On the Provable Security of an Efficient RSA-Based Pseudorandom Generator

Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang

Centre for Advanced Computing – Algorithms and Cryptography (ACAC)
Dept. of Computing, Macquarie University, North Ryde, Australia
{rons, josef, hwang}@comp.mq.edu.au
http://www.ics.mq.edu.au/acac/

Abstract. Pseudorandom Generators (PRGs) based on the RSA inversion (one-wayness) problem have been extensively studied in the literature over the last 25 years. These generators have the attractive feature of provable pseudorandomness security assuming the hardness of the RSA inversion problem. However, despite extensive study, the most efficient provably secure RSA-based generators output asymptotically only at most $O(\log n)$ bits per multiply modulo an RSA modulus of bitlength n, and hence are too slow to be used in many practical applications.

To bring theory closer to practice, we present a simple modification to the proof of security by Fischlin and Schnorr of an RSA-based PRG, which shows that one can obtain an RSA-based PRG which outputs $\Omega(n)$ bits per multiply and has provable pseudorandomness security assuming the hardness of a well-studied variant of the RSA inversion problem, where a constant fraction of the plaintext bits are given. Our result gives a positive answer to an open question posed by Gennaro (J. of Cryptology, 2005) regarding finding a PRG beating the rate $O(\log n)$ bits per multiply at the cost of a reasonable assumption on RSA inversion.

Keywords: Pseudorandom generator, RSA, provable security, lattice attack.

1 Introduction

Background. The RSA Pseudorandom bit generator (RSA PRG) works by iterating the RSA encryption mapping $x \rightarrow x^e \bmod N$ (with public RSA modulus N of length n bits and public exponent e coprime to $\phi(N)$) on a secret random initial seed value $x_0 \in \mathbb{Z}_N$ to compute the intermediate state values $x_{i+1} = x_i^e \bmod N$ (for $i = 0, 1, 2, \ldots$) and outputting r least-significant bits of the state value x_i per iteration. The pseudorandomness of the RSA PRG (especially the case $r = 1$) was studied extensively by several researchers [19,2,30,1,14]. However, even the best security proof so far [14,28] only applies to the case when only a very small number of bits $r = O(\log n)$ is output per iteration. Consequently, even with small public exponent e, these proven RSA PRG variants only output $O(\log n)$ bits per multiply modulo N and hence are too slow for most practical applications. As far as we are aware, these are currently the most efficient RSA-based PRGs with proven pseudorandomness security.

Our Approach. Our approach to studying the provable security of efficient variants of the RSA PRG is based on two observations.

First, we observe that existing security proofs of the RSA PRG have always attempted to prove the security assuming the hardness of the classical RSA one-wayness problem (given RSA modulus N and $y = x^e \bmod N$ for random $x \in \mathbb{Z}_N$, find x). If we instead make a stronger hardness assumption, we can hope to prove the security of much more efficient and practical variants of the RSA PRG, with $r = \Omega(n)$. But we must be careful in choosing this stronger hardness assumption to ensure that it is based on substantial evidence – it must be a hard problem which has been undoubtedly studied extensively by experts. This leads to our second observation.

Our second observation is that over the last decade, beginning with the work of Coppersmith [11], the following variant of the RSA one-wayness problem has been studied explicitly:

(δ, e)-Small Solution RSA ((δ, e)-SSRSA) Problem. Given a random n-bit RSA modulus N, the coefficients of a univariate polynomial $f(z) = a_e z^e + a_{e-1} z^{e-1} + \cdots + a_0 \in \mathbb{Z}_N[z]$ of degree e (with $a_e \in \mathbb{Z}_N^*$) and $y = f(\bar{z}) \bmod N$ for a random integer $\bar{z} < N^\delta$ (with $0 < \delta < 1$), find $\bar{z}$ (note that we will only be interested in instances where f is such that $\bar{z}$ is uniquely determined by (N, f, y)).

