{m}{2}$ and 1 otherwise $]$ END while
3. OUTPUT $x := a_n^{-1}\lfloor u_n N + \frac{1}{2} \rfloor \bmod N$

Correctness. If $\ell(a_t x)$ is always correctly determined the rational approximation $u_t N$ to $[a_t x]_N$ tightens from stage $t-1$ to stage t by a factor $\frac{1}{2}$. As $a_t = \frac{1}{2}a_{t-1} \bmod N$ we have $[a_t x]_N = \frac{1}{2}[a_{t-1}x]_N$ for even $[a_{t-1}x]_N$, $[a_t x]_N = \frac{1}{2}([a_{t-1}x]_N + N)$ for odd $[a_{t-1}x]_N$. Hence

$$[a_t x]_N - u_t N = [a_t x]_N - \tfrac{1}{2}(u_{t-1} + \ell(a_{t-1}x))N = \tfrac{1}{2}([a_{t-1}x]_N - u_{t-1}N). \quad (1)$$

Probability of success. We call $w_{t,i}$ *correct* if $0 \leq [a_t x]_N + i[a_{t-1}x]_N + [bx]_N - w_{t,i}N < N$. Correct $w_{t,i}$ satisfy the equation (as N is odd we have $-w_{t,i}N = w_{t,i} \bmod 2$) : $\ell((a_t + ia_{t-1} + b)x) = \ell(a_t x) + i\,\ell(a_{t-1}x) + \ell(bx) + w_{t,i} \bmod 2$. In the majority decision we replace in this equation $\ell((a_t + ia_{t-1} + b)x)$ by $O_1(E_N((a_t + ia_{t-1} + b)x))$, and we determine $\ell(a_t x)$ so that the equation holds for the majority of the $i \in A_m$. The algorithm succeeds if step 1 guesses correctly and if the majority decisions for $\ell(a_t x)$ are all correct. In this case we have $|[a_n x]_N - u_n N| \leq \frac{\varepsilon N}{4 \cdot 2^n} < \frac{1}{2}$ and thus $a_n x = \lfloor u_n N + \frac{1}{2} \rfloor \bmod N$ and the output is correct. All probabilities refer to the random pair $(a, b) \in_R (\mathbb{Z}_N^*)^2$ and to the coin tosses of the oracle. We use the conditional probability for the case that we are in the right alternative, where step 1 guesses correctly and the bits $\ell(a_t x)$ of previous stages have been correctly determined.

Error probability of $w_{t,i}$. Let us denote $w'_{t,i} = u_t + iu_{t-1} + v$ so that $w_{t,i} = \lfloor w'_{t,i} \rfloor$. In the right alternative we have by iteration of equation (1) $[a_j x]_N - u_j N = 2^{-j}([ax]_N - uN)$ for all $j \leq t$. Therefore and since $2^{-t}\varepsilon^2|1+2i| \leq 1$ for $i \in A_m$ we have

$$|[a_t x]_N + i[a_{t-1}x]_N + [bx]_N - w'_{t,i}N| \leq \tfrac{\varepsilon}{8}(2^{-t}\varepsilon^2|1+2i|+1)N \leq \tfrac{\varepsilon}{4}N.$$

Hence $w_{t,i}$ is correct except that there exists an integer between $w'_{t,i}N$ and $[a_t x]_N + i[a_{t-1}x]_N + [bx]_N$. Therefore $w_{t,i}$ errs with probability at most $\frac{\varepsilon}{4}$. By using the m integers $i \in A_m$ instead of $i = 1, ..., m$ we save a factor 2 in $|i|$ and in the error probability of $w_{t,i}$.

Error probability of the majority decision. The multipliers $(\frac{1}{2} + i)a + b$ are pairwise independent for $|i| < \frac{1}{2}\min(p, q)$ since the matrix of the $\mathbb{Z}_N$-linear transformation $\begin{bmatrix} 1, & \frac{1}{2}+i \\ 1, & \frac{1}{2}+j \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$ has determinant $j - i \neq 0 \bmod N$ and (a, b) is random in $(\mathbb{Z}_N^*)^2$. A similar argument shows that the errors of the $w_{t,i}$ for $i \in A_m$ are pairwise independent if we are in the right alternative. The i-th measurement is *correct* iff

$$O_1(E_N((a_t + ia_{t-1} + b)x)) = \ell(a_t x) + i\,\ell(a_{t-1}x) + \ell(bx) + w_{t,i} \bmod 2.$$

This is the case if the oracle guesses correctly and $w_{t,i}$ is correct. The error of the i-th measurement can be dominated by 0, 1-valued random variables X_i with $\mathrm{E}[X_i] = \mathrm{E}[\,O_1(E_N((a_t + ia_{t-1} + b)x)) \neq \ell((a_t + ia_{t-1} + b)x)\,] + \mathrm{E}[\,w_{t,i}\ \mathrm{errs}\,]$ so that the X_i are pairwise independent for $i \in A_m$. Hence $\mathrm{E}[X_i] \leq \frac{1}{2} - \frac{3}{4}\varepsilon$, $\mathrm{Var}[X_i] \leq \frac{1}{4}$.

