J. Cryptology (2000) 13: 221–244
DOI: 10.1007/s001459910008

Journal of
CRYPTOLOGY

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
{fischlin, schnorr}@cs.uni-frankfurt.de
http://www.mi.informatik.uni-frankfurt.de

Communicated by Oded Goldreich

Received 12 July 1996 and revised 8 January 1999

Abstract. The RSA and Rabin encryption functions 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 simpler and tighter proof of the result of Alexi et al. [ACGS] 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 $\frac{1}{2} + 1/\text{poly}(n)$, 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 modules N of practical size.

Key words. RSA function, Rabin function, RSA random number generator, Perfect pseudorandom number generator.

1. Introduction

Randomness is a fundamental computational resource and the efficient generation of provably secure pseudorandom bits is a basic problem. Yao [Y] and Blum and Micali [BM] have shown that perfect *random number generators* (RNG) exist under reasonable complexity assumptions. Some perfect RNGs are based on the RSA function $E_N(x) = x^e \bmod N$ [RSA] and the Rabin function $E_N(x) = x^2 \bmod N$, where the n-bit integer N is a product of two large random primes p, q and e is relatively prime to $\varphi(N) = (p-1)(q-1)$ and $e \neq 1 \bmod \varphi(N)$. 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 $b_i := x_i \bmod 2$, $x_i := E_N(x_{i-1})$. The security of these RNGs was established in a series of works [GMT], [BCS], [ACGS], [VV]: 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 $\frac{1}{2} + 1/\text{poly}(n)$. 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 RNGs 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 reduction from bit prediction to full inversion of the RSA/Rabin function. It yields a security guarantee for modules N of practical size.

Notation. Let N be product of two large primes $p, q, 2^{n-1} < N < 2^n$. Let $\mathbf{Z_N} = \mathbf{Z}/N\mathbf{Z}$ be the ring of integers modulo N and let $\mathbf{Z_N^*}$ denote the multiplicative subgroup of invertible elements in $\mathbf{Z_N}$. We represent elements $x \in \mathbf{Z_N}$ by their least nonnegative residue in the interval $[0, N)$, i.e., $\mathbf{Z_N} = [0, N)$. We let $[ax]_N \in [0, N)$ denote the least nonnegative residue of $ax \pmod N$. We use $[ax]_N$ for arithmetic expressions over $\mathbf{Z}$ while the arithmetic for $a, x \in \mathbf{Z_N} = [0, N)$ is done modulo N. We let $\ell(z) = z \bmod 2$ denote the *least-significant bit* of $z \in \mathbf{Z_N}$. Let e be relatively prime to $\varphi(N) = (p-1)(q-1)$ and $e \neq 1 \bmod \varphi(N)$. The RSA cryptosystem enciphers a message $x \in \mathbf{Z_N}$ 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 e, 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 time for the evaluation of the function E_N. Throughout the paper we assume that ε^{-1}, n are powers of 2 and $n \geq 2^9$. We let lg denote the logarithm function with base 2. All intervals $[0, N)$, $[0, 8\varepsilon^{-1})$, etc., are over the integers. 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 $\lg(n\varepsilon^{-1}) + O(1)$ bits. We use integers of that size for counting the votes in majority decisions.

Halving approximations via binary division. Consider the problem of computing $x \in \mathbf{Z_N}$ from $E_N(x)$ and N with the help of the oracle O_1 for $\ell(\cdot)$, but without knowing the factorization of N. The new method inverts E_N by iteratively halving approximations uN of random multiples $[ax]_N$ with known multiplier a, via *binary division*, see Fig. 1.

The basic idea is that given an interval containing $[ax]_N$, an interval of half-width containing $[\frac{1}{2}ax]_N$ can be computed given the least-significant bit $\ell(ax)$ of $[ax]_N$—Fig. 1 shows these two intervals, the half-width interval is shown for the two values of $\ell(ax)$.

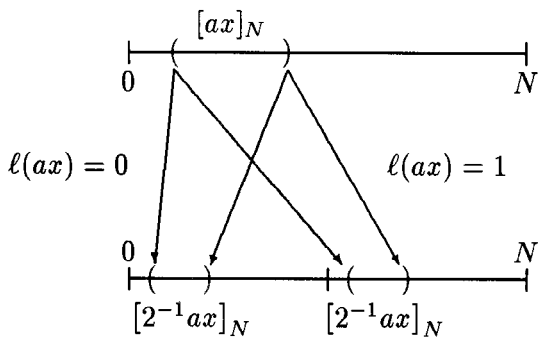

Fig. 1. Binary division.

Repeating this process for at most n iterations the interval narrows down to containing exactly one element $[2^{-n}ax]_N$. At this point, ax and therefore x can be found exactly. More formally we have $[\frac{1}{2}ax]_N = \frac{1}{2}[ax]_N$ for even $[ax]_N$ and $[\frac{1}{2}ax]_N = \frac{1}{2}([ax]_N + N)$ for odd $[ax]_N$. Given the approximation uN for $[ax]_N$ we approximate $[\frac{1}{2}ax]_N$ by $\frac{1}{2}(u + \ell(ax))N$. One binary division halves the approximation error: $[\frac{1}{2}ax]_N - \frac{1}{2}(u + \ell(ax))N = \frac{1}{2}([ax]_N - uN)$. The bits $\ell(ax)$ can be obtained for known a's with the help of O_1 as in [ACGS]. Actually we improve the relevant procedure in various ways.

Binary division has already been used by Goldwasser et al. [GMT] together with a perfect oracle. However, we face the difficulty that the oracle is faulty with a small advantage. The subsequent works [BCS] and [ACGS] have replaced binary division by the binary gcd algorithm of Brent and Kung [BK]. On the other hand the analysis of the discrete log generator [HSS] is based on binary division. We reset binary division together with halving approximations into the setup of [ACGS]. A cornerstone is majority decisions via pairwise independent sampling as introduced in [ACGS]. We introduce the *canonical multipliers* $2^{-t}a$ into this sampling method. This leads to an algorithm for RSA inversion that is much more uniform than the AGCS algorithm because $2^{-t}a$ does not depend on the dynamics of the algorithm. This higher uniformity is the basis of our optimizations.

Our results. In Section 2 we present the core of the new algorithm for inverting the RSA function E_N. It runs in expected time $O(n^2\varepsilon^{-2}\,T + n^2\varepsilon^{-6})$, where T is the time and ε is the advantage of oracle O_1. The expectation refers to the internal coin tosses of the oracle and the inversion algorithm. This improves the ACGS time bound of $O(n^3\varepsilon^{-8}\,T)$ for RSA inversion using such O_1. The new time bound differentiates the costs $O(n^2\varepsilon^{-2}\,T)$ induced by the oracle calls and all other steps $O(n^2\varepsilon^{-6})$ which we call the *additional overhead*. Subsequent optimizations in Sections 3 and 4 minimize the number of oracle calls and reduce the additional overhead. Our security result extends to the jth least-significant message bit for arbitrary j. In the extension to arbitrary j the additional overhead is proportional to 2^{2j}, whereas the number of oracle calls does not depend on j.

In Section 3 we introduce the *subsample majority rule*, a trick that improves the efficiency of majority decisions via pairwise independent votes. In various applications it is computationally easy to generate pairwise independent votes as the base of a majority decision, while mutually independence is not available or too expensive. Following [ACGS] we can easily generate with the help of the oracle 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 $1/m\varepsilon^2$ of the majority decision sufficiently small. To reduce the computational costs of the large sample we only use a small random subsample of it. The votes of the random subsample are *mutually independent*, and so a smaller subspace suffices to maintain the error probability of a majority decision. 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 $n/\lg n$ in the number of oracle calls and in the time for RSA inversion. The reduced number of oracle calls is optimal up to factor $O(\lg n)$.

In Section 4 we process all possible initial guesses for the approximate locations of $[ax]_N, [bx]_N$ simultaneously. This reduces the additional overhead in the time for RSA

inversion to $O(n^2\varepsilon^{-4}\lg(n\varepsilon^{-1}))$. Section 5 contains conclusions for the security of RSA message bits and of the RSA RNG for modules N of practical size. These conclusions are preliminary as the additional overhead can be further reduced. In Section 6 we extend the oracle algorithm from inverting the RSA function to inverting the Rabin function and we derive a security guarantee for the $x^2 \bmod N$ generator under the assumption that factoring is hard. We consider two versions of Rabin's function, improving previously known results of [ACGS] and [VV].

2. RSA Inversion by Binary Division

We introduce a novel method for inverting the RSA function with the help of an oracle O_1 that predicts the least-significant message bit with advantage ε, but without knowing the factorization of N. The core of the new method is the algorithm *RSA inversion* presented below.

High-level description of RSA inversion. We want to compute x from $E_N(x)$ using oracle O_1 that has an ε-advantage for $\ell(x)$. We invert $E_N(x)$ using the method of binary division explained in the Introduction. By that method we get from an approximate location uN of $[ax]_N$ approximate locations u_tN of $[a_tx]_N$ where the error of $u_{t+1}N$ is only half that of u_tN. Formally let $a_0 = a$, $u_0 = u$, $a_t = [2^{-t}a]_N$, and $u_t = \frac{1}{2}(u_{t-1} + \ell(a_tx))$. The main work of *stage t* is to determine the bit $\ell(a_tx)$ by majority decision using oracle O_1. For this we use a second independent multiplier b and an approximate location vN of $[bx]_N$. So upon initiation the algorithm picks two random multipliers $a, b \in \mathbf{Z_N}$, it guesses $\ell(ax)$, $\ell(bx)$ and approximate locations uN, vN of $[ax]_N$, $[bx]_N$. More precisely, it guesses the closest rationals u, v to $(1/N)[ax]_N$, $(1/N)[bx]_N$ so that $8\varepsilon^{-3}u$, $8\varepsilon^{-1}v$ are integers.

Our majority decision for $\ell(a_tx)$ further develops the procedure of [ACGS]. It predicts $\ell(a_tx)$ using $O_1(E_N(c_{t,i}x))$ for multipliers $c_{t,i} =_{\text{def}} a_t(1+2i)+b \in \mathbf{Z_N}$, where i ranges over the set $A_m =_{\text{def}} \{i : |1+2i| \le m\}$ of size m. Note that the equation $[c_{t,i}x]_N = [(1+2i)a_tx + bx]_N$ induces a uniquely defined integer $w_{t,i}$ with $|w_{t,i}| \le m$ satisfying

$$[c_{t,i}x]_N = [a_tx]_N(1+2i) + [bx]_N - w_{t,i}N. \tag{1}$$

Hence $\ell(c_{t,i}x) = \ell(a_tx)(1+2i) + \ell(bx) - w_{t,i}N = \ell(a_tx) + \ell(bx) + w_{t,i} \bmod 2$.

If $w_{t,i}$ and $\ell(bx)$ are given, then $\ell(c_{t,i}x)$ and $\ell(a_tx)$ are linearly related. Therefore a prediction of $\ell(c_{t,i}x)$ yields a prediction of $\ell(a_tx)$ with the same advantage. Importantly, the least-significant bit of $[a_tx]_N(1+2i)$ and $[a_tx]_N$ coincide because $1+2i$ is odd—an even factor $2i$ would cancel out the least-significant bit of $[a_tx]_N$ in (1), and thus the least-significant bits of $[a_tx]_N$ and $[(a_t(2i)+b)x]_N$ are uncorrelated. This shows that multipliers of the form $a_t(2i)+b$ are useless.