The celebrated lattice-based attack of Coppersmith [11] shows that for small e, the (δ, e)-SSRSA problem can be solved in polynomial time (in n) whenever $\delta < 1/e$. But when $\delta > 1/e + \epsilon$ for some constant $\epsilon > 0$, the lattice attack fails, and the only known attack (beyond factoring N) is to run the lattice attack $O(N^\epsilon)$ times for each guess of the $\epsilon \cdot n$ most-significant bits of $\bar{z}$. Hence, when ϵ is made sufficiently large to make the above lattice attack slower than factoring N (namely even $\epsilon = O((\log n/n)^{2/3})$ suffices), the best known attack against $(1/e + \epsilon, e)$-SSRSA problem is to factor N. Importantly, this hardness assumption is supported by explicit evidence in the literature that the $(1/e+\epsilon, e)$-SSRSA problem has been studied by experts [12,26,10], yet these studies have not yielded an efficient algorithm for the $(1/e + \epsilon, e)$-SSRSA problem.

Our Result. We present a simple modification to the proof of security of the RSA PRG by Fischlin and Schnorr [14] which shows that assuming the hardness of a certain specific $(1/e + \epsilon, e)$-SSRSA one-wayness problem suffices to prove the pseudorandomness of the RSA PRG outputting $r = (1/2 - 1/e - \epsilon - o(1)) \cdot n$ LS bits per iteration. Our specific $(1/e + \epsilon, e)$-SSRSA one-wayness problem can be posed as RSA inversion with some known plaintext bits, namely: Given N, $y = [x^e]_N$, r LS bits of x and $w \approx n/2$ MS bits of x, for $x \in_R \mathbb{Z}_N$, find x. For small (constant) $e \geq 3$ we therefore obtain a throughput of $\Omega(n)$ output pseudorandom bits per multiply modulo the RSA modulus N, which is a significant improvement over the $O(\log n)$ bits per multiply throughput obtained using previous proof of security relative to the RSA assumption. We believe this answers in the positive an open question raised by Gennaro [15], who asked whether one can obtain a PRG which beats the rate $O(\log n)$ bits per multiply at the cost of a stronger but reasonable assumption on RSA inversion.

Organization. In Section 1.1 we discuss additional related work. Section 2 contains definitions and notations. In Section 3, we review the RSA PRG construction and its proof of security by Fischlin and Schnorr [14]. Section 4 presents our modified security proof for the RSA PRG assuming the hardness of a $(1/e+\epsilon, e)$-SSRSA problem. In Section 5, we estimate concrete parameters and associated PRG performance for given proven security level and security assumptions. In Section 6 we investigate the potential for performance improvements using a stronger hardness assumption. Section 7 concludes the paper with some open problems.

1.1 Additional Related Work

Related PRG constructions can be divided in two classes.

The first class contains PRGs based on related hardness assumptions. The well known Blum-Blum-Shub (BBS) generator [6] has the same structure as the RSA PRG, but uses the Rabin squaring iteration function instead. Similar security results as for the RSA PRG are known for this generator [14], but we need a less known assumption to prove the security of efficient variants of this generator (see Section 6). The factoring-based construction by Goldreich and Rosen [17] has a throughput of $O(1)$ bits per multiply modulo an n bit modulus. The Micali-Schnorr RSA-based constructions [24] have a throughput of $\Omega(n)$ bits per multiply, but their pseudorandomness security is only proven assuming the *pseudorandomness* of the RSA function with small inputs whereas for our construction we can prove pseudorandomness assuming only a much weaker assumption of *one-wayness* of RSA with small inputs. The PRG of Boneh et al [9] also achieves a throughput of $\Omega(n)$ bits per multiply (and in fact may use a smaller *prime* modulus), but its provable pseudorandomness security also relies on a pseudorandomness assumption rather than a one-wayness assumption.

The second class of PRGs achieve provable pseudorandomness based on different one-wayness assumptions. The construction by Impagliazzo and Naor [21] is based on the hardness of the Subset Sum problem. Although this construction is potentially very efficient, its concrete security against lattice-based subset sum attacks is difficult to estimate and requires carefully chosen large parameters with a small number of bits output per function evaluation. Very recently, a more practical 'QUAD' construction by Berbain et al [3] was proposed, using similar ideas to [21] in its security proof, but based on the hardness of solving a random system of multivariate quadratic equations over a finite field ('MQ' problem). We compare the practical performance of our construction with QUAD in Section 5. The fastest PRG based on the hardness of a variant of the Discrete-Log one-wayness problem is due to Gennaro [15] (improving on earlier work by Patel and Sundaram [27]), but its throughput is at most $O((\frac{n}{\log n})^{2/3}) = o(n)$ bits per multiply, compared to $\Omega(n)$ bits per multiply for our construction with same modulus length n and conjectured security level.