A majority decision is correct iff the majority of the m measurements is correct. A majority decision errs only if $\frac{1}{m}\sum_i X_i - \mu \geq \frac{3}{4}\varepsilon$, where $\mu := \frac{1}{m}\sum_i \mathrm{E}[X_i]$. We apply Chebyshev's inequality to the m pairwise independent error variables X_i with $i \in A_m$.

Chebyshev's inequality.

$$\Pr[\,|\tfrac{1}{m}\textstyle\sum_i X_i - \mu| \geq \tfrac{3}{4}\varepsilon\,] \leq \textstyle\sum_i \mathrm{Var}[X_i](m\tfrac{3}{4}\varepsilon)^{-2} \leq \tfrac{4}{9m\varepsilon^2}.$$

By $m = \min\{2^t, 2n\}\varepsilon^{-2}$ the majority decisions for $\ell(a_t x)$ errs with probability $\frac{4}{2^t 9}$ for $t \leq 1 + \lg n$ and with probability $\frac{2}{9n}$ for $t \geq 1 + \lg n$. The majority decision for $t = 1, ..., n$ have error probability $\sum_{t \geq 1} \frac{4}{2^t 9} + (2n - \lg n)/(9n) \leq \frac{4}{9} + \frac{2}{9} = \frac{2}{3}$.

Running time. We give an upper bound for the expected number of steps required to compute x when given $E_N(x)$ and N. We separately count the steps of the oracle calls and the other steps which form the *additional overhead.*

The oracle is queried about $E_N((a_t + ia_{t-1} + b)x)$ for $t = 1, ..., n$ for the $i \in A_m$. The oracle calls depend on a, b but not on $u, v, \ell(ax), \ell(bx)$. So we keep a, b fixed while we try all possibilities for $u, ..., \ell(bx)$. As the algorithm has success rate $\frac{1}{3}$ and calls the oracle at most $m \leq 2n\varepsilon^{-2}$ times per stage, there are in total at most $3 \cdot 2\, n^2\varepsilon^{-2}T$ oracle calls.

Each majority decision contributes to the additional overhead at most $2n\varepsilon^{-2}$ steps that are performed with all oracle replies given. The algorithm does not need the exact rational $u_t + v$ and merely computes $w_{t,i} = \lfloor u_t + v + iu_{t-1} \rfloor$ using

$\lg(n\varepsilon^{-1}) + O(1)$ precision bits from $u_t + v$ and iu_{t-1}. We see that the additional overhead is at most the product of the following factors

1. # of quadruples $(u, v, \ell(ax), \ell(bx))$	$4^2\varepsilon^{-4}\,2^2$
2. # of stages	n
3. # of steps per majority decision	$2n\varepsilon^{-2}$
4. the inverse of the success rate	3

Hence the additional overhead is at most $3 \cdot 2^7 n^2 \varepsilon^{-6}$, and thus the expected time for the inversion of E_N is $3n^2\varepsilon^{-2}(2T + 2^7\varepsilon^{-4})$.

Using an oracle for the j-th least-significant message bit. The j-th least-significant message bit $\ell_j(x)$ is called *secure* if E_N can be inverted in polynomial time via an oracle O_j that predicts $\ell_j(x)$ when given $E_N(x)$. Let oracle O_j predict $\ell_j(x)$ with advantage ε in expected time T. With the oracle O_j the RSA-inversion proceeds in a similar way as for $j = 1$. It guesses initially $L_j(ax), L_j(bx) \in [0, 2^j)$, the integers that consist of the j least-significant bits of $[ax]_N, [bx]_N$. A main point is that the majority decision for $\ell_j(a_t x)$ takes into account carry overs from the $j-1$ least-significant bits. The equation

$$L_{j-1}((a_t + i\,a_{t-1} + b)x) + 2^{j-1}\ell_j((a_t + i\,a_{t-1} + b)x) \;=\; L_{j-1}(a_t x) +$$

$$iL_{j-1}(a_{t-1}x) + L_{j-1}(bx) + 2^{j-1}(\,\ell_j(a_t x) + i\ell_j(a_{t-1}x) + \ell_j(bx)\,) - w_{t,i}N \bmod 2^j$$

holds for correct $w_{t,i}$. In order to predict $\ell_j(a_t x)$ we replace in this equation $\ell_j((a_t + i\,a_{t-1} + b)x)$ by $O_j(E_N((a_t + i\,a_{t-1} + b)x))$ and we recover $L_{j-1}((a_t + i\,a_{t-1} + b)x)$, $L_{j-1}(a_t x)$ and $L_{j-1}(a_{t-1}x)$ recursively from the initial values $L_j(ax), L_j(bx)$, the approximate locations uN, vN and N. We choose $\ell_j(a_t x)$ so that the equation holds for the majority of $i \in A_m$.