We compute $w_{t,i}$ from the approximations u_t and v, subject to some error due to the inaccuracy of u_t, v. We call the computed $w_{t,i}$ *correct* if it coincides with the $w_{t,i}$ defined by (1). The ith *measurement* of the majority decision for $\ell(a_tx)$ guesses $O_1(E_N(c_{t,i}x))$ for the left-hand side and computes $\ell(a_tx)+\ell(bx)+w_{t,i} \bmod 2$ for the right-hand side, and guesses $\ell(a_tx)$ accordingly: "$\ell(a_tx) = 0$" iff $O_1E_N(c_{t,i}x) = \ell(bx) + w_{t,i} \bmod 2$.

This is correct if the oracle reply $O_1 E_N(c_{t,i}x)$ and $w_{t,i}$ are both correct. The majority decision for $\ell(a_t x)$ makes measurements over the m points $c_{t,i}$, it samples over the oracle replies $O_1 E_N(c_{t,i}x)$ for $i \in A_m$. A main point will be to have the errors of the measurements pairwise independent for distinct i. This pairwise independence is induced by the pairwise independence of the multipliers $c_{t,i} = 2^{-t} a_t(1+2i) + b \in \mathbf{Z_N}$ for random $a, b \in \mathbf{Z_N}$ and fixed t.

The number of measurements. What is a good choice for the number m_t of measurements at stage t ? On the one hand we want to minimize the number of oracle calls $\sum_{t=1}^{n} m_t$ over all stages. On the other hand we need that the error probability of the guessed $\ell(a_t x)$, summed over the stages $t = 1, \ldots, n$, is bounded away from 1. That error probability depends on m_t and on the error $|u_t - (1/N)[a_t x]_N|$ of stage t.

Suppose we initially guess u so that the approximation error $|u - (1/N)[ax]_N|$ is at most δ—we justify the choice $\delta = \varepsilon^3/16$. Then we have via binary division $|u_t - (1/N)[ax]_N| \le 2^{-t}\delta$ at stage t. The numerical error induced into the above computation of $w_{t,i}$ is at most $2\,|1+2i|\,2^{-t}\delta$. Using $|1+2i| \le m_t$ this error is at most $2m_t\,2^{-t}\delta$. In order to preserve the ε-advantage of oracle O_1 we require that $2m_t\,2^{-t}\delta \le \varepsilon/4$.

Under theses premises we show below that the majority decision for $\ell(a_t x)$ errs with probability at most $4/(9\,m_t\varepsilon^2)$. So we need that $\sum_{t=1}^{n} 4/(9\,m_t\varepsilon^2) < 1$. This goal can be achieved by setting $m_t = n\varepsilon^{-2}$. However, this implies that $\delta \le \varepsilon^3/(8\,n)$ and δ^{-1} is a factor of the additional overhead which we want to be small. An alternative choice is $m_t = 2^t\varepsilon^{-2}$, $\delta = \varepsilon^3/16$ which yields $\sum_{t=1}^{n} 4/(9\,m_t\varepsilon^2) < \frac{4}{9}$. Here the problem is that we cannot have an exponential number $m_t = 2^t\varepsilon^{-2}$ of oracle calls for large t. To cap these costs we replace $2^t\varepsilon^{-2}$ for $t \ge 1 + \lg n$ by $2n\varepsilon^{-2}$. Thus our choice is $m_t := \min\{2^t, 2n\}\varepsilon^{-2}$. The deviation from $2^t\varepsilon^{-2}$ for $t \ge 1 + \lg n$ adds at most $\frac{2}{9}$ to $\sum_{t=1}^{n} 4/(9\,m_t\varepsilon^2)$, and the number of oracle calls is at most $\sum_{t=1}^{n} m_t < 2n\varepsilon^{-2}$, i.e., it is polynomial in n, ε^{-1}. Our choice of m_t, δ saves a factor n in the additional overhead compared with the choice $m_t = n\varepsilon^{-2}$, $\delta = \varepsilon^3/(8\,n)$.

Novelties. We introduce the *canonical multipliers* a_t, $c_{t,i}$ and the calculation of u_t, $w_{t,i}$ via binary division. We recursively get the approximate location $u_t N$ of $[a_t x]_N$ as $u_t := \frac{1}{2}(u_{t-1} + \ell(a_{t-1}x))$. This in turn yields $w_{t,i} := \lfloor u_t(1+2i) + v \rfloor$. We may get a faulty $w_{t,i}$ when $[a_t x]_N(1+2i) + [bx]_N$ is close to an integer multiple of N. The corresponding error of $w_{t,i}$ will be analyzed in detail.

Comparison with the ACGS method. The ACGS algorithm uses binary division within the gcd calculation, which for given a, b searches for integers k, l so that $[(ak+bl)x]_N = \pm 1$. That use of binary division gives away the advantage that binary division halves the approximation error. Our *canonical multipliers* $c_{t,i}$ provide higher uniformity than the *dynamic* multipliers $ak + bl$—that reduce $[(ak+bl)x]_N$ in the gcd algorithm. The higher uniformity of the new method opens the door to various optimizations that improve efficiency. The oracle is only queried about the canonical points $E_N(c_{t,i}x)$, this reduces the number of oracle calls to almost its information–theoretical minimum. Furthermore, the new algorithm guesses the least-significant bits and approximate locations

of two message multiples $[ax]_N, [bx]_N$, whereas the gcd method requires four such multiples.

RSA inversion

1. INPUT $E_N(x),\ N,\ \varepsilon$
   Initiation Pick random integers $a, b \in_R \mathbf{Z_N} \cong [0, N)$, guess rational integers $u \in (\varepsilon^3/8)\,[0, 8\varepsilon^{-3})$, $v \in (\varepsilon/8)\,[0, 8\varepsilon^{-1})$ satisfying

   $$\left|\frac{1}{N}[ax]_N - u\right| \le \frac{\varepsilon^3}{16}, \quad \left|\frac{1}{N}[bx]_N - v\right| \le \frac{\varepsilon}{16}, \quad \text{set } a_0 := a,\ u_0 := u.$$

   Guess $\sigma_0, \tau \in \{0, 1\}$ such that $\sigma_0 = \ell(ax)$, and $\tau = \ell(bx)$. (The above guesses are made at random and the condition refers to what we hope to be the outcome.)
2. FOR $t = 1$ TO n DO
   $a_t := \frac{1}{2}a_{t-1}$, $u_t := \frac{1}{2}(u_{t-1} + \sigma_{t-1})$, $m := \min\{2^t, 2n\}\varepsilon^{-2}$
   $c_{t,i} := a_t(1 + 2i) + b$, $w_{t,i} := \lfloor u_t(1 + 2i) + v \rfloor$ for all $i \in A_m = \{i\colon |1 + 2i| \le m\}$.
   $z := \#\{i \in A_m | O_1 E_N(c_{t,i}x) = \tau + w_{t,i} \bmod 2\}$
   Majority decision $\sigma_t := [0$ if $z \ge m/2$ and 1 otherwise$]$
3. OUTPUT $x := a_n^{-1}\lfloor u_n N + \frac{1}{2}\rfloor \bmod N$

In the following analysis we use the conditional probability that step 1 guesses correctly. Furthermore, when analyzing stage t, we assume that $\sigma_i = \ell(a_i x)$ for all $i < t$. We refer to that condition as the *right alternative*. If we are in the right alternative u, v, and u_t are uniquely determined by a, b—we eliminate ties by requiring $[ax]_N < uN + (\varepsilon^3/16)N$ and $[bx]_N < vN + (\varepsilon/16)N$. All probabilities refer to the random pair $(a, b) \in_R (\mathbf{Z_N})^2$ and to the coin tosses of the oracle.

Halving the approximation error. In the right alternative the approximation error of $u_t N$ to $[a_t x]_N$ halves with each iteration: $[a_t x]_N = \frac{1}{2}([a_{t-1}x]_N + \ell(a_{t-1}x)N)$. Hence

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

Correctness of output. By (2) RSA inversion succeeds in the right alternative. In this case we have $|(1/N)[a_n x]_N - u_n| = 2^{-n}|(1/N)[a_0 x]_N - u_0| < \frac{1}{2}$. Thus $a_n x = \lfloor u_n N + \frac{1}{2}\rfloor \bmod N$ and the output is correct.

Correctness of $w_{t,i}$. We call $w_{t,i}$ *correct* if $[c_{t,i}x]_N = [a_t x]_N(1+2i) + [bx]_N - w_{t,i}N$. Correct $w_{t,i}$ satisfy the equation

$$\ell(c_{t,i}x) = \ell(a_t x) + \ell(bx) + w_{t,i} \bmod 2,$$

where we use that $-w_{t,i}N = w_{t,i} \bmod 2$ holds for odd N. Majority decision replaces in the latter equation $\ell(c_{t,i}x)$ by $O_1 E_N(c_{t,i}x)$ and determines $\ell(a_t x)$ so that the equation holds for the majority of the $i \in A_m$. The error probability of this majority decision is analyzed below.

Error probability of $w_{t,i}$. We show that $\Pr_{a,b}[w_{t,i} \text{ errs}] \le \varepsilon/4$, where "$w_{t,i}$ *errs*" means that it is not correct. The error probability of $w_{t,i}$ depends on the *numerical error*

$$\triangle_{t,i} =_{\text{def}} (u_t(1+2i)+v) - \frac{1}{N}([a_t x]_N(1+2i)+[bx]_N)$$

of the rational number $u_t(1+2i)+v$. While the "correct" value of $w_{t,i}$ is the integer part of $(1/N)([a_t x]_N(1+2i)+[bx]_N)$, we compute $w_{t,i}$ from $u_t(1+2i)+v$. If $w_{t,i}$ errs we must have $(1/N)|c_{t,i}x|_N \le |\triangle_{t,i}|$, where $|z|_N = \min([z]_N, N-[z]_N)$ denotes the *absolute value* of z in $\mathbf{Z}_N$. In fact, the numerical error $\triangle_{t,i}$ can only affect the mod N reduction of $[a_t x]_N(1+2i)+[bx]_N$ if that integer has distance at most $N \cdot |\triangle_{t,i}|$ to the nearest integer multiple of N.

In the right alternative, iterating (2) yields $(1/N)[a_t x]_N - u_t = 2^{-t}((1/N)[ax]_N - u)$. Using $2^{-t}\varepsilon^2|1+2i| \le 1$ for $i \in A_m$ and the triangular inequality we get

$$\begin{aligned} |\triangle_{t,i}| &= \left| u_t(1+2i) - \frac{1}{N}[a_t x]_N(1+2i) + v - \frac{1}{N}[bx]_N \right| \\ &\le \frac{\varepsilon}{16}(2^{-t}\varepsilon^2|1+2i|+1) \le \frac{\varepsilon}{8}. \end{aligned}$$

So far we have shown that incorrect $w_{t,i}$ implies that $(1/N)|c_{t,i}x|_N \le |\triangle_{t,i}| \le \varepsilon/8$. Thus, the event $\text{Err}_{t,i} =_{\text{def}} [(1/N)|c_{t,i}x|_N \le \varepsilon/8]$ covers errors of $w_{t,i}$. As $c_{t,i}$ is random in $\mathbf{Z_N}$ we get $\Pr_{a,b}[w_{t,i} \text{ errs}] \le \Pr_{a,b}[\text{Err}_{t,i}] = \varepsilon/4$.