Finally, we also wish to mention the lattice-based attacks of Blackburn et al [5,4] on a class of PRGs having the same iterative structure as our RSA PRG. These attacks show that the RSA PRG is insecure when the number of bits output per iteration r is larger than about $\frac{2}{3}n$ [5] for $e = 2$, and about

$(1-\frac{1}{e(e+1)/2+2})n$ [4] in the general case (these results are obtained for r MS bits output per iteration and prime moduli, but we believe that with appropriate modifications they hold also for r LS bits and RSA moduli). We remark that the general case attacks in [4] use low-dimension lattices and are rigorously proven. A heuristic extension of these attacks to high dimension lattices using the Coppersmith method [11] suggests that the RSA PRG is insecure asymptotically with $r \geq (1-\frac{1}{e+1})n$ (we omit details of these calculations here). These lower bounds for insecure values of r are greater by a factor of about 2 than the upper bounds on r for which our security proof applies. Closing this remaining gap between best attack and best proof is an interesting open problem.

2 Preliminaries

Notation. For integers x and N, we use $[x]_N$ to denote the remainder x mod N. We use $L_r(x) = [x]_{2^r}$ to denote the r least significant bits of the binary representation of x. Similarly, we use $M_r(x) = (x - L_{n-r}(x))/2^{n-r}$ (where n is the bit length of x) to denote the r most significant bits of the binary representation of x. For $x \in \mathbb{Z}_N$, we use $\widehat{M}_{N,r}(x)$ to denote any approximation of x with additive error $|x - \widehat{M}_{N,r}(x)| \leq N/2^r$.

Probability Distributions and Distinguishers. Let $\mathcal{D}$ denote a probability distribution over $\{0,1\}^\ell$. We denote by $s \leftarrow \mathcal{D}$ the assignment to s of a random element sampled from the distribution $\mathcal{D}$. If S denotes a set then we let $s \in_R S$ denote the assignment to s of a *uniformly* random element sampled from S. Let $\mathcal{D}_1$ and $\mathcal{D}_2$ denote two probability distributions on some finite set. We say that an algorithm D is a (T,δ) distinguisher between $\mathcal{D}_1$ and $\mathcal{D}_2$ if D runs in time at most T and has distinguishing advantage at least δ between $\mathcal{D}_1$ and $\mathcal{D}_2$, i.e. $|\Pr_{s\leftarrow\mathcal{D}_1}[\mathsf{D}(s)=1] - \Pr_{s\leftarrow\mathcal{D}_2}[\mathsf{D}(s)=1]| \geq \delta$. The *statistical distance* between two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ is $\frac{1}{2}\sum_s |\mathcal{D}_1(s) - \mathcal{D}_2(s)|$. It gives an upper bound on the distinguishing advantage of any distinguisher between $\mathcal{D}_1$ and $\mathcal{D}_2$, regardless of run-time.

Pseudorandom Bit Generators (PRGs). We use the following definition of pseudorandom generators and their concrete pseudorandomness.

Definition 1 ((T,δ) PRG). *A (T,δ) Pseudorandom Generator (family)* PRG *is a collection of functions $G_N : \mathcal{S}_N \to \{0,1\}^\ell$ indexed by $N \in \mathcal{I}_n$. Here $\mathcal{I}_n$ (PRG function index space) and $\mathcal{S}_N$ (PRG seed domain) are both efficiently samplable subsets of $\{0,1\}^n$, where n is the security parameter. We require that any (probabilistic) distinguisher algorithm* D *running in time T has distinguishing advantage at most δ between the* pseudorandom distribution $\mathcal{D}_{P,\ell}$ *and the* random distribution $\mathcal{D}_{R,\ell}$ *on ℓ-bit strings, which are defined as follows:*

$$\mathcal{D}_{P,\ell} = \{s : N \in_R \mathcal{I}_n; x_0 \in_R \mathcal{S}_N; s = G_N(x_0)\}$$

while

$$\mathcal{D}_{R,\ell} = \{s : s \in_R \{0,1\}^\ell\}.$$

If algorithm D *runs in time* T *and has distinguishing advantage at least* δ *between* $\mathcal{D}_{P,\ell}$ *and* $\mathcal{D}_{R,\ell}$*, we say that* D *is a* (T,δ) *distinguisher for* PRG.