The time of the inversion algorithm does not change from the case $j = 1$ to arbitrary j, except that the factor under 1. increases to $2^{2j}4^2\varepsilon^{-4}$ as we have to guess $L_j(ax), L_j(bx) \in [0, 2^j)$. Now the time bound for RSA-inversion via O_j is $O(n^2\varepsilon^{-2}(T + 2^{2j}\varepsilon^{-4}))$ while it is $O(2^{4j}n^3\varepsilon^{-8}T)$ for the ACGS-algorithm. There is a double advantage in the new time bound. The factor 2^{4j} decreases to 2^{2j} and it only affects the additional overhead. The additional overhead can be reduced by the method in section 4 to $O(n^2\varepsilon^{-4}\lg(n\varepsilon^{-1}))$.

3 From pairwise to mutually independent votes.

We introduce the *subsample majority decision*, a trick that reduces the number of oracle calls for RSA-inversion by a factor $\lg n/n$. Suppose we have m pairwise independent 0,1-valued random variables (votes) V_i for $i \in A_m$ that have advantage ε in predicting the target bit $\ell(a_t x)$. The error probability of a majority decision is $\frac{1}{m\varepsilon^2}$, so we need a large m to make this error small. To reduce the computational costs of the large sample we only use a small random subsample consisting of $m' \ll m$ votes that are selected uniformly at random. Now the votes of the subsample are mutually independent, even though the original votes are merely pairwise independent, and their advantage ε' is close to ε. While the subsample induces only a small additional error probability $\exp(-2m'\varepsilon'^2)$ the time

for the subsample majority decision is only m'. The large sample only appears in the mental error analysis, it does not enter into the computation. We can even fix a random subset $A'_{m'} \subset A_m$ for all SMAJ-calls, where $A_m := \{i \mid |1+2i| \leq m\}$ as in section 2. Theorem 3 in section 4 uses such a fixed subset $A_{m'}$.

***Subsample Majority Decision* (SMAJ).** Pick $(\nu(1), ..., \nu(m')) \in_R (A_m)^{m'}$ and guess that $\ell(a_t x)$ is $[\sum_{i=1}^{m'} V_{\nu(i)} \geq \frac{m'}{2}]$.

As in section 2 let X_i be the error of the vote V_i so that $\mathrm{E}[X_i] \leq \frac{1}{2} - \frac{3}{4}\varepsilon$. We denote $\mu = \frac{1}{m}\sum_{i\in A_m} \mathrm{E}[X_i]$ and $\mu' = \frac{1}{m}\sum_{i\in A_m} X_i$. Consider the case that $|\mu' - \mu| < \frac{1}{4}\varepsilon$ which by Chebyshev's inequality holds except with probability $\frac{\mathrm{Max}_i \mathrm{Var}[X_i]}{m\,(\varepsilon/4)^2} \leq \frac{4}{m\,\varepsilon^2}$. The SMAJ-rule errs in this case only if $\frac{1}{m'}\sum_{i=1}^{m'} X_{\nu(i)} \geq \mu' + \frac{1}{2}\varepsilon$. For fixed values X_i with $i \in A_m$ the variables $X_{\nu(1)}, \ldots, X_{\nu(m')}$ are identically distributed and *mutually independent* with mean value μ'. So we use

Bernstein's law of large numbers. For random $(\nu(1), ..., \nu(m')) \in_R (A_m)^{m'}$:

$$\Pr[\tfrac{1}{m'}\textstyle\sum_{i=1}^{m'} X_{\nu(i)} \geq \mu' + \frac{1}{2}\varepsilon] < \exp(-2(\tfrac{1}{2}m'\varepsilon)^2).$$

Proposition 1. *If the errors X_i of the votes are pairwise independent and $\mathrm{E}[X_i] \leq \frac{1}{2} - \frac{3}{4}\varepsilon$ then* SMAJ *errs with probability at most $\frac{4}{m\,\varepsilon^2} + \exp(-\frac{1}{2}m'\varepsilon^2)$.*

Proof. If $\frac{1}{m'}\sum_{i=1}^{m'} X_{\nu(i)} \geq \frac{1}{2}$ we either have $|\mu - \mu'| \geq \frac{1}{4}\varepsilon$ or $\frac{1}{m'}\sum_{i=1}^{m'} X_{\nu(i)} \geq \mu' + \frac{1}{2}\varepsilon$. The first event has probability $\leq \frac{4}{m\,\varepsilon^2}$ and the second $\leq \exp(-2m'(\frac{1}{2}\varepsilon)^2)$.