Pairwise independence of oracle and approximation errors. The matrix of the $\mathbf{Z_N}$ linear transformation

$$\begin{bmatrix} c_{t,i} \\ c_{t,j} \end{bmatrix} = \begin{bmatrix} 1, 2^{-t}(1+2i) \\ 1, 2^{-t}(1+2j) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

has determinant $2^{-t+1}(j-i) \not\equiv 0 \bmod N$ for $|2i| < \min(p,q)$. This shows that the multipliers $c_{t,i}$'s are pairwise independent for random (a,b). The oracle errors as well as the events $\text{Err}_{t,i}$ are pairwise independent since these events depend only on the multipliers $c_{t,i}$.

Remark. In [ACGS] and [FS] the approximation errors for $w_{t,i}$ have not been treated correctly. These errors are—the way they are defined—not pairwise independent for distinct measurements. This can be corrected by enlarging the approximation error to an event—like $\text{Err}_{t,i}$—that only depends on the multipliers. The correction increases the constant factors in the time bounds. We thank D. Knuth for pointing out this mistake and O. Goldreich for his help in correcting it.

Error probability of the majority decision. The ith measurement—the ith guess of $\ell(a_t x)$—is *correct* if the oracle reply and $w_{t,i}$ are both correct. We define 0,1-valued *error variables* X_i that cover the error of the ith measurement:

$$X_i = 1 \qquad \text{iff} \quad O_1 E_N(c_{t,i}x) \ne \ell(c_{t,i}x) \text{ OR } \text{Err}_{t,i}\,.$$

The X_i's are pairwise independent as the $c_{t,i}$'s are pairwise independent, and $\text{Err}_{t,i}$ depends only on $c_{t,i}x$. We have $\text{E}[X_i] \le \frac{1}{2} - \frac{3}{4}\varepsilon$ since the oracle has advantage ε

and $\mathrm{Err}_{t,i}$ has probability at most $\varepsilon/4$. Moreover, $\mathrm{Var}[X_i] < \frac{1}{4}$. A majority decision is correct iff the majority of the m measurements is correct. A majority decision errs only if $(1/m)\sum_i X_i - \mu \geq \frac{3}{4}\varepsilon$, where $\mu =_{\mathrm{def}} (1/m)\sum_i \mathrm{E}[X_i] \leq \frac{1}{2} - \frac{3}{4}\varepsilon$. *Chebyshev's inequality* for the the m random variables X_i with $i \in A_m$ shows that

$$\Pr\left[\left|\frac{1}{m}\sum_i X_i - \mathrm{E}[X]\right| \geq \frac{3}{4}\varepsilon\right] \leq \left(\frac{3}{4}\varepsilon\right)^{-2} \mathrm{Var}\left[\sum_i X_i\right] \leq \frac{4}{9m\varepsilon^2}.$$

Here, the last inequality follows from the identity $\mathrm{Var}[\sum_i X_i] = \sum_i \mathrm{Var}[X_i]/m^2$, which holds for any m pairwise independent random variables X_i. At this point we need that the X_i are pairwise independent.

As $m = \min\{2^t, 2n\}\varepsilon^{-2}$ the majority decision for $\ell(a_t x)$ errs with probability $4/(9\cdot 2^t)$ for $t \leq 1+\lg n$ and with probability $2/9n$ for $t \geq 1+\lg n$. Thus, the probability that some majority decision is wrong is at most $\sum_{t\geq 1} 4/(9\cdot 2^t) + (n - \lg n)(2/9n) \leq \frac{4}{9} + \frac{2}{9} = \frac{2}{3}$. Therefore, if step 1 guesses correctly, RSA inversion succeeds with probability at least $\frac{1}{3}$.

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

The oracle is queried about $E_N(c_{t,i}x)$ for $t = 1, \ldots, n$ and $i \in A_m$. The oracle calls depend on a, b but not on u, v, σ_0, τ. So we keep a, b fixed while we try all relevant possibilities for u, v, σ_0, τ. As the algorithm has success rate $\frac{1}{3}$ and calls the oracle at most $m \leq 2n\varepsilon^{-2}$ times per stage, the expected number of oracle calls is at most $3\cdot 2n^2\varepsilon^{-2}$. They require $3 \cdot 2n^2\varepsilon^{-2}T$ steps.

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 and merely computes $w_{t,i} = \lfloor u_t(1+2i)+v \rfloor$ using $\lg(n\varepsilon^{-1}) + O(1)$ precision bits from u_t. We see that the additional overhead is at most the product of the following factors:

1. Number of quadruples (u, v, σ_0, τ):	$8\varepsilon^{-3} \cdot 8\varepsilon^{-1} \cdot 2 \cdot 2$.
2. Number of stages:	n.
3. Number of steps per majority decision:	$2n\varepsilon^{-2}$.
4. The inverse of the success rate:	3.

Hence the additional overhead is at most $3 \cdot 2^9 n^2 \varepsilon^{-6}$, and thus the expected time for RSA inversion is $3n^2\varepsilon^{-2}(2T + 2^9\varepsilon^{-4})$.

Recall that our time bounds count arithmetic steps using integers with $\lg(n\varepsilon^{-1}) + O(1)$ bits. Integers of bit length $\lg(n\varepsilon^{-1}) + 1$ are used for counting the $n\varepsilon^{-2}$ votes of a majority decision. We use $\lg(n\varepsilon^{-1}) + 3$ precision bits from u_t when we compute $w_{t,i}$. While the n stages of RSA inversion are done with $\lg(n\varepsilon^{-1}) + 3$ bit integers, we need for the computation of the output n precision bits of u_n. For this we store n precision bits of each u_t. These bits incur only minor costs because we simply shift them at each stage.

Using an oracle for the jth least-significant message bit. The jth 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 $\ell_j(x)$ for given $E_N(x)$. Suppose that oracle O_j predicts $\ell_j(x)$ with

advantage ε in expected time T. RSA inversion using oracle O_j for arbitrary j proceeds in a similar way as for $j = 1$. It guesses initially u, v and $L_j(ax), L_j(bx) \in [0, 2^j)$, 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 carryovers from the $j-1$ least-significant bits. By the linearity of L_j the equation $L_j(c_{t,i}x) = L_j(a_t x)(1+2i) + L_j(bx) - w_{t,i}N \bmod 2^j$ holds for correct $w_{t,i}$. This implies the equation

$$L_{j-1}(c_{t,i}x)+2^{j-1}\ell_j(c_{t,i}x) = L_{j-1}(a_t x)(1+2i)+L_j(bx)+2^{j-1}\ell_j(a_t x)-w_{t,i}N \bmod 2^j.$$

In order to predict $\ell_j(a_t x)$ we replace in the latter equation $\ell_j(c_{t,i}x)$ by $O_j E_N(c_{t,i}x)$ and we recover $L_{j-1}(c_{t,i}x)$, $L_{j-1}(a_t x)$ 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 $i \in A_m$. We can apply binary division since $\ell(a_t x)$ is always given via $L_j(a_t x)$.

Next we consider how the time of RSA inversion for the case of arbitrary j compared with the particular case $j = 1$. The first factor $8^2\varepsilon^{-4}2^2$ of the additional overhead increases to the number of quadruples $(u, v, L_j(ax), L_j(bx))$ which is at most $8^2\varepsilon^{-4}2^{2j}$. 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 number of oracle calls and the additional overhead can be further reduced by the methods in Sections 3 and 4.

Simultaneous security of RSA message bits. The m least-significant message bits $L_m(x) \in [0, 2^m)$ are by definition *simultaneously secure* if given $E_N(x)$ they are polynomial-time indistinguishable from a random number $y \in_R [0, 2^m)$. In Section 5.1 of [ACGS] it is shown that the $O(\lg n)$ least-significant RSA message bits are secure or else the RSA function can be inverted in polynomial time. Here we give a stronger proof of this result, improving the ACGS time bound of oracle RSA inversion. Now RSA inversion uses a *distinguishing oracle* D which given $E_N(x)$ distinguishes $L_m(x)$ from a random $y \in_R [0, 2^m)$ at tolerance level δ:

$$|\Pr[D(L_{j+1}(x), E_N(x)) = 1] - \Pr[D(y, E_N(x)) = 1]| \geq \delta$$

To extend the algorithm RSA inversion from using oracle O_j to using a distinguishing oracle D we follow Section 5.1 of [ACGS]. Yao has shown that every distinguishing algorithm D yields for some $j \leq m$ an oracle O_j that predicts $\ell_j(x)$ when given $L_{j-1}(x)$ and $E_N(x)$, see Section 3.5, Lemma P1 of [K]. The time bound T of O_j is essentially the time bound of D and its advantage is $\varepsilon = \delta/m$. Consider RSA inversion via oracle O_j. For the prediction of $\ell_j(c_{t,i}x)$ the oracle O_j must be given $L_{j-1}(c_{t,i}x)$. As this value is to be recovered from the initial guesses $L_j(ax), L_j(bx)$ and uN, vN, the oracle depends on the correctness of the initial guess. In particular O_j must be tried for all $8^2\varepsilon^{-4}2^{2j}$ tuples $(u, v, L_j(ax), L_j(bx))$ which increases the number of oracle calls by that factor.

We see that the RSA function can be inverted using the oracle D in expected time $O(2^{2m}n^2m^6\delta^{-6}T)$. This improves the corresponding ACGS time bound $O(2^{4m}n^3m^8\delta^{-8}T)$. As the new time bound decreases the factor 2^{4m} to 2^{2m} it doubles the number m of simultaneously secure least-significant RSA message bits compared with the ACGS result.

Another approach is to apply the XOR Lemma of Vazirani and Vazirani [VV], [G]. Let $H \subset \{1, \ldots, m\}$ be a random nonempty subset. A distinguishing algorithm D yields an oracle O_H which given $E_N(x)$ predicts $\sum_{k \in H} \ell_k(x) \bmod 2$ with advantage $\varepsilon = \delta 2^{-m}$. The time bound T of O_H is essentially that of D, see the computational XOR Proposition in [G]. So we have an oracle O_j with ε–advantage for the jth message bit $\ell_j(x)$ for $j = \max(H)$, provided we know the kth bit for all $k \in H \backslash \{j\}$. Recall that this condition holds when we apply the oracle inversion algorithm using O_j. This way RSA inversion using oracle D runs in expected time $O(2^{2m} n^2 \delta^{-2}(T + 2^{4m}\delta^{-4}))$. Thus, the number of oracle calls reduces from $2^{2m} n^2 m^6 \delta^{-6}$ to $2^{2m} n^2 m^2 \delta^{-2}$ at the expense of an increased additional overhead of $O(2^{6m} n^2 \delta^{-6})$.