The RSA Inversion Problem. The classical RSA inversion problem is defined as follows.

Definition 2 ((n,e)-RSA problem). *Let* e *be a fixed integer. Let* $\mathcal{I}_n$ *denote the set of all* n*-bit RSA moduli* $N = pq$ *(for p,q primes of* $n/2$ *bits each) such that* $\gcd(e,(p-1)(q-1)) = 1$. *The* (n,e)*-RSA inversion problem is the following: given* $N \in_R \mathcal{I}_n$ *and* $y = [x^e]_N$ *for* $x \in_R \mathbb{Z}_N$*, find* x. *We say that algorithm* A *is a* (T,ϵ) *inversion algorithm for* (n,e)*-RSA if* A *runs in time* T *and succeeds with probability* ϵ *over the choice of* $N \in_R \mathcal{I}_n$, $x \in_R \mathbb{Z}_N$ *and the random coins of* A.

Lattices. Let $\{\mathbf{b}_1,\ldots,\mathbf{b}_n\}$ be a set of n linearly independent vectors in $\mathbb{R}^n$. The set

$$\mathcal{L} = \{\mathbf{z}\colon \mathbf{z} = c_1\mathbf{b}_1 + \ldots + c_n\mathbf{b}_n; c_1,\ldots,c_n \in \mathbb{Z}\}$$

is called an *n-dimensional (full-rank) lattice* with *basis* $\{\mathbf{b}_1,\ldots,\mathbf{b}_n\}$. Given a basis $\mathbf{B} = \{\mathbf{b}_1,\ldots,\mathbf{b}_n\}$ for a lattice $\mathcal{L}$, we define the associated *basis matrix* $M_{\mathcal{L},\mathbf{B}}$ to be the (full-rank) $n \times n$ matrix whose ith row is the ith basis vector $\mathbf{b}_i$ for $i = 1,\ldots,n$. The quantity $|\det(M_{\mathcal{L},\mathbf{B}})|$ is independent of $\mathbf{B}$. It is called the *determinant* of the lattice $\mathcal{L}$ and denoted by $\det(\mathcal{L})$. Given any basis of a lattice $\mathcal{L}$, the well-known LLL algorithm [22] outputs in polynomial time a *reduced basis* for $\mathcal{L}$ consisting of short vectors. We use the following result [8] bounding the length of those vectors.

Lemma 1. *Let* $\mathcal{L}$ *be a lattice of dimension* d *with basis matrix* $\mathbf{B}_{\mathcal{L}}$ *in lower diagonal form whose diagonal elements are greater or equal to 1. Then the Euclidean norm of the first two vectors in the LLL reduced basis for* $\mathcal{L}$ *is at most* $2^{d/2}(\det(\mathcal{L}))^{\frac{1}{d-1}}$.

3 Overview of the Fischlin-Schnorr Security Proof

The RSA PRG. We begin by recalling the RSA PRG construction.

Definition 3 ((n,e,r,ℓ)-RSAPRG Pseudorandom Generator). *The psuedorandom generator family* (n,e,r,ℓ)-RSAPRG *is defined as follows. The PRG function index space* $\mathcal{I}_n$ *is the set of all* n*-bit RSA moduli* $N = pq$ *(for p,q primes of* $n/2$ *bits each) such that* $\gcd(e,(p-1)(q-1)) = 1$. *Given index* $N \in \mathcal{I}_n$ *the PRG seed domain is* $\mathbb{Z}_N$. *Assume that* ℓ *is a multiple of* r. *Given a seed* $x_0 \in_R \mathbb{Z}_N$, *the PRG function* $G_N : \mathbb{Z}_N \to \{0,1\}^\ell$ *is defined by*

$$G_N(x_0) = (s_0,\ldots,s_{\ell/r-1}) : s_i = L_r(x_i), x_{i+1} = [x_i^e]_N \text{ for } i = 0,\ldots,\ell/r-1.$$

As will become clear below, our result builds on the Fischlin-Schnorr result in essentially a 'black box' way, so our result can be understood without knowing most of the internal details of the reduction in [14]. Hence, in this section we provide only a very high-level overview of the basic security reduction [14] for the RSA PRG from the RSA assumption, in the case of r LS bits output per iteration (refer to the full version of the paper [29] for more details).