RSA-inversion using the SMAJ-rule. Let us modify the stages $t \geq 4 + \lg n$ of RSA-inversion so that at these stages the SMAJ-rule is used with $m = 2^4\varepsilon^{-2}n$ and the multipliers $a_t + ia_{t-1} + b$ with $i \in A_m$ — at stages $t \leq 3 + \lg n$ the set A_m is too small for SMAJ. We apply Proposition 1 with this m and $m' = 2\varepsilon^{-2}\lg n$. Then $\frac{1}{2}m'\varepsilon^2 = \lg n > 1.4426 \ln n$, and thus a single SMAJ-call at stage $t \geq 4 + \lg n$ fails with probability $\frac{4}{m\varepsilon^2} + n^{-1.4426} < \frac{1}{3n}$ for $n \geq 2^9$. All SMAJ-calls together fail with probability $\frac{4}{9} + \frac{1}{3} = \frac{7}{9}$. As the number of oracle calls and the additional overhead decrease by a factor $\lg n/n$ we get

Theorem 2. *Using an oracle O_1 that, given $E_N(x)$ and N, predicts $\ell(x)$ with advantage ε in time T, the* RSA*-function E_N can be inverted in expected time $9n(\lg n)\,\varepsilon^{-2}(T + 2^6\varepsilon^{-4})$.*

A main point is that the number of oracle calls for RSA-inversion is at most $9n\varepsilon^{-2}\lg n$, whereas the ACGS-algorithm requires $(64)^3\frac{\pi^2}{3}n^3\varepsilon^{-8}$ oracle calls, where $(64)^3\frac{\pi^2}{3} \approx 2^{19.7}$. We can further reduce the factor 9 in Theorem 2 by guessing upon initiation closer approximations uN, vN — this merely increases the additional overhead. On the other hand the number of oracle calls is nearly minimal.

Oracle optimality. Goldreich [G96] observed that the number $9n\varepsilon^{-2}\lg n$ of oracle calls in Theorem 2 is minimal up to a factor $O(\lg n)$.

4 Processing all possible locations together.

We sketch a first step in reducing the additional overhead in the time for RSA-inversion. So far RSA-inversion processes all pairs of locations uN, vN separately. Together these pairs can be processed much faster. We simulate the algorithm RSA-inversion for fixed a, b and for all $u \in \frac{\varepsilon^3}{4}[0, 4\varepsilon^{-3})$, $v \in \frac{\varepsilon}{4}[0, 4\varepsilon^{-1})$ with all oracle replies $O_{i,t} := O_1(E_N((a_t + ia_{t-1} + b)x))$ given. The majority decision sets $\ell(a_t x)$ to 0 iff the equation (2) holds for the majority of the $i \in A'_{m'}$.

$$O_{i,t} = i\,\ell(a_{t-1}x) + \ell(bx) + \lfloor u_t + v + iu_{t-1} \rfloor \bmod 2. \tag{2}$$

The main work of RSA-inversion is to compute for all $\bar{u} \in \frac{\varepsilon^3}{2^6 n}[0, 2^6 n\varepsilon^{-3})$, $u_{t-1} := 2u_t \bmod 1$, all v, all t and $l = (l_1, l_2) := (\ell(a_{t-1}x), \ell(bx)) \in \{0,1\}^2$:

$$\Gamma(\bar{u}, v, l, t) := \#\{i \in A'_{m'} \mid \text{equation (2) holds with } \bar{u} = u_t, v, l, t\}.$$

This requires some technical algorithms and a tedious analysis that are contained in the full version of this paper. A main point is to separate in equation (2) the influence of $u_t + v$ — we only use a few precision bits of $u_t + v$ — and that of u_{t-1}. A key observation is that counting the i that satisfy equation (2) can easily be done simultaneously for u_{t-1} and $u_{t-1} + \frac{1}{2}$ if we separately count even and odd i. By exploiting and extending these ideas we can prove

Theorem 3. If all pairs (u, v) are processed together, the additional overhead in RSA-inversion requires at most expected time $O(n^2\varepsilon^{-4}\lg(n\varepsilon^{-1}))$.

The additional overhead in Theorem 3 can be further reduced. We can discard all pairs (u, v) for which $\Gamma(u, v, l, t)/m'$ is not in the *correct* range of numbers that differ from $\frac{1}{2} \pm \varepsilon$ by at most $\frac{3}{4}\varepsilon$, where ε is the exact advantage of O_1. Thus we can restrict the set of pairs (u, v) to a small subset of $\frac{\varepsilon^3}{4}[0, 4\varepsilon^{-3}) \times \frac{\varepsilon}{4}[0, 4\varepsilon^{-1})$.

5 Security of RSA-message bits and of the RSA-RNG.

An important question of practical interest is how to generate efficiently many pseudorandom bits that are provably good under weak complexity assumptions. Provable security for the RSA-RNG follows from Theorems 2 and 3. Under the assumption that there is no breakthrough in algorithms for inverting the whole RSA-function Theorems 2 and 3 yield provable security for RSA-message bits and for the RSA-RNG for moduli N of practical size — $n = 1\,000$ and $n = 5\,000$.