3. Subsample Majority Decision

According to the ACGS method we have generated a large sample of pairwise independent points $[a_t(1 + 2i) + b)x]_N$ from two random points $[a_t x]_N, [bx]_N$. By that method we get via oracle O_1 a large number of pairwise independent guesses for the unknown bit $\ell(a_t x)$ given sufficiently close approximations of $[a_t x]_N, [bx]_N$. Next, we reduce the computational costs of the ACGS method maintaining the structure of the pairwise independent sample of ACGS:

We introduce the *subsample majority decision*, a trick that reduces the number of oracle calls for RSA inversion by a factor $(1/n) \lg n$. Consider the error variables X_i that cover the error of the ith measurement in a majority decision for $\ell(a_t x)$. The error probability of a majority decision is $4/(9m\varepsilon^2)$, so we need a large sample size m to make this error small. To reduce the computational costs of the large sample we only use a small random subsample $A'_{m'}$ of $m' \ll m$ randomly selected $i \in A_m$. The randomly selected $X_{i(k)}$ are *mutually independent*, even though the original X_i are merely pairwise independent. While the subsample induces only a small additional error probability, the time for the subsample majority decision is reduced from m to m'. The large sample merely appears in the mental error analysis, it does not enter into the computation. We can even fix a random multiset $A'_{m'} \subset \{i : |1 + 2i| \leq m\}$ for all SMAJ calls, where a *multiset* is a set with multiplicities. Theorem 3 uses such a fixed multiset $A'_{m'}$.

Subsample Majority Decision (SMAJ). Pick a random $(i(1), \ldots, i(m')) \in_R (A_m)^{m'}$ and let $A'_{m'} = \{i(1), \ldots, i(m')\}$ *with multiplicities*. Decide that "$\ell(a_t x) = 0$" iff $O_1 E_N(c_{t,i(k)} x) = \ell(bx) + w_{t,i(k)} \bmod 2$ holds for at least half of the $k = 1, \ldots, m'$.

We randomly select $i(1), \ldots, i(m')$ *with repetition* from A_m so that the $X_{i(k)}$ are mutually independent. Consider the error variables X_i of Section 2 with $\mathrm{E}[X_i] \leq \frac{1}{2} - \frac{3}{4}\varepsilon$ and $\mathrm{Var}[X_i] < \frac{1}{4}$ and let $X =_{\mathrm{def}} (1/m) \sum_{i \in A_m} X_i$. The SMAJ rule errs only if $(1/m') \sum_{k=1}^{m'} X_{i(k)} \geq \frac{1}{2}$, i.e., if the majority of measurements of the subsample err. Since $\mathrm{E}[X] \leq \frac{1}{2} - \frac{3}{4}\varepsilon$ the SMAJ rule errs only if either $X \geq \mathrm{E}[X] + \frac{1}{4}\varepsilon$ or if $(1/m') \sum_{k=1}^{m'} X_{i(k)} \geq X + \frac{1}{2}\varepsilon$. For the second event we let the values X_i with $i \in A_m$ be fixed while the randomization of $X_{i(k)}$ is over the random selection $i(k) \in_R A_m$ of $X_{i(1)}, \ldots, X_{i(m')}$.

Then the variables $X_{i(1)}, \ldots, X_{i(m')}$ are identically distributed and *mutually independent* with mean value X. We use the Hoeffding bound [H] as in Exercise 4.7 [MR]:

Hoeffding's Bound. For fixed X_i's and random $(i(1), \ldots, i(m')) \in_R (A_m)^{m'}$,

$$\Pr\left[\frac{1}{m'}\sum_{k=1}^{m'} X_{i(k)} \geq X + \tfrac{1}{2}\varepsilon\right] < \exp(-2m'(\tfrac{1}{2}\varepsilon)^2).$$

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

Proof. We have seen that the SMAJ rule errs only if either $X \geq \mathrm{E}[X] + \frac{1}{4}\varepsilon$ or $(1/m')\sum_{k=1}^{m'} X_{i(k)} \geq X + \frac{1}{2}\varepsilon$. The probability of the first event is at most $\mathrm{Max}_i\, \mathrm{Var}[X_i]/m(\varepsilon/4)^2 \leq 4/m\varepsilon^2$ by Chebyshev's inequality for the pairwise independent events X_i. Then the variables $X_{i(1)}, \ldots, X_{i(m')}$ are identically distributed and *mutually independent* with mean value X. By Hoeffding's bound the probability of the second event is at most $\exp(-\frac{1}{2}m'\varepsilon^2)$. □

RSA inversion using the SMAJ rule. We modify the stages $t \geq 4 + \lg n$ of RSA inversion as follows. Apply the SMAJ rule and Proposition 1 with $m = 2^4\varepsilon^{-2}n$ (rather than $m = 2\varepsilon^{-2}n$) and $m' = 2\varepsilon^{-2}\lg n$ and use the multipliers $c_{t,i}$ with $i \in A'_{m'} \subset A_m$. At stages $t \leq 3 + \lg n$ we use $m = 2^t\varepsilon^{-2}$ and no subspace sampling. The algorithm of Section 4 starts at stage $4 + \lg n$.

We have $\frac{1}{2}m'\varepsilon^2 = \lg n > 1.4426 \ln n$. A single SMAJ call at stage $t \geq 4 + \lg n$ fails by Proposition 1 with probability $4/m\varepsilon^2 + n^{-1.4426} < 1/3n$ for $n \geq 2^9$. Thus, all SMAJ calls of stages $t \geq 4 + \lg n$ succeed except with probability $\frac{1}{3}$. They require at most $2n \lg n\, \varepsilon^{-2}$ oracle calls. These bounds are used in Section 4.

All majority decisions at stages $t \leq 3 + \lg n$ have error probability $\frac{4}{9}$. Thus RSA inversion succeeds at least with probability $1 - \frac{4}{9} - \frac{1}{3} = \frac{2}{9}$. Neglecting the $\sum_{t \leq 3+\lg n} 2^t\varepsilon^{-2} = 16n\varepsilon^{-2}$ oracle calls of stages $t \leq 3 + \lg n$ 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* $9n(\lg n)\varepsilon^{-2}\cdot(T + 2^8\varepsilon^{-4})$.

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 $(64)^3\,(\pi^2/3)n^3\varepsilon^{-8}$ oracle calls, with $(64)^3\,(\pi^2/3) \approx 2^{19.7}$. We can further reduce in Theorem 2 the factor 9 to 2: By guessing upon initiation closer approximations uN, vN to $[ax]_N$, $[bx]_N$ we can raise the success rate $\frac{2}{9}$ closely to 1. This merely increases the additional overhead. On the other hand the inversion algorithm makes almost optimal use of the oracle calls, as argued below.

Oracle optimality. We want to invert $E_N(x)$ with the help of an oracle O_1 where $O_1E_N(x)$ predicts $\ell(x)$ with advantage ε for random x. We are not interested in the known subexponential time algorithms that invert E_N without using the oracle. We exclude these

algorithms by restricting the access to $E_N(x)$ exclusively to oracle queries $O_1E_N(ax)$ for multipliers a of the algorithms choice. We let the multipliers a be computed with unlimited computational costs. This makes the oracle replies the only possible source of information for the recovering of x. We call this computational model the *oracle access model*. It covers all known oracle algorithms for RSA inversion.

Theorem 3. *Inverting $E_N(x)$ in the oracle access model requires $(\ln 2/4)n\varepsilon^{-2}$ oracle calls.*

Theorem 3 has been suggested by Goldreich. By that theorem, the number $9n\varepsilon^{-2}\lg n$ of oracle calls in Theorem 2 is minimal up to a factor $O(\lg n)$. Informally an oracle call reveals only an ε^2-fraction of a random bit while we must recover an n-bit random string x.

We need some concepts from information theory, see, e.g., Section 1.11 of [LV]. The Shannon *entropy* $H(X) = -\sum_\alpha p_\alpha \lg p_\alpha$ measures the information content (or uncertainty) of a discrete random variable X with probabilities $p_\alpha = \Pr[X = \alpha]$. The *conditional entropy* $H(X|Y)$ of X given Y is defined the same way with the p_α's replaced by the conditional probabilities $\Pr[X = \alpha \mid Y]$. It is well known that $H(X|Y) = H(X, Y) - H(Y)$, where the probability distribution of (X, Y) is the *joint* distribution of X and Y with probabilities $p_{\alpha,\beta} = \Pr[X = \alpha, Y = \beta]$. Moreover, the information in X about Y is defined as $I(X : Y) = H(Y) - H(Y|X)$, and thus $I(X : Y) = H(X) + H(Y) - H(X, Y)$. By definition $I(X : Y)$ is the part of $H(Y)$ that is complementary to $H(Y|X)$. We clearly have $I(X : Y) = I(Y : X)$ and by that symmetry $I(X : Y)$ is called the *mutual information* in X and Y. Finally, let $H_2(p) = -p \lg p - (1 - p)\lg(1 - p)$ denote the *binary entropy function*.

Proof. Let the random variable X be uniformly distributed over $\mathbf{Z_N}$ representing random RSA messages. When queried about $E_N(cX)$, oracle O_1 replies by the bit $O_1E_N(cX)$. The entropy of that bit is at most 1. We study the mutual information

$$I(X : O_1E_N(cX)) = H(X) - H(X \mid O_1E_N(cX))$$

of X and the oracle's reply. As $O_1E_N(cX)$ has ε-advantage in predicting $\ell(cX)$, this mutual information satisfies

$$\begin{aligned} H(X) - H(X \mid O_1E_N(cX)) &\geq H(\ell(cX)) - H(\ell(cX) \mid O_1E_N(cx)) \\ &\geq H_2(\tfrac{1}{2}) - H_2(\tfrac{1}{2} + \varepsilon) = 1 - H_2(\tfrac{1}{2} + \varepsilon). \end{aligned}$$

Here we use that $I(X : O_1E_N(cX)) \geq I(\ell(cX) : O_1E_N(cX))$, the latter is the mutual information of $\ell(cX)$ and $O_1E_N(cX)$. Moreover, $H(\ell(cX)) = H_2(\frac{1}{2}) = 1$, and the conditional entropy $H(\ell(cX) \mid O_1E_N(cX))$ is $H_2(p)$ where $p = \Pr_{x,w}[O_1E_N(cX) = \ell(cX)] \geq \frac{1}{2} + \varepsilon$. As $H_2(\frac{1}{2} + p)$ is monotonously decreasing for $p > \frac{1}{2}$ we get the lower bound $1 - H_2(\frac{1}{2} + \varepsilon)$.

Moreover, if $O_1E_N(cX)$ reveals no further information on cX—information about the bits other than $\ell(cX)$—then $I(X : O_1E_N(X)) = 1 - H_2(\frac{1}{2} + \varepsilon)$. In particular there exist oracles O_1 having ε-advantage for $\ell(X)$, where $I(X : O_1E_N(X)) = 1 - H_2(\frac{1}{2} + \varepsilon)$

$\approx (4/\ln 2)\,\varepsilon^2 + O(\varepsilon^4)$. Using such oracle O_1, every RSA inversion algorithm must perform at least $(\ln 2/4)\,n\,\varepsilon^{-2}\,(1 - O(\varepsilon^2))$ oracle calls as it must recover n bits of information of X. This holds because $H(X) = n$ while each oracle call provides at most $I(X : O_1 E_N(cX)) \approx (4/\ln 2)\,\varepsilon^2 + O(\varepsilon^4)$ information about X. □

4. Processing All Approximate Locations Simultaneously

So far RSA inversion processes all pairs of locations (u, v) separately. Simultaneously these pairs can be processed much faster. We simulate for all u, v the algorithm RSA inversion using the SMAJ rule of Section 3, where $m = 2^4 n\varepsilon^{-2}$ and $m' = 2\varepsilon^{-2}\lg n$. All SMAJ calls are performed with the same random multiset $A'_{m'} \subset A_m$ of size m'.