Using our notation, the Fischlin-Schnorr security result can be stated concretely as follows.

Theorem 1 (Fischlin-Schnorr [14]). *For all $n \geq 2^9$, any (T, δ) distinguisher D for (n, e, r, ℓ)-RSAPRG can be converted into a $(T_{INV}, \delta/9)$ inversion algorithm A for the (n, e)-RSA problem with run-time at most*

$$T_{INV} = 2^{2r+14}(\ell/\delta)^6 n \log(n) \cdot (T + O(\ell/r \log(e) n^2)). \tag{1}$$

Proof. We are given a distinguisher D with run-time T and distinguishing advantage $\mathsf{Adv}(\mathsf{D}) \geq \delta$ between the *pseudorandom distribution* $\mathcal{D}_{P,\ell}$ (obtained by iterating $m = \ell/r$ times and outputting r LS bits per iteration) and the *random* distribution $\mathcal{D}_{R,\ell}$ on ℓ bit strings, namely:

$$\mathcal{D}_{P,\ell} = \{G_N(x_0) : N \in_R \mathcal{I}_n; x_0 \in_R \mathbb{Z}_N\}$$

while

$$\mathcal{D}_{R,\ell} = \{s : s \in_R \{0,1\}^\ell\}.$$

We use D to construct the (n, e)-RSA inversion algorithm A as follows.

As a first step, we note that the pseudorandom distribution $\mathcal{D}_{P,\ell}$ is taken over the random choice of modulus $N \in_R \mathcal{I}_n$ as well as random seed $x_0 \in_R \mathbb{Z}_N$. For the remainder of the proof, we wish to fix N and find a lower bound on the distinguishing advantage $\mathsf{Adv}_N(\mathsf{D})$ between $\mathcal{D}_{R,\ell}$ and the pseudorandom distribution $\mathcal{D}_{P,\ell,N}$ taken over just the random choice of $x_0 \in_R \mathbb{Z}_N$ for this fixed N, that is:

$$\mathcal{D}_{P,\ell,N} = \{G_N(x_0) : x_0 \in_R \mathbb{Z}_N\}.$$

To do so, we use an averaging argument over N.

Lemma 2. *There exists a subset $\mathcal{G}_n \subseteq \mathcal{I}_n$ of size at least $|\mathcal{G}_n| \geq \delta/2|\mathcal{I}_n|$ such that D has distinguishing advantage at least $\delta/2$ between the distributions $\mathcal{D}_{P,\ell,N}$ and $\mathcal{D}_{R,\ell}$ for all $N \in \mathcal{G}_n$.*

From now on we assume that $N \in \mathcal{G}_n$ (which happens with probability at least $\delta/2$ over $N \in_R \mathcal{I}_n$) so that D has distinguishing advantage at least $\delta/2$ between $\mathcal{D}_{P,\ell,N}$ and $\mathcal{D}_{R,\ell}$ (We remark that this first step is actually omitted in [14] which always assumes a fixed N; however we add this step since we believe it is essential for a meaningful security proof: to demonstrate an efficient algorithm for RSA inversion contradicting the RSA assumption, one must evaluate its success probability over the random choice of modulus N, since for any fixed N an efficient algorithm always exists; it has built into it the prime factors of N).

We now convert ℓ/r-iteration distinguisher D into a 1-iteration distinguisher D'. This is a 'hybrid' argument using the fact that the mapping $x \rightarrow [x^e]_N$ is a permutation on $\mathbb{Z}_N$. Note that the 'hybrid' argument underlying this reduction has been known since the work of [18,7] and it is not explicitly included in [14].