Practical security of RSA-message bits. For given $E_N(x)$ it is impossible to predict $\ell(x)$ with advantage $\frac{1}{100}$ within one MIP-year ($3.16 \cdot 10^{13}$ instructions) or else the RSA-function E_N can be inverted faster than is possible by factoring N using the fastest known algorithm. For this we choose $T := 3.16 \cdot 10^{13}$, $n := 1\,000$, $\varepsilon := \frac{1}{100}$. As the O-constant in Theorem 3 is about 2^{10}, Theorems 2 and 3 yield a time bound $3 \cdot 10^{22}$ for factoring N that is clearly smaller than

$10^{25.5} \approx L_N[\frac{1}{3}, 1.9]$, the time for the fastest known factoring algorithm, see the next paragraph.

Each of the 10 least-significant RSA-message bits is individually secure for RSA-moduli N with 1 000 bits. This is because we can — see the end of section 2 — invert E_N in time $9n \lg n\, \varepsilon^{-2} T + O(n^2 \varepsilon^{-4} \lg(n\varepsilon^{-1}))$ using an oracle O_j that predicts the j-th message bit $\ell_j(x)$.

On the other hand the ACGS-result does not give any security-guarantee for moduli N of bit length 1 000, not even against one-step attackers with $T = 1$, as $2^{19.7} 1000^3 100^8 \approx 8.5 \cdot 10^{30} \gg 10^{25.5}$.

The fastest known factoring method. The fastest known algorithm for factoring N or for breaking the RSA cryptoscheme requires at least $L_N[\frac{1}{3}, 1.9]^{1+o(1)}$ steps, where $L_N[v, c] = \exp(c \cdot (\ln N)^v (\ln \ln N)^{1-v})$. $L_N[\frac{1}{3}, 1.9]$ is the conjectured run time of the number field sieve method with Coppersmith's modification using several number fields [BLP93]. Factoring even a non-negligible fraction of random RSA-moduli N requires $L_N[\frac{1}{3}, 1.9]$ steps by this algorithm.

Practical and provably secure random bit generation. Let $N = p \cdot q$ be a random RSA-modulus with primes p, q, e an RSA-exponent and let $x_0 \in_R [0, N)$. The RSA-RNG produces from random seeds (x_0, N) the bit string $b = (b_1, \ldots, b_m)$ as

$$x_i = x_{i-1}^e \bmod N, \quad b_i = x_i \bmod 2 \quad \text{for } i = 1, \ldots, m.$$

A *statistical test* A rejects b at *tolerance level* ε if for random $a \in_R \{0,1\}^m$

$$| \Pr_b[A(b) = 1] - \Pr_a[A(a) = 1] | \ \geq \ \varepsilon.$$

A tolerance level $\frac{1}{100}$ is considered to be sufficient for practical purposes.

Theorem 4. *Let the* RSA-*RNG produce from random seeds (x_0, N) of length $2n$ an output $b = (b_1, ..., b_m)$ of length m. Every statistical test A, that rejects the output at tolerance level ε, yields an algorithm that inverts the whole RSA-function E_N in expected time $9n \lg n\, (m/\varepsilon)^2 T(A) + O(n^2 (m/\varepsilon)^4 \lg(nm/\varepsilon))$ for a non-negligible fraction of N.*

Proof. Suppose the bit string $b \in \{0,1\}^m$ is rejected by some test A in time $T(A)$ and tolerance level ε. By Yao's argument, see eg. [K97, section 3.5, Lemma P1], and since the distribution of b is shift-invariant, there is an oracle O_1, which given $E_N(x)$ and N, predicts $\ell(x)$ in time $T(A) + mn^2$ with advantage ε/m for a non-negligible fraction of N. By Theorems 2 and 3, and assuming that $T(A)$ dominates mn^2, we can invert E_N in the claimed expected time. □

Corollary 5. *The RSA-random generator produces for $n = 5\,000$ from random seeds (x_0, N) of bit length 10^4 at least $m = 10^7$ pseudorandom bits that withstand all statistical tests doable with the 1995 world computing power at tolerance level $\frac{1}{100}$, or else the whole RSA-function E_N can be inverted in less than $L_N[\frac{1}{3}, 1.9]$ steps for a non-negligible fraction of N.*

Proof. ODLYZKO rates the 1995 yearly world computing power to $3 \cdot 10^8$ MIP-years, where a MIP-year corresponds to $3.16 \cdot 10^{13}$ instructions. Then $3 \cdot 10^8$ MIP-years correspond to 10^{22} instructions. By Theorem 4 with a O-constant of 2^{10} we can invert E_N using less than 10^{48} steps while $L_N[\frac{1}{3}, 1.9] > 3.7 \cdot 10^{50}$. □