We skip the first $3 + \lg n$ stages of the algorithm of Section 3, these stages iteratively improve the precision of the approximate locations and differ from the rest. So our simulation starts at stage $t = 4 + \lg n$ and ends at stage $t = n$. It picks random $a, b \in_R \mathbf{Z_N}$ and precomputes all oracle replies $O_{t,i} := O_1 E_N(c_{t,i} x)$ for $i \in A'_{m'}$ and $c_{t,i} = a_t(1+2i)+b$. It tries for the fixed a, b all $u \in B := (\varepsilon^3/2^7 n)\,[0, 2^7 n\varepsilon^{-3})$ and all $v \in (\varepsilon/8)\,[0, 8\varepsilon^{-1})$. The interval B is chosen so that $|(1/N)[ax]_N - u| \le \varepsilon^3/2^8 n$ holds for some $u \in B$. This is the precision required at stage $t = 4 + \lg n$ of RSA inversion, where $u_t \in (\varepsilon^3/8 \cdot 2^t)\,[0, 8 \cdot 2^t \varepsilon^{-3})$. At stages $t > 4 + \lg n$ we stick to the precision level of stage $4 + \lg n$, we round u_t to the nearest rational in B. The rounding induces no extra error of $w_{t,i}$.

Recall that the SMAJ rule sets, in the i-measurement, σ_t to 0 iff (3) holds for at least half of the $i \in A'_{m'}$:

$$O_{t,i} = \tau + \lfloor u_t(1 + 2i) + v \rfloor \bmod 2. \tag{3}$$

The main work is to compute, for all $u \in B$, $v \in (\varepsilon/8)\,[0, 8\varepsilon^{-1})$, all t, and $\tau \in \{0, 1\}$,

$$\Gamma(u, v, \tau, t) =_{\text{def}} \#\{i \in A'_{m'} \mid O_{t,i} = \tau + \lfloor u(1 + 2i) + v \rfloor \bmod 2\}.$$

Once we are given all Γ-values we easily simulate the algorithm RSA inversion for *all* $u \in B$, $v \in (\varepsilon/8)\,[0, 8\varepsilon^{-1})$ and $\sigma_0, \tau \in \{0, 1\}$ in time $O(n^2\varepsilon^{-4})$ because there are $O(n\varepsilon^{-4})$ quadruples (u, v, σ_0, τ) to start with, and RSA inversion sets "$\sigma_t = 0$" iff $\Gamma(u_{t-1}, v, \tau, t) \ge m'/2$.

So it remains to compute all Γ-values. Recall that there are $|B| \cdot 8\varepsilon^{-1} \cdot 2 \cdot (n - \lg n) < 2^{11} n^2 \varepsilon^{-4}$ such values, each depending on $m' = 2\varepsilon^{-2}\lg n$ equations. Our aim is to compute all these values in time almost linear in the number of values.

Computation of all Γ*-values.* Equation (3) can be written with $u_t = u$ as

$$O_{t,i} = \tau + \lfloor (s + \bar{s})\varepsilon/8 \rfloor \bmod 2, \tag{4}$$

where we define $s := \lfloor (2iu \bmod 1) 8\varepsilon^{-1} \rfloor \in [0, 8\varepsilon^{-1})$ and $\bar{s} := \lfloor (u + v \bmod 1) 8\varepsilon^{-1} \rfloor \in [0, 8\varepsilon^{-1})$. Obviously (3) and (4) are equivalent, except when $s + \bar{s} = -1 \bmod 8\varepsilon^{-1}$. This equivalence is due to the additivity $\lfloor (s + \bar{s})\varepsilon/8 \rfloor = \lfloor s\varepsilon/8 \rfloor + \lfloor \bar{s}\varepsilon/8 \rfloor$ which holds for all integers $s, \bar{s}$ with $s + \bar{s} \ne -1 \bmod 8\varepsilon^{-1}$. For simplicity we neglect the condition $s + \bar{s} \ne -1 \bmod 8\varepsilon^{-1}$. [1]

[1] In the case $s + \bar{s} = -1 \bmod 8\varepsilon^{-1}$ we must have $|((1 + 2i)a_t + b)x|_N \le \varepsilon/4$. If the i with $s + \bar{s} = -1 \bmod 8\varepsilon^{-1}$ are all counted incorrectly, the error probability of $w_{t,i}$ increases at most by $\varepsilon/4$.

As the values $\tau, \bar{s}$ in (4) do not depend on i we can eliminate $\tau, \bar{s}$ from the problem. We reduce the Γ-values to the following Φ_ν-values which do not depend on $\tau, \bar{s}, v$:

$$\Phi_\nu(s,u,t) := \#\left\{ i \in A'_{m'} \;\middle|\; \begin{array}{l} \lfloor (2iu \bmod 1) 8\varepsilon^{-1} \rfloor = s \bmod 8\varepsilon^{-1} \\ O_{t,i} = \nu \end{array} \right\},$$

for $\nu = 0, 1$, $s \in [0, 8\varepsilon^{-1})$, $u \in B$. By (4) we have

$$\Gamma(u,v,\tau,t) = \sum_{(\nu,s)} \Phi_\nu(s,u,t),$$

where the sum ranges over all pairs $(\nu, s) \in \{0,1\} \times [0, 8\varepsilon^{-1})$ with $\nu = \tau + \lfloor (s + \bar{s})\varepsilon/8) \rfloor \bmod 2$ and $\bar{s} := \lfloor (u + v \bmod 1) 8\varepsilon^{-1} \rfloor$.

Given all Φ_ν–values we first compute $\Phi^*_\nu(S,u,t) := \sum_{0 \le s < S} \Phi_\nu(s,u,t)$ for all $S \in [0, 8\varepsilon^{-1})$ and all ν, u. Rewriting the above sum $\sum_{(\nu,s)} \Phi_\nu(s,u,t)$ we get the equation

$$\Gamma(u,v,\tau,t) = \Phi^*_\tau(8\varepsilon^{-1} - \bar{s}, u, t) + (\Phi^*_{1-\tau}(8\varepsilon^{-1}, u, t) - \Phi^*_{1-\tau}(8\varepsilon^{-1} - \bar{s}, u, t)),$$

where we separately sum with $\nu = \tau$ over the s satisfying $0 \le s + \bar{s} < 8\varepsilon^{-1}$ and with $\nu = 1 + \tau \bmod 2$ over the s with $8\varepsilon^{-1} \le s + \bar{s} < 16\varepsilon^{-1}$. We need one addition for each of the $2^{11} n^2 \varepsilon^{-4}$ Φ^*_ν–values and two additions per Γ–value. So we get all Γ-values using $3 \cdot 2^{11} n^2 \varepsilon^{-4}$ steps. It remains to compute the Φ_ν–values.

Computation of all Φ_ν–values. 1. We must compute $2^{11} n^2 \varepsilon^{-4}$ Φ_ν–values. The ν, s in the definition of $\Phi_\nu(s,u,t)$ are determined by i, u, t and thus, for fixed u, t, the m' i's distribute into $16\varepsilon^{-1}$ *boxes* (ν, s, u, t) corresponding to the pairs (ν, s). Specifically $\Phi_\nu(s,u,t)$ is the number of i's in box (ν, s, u, t). We partition each box into $8\varepsilon^{-1}$ *subboxes* containing the i with $i = c \bmod 8\varepsilon^{-1}$ for $c \in [0, 8\varepsilon^{-1})$. Let $\Psi_\nu(c,s,u,t)$ denote the number of i's in subbox (ν, c, s, u, t). We compute the Ψ_ν-values first. The point is that many subboxes coincide for distinct s, u for the same t. For example, if we form subboxes for the even and the odd i, the same subboxes appear for u and $u + \frac{1}{2}$ because $((2\,i\,\frac{1}{2}) \bmod 1) 8\varepsilon^{-1} = 1$ in the definition of Φ_ν.

In general we write $u \in B$ uniquely in the form $u = u' + j\varepsilon/8$ with $u' \in (\varepsilon^3/2^7 n)\,[0, 2^4 n\varepsilon^{-2})$ and $j \in [0, 8\varepsilon^{-1})$. Clearly, $\Psi_\nu(c,s,u'+j\varepsilon/8,t) = \Psi_\nu(c, s - 2jc, u', t)$, where the s-coordinates are always taken mod $8\varepsilon^{-1}$, and which follows from $(2c\,(j\varepsilon/8) \bmod 1) 8\varepsilon^{-1} = 2cj \bmod 8\varepsilon^{-1}$. Thus we only need to compute the $\Psi_\nu(c,s,u',t)$ for all $u' \in B \cap [0, \varepsilon/8)$. We distribute, for fixed $u' \in B \cap [0, \varepsilon/8)$ and fixed t, the m' i's into the subboxes (ν, c, s, u, t) and we count accordingly. This requires at most $m' \cdot |B| \cdot 2 \cdot n/8\varepsilon^{-1} = 2^6 n^2 \varepsilon^{-4} \lg n$ steps. Note that most of the $O(n^2\varepsilon^{-5})$ subboxes remain empty without incurring any costs.

2. Finally we compute $\Phi_\nu(s,u,t) = \sum_c \Psi_\nu(c,s,u,t)$ for all $\nu \in \{0,1\}$, all s, u, t summing up over $c \in [0, 8\varepsilon^{-1})$. This is done by an FFT–like network, where subsums are used repeatedly. In depth e of the network we form the sum over the $\Psi_\nu(c,s,u,t)$ for the 2^e c's that coincide modulo $8\varepsilon^{-1}/2^e$, we do this for all residue classes modulo $8\varepsilon^{-1}/2^e$. Each of these sums is the sum of two subsums of the preceding layer in depth $e-1$. Each subsum in depth e repeats 2^e times as the values for u, c and $\bar{u}, \bar{c}$ coincide, $\Psi_\nu(c,s,u,t) = \Psi_\nu(\bar{c}, s, \bar{u}, t)$ if $c = \bar{c} \bmod 8\varepsilon^{-1}/2^e$ and $2^{e-1}u = 2^{e-1}\bar{u} \bmod 1$. (In this

case the values $(2iu \bmod 1)8\varepsilon^{-1} = (2cu \bmod 1)8\varepsilon^{-1}$ in the definition of Φ_v coincide for c, u and $\bar{c}, \bar{u}$.)

The FFT network has depth $\lg(8\varepsilon^{-1})$ and after elimination of the redundancies there are $2^{10}n^2\varepsilon^{-4}$ vertices per layer. We have reduced the number of vertices per layer by a factor 2 due to the identity $\Psi_v(c, s, u, t) = \Psi_v(c, s, u + \frac{1}{2}, t)$. Another reduction by a factor 2 is possible since $\Phi_0(s, u, t) + \Phi_1(s, u, t)$ does not depend on the oracle calls. The resulting network has size $2^9n^2\varepsilon^{-4}\lg(8\varepsilon^{-1})$.