Lemma 3 ($m = \ell/r$ iterations to 1 iteration.). *Any (T, δ) distinguisher D between the m-iteration pseudorandom distribution $\mathcal{D}_{P,\ell,N}$ and the random distribution $\mathcal{D}_{R,\ell}$ can be converted into a $(T + O(m \log(e) n^2), \delta/m)$ 1-iteration distinguisher D' between the distributions*

$$\mathcal{D}'_{P,r,N} = \{(y = [x^e]_N, s = L_r(x)) : x \in_R \mathbb{Z}_N\}$$

and

$$\mathcal{D}'_{R,r,N} = \{(y = [x^e]_N, s) : x \in_R \mathbb{Z}_N; s \in_R \{0,1\}^r\}.$$

The main part of the Fischlin-Schnorr reduction [14] is the conversion of the distinguisher D' into an inversion algorithm that recovers the RSA preimage x from $y = [x^e]_N$ *with the help of some additional information on* x, namely r least-significant bits of $[ax]_N$ and $[bx]_N$ for some randomly chosen known $a, b \in \mathbb{Z}_N$, as well as rough approximations to $[ax]_N$ and $[bx]_N$. This is stated more precisely as follows.

Lemma 4 (Distinguisher to Inverter). *For all $n \geq 2^9$, any (T, δ) distinguisher D' between the distributions $\mathcal{D}'_{P,r,N}$ and $\mathcal{D}'_{R,r,N}$ (see Lemma 3) can be converted into an inversion algorithm A' that, given N and $(y = [x^e]_N, a \in_R \mathbb{Z}_N, s_1 = L_r([ax]_N), u_1 = \widehat{M}_{N,k}([ax]_N), b \in_R \mathbb{Z}_N, s_2 = L_r([bx]_N), u_2 = \widehat{M}_{N,l}([bx]_N))$, for any $x \in \mathbb{Z}_N$ with $k = 3\log(r/\delta) + 4$ and $l = \log(r/\delta) + 4$, outputs x with probability $\epsilon'_{INV} \geq 2/9$ (over the choice of $a \in_R \mathbb{Z}_N$, $b \in_R \mathbb{Z}_N$ and the random coins of A') and runs in time $T'_{INV} = 4n\log(n)(r/\delta)^2 \cdot (T + O(n^2))$. Here $\widehat{M}_{N,k}(x)$ denotes any approximation of x with additive error $|\widehat{M}_{N,k}(x) - x| \leq 2^{n-k}$.*

Putting it Together. On input $(N, y = [x^e]_N)$, the RSA inversion algorithm A runs as follows. It applies Lemmas 2 and 3 to convert the (T, δ) distinguisher D into a $(T + O(m\log(e)n^2), \delta/(2m))$ distinguisher D' between distributions $\mathcal{D}'_{P,r,N}$ and $\mathcal{D}'_{R,r,N}$ which works for at least a fraction $\delta/2$ of $N \in \mathcal{I}_n$. Then A applies Lemma 4 to convert D' into the inversion algorithm A'. A now chooses random a and b in $\mathbb{Z}_N$. Since A does not know the 'extra information' $s_1 = L_r([ax]_N)$, $u_1 = \widehat{M}_{N,k}([ax]_N)$, $s_2 = L_r([bx]_N)$ and $u_2 = \widehat{M}_{N,l}([bx]_N))$ required by A', A just exhaustively searches through all N_G possible values of (s_1, u_1, s_2, u_2) and runs A' on input $(N, y = [x^e], \widehat{s}_1, \widehat{u}_1, \widehat{s}_2, \widehat{u}_2)$ for every guessed possibility $(\widehat{s}_1, \widehat{u}_1, \widehat{s}_2, \widehat{u}_2)$ until A' succeeds to recover x. Note that to find an approximation $\widehat{M}_{N,k}([ax]_N)$ correct within additive error $N/2^k$ it is enough to search through 2^{k-1} uniformly spaced possibilities $(N/2^{k-1})i$ for $i = 0, \ldots, 2^{k-1} - 1$. Since $k = 3\log(2mr/\delta) + 4 = 3\log(2\ell/\delta) + 4$ and $l = \log(2\ell/\delta) + 4$, there are at most

$$N_G = 64(2\ell/\delta)^4 2^{2r} \tag{2}$$

guessing possibilities for $L_r([ax]_N)$, $\widehat{M}_{N,k}([ax]_N)$, $L_r([bx]_N)$, $\widehat{M}_{N,l}([bx]_N)$ to search through. So the run-time bound of A is

$$\begin{aligned} T_{INV} &= N_G \cdot (4n\log(n)(2\ell/\delta)^2) \cdot (T + O(m\log(e)n^2)) \\ &= 2^{2r+14}(2\ell/\delta)^6 n\log(n) \cdot (T + O(m\log(e)n^2)). \end{aligned} \tag{3}$$