6 The x^2 mod N generator and the Rabin-function.

The x^2 mod N generator has been proved to be secure under the assumption that factoring integers is hard. Here we show that this even holds for moduli N of practical size. The x^2 mod N generator transforms a random seed (x_0, N) into a bit string $(b_1, ..., b_m)$ as $x_i := E_N(x_{i-1})$, $b_i := \ell(x_i)$ for $i = 1, ..., m$. Here E_N is the Rabin-function, N is a random *Blum integer* — a product of two primes p, q that are congruent 3 mod 4 — and x_0 is a random number in $\mathbb{Z}_N$. We distinguish three variants of this generator, the *absolute*, the *centered* and the *uncentered* RNG, according to the following variants of the Rabin-function:

- the *absolute* Rabin-function $E_N^a(x) = |x^2 \bmod N| \in (0, N/2)$,
- the *centered* Rabin-function $E_N^c(x) = x^2 \underline{\bmod} N \in (-N/2, N/2)$,
- the *uncentered* Rabin-function $E_N^u(x) = x^2 \bmod N \in [0, N)$.

The centered function E_N^c outputs $x^2 \underline{\bmod} N$, the absolute smallest residue of x^2 modulo N in $(-N/2, N/2)$ whereas E_N^u outputs the residue in $[0, N)$. Historically the uncentered RNG has been introduced as the x^2 mod N generator [BBS86]. However, the absolute and the centered RNG coincide and are more natural than the uncentered RNG. We note that

$$E_N^c(x) = \pm E_N^a(x), \quad E_N^a(x) = |E_N^u(x)|, \quad E_N^u(x) \in \{E_N^c(x), E_N^c(x) + N\},$$

where $|y| = \min(y, N - y)$ for $y \in \mathbb{Z}_N = [0, N)$. Thus E_N^c extends the output of E_N^a by one bit, the sign.

The absolute and the centered RNG coincide in the output. Let x_i^a, x_i^c, x_i^u denote the integer x_i in the i-th iteration with E_N^a, E_N^c, E_N^u and input $x_0 = x_0^a = x_0^c = x_0^u$. Using $E_N^c(x) = \pm E_N^a(x)$ we see by induction on i that $x_i^c = \pm x_i^a$ and $\ell(x_i^c) = x_i^c \bmod 2 = x_i^a \bmod 2 = \ell(x_i^a)$.

On the other hand the uncentered RNG is quite different. It outputs the xor of $\ell(x_i^c)$ and the sign-bit $[x_i^c > 0]$. The uncentered RNG is less natural. Consider the group $\mathbb{Z}_N^*(+1)$ of elements in $\mathbb{Z}_N^*$ with Jacobi symbol 1. $\mathbb{Z}_N^*(+1)$ is a subgroup of $\mathbb{Z}_N^*$ of index 2 that contains the group QR_N of quadratic residues modulo N. We see from $-1 \in \mathbb{Z}_N^*(+1) \setminus QR_N$ that E_N^a permutes the set $S_N = \mathbb{Z}_N^*(+1) \cap [1, N/2)$, E_N^c permutes the set $QR_N \cap (-N/2, N/2)$ and E_N^u permutes $QR_N \cap (0, N)$. The whole point is that $\mathbb{Z}_N^*(+1)$ can be decided in polynomial time whereas QR_N may be difficult to decide. So E_N^a permutes a nice set S_N whereas E_N^c, E_N^u permute complicated sets. It comes as no surprise that we get better security results for the absolute/centered RNG than for the uncentered one.

Oracle inversion of the absolute Rabin-function. The algorithm RSA-inversion can be directly extended from the RSA-function to the permutation E_N^a acting on $S_N = \mathbb{Z}_N^*(+1) \cap [1, N/2)$. This extension uses an oracle O_1 which given $E_N^a(x)$ and N predicts for random $x \in \mathbb{Z}_N^*(+1)$ the bit $\ell(x)$ with advantage ε. A main point is that the majority decisions must use multipliers $\bar{a} = a_t + ia_{t-1} + b$ in $\mathbb{Z}_N^*(+1)$ as we can only interpret the oracle for such inputs $E_N(\bar{a}x)$ with $x \in \mathbb{Z}_N^*(+1)$. On the average half of the multipliers $\bar{a}$ are in $\mathbb{Z}_N^*(+1)$, the usable multipliers are nearly uniformly distributed, see[P92]. For compensation the inversion algorithm guesses initially an approximate location uN for $[ax]_N$ of half the previous distance. This doubles the additional overhead, but does not affect the number of oracle calls. With these remarks Theorems 2 and 3 extend from the RSA-function to the absolute Rabin-function E_N^a, Theorem 4 and Corollary 5 extend from the RSA-RNG to the absolute/centered x^2 mod N generator. The extended results prove security if factoring integers is hard, as the problems of inverting E_N^a and of factoring N are equivalent.