Time bounds. The dominant part of the time bound for Γ is the size $2^9n^2\varepsilon^{-4}\lg(8\varepsilon^{-1})$ of the FFT network for step 2. In the right alternative each SMAJ call errs by the analysis of Section 3 with probability at most $1/3n$. We perform less than n consecutive stages starting in stage $4+\lg n$. The inversion of the RSA function succeeds with probability $\frac{2}{3}$. The elimination of the first $3+\lg n$ stages has doubled the success probability. Neglecting minor terms, E_N can be inverted in expected time

$$3n(\lg n)\varepsilon^{-2}T + 3 \cdot 2^8n^2\varepsilon^{-4}\lg(8n\varepsilon^{-1}). \tag{5}$$

Theorem 4. *Processing all pairs (u, v) simultaneously, the additional overhead of RSA inversion is at most $O(n^2\varepsilon^{-4}\lg(n\varepsilon^{-1}))$.*

The additional overhead in Theorem 4 is quadratic in n, while in Theorem 2 it is linear in n. We believe that a factor n can be saved in Theorem 4. In the right alternative the fraction of measurements that predict "$\ell(a_t x) = 0, 1$" is $\Gamma(u, v, l, t)/m'$. By Chebyshev's inequality this fraction must be close to $\frac{1}{2} \pm \varepsilon$, where ε is the exact advantage of O_1. We can discard the pairs (u, v) for which $\Gamma(u, v, l, t)/m'$ differs more than $\varepsilon/2$ from both $\frac{1}{2} \pm \varepsilon$.

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 4. Under the assumption that there is no breakthrough in algorithms for inverting the whole RSA function Theorems 2 and 4 yield provable security for RSA message bits and for the RSA RNG for modules N of practical size—$n = 1000$ and $n = 5000$, respectively.

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 [BLP]. Factoring even a nonnegligible fraction of random RSA modules N requires $L_N[\frac{1}{3}, 1.9]$ steps by this algorithm.

Practical security of RSA message bits. Consider the time bound (5) for RSA inversion with the oracle time bound $T := 3.16 \cdot 10^{13}$, $n := 1000$, $\varepsilon := \frac{1}{100}$. The time bound (5) for

RSA inversion is about 10^{22} which is clearly less than $L_N[\frac{1}{3}, 1.9] \approx 10^{25.5}$, for $N \approx 2^n$, the time of the fastest known algorithm for factoring N. This yields a security guarantee for the least-significant message bit: 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.

The contribution of the additional overhead to the time bound (5) is about $1.4 \cdot 10^{18}$ for $n = 1000$, $\varepsilon = \frac{1}{100}$. There is an interesting consequence. Each of the eight least-significant RSA message bits satisfies essentially the same security guarantee as the least-significant one because the additional overhead of oracle RSA inversion via oracle O_j is proportional to 2^{2j}, see the end of Section 2, the claim holds since $1.4 \cdot 10^{18} 2^{14} \approx 2.3 \cdot 10^{22}$.

On the other hand the ACGS result does not give any security guarantee for modules N of bit length 1000, not even against a one-step attacker with $T = 1$. The ACGS time bound for inversion is $2^{19.7} 1000^3 100^8 \approx 8.5 \cdot 10^{30} \gg 10^{25.5}$, which means that the ACGS time for RSA inversion does not beat the time for factoring N by known algorithms.

Practical and provably secure random bit generation. Let $N = p \cdot q$ be a random RSA modulus with primes p, q, let $x_0 \in_R [0, N)$, and let e be a fixed RSA exponent—e is relatively prime to $\varphi(N) = (p-1)(q-1)$ and $e \not\equiv 1 \bmod \varphi(N)$. Based on the Blum–Micali construction [BM], the RSA RNG produces from random seeds (x_0, N) the bit string $\mathbf{b} = (b_1, \ldots, b_m)$ as

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

A *distinguishing algorithm* D rejects $\mathbf{b}$ produced as above at *tolerance level* δ if, for random $\mathbf{a} \in_R \{0,1\}^m$,

$$|\Pr_{\mathbf{b}}[D(\mathbf{b}) = 1] - \Pr_{\mathbf{a}}[D(\mathbf{a}) = 1]| \geq \delta.$$

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

Theorem 5. *Let the RSA RNG produce from random seeds (x_0, N) of length $2n$ an output $\mathbf{b} = (b_1, \ldots, b_m)$ of length m. Every distinguishing algorithm D of running time T, that rejects the output at tolerance level δ, yields an algorithm that inverts the RSA function E_N in expected time $3n(\lg n)\, m^2 \delta^{-2} T + O(n^2 m^4 \delta^{-4} \lg(nm\delta^{-1}))$ for a polynomial fraction of N.*

Proof. Suppose the bit string $\mathbf{b} \in \{0,1\}^m$ is rejected by some test A in time $T(A)$ and tolerance level δ. By Yao's argument, see, e.g., Section 3.5, Lemma P1, of [K], and since the distribution of $\mathbf{b}$ is shift-invariant (E_N is a permutation), there is an oracle O_1, which given $E_N(x)$ and N, predicts $\ell(x)$ in time $T(A) + mn^2$ with advantage $\varepsilon := \delta/m$ for a nonnegligible fraction of N. By the time bound (5), and assuming that $T(A)$ dominates mn^2, we can invert E_N in the claimed expected time. □

Odlyzko [O] rates the 1995 yearly world computing power to $3 \cdot 10^8$ MIP-years, where an MIP-year corresponds to $3.16 \cdot 10^{13}$ instructions. Then $3 \cdot 10^8$ MIP-years correspond to 10^{22} instructions.

Corollary 6. *The RSA random generator produces for $n = 5000$, from random seeds (x_0, N) of bit length 10^4, at least $m = 10^7$ pseudorandom bits that withstand all statistical tests performable with at most 10^{22} steps at tolerance level $\delta = \frac{1}{100}$, or else the whole RSA function E_N can be inverted in less than $L_N[\frac{1}{3}, 1.9]$ steps for a nonnegligible fraction of N.*

Proof. We apply Theorem 4 with the O-constant $3 \cdot 2^8$ of the time bound (5). This inverts the RSA function E_N using about $8 \cdot 10^{47}$ steps while $L_N[\frac{1}{3}, 1.9] > 3.7 \cdot 10^{50}$ for $N \approx 2^{5000}$. □

6. The Rabin Function and the x^2 mod N Generator

We extend the results of the previous sections from the RSA function $E_N(x) = x^e \bmod N$ to Rabin's encryption function [R] where $e = 2$. The corresponding RNG is the $x^2 \bmod N$ generator. The least-significant bit of the Rabin function and the $x^2 \bmod N$ generator have been proved to be secure under the assumption that factoring integers is hard [ACGS], [VV]. We show that this security even holds for modules N of practical size, i.e., we improve the time bound of the oracle factoring algorithm.

Throughout the section let N be a *Blum integer*—a product of two primes p and q that are congruent to 3 mod 4. Let QR_N and $\mathbf{Z}_N^*(+1)$ be the groups of residues mod N that are squares, respectively have Jacobi symbol +1. Then -1 is a quadratic nonresidue modulo N, $-1 \in \mathbf{Z}_N^*(+1)$. We have a chain of groups $QR_N \subset \mathbf{Z}_N^*(+1) \subset \mathbf{Z}_N^*$ that increase by a factor 2 with each inclusion. For further details on the Jacobi symbol see [NZ].

We distinguish three variants of the Rabin function $x \mapsto x^2 \bmod N$, the *uncentered*, the *centered* and the *absolute* Rabin function:

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

The uncentered Rabin function E_N^{u} outputs the residue in $(0, N)$ whereas E_N^{c} outputs $x^2 \underline{\bmod} N$, the residue in the symmetrical interval $(-N/2, N/2)$. Note that $|x^2 \underline{\bmod} N| = \min([x^2]_N, [N - x^2]_N) \in [0, N/2)$ is the *absolute* value of $x^2 \in \mathbf{Z_N}$. As N is a Blum integer we have $-1 \in \mathbf{Z_N^*}(+1) \backslash QR_N$. Moreover, the set $M_N =_{\mathrm{def}} \mathbf{Z_N^*}(+1) \cap (0, N/2)$ has cardinality $|\mathbf{Z_N^*}|/4$, and is a group under the operation $a \circ b =_{\mathrm{def}} |ab|_N$. It is important that:

- The uncentered Rabin function E_N^{u} permutes the set $QR_N \cap (0, N)$.
- The centered Rabin function E_N^{c} permutes the set $QR_N \cap (-N/2, N/2)$.
- The absolute Rabin function E_N^{a} permutes the set $M_N = \mathbf{Z_N^*}(+1) \cap (0, N/2)$.

The whole point is that $\mathbf{Z_N^*}(+1)$ can be decided in polynomial time whereas QR_N is presumably difficult to decide. So E_N^{a} permutes a nice, polytime recognizable set M_N, whereas both E_N^{c}, E_N^{u} permute "complicated" sets. We note that

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

Thus E_N^{c} extends the output of E_N^{a} by one bit—the sign.

Previous oracle inversion algorithms for E_N^{a} have been proposed in [ACGS] and for E_N^{u} in [VV]. We improve the previous time bounds.

The x^2 mod N generator. The x^2 mod N generator transforms a random seed (x_0, N) into a bit string $(b_1, \ldots, b_m)$ as $x_i := E_N(x_{i-1})$, $b_i := \ell(x_i)$ for $i = 1, \ldots, m$. We distinguish three variants of this generator, the *uncentered*, the *centered*, and the *absolute* RNG, according to the three variants of the Rabin function E_N. Specifically, the *seed* x_0 is random in the set $QR_N \cap (0, N)$, $QR_N \cap (-N/2, N/2)$, respectively M_N for the uncentered, centered, respectively absolute Rabin function. Historically the uncentered RNG has been introduced as the x^2 mod N generator [BBS]. However, the absolute and the centered RNG coincide and yield better results.

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

On the other hand, the uncentered RNG is quite different and unsymmetrical. It outputs the XOR of $\ell(x_i^{\mathrm{c}})$ and the sign-bit $[x_i^{\mathrm{c}} > 0]$. It comes as no surprise that we can establish better security for the absolute—and the equivalent centered—RNG than for the uncentered one.

Oracle inversion of the absolute Rabin Function. We would like to modify the oracle algorithm for RSA inversion from the RSA function to the permutation E_N^{a} on M_N. The modified algorithm uses an oracle O_1 which given $E_N^a(x)$ and N predicts for random $x \in_R M_N$ the bit $\ell(x)$ with advantage ε. The inversion algorithm uses the canonical multipliers $c_{t,i} = a_t(1 + 2i) + b$ of Section 2.

How to interpret the oracle. The difficulty is that the queries to the oracle may be of the wrong form. Namely, if we feed the oracle with $E_N^{\mathrm{a}}(c_{t,i}x)$—where $c_{t,i}x \notin M_N$—then the oracle's answer does not correspond to $c_{t,i}x$ but rather to the square root of $(c_{t,i}x)^2$ that resides in M_N. This may happen if either $c_{t,i} \notin \mathbf{Z}_{\mathbf{N}}^*(+1)$ or $[c_{t,i}x]_N > N/2$. The case $c_{t,i} \notin \mathbf{Z}_{\mathbf{N}}^*(+1)$ is easy to detect, in this case we discard the multiplier $c_{t,i}$.