For at least a fraction $\delta/2$ of $N \in \mathcal{I}_n$, with the correct guessed value of the 'extra information', A' succeeds with probability at least $2/9$ over the choice of a, b. Hence we conclude that the success probability of A is at least $\epsilon_{INV} \geq \delta/9$, as claimed. □

We can interpret Theorem 1 as follows. Suppose we assume that the expected run-time T_{INV}/ϵ_{INV} of any $(T_{INV}, \epsilon_{INV})$ RSA inversion algorithm is at least T_L. Then Theorem 1 can be used to convert a (T, δ) distinguisher for (n, e, r, ℓ)-RSAPRG to an RSA inverter contradicting our hardness assumption only if we output at most r bits per iteration, where

$$r < \frac{1}{2} \log \left(\frac{1}{9 \cdot 2^{14} \cdot n \log n \ell^6 \delta^{-7}} \cdot \frac{T_L}{T} \right). \tag{4}$$

Hence asymptotically, if we take $T_L = poly(n)$ (i.e. assume no poly-time RSA algorithm) then we get $r = O(\log(n))$ bits per iteration. If we assume that $T_L = O(2^{cn^{1/3}(\log n)^{2/3}})$ for constant c (run-time of the Number Field Sieve factoring algorithm [23]) then we can have $r = O(n^{1/3} \log^{2/3} n)$. But in any case, $r = o(n)$.

4 Our Modified Security Proof from an SSRSA Problem

We now explain how we modify the above reduction to solve a well-studied SSRSA problem and the resulting improved PRG efficiency/security tradeoff.

Our goal is to remove the search factor $N_G = 64 \cdot 2^{2r}(\ell/\delta)^4$ from the run-time bound (3) of the reduction in the proof of Theorem 1. The simplest way to do so is to provide the inversion algorithm A with the correct values for the 'extra information' required by the inversion algorithm A' of Lemma 4. This leads us to consider the following (not well-known) inversion problem that we call (n, e, r, k, l)-FSRSA :

Definition 4 ((n, e, r, k, l)-FSRSA Problem.). *Given RSA modulus N, and $(y = [x^e]_N, a \in_R \mathbb{Z}_N, s_1 = L_r([ax]_N), u_1 = \widehat{M}_k([ax]_N), b \in_R \mathbb{Z}_N, s_2 = L_r([bx]_N), u_2 = \widehat{M}_l([bx]_N))$, for $x \in_R \mathbb{Z}_N$, find x (here $\widehat{M}_{N,k}(x)$ denotes any approximation to x with additive error $|\widehat{M}_{N,k}(x) - x| \leq N/2^k$). We say that algorithm A is a (T, η) inversion algorithm for (n, e, r, k, l)-FSRSA if A runs in time at most T and has success probability at least η (over the random choice of $N \in_R \mathcal{I}_n$, $x, a, b \in_R \mathbb{Z}_N$ and the random coins of A, where $\mathcal{I}_n$ is the same as in Definition 2).*

With the search factor N_G removed from the Fischlin-Schnorr reduction we therefore have that the hardness of the inversion problem (n, e, r, k, l)-FSRSA (with $k = 3\log(2\ell/\delta) + 4$ and $l = \log(2\ell/\delta) + 4$) suffices for the 'simultaneous security' of the r least-significant RSA message bits (i.e. indistinguishability of distributions $\mathcal{D}'_{P,r,N}$ and $\mathcal{D}'_{R,r,N}$ in Lemma 3) and hence the pseudorandomness of (n, e, r, ℓ)-RSAPRG, with a much tighter reduction than the one of Theorem 1 relative to the RSA problem.