Theorem 6. *The assertions of Theorems 2 and 3 hold for the absolute Rabin-function E_N^a in place of the RSA-function E_N. Theorems 4 and Corollary 5 hold for the absolute/centered x^2 mod N generator in place of the RSA-generator.*

Comparison with the muddle square method. It is interesting to compare the centered x^2 mod N generator with the randomized x^2 mod N generator proposed by GOLDREICH and LEVIN [GL89,L93]: iteratively square x_i mod N and output the scalar products $b_i = \langle x_i, z \rangle$ mod 2 for $i = 1, .., m$ with a random bit string z. Following [GL89, L93] KNUTH shows that N can be factored in expected time $O(n^2\varepsilon^{-2}m^2T(A) + n^4\varepsilon^{-2}m^3)$ for a non-negligible fraction of the N if we are given a statistical test A that rejects $(b_1, ..., b_m)$ at tolerance level ε, see [K97, section 3.5, Theorem P]. This yields a security guarantee for the *muddle square method* that is similar to the one of Corollary 5.

The problem of inverting of the (un)centered Rabin-function. Consider the permutations E_N^c, E_N^u acting on the set of quadratic residues. The problems of inverting E_N^c and E_N^u are equivalent as we can easily transform one output into the other using that $E_N^u(x) - E_N^c(x) \in \{0, N\}$. We consider the oracle inversion of E_N^c. The problem we face in the oracle inversion of E_N^c is that for given $\pm y \in \mathbb{Z}_N^*(+1)$ we do not know which of $\pm y$ is in QR_N. A solution has been found by VAZIRANI and VAZIRANI [VV84]. We can determine the quadratic character of $\pm y$ using the oracle that predicts $\ell(z)$ for the inverse image $z \in QR_N$ with $E_N^c(z) = \pm y$.

Let O_1 be an oracle which, given $E_N^c(x)$ and N, predicts the least-significant bit of $x \in QR_N$ with advantage ε, $\Pr_{x,w}[O_1(E_N^c(x)) = \ell(x)] \geq \frac{1}{2} + \varepsilon$ for $x \in_R QR_N$ and the coin tosses w of O_1. The main problem in extending the RSA-inversion to the Rabin-function is that we can only use multipliers $\bar{a} = a_t + ia_{t-1} + b$ that are in QR_N as we can only interpret oracle values $O_1(E_N^c(\bar{a}x))$ with $\bar{a}x \in QR_N$. QR_N is a subgroup of $\mathbb{Z}_N^*$ with index 4.

Let us first suppose that 2 is in QR_N and that we are given $n\varepsilon^{-2}$ multipliers in QR_N of each of the two types $(\frac{1}{2}+i)a+b$ and $\frac{1}{2}(ia+b)$. Hereafter we show how to get rid of this assumption.

Inverting the centered Rabin-function. We describe how the algorithm differs from RSA-inversion if $2 \in QR_N$.

Initially pick random $a, b \in_R \mathbb{Z}_N^*$ and produce about $n\varepsilon^{-2}$ quadratic residues of either type $(\frac{1}{2}+i)a+b$, $\frac{1}{2}(ia+b)$ — with $|1+2i| \leq 4m\varepsilon^{-2}$ — in QR_N. On the average there are $n\varepsilon^{-2}$ residues in QR_N of either type. Guess the closest approximations uN, vN to $[ax]_N, [bx]_N$ with $u \in \frac{\varepsilon^3}{24}[0, 2^4\varepsilon^{-3})$, $v \in \frac{\varepsilon}{4}[0, 4\varepsilon^{-1})$.

At stage t determine $\ell(\frac{1}{2}ax)$ by majority decision using oracle O_1 and all sample points $(\frac{1}{2}+i)a+b \in QR_N$. Given $\ell(\frac{1}{2}ax)$ we can in the same way determine $\ell(\frac{1}{2}bx)$ using the sample points $\frac{1}{2}(ia+b)$ in QR_N. Then replace a, b by $\frac{1}{2}a \bmod N$, $\frac{1}{2}b \bmod N$ and go to the next stage. The new sample points $(i+\frac{1}{2})a+b$ and $\frac{1}{2}(ia+b)$ are again in QR_N since we only divide by the quadratic residue 2.

The case that 2 is a quadratic nonresidue. In this case we determine the quadratic residues $(\frac{1}{2}+i)a+b$ and $\frac{1}{2}(ia+b)$ at stages $t=1$ and $t=2$. We use the quadratic residues of stage 1 at the odd stages and the quadratic residues of stage 2 at the even stages. This is possible since we divide the residues by a power of 4 compared to stages 1 and 2.

Determining quadratic residuosity. Suppose $\bar{a} \in \mathbb{Z}_N^*$ has Jacobi symbol 1, then we have $\bar{a} \in QR_N$ iff $\Pr_z[O_1 E_N^c(\bar{a}z) = \ell(\bar{a}z)] \geq \frac{1}{2}+\varepsilon$ for $z \in_R QR_N$. This yields an oracle that predicts quadratic residuosity with advantage ε.