We detect the case $[c_{t,i}x]_N > N/2$ with high probability via the approximation $w_{t,i}N$ of $[c_{t,i}x]_N$. If $c_{t,i}x \in \mathbf{Z}_N^*(+1)$ and $[c_{t,i}x]_N > N/2$, then the oracle's answer corresponds to $-c_{t,i}x$ since $-c_{t,i}x \in M_N$ and $(-c_{t,i}x)^2 = (c_{t,i}x)^2$. In this case the oracle guess corresponds to

$$\ell(-c_{t,i}x) = \ell(N - [c_{t,i}]_N) = 1 + \ell(c_{t,i}x) \bmod 2.$$

So if $[c_{t,i}x]_N > N/2$ we must reverse the guess $O_1 E_N^{\mathrm{a}}(c_{t,i}x)$. Figure 2 shows "$[ax/2]_N > N/2$" as a function of $\ell(ax)$.

The distribution of multipliers with Jacobi symbol $+1$. On average half of the multipliers $c_{t,i} = a_t(1 + 2i) + b$ are in $\mathbf{Z}_N^*(+1)$, which follows from $|\mathbf{Z}_N^*(+1)| = |\mathbf{Z}_N^*|/2$. However, we have to get sufficiently close to the density $\frac{1}{2}$ for a specific subset, corresponding to the $i \in A_m$, and for *all* stages t. Otherwise we need higher accuracy for the

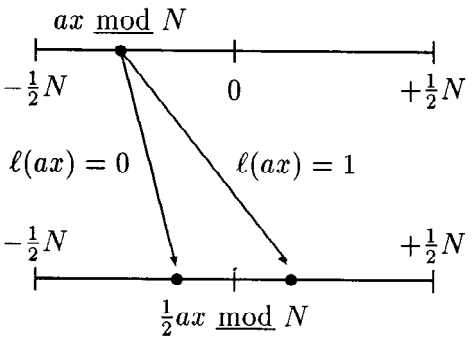

Fig. 2. Binary division via $\bmod N$.

approximate location uN of $[ax]_N$ that has to be guesssed upon initiation. This point affects the time analysis, it has been neglected in [ACGS].

Peralta [P] shows that for every prime number P, for distinct fixed integers $A_1, \ldots, A_m \in \mathbf{Z}_P^*$, and random $X \in_R \mathbf{Z}_P^*$ the distribution of the sequence of quadratic characters of $X + A_1, \ldots, X + A_m$ deviates from the uniform distribution on $\{\pm 1\}^m$ by at most $m(3+\sqrt{P})/P$. We apply this result to the prime factors p, q of N with $X = b$ and $A_i = a_t(1+2i)$ for the $i \in A_m$, and we use that $b \bmod p, b \bmod q$ are independent for random $b \in_R \mathbf{Z_N}$. Then Peralta's result shows that for random $b \in_R \mathbf{Z_N}$ the fraction of multipliers $c_{t,i} = a_t(1+2i) + b$ with $|1+2i| \leq m$ that are in $\mathbf{Z}_N^*(+1)$ is $\frac{1}{2} + O((m(3+\sqrt{\bar{p}})/\bar{p}))$, where $\bar{p} := \min(p, q)$, for every $t = 1, \ldots, n$. The difference $O((m(3+\sqrt{\bar{p}})/\bar{p}))$ of this fraction to $\frac{1}{2}$ is so small that its effect is negligible over all n stages of the inversion algorithm.

Technical details that deviate from RSA inversion. The following changes compared with RSA inversion accompany the smaller density $\approx \frac{1}{2}$ of the multipliers $c_{t,i} \in \mathbf{Z}_N^*(+1)$:

- Double m. With high probability there are about m multipliers $c_{t,i} \in \mathbf{Z}_N^*(+1)$ with $|1+2i| \leq 2m$.
- The numerical error $\triangle_{t,i}$ of $u_t(1+2i) + v$ doubles as i and m double.
- Use an initial approximation v for $(1/N)[bx]_N$ of double distance, the numerical error $\triangle_{t,i}$ doubles anyway. So initially guess the $v \in (\varepsilon/4)\,[0, 4\varepsilon^{-1})$ that is closest to $(1/N)[bx]_N$. The number of v–values halves.
- In case $w_{t,i}$ errs, then $(1/N)|c_{t,i}x|_N$ may be twice as large, as the numerical error $\triangle_{t,i}$ doubles. If $w_{t,i}$ errs, we have $(1/N)|c_{t,i}x|_N \leq \varepsilon/4$.
- The new error event $\mathrm{Err}_{t,i} = [(1/N)|c_{t,i}x|_N \leq \varepsilon/4]$ has probability at most $\varepsilon/2$ ($\varepsilon/4$ for RSA inversion). The increased error $\mathrm{Err}_{t,i}$ will be reduced by an "additional" error.

Compute $a_t, c_{t,i}, u_t, w_{t,i}$ as for RSA inversion. Decide that $2[c_{t,i}x]_N < N$ iff $\lfloor 2(u_t(1+2i)+v)\rfloor$ is even. If $2[c_{t,i}x]_N > N$, reverse the guess for $\ell(a_t x)$ by simply adding $w'_{t,i} := \lfloor 2(u_t(1+2i)+v)\rfloor \bmod 2$ in the equation of the ith measurement. Thus, guess that "$\ell(a_t x) = 0$" iff at least half of the i satisfy $O_1 E_N^{\mathrm{a}}(c_{t,i}x) = \ell(bx) + w_{t,i} + w'_{t,i} \bmod 2$.

Error analysis. There is an "additional" error if the equivalence $2[c_{t,i}x]_N < N \Leftrightarrow \lfloor 2(u_t(1+2i)+v)\rfloor = 0 \bmod 2$ does not hold. From the error analysis of $w_{t,i}$ in Section 2 we see that this event occurs only if $|2c_{t,i}x|_N \le \varepsilon/4$. The latter event is part of the error event $\mathrm{Err_{t,i}} = [|c_{t,i}x|_N \le (\varepsilon/4)N]$ and it cancels out: if the parities of $w_{t,i}$ and $w'_{t,i}$ are both incorrect these errors cancel each other. Therefore the error of the ith measurement is covered by the event $\varepsilon/8 < |c_{t,i}x|_N \le \varepsilon/4$ which has probability at most $\varepsilon/4$, the same as for RSA inversion.

Time bounds. We use half the number of v and the same number of u as for RSA inversion and thus the additional overhead for E_N^{a} inversion is half that for RSA inversion, while the number of oracle calls is the same. On the other hand, the method of [ACGS] makes E_N^{a} inversion 4^4 times slower than RSA inversion.

Furthermore, the improvements of Sections 3 and 4 carry over to E_N^{a}. This extends Theorems 2 and 3 from the RSA function to the absolute Rabin function E_N^{a} and Theorem 5 and Corollary 6 from the RSA RNG to the absolute/centered $x^2 \bmod N$ generator. We neglect the difference between the uniform distribution on $\mathbf{Z_N}$ and $\mathbf{Z}_N^*$. Note that the time to compute two Jacobi symbols—that comes with each oracle call—is negligible provided that $T \gg n \lg n$.

Theorem 7. *The assertions of Theorems* 2 *and* 3 *hold for the absolute Rabin function* E_N^{a} *in place of the RSA function* E_N. *Theorem* 4 *and Corollary* 6 *hold for the absolute/centered* $x^2 \bmod N$ *generator in place of the RSA generator.*

The strength of Theorem 7 is that it proves security provided that factoring Blum integers is hard. This follows because the problems of inverting E_N^{a} and of factoring N are equivalent [R]. Here is the standard way of factoring $N = p \cdot q$ using an inversion algorithm for E_N^{a}. Let D_N^{a} denote the permutation on M_N that is inverse to E_N^{a}. Obviously, $D_N^{\mathrm{a}} E_N^{\mathrm{a}}$ yields a 4–1 mapping from $\mathbf{Z}_\mathbf{N}^*$ to M_N. Two inputs x, y collide under this mapping if and only if $x = \pm y \bmod p$ and $x = \pm y \bmod q$. Exactly one of four colliding inputs is in M_N. Therefore the events $p \mid (z - D_N^{\mathrm{a}} E_N^{\mathrm{a}}(z))$ and $q \mid (z - D_N^{\mathrm{a}} E_N^{\mathrm{a}}(z))$ are independent for random $z \in \mathbf{Z}_\mathbf{N}^*$, and each event occurs with probability $\frac{1}{2}$. This shows that $\{\gcd(z \pm D_N^{\mathrm{a}} E_N^{\mathrm{a}}(z), N)\} = \{p, q\}$ holds with probability $\frac{1}{2}$ for random $z \in \mathbf{Z}_\mathbf{N}^*$.

In particular, N can be factored in expected time $2{\cdot}3n(\lg n)\varepsilon^{-2}T + 3{\cdot}2^8 n^2 \varepsilon^{-4} \lg(8n\varepsilon^{-1})$ using an oracle with ε–advantage in predicting $\ell(x)$ from given $E_N^{\mathrm{a}}(x)$. This improves the $O(n^3\varepsilon^{-8}T)$ time bound of [ACGS].

Comparison with the muddle square method. It is interesting to compare the centered $x^2 \bmod N$ generator with the randomized $x^2 \bmod N$ generator implied by the general result of Goldreich and Levin [GL], [L]: iteratively square $x_i \bmod N$ and output the scalar products $b_i = \langle x_i, z\rangle \bmod 2$ for $i = 1, \ldots, m$ with a fixed random bit string z. Knuth [K, Section 3.5, Theorem P] shows that N can be factored in expected time $O(n^2\delta^{-2}m^2T(A) + n^4\delta^{-2}m^3)$ for a nonnegligible fraction of the N if we are given a statistical test A that rejects $(b_1, \ldots, b_m)$ at tolerance level δ. This yields a security guarantee for the *muddle square method* that is similar to the one of Corollary 6.

The problem of inverting the uncentered 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 $E_N^{\mathrm{u}}(x) - E_N^{\mathrm{c}}(x) \in \{0, N\}$. We would like to modify the oracle algorithm for RSA inversion to the inversion of E_N^{u}. Let O_1 be an oracle which, given $E_N^{\mathrm{u}}(x)$ and N, predicts the least-significant bit of $x \in QR_N$ with advantage ε, $\Pr_{x,w}[O_1(E_N^{\mathrm{u}}(x)) = \ell(x)] \geq \frac{1}{2} + \varepsilon$ for $x \in_R QR_N$, and the coin tosses w of O_1. We face the problem that the correct interpretation of the oracle replies $O_1 E_N^u(c_{t,i}x)$ requires the quadratic residuosity of $c_{t,i}x$. For simplicity we only use multipliers $c_{t,i}$ that are quadratic residues. The subgroup QR_N of $\mathbf{Z}_N^*$ has order $|\mathbf{Z}_N^*|/4$. To compensate the reduced density of the usable multipliers the inversion algorithm guesses initially an approximate location uN for $[ax]_N$ with $\frac{1}{4}$ times the distance for RSA inversion.