Theorem 2. *For all $n \geq 2^9$, any (T, δ) distinguisher* D *for* (n, e, r, ℓ)-RSAPRG *can be converted into a $(T_{INV}, \delta/9)$ inversion algorithm* A *for the (n, e, r, k, l)-FSRSA problem (with $k = 3\log(2\ell/\delta) + 4$ and $l = \log(2\ell/\delta) + 4$) with run-time at most*

$$T_{INV} = 16 \cdot (\ell/\delta)^2 n \log(n) \cdot (T + O(\ell/r \log(e) n^2)). \tag{5}$$

Proof. We use the same inversion algorithm A as in the proof of Theorem 1, except that when applying Lemma 4, A runs inversion algorithm A' just once using the correct values of $(a, b, s_1 = L_r([ax]_N), u_1 = \widehat{M}_{N,k}([ax]_N), s_2 = L_r([bx]_N), u_2 = \widehat{M}_{N,l}([bx]_N))$ given as input to A, eliminating the search through $N_G = 64(2\ell/\delta)^4 2^{2r}$ possible values for (s_1, u_1, s_2, u_2). □

We defer to Section 6.1 our cryptanalysis of the (n, e, r, k, l)-FSRSA problem using the lattice-based method introduced by Coppersmith [11], which leads us to conjecture that the problem is hard whenever $r/n \leq 1/2 - 1/(2e) - (k + l)/2n - \epsilon$ for constant $\epsilon > 0$. This assumption together with the above reduction already implies the security of the efficient variants of (n, e, r, ℓ)-RSAPRG with $r = \Omega(n)$. Unfortunately, (n, e, r, k, l)-FSRSA is a new problem and consequently our conjecture on its hardness is not currently supported by extensive research. However, we will now show that in fact for $r/n = 1/2 - \max(k, l)/n - 1/e - \epsilon$ (note that this is smaller by $(\max(k, l) - (k + l)/2)/n + 1/(2e)$ than the largest secure value of r/n conjectured above), the problem (n, e, r, k, l)-FSRSA is at least as hard as a specific $(1/e + \epsilon, e)$-SSRSA problem (i.e. with a specific univariate polynomial f of degree e) which we call (n, e, r, w)-CopRSA and define as follows:

Definition 5 ((n, e, r, w)-CopRSA Problem.). *Given RSA modulus N, and $(y = [x^e]_N, s_L = L_r(x), s_H = M_{n/2+w}(x))$, for $x \in_R \mathbb{Z}_N$, find x (here $M_k(x)$ denotes the k most-significant bits of the binary representation of x). We say that algorithm* A *is a (T, η) inversion algorithm for (n, e, r, w)-CopRSA if* A *runs in time at most T and has success probability at least η (over the random choice of $N \in_R \mathcal{I}_n$, $x \in_R \mathbb{Z}_N$ and the random coins of* A*, where $\mathcal{I}_n$ is the same as in Definition 2).*

To see that (n, e, r, w)-CopRSA problem is a specific type of SSRSA problem, note that it is equivalent to finding a small solution $\bar{z} < 2^{n/2-(r+w)}$ (consisting of bits $r + 1, \ldots, (n/2 - w)$ of the randomly chosen integer x) to the equation $f(\bar{z}) \equiv y \bmod N$, where the degree e polynomial $f(z) = (2^r z + s)^e$, where $s = s_H \cdot 2^{n/2-w} + s_L$ is known. Hence (n, e, r, w)-CopRSA is a $(1/e + \epsilon, e)$-SSRSA problem when $1/2 - (r + w)/n = 1/e + \epsilon$, i.e. $r/n = 1/2 - 1/e - \epsilon - w/n$.

Theorem 3. *Let* A' *be a (T', η') attacker against $(n, e, r, w - 1, w - 1)$-FSRSA. Then we construct a (T, η) attacker* A *against (n, e, r, w)-CopRSA with*

$$T = 4T' + O(n^2) \text{ and } \eta = \eta' - 4/2^{n/2}.$$

Proof. On input $(N, y = [x^e]_N, s_L = L_r(x), s_H = M_{n/2+w}(x))$, for $N \in_R \mathcal{I}_n$ and $x \in_R \mathbb{Z}_N$, the attacker A runs as follows:

– Choose a uniformly random $b \in_R \mathbb{Z}_N$.
– Compute an integer c coprime to N with $|c| < N^{1/2}$ such that $|[b{\cdot}c]_{\underline{N}}| < N^{1/2}$ (here $[z]_{\underline{N}} \in (-N/2, N/2)$ denotes the 'symmetrical' residue of z modulo N, i.