The algorithm for inverting the Rabin-function requires $O(n\varepsilon^{-4}\lg(n\varepsilon^{-1})T)$ extra steps for the determination of the quadratic residues $(\frac{1}{2}+i)a+b$, $\frac{1}{2}(ia+b)$. There is an extra factor 4 induced by the density $\frac{1}{4}$ of QR_N in $\mathbb{Z}_N^*$. To compensate for the smaller density the inversion algorithm guesses initially an approximate location uN for $[ax]_N$ with $\frac{1}{4}$ times the previous distance. We reduce the additional overhead by the method of section 4. Assuming that T dominates n we get

Theorem 7. *The centered Rabin-function E_N^c can be inverted in expected time $O(n\varepsilon^{-4}\lg(n\varepsilon^{-1})T)$ with the help of an oracle that predicts $\ell(x)$ with advantage ε in time T when given N and $E_N^c(x)$.*

Conclusion. We have given a stronger security proof for RSA/Rabin bits. Our proof yields provable security for RSA-message bits, for the RSA-RNG and for the centered $x^2 \bmod N$ generator for moduli N of practical size, e.g. of bit length 1 000 and 5 000. For the first time this yields provably secure and practical RNG's under the assumption that factoring integers is hard. On the other hand there are more efficient and provably secure RNG's based on stronger complexity assumptions, e.g. [MS91], [FS96].

Acknowledgement. We gratefully acknowledge the comments of D.E. Knuth and that of an anonymous referee that led to a considerably improved presentation of the material.

References

[ACGS88] W. Alexi, B. Chor, O. Goldreich and C.P. Schnorr: RSA and Rabin Functions: certain parts are as hard as the whole. Siam J. Comp. 17 (1988), pp. 194–209.

[BCS83] M. Ben-Or, B. Chor and A. Shamir: On the Cryptographic Security of Single RSA-Bits. Proc. 15th ACM Symp. on Theory of Computation, April 1983, pp. 421–430.

[BBS86] L. Blum, M. Blum and M. Shub: A Simple Unpredictible Pseudo-Random Number Generator. Siam J. Comp. 15 (1986), pp. 364–383.

[BLP93] J.P. Buhler, H.W. Lenstra, Jr. and C. Pomerance: Factoring Integers with the Number Field Sieve. in: The Development of the number field sieve, (Ed. A.K. Lenstra, H.W. Lenstra, Jr.) Springer LNM 1554 (1993), pp. 50–94.

[BM84] M. Blum and S. Micali: How to Generate Cryptographically Strong Sequences of Pseudorandom Bits. Siam J. Comp., 13 (1984), pp. 850–864.

[FS96] J.B. Fischer and J. Stern: An Efficient Pseudo-Random Generator Provably as Secure as Syndrome Decoding. Proc. EUROCRYPT'96, Springer LNCS 1070 (1996) pp. 245–255.

[G96] O. Goldreich: personal information at the Oberwolfach workshop on Complexity Theory, November 10–16, 1996.

[GL89] O. Goldreich and L.A. Levin: Hard Core Bit for any One Way Function. Proc. of ACM Symp. on Theory of Computing (1989) pp. 25–32.

[GMT82] S. Goldwasser, S. Micali and P. Tong: Why and How to Establish a Private Code on a Public Network. Proc. 23rd IEEE Symp. on Foundations of Computer Science, Nov. 1982, pp. 134–144.

[HSS93] J. Håstad, A.W. Schrift and A. Shamir: The Discrete Logarithm Modulo a Composite Hides $O(n)$ bits. J. of Computing and Systems Science 47 (1993), pp. 376–404.

[K97] D.E. Knuth: Seminumerical Algorithms, 3rd edn. Addison-Wesley, Reading, MA (1997). Also Amendments to Volume 2. January 1997. http://www-cs-staff.Stanford.EDU/~uno/taocp.html

[L93] L.A. Levin: Randomness and Nondeterminism. J. Symbolic Logic 58 (1993), pp. 1102–1103.

[MS91] S. Micali and C.P. Schnorr: Efficient, Perfect Polynomial Random Number Generators. J. Cryptology 3 (1991), pp. 157–172.

[O95] A.M. Odlyzko: The Future of Integer Factorization. CryptoBytes, RSA Laboratories, 1 (1995), pp. 5–12.

[P92] R. Peralta: On the Distribution of Quadratic Residues and Non-residues Modulo a Prime Number. Math. Comp., 58

[R79] M.O. Rabin: Digital signatures and public key functions as intractable as factorization. TM-212, Laboratory of Computer Science, MIT, 1979.

[RSA78] R.L. Rivest. A. Shamir and L. Adleman: A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Comm. ACM, 21 (1978), pp. 120–126.

[VV84] U.V. Vazirani and V.V. Vazirani: Efficient and Secure Pseudo-Random Number Generation. In Proc. 25th Symp. on Foundations of Computing Science (1984) IEEE, pp. 458–463.

[Y82] A.C. Yao: Theory and Application of Trapdoor Functions. Proc. of IEEE Symp. on Foundations of Computer Science (1982), pp. 80–91.