Deciding quadratic residuosity. Blum et al. [BSS] have shown in Lemma 2 of [BBS] that every oracle O_1 as above—that given $E_N^{\mathrm{u}}(x)$ predicts with ε-advantage the least-significant bit of x—can be used to predict quadratic residuosity with ε-advantage:

For all $\bar{a} \in \mathbf{Z}_N^*(+1)$ and random $z \in_R QR_N$ the following equivalence holds with probability $\frac{1}{2} + \varepsilon$:

$$\bar{a} \in QR_N \quad \Longleftrightarrow \quad O_1 E_N^{\mathrm{c}}(\bar{a}z) = \ell(\bar{a}z).$$

So we predict "$\bar{a} \in QR_N$" iff $O_1 E_N^{\mathrm{c}}(\bar{a}z) = \ell(\bar{a}z)$. The resulting ε-advantage for quadratic residuosity can be easily amplified. Specifically, to predict quadratic residuosity with error probability at most $1/8m$ we pick $\overline{m} := \varepsilon^{-2}\lceil \ln m \rceil$ independent $z_1, \ldots, z_{\overline{m}} \in_R QR_N$, and we decide that "$\bar{a} \in QR_N$" iff $O_1 E_N(\bar{a}z_i) = \ell(\bar{a}z_i)$ holds for the majority of the i. By Hoeffding's bound this majority decision has error probability at most $\exp(-2\overline{m}\varepsilon^2) \leq \exp(-2\ln m) = m^{-2} \leq 1/8m$ for $m \geq 8$.

How to interpret the oracle. If we feed the oracle with $E_N^{\mathrm{u}}(c_{t,i}x)$—where $c_{t,i}x \notin QR_N$—then the oracle's answer does not correspond to $c_{t,i}x$ but rather to the square root of $(c_{t,i}x)^2$ that resides in $QR_N \cap (0, N)$. This may happen if either $c_{t,i} \in \mathbf{Z}_N^*(+1) \setminus QR_N$ or $c_{t,i}x \notin \mathbf{Z}_N^*(+1)$. The case $c_{t,i} \notin \mathbf{Z}_N^*(+1)$ is easy to detect, in this case we discard the multiplier $c_{t,i}$.

If $c_{t,i}x \in \mathbf{Z}_N^*(+1)$, then the oracle's answer corresponds to either $c_{t,i}x$ or to $-c_{t,i}x$ since $(-c_{t,i}x)^2 = (c_{t,i}x)^2$. If $c_{t,i}x \notin QR_N$ then we must reverse the guess $O_1 E_N^{\mathrm{u}}(c_{t,i}x)$.

Inverting the uncentered Rabin function. We describe how the algorithm differs from the RSA inversion. Let m be as for RSA inversion. We first assume that 2 is in QR_N.

Initially pick random $a, b \in_R \mathbf{Z_N}$ and produce all quadratic residues of either type $c_{1,i} := (a/2)(1+2i)+b$ or $c'_{1,i} := ai+b/2$ with $|1+2i| \leq 4m$. On average there are about m quadratic residues of either type. Guess the closest rationals uN, vN to $[ax]_N$, $[bx]_N$ so that $32\varepsilon^{-3}u$, $8\varepsilon^{-1}$ are integers. Also guess $\ell(ax)$, $\ell(bx)$. At stage t determine $\ell(ax/2)$ by majority decision using oracle O_1 and the multipliers $c_{1,i} = (a/2)(1+2i)+b \in QR_N$.

Details of the majority decision of $\ell(ax/2)$. Compute from u, v the integer $w_{1,i}$ which most likely satisfies

$$[c_{1,i}x]_N = [ax/2]_N(1+2i) + [bx]_N + w_{1,i} \bmod 2.$$

Decide that $\ell(ax/2) = 0$ iff the majority of the multipliers $c_{1,i}$ satisfies

$$O_1 E_N(c_{1,i}x) = \ell(ax/2) + \ell(bx) + w_{1,i} \bmod 2.$$

Given $\ell(ax/2)$ we can in the same way determine $\ell(bx/2)$ using the multipliers $a'_{1,i} = ai + (b/2) \in QR_N$ with $|1+2i| \le 4m$. Then replace a, b by $a/2, b/2 \in \mathbf{Z_N}$ and go to the next stage. The new multipliers $c_{1,i}$ and $c'_{1,i}$ are again in QR_N since we divide them by the quadratic residue 2.

The case that 2 *is a quadratic nonresidue*. In this case we separately determine the quadratic residues of both types $c_{1,i}, c'_{1,i}$ for the stages $t = 1$ and $t = 2$. We use the quadratic residues of stage 1 at the odd stages t and the quadratic residues of stage 2 at the even stages t. This is correct since we divide the residues by a power of 4 compared with stages 1 and 2.

Time bounds. The extra work in E_N^{u} inversion—versus RSA inversion—is to decide which of the $c_{t,i}$ and $c'_{t,i}$, x are in QR_N. We first discard the multipliers with Jacobi symbol -1. Then we are left with about $4m+1 = O(n\varepsilon^{-2})$ quadratic residuosity decisions. We have shown above that we can decide each of these quadratic residuosities—with error probability at most $1/8m$—using $\varepsilon^{-2}\lceil \ln n \rceil$ calls to the oracle O_1. So we get all $4m+1$ decisions using $O(n\varepsilon^{-4} \lg(n\varepsilon^{-1}))$ oracle calls. Reducing the additional overhead as in Section 4 we get the following theorem which improves the $O(n^3\varepsilon^{-11}T)$ time bound of [VV].

Theorem 8. *Using an oracle which given* $E_N^{\mathrm{c}}(x)$ *predicts* $\ell(x)$ *with advantage* ε *in time* T *the* (*un-*)*centered Rabin function* E_N^{c} *can be inverted in expected time* $O(n\varepsilon^{-4} \lg(n\varepsilon^{-1})T)$.

7. Conclusion

We give stronger security proofs for the input bits of the RSA/Rabin function given the value of the function. Assuming that there are no faster algorithms for inverts of the RSA/Rabin function than via the currently known factoring algorithms we prove security for RSA message bits, for the entire RSA RNG and for the centered $x^2 \bmod N$ generator for modules N of practical size, e.g., of bit length 1000 and 5000, respectively. For the first time this yields provably secure and practical RNG under the assumption that factoring integers is hard. Other efficient RNGs have been proved to be secure under complexity assumptions that are less widely studied, e.g., [MS] and [FS1].

Our main result shows that the asymptotic theory of perfect RNGs originating from the work of Yao [Y] and Blum and Micali [BM] has a practical impact. Note that the statements of this theory only hold for *sufficiently long* seeds of the random generator,

whereas in applications the seeds are limited by precise bounds, e.g., by bit length 1000 or 5000. Our results bridge the gap between asymptotics and praxis. To close this gap we give explicit constant factors for all time bounds. In addition we use a complexity assumption for the problem of factoring integers of bit length 1000 and 5000. Such a precise assumption is possible by the extensive work on the factoring problem done over the last years, where the results of theoretical analysis and of practical implementations of algorithms are surprisingly close, see [LL]. The whole argument set forth in [ACGS] and finalized in the present paper is not at all trivial. It is all the more surprising that this has a practical impact, a well-designed theory can be practical.

The present paper relies heavily on its precursor [ACGS]. Specifically we use the method of pairwise independent sampling for majority decision of [ACGS]. We further refine this method by the rule of subsample majority decision (SMAJ). SMAJ may be of independent interest and may have applications beyond this paper. The other main improvement over [ACGS] originates from the observation that binary division is more appropriate for RSA inversion and easier to analyze than the binary gcd method used in [BCS] and [ACGS]. In fact we have reinvented binary division which has already been used in [GMT]. Binary division bears fruits beyond the present work. In a subsequent paper [S] we introduce a novel bit representation of modulo N integers so that the RSA message bits in the new representation are *all* individually secure.

In a sense we not only improve the results of [ACGS], we present the essentially optimal reduction from RSA bit prediction to full RSA inversion—optimal in the number of calls of the prediction oracle. The reduction of the number of oracle calls is possible by the SMAJ rule. The reduced number of oracle calls is minimal up to a factor $O(\lg n)$ for all algorithms for RSA inversion in a suitable model of algorithms.

Our complexity analysis separates the number of oracle calls from the remaining steps which we call the *additional overhead*. We substantially reduce the additional overhead by simulating the algorithm RSA inversion in parallel for all values of the unknown approximation points (u, v). Our parallel simulation via an FFT network is almost linear time in the number of values of (u, v) times the number n of stages (n is the bit length of the RSA message). We believe that the latter factor n can still be removed.

Acknowledgments

We gratefully acknowledge the comments of D. E. Knuth and of an anonymous referee. We thank O. Goldreich for his sustained and insisting suggestions for improvements. All this led to a considerably improved presentation of the material.

References

[ACGS] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr: RSA and Rabin Functions: Certain Parts Are as Hard as the Whole. *Siam J. Comput*. 17 (1988), 194–209.

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

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

[BLP] 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* (A.K. Lenstra and H.W. Lenstra, Jr., eds.), pp. 50–94. LNM 1554. Springer-Verlag, Berlin, 1993.

[BK] R. P. Brent and H. T. Kung: Systolic VLSI Arrays for Linear Time GCD Computation. In *VLSI* 83, *IFIP* (F. Anceau and E.J. Aas, eds.), pp. 145–154. Elsevier, Amsterdam, 1983.

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

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

[FS] R. Fischlin and C. P. Schnorr: Stronger Security Proofs for RSA and Rabin Bits. *Proc. Eurocrypt* '97, pp. 267–279. LNCS 1233. Springer-Verlag, Berlin, 1997. This is a preliminary version of the current paper.

[G] O. Goldreich: Three XOR Lemmas—An Exposition. ECCC TR95-056, http://www.eccc.uni-trier.de/eccc/.

[GL] O. Goldreich and L. A. Levin: Hard Core Bit for Any One Way Function. *Proc. ACM Symp. on Theory of Computing*, 1989, pp. 25–32.

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

[H] W. Hoeffding: Probability in Equalities for Sums of Bounded Random Variables. *J. Amer. Statist. Assoc.* 58 (1963), 13–30.

[K] 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

[HSS] J. Håstad, A. W. Schrift, and A. Shamir: The Discrete Logarithm Modulo a Composite Hides $O(n)$ bits. *J. Comput. Systems Sci.* 47 (1993), 376–404.

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

[LL] A. K. Lenstra and H. W. Lenstra, Jr. (eds): *The Development of the Number Field Sieve*. LNM 1554. Springer-Verlag, Berlin, 1993.

[MR] R. Motwani and P. Raghavan: *Randomized Algorithms*. Cambridge University Press, Cambridge, 1995.

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

[LV] M. Li and P. Vitanyi: *An Introduction to Kolmogorov Complexity and Its Applications*, 2nd edn. Springer-Verlag, New York, 1997.

[NZ] I. Niven and H. S. Zuckermann: *An Introduction into the Theory of Numbers*. Wiley, New York, 1980.

[O] A. M. Odlyzko: The Future of Integer Factorization. *CryptoBytes*, 1 (1995), 5–12.

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

[R] M. O. Rabin: Digital Signatures and Public Key Functions as Intractable as Factorization. TM-212, Laboratory of Computer Science, MIT, 1979.

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

[S] C. P. Schnorr: Security of Arbitrary RSA and of "All" Discrete Log Bits. Preprint, University Frankfurt, 1998.

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

[Y] A. C. Yao: Theory and Application of Trapdoor Functions. *Proc.* 23*rd IEEE Symp. on Foundations of Computer Science*, 1982, pp. 80–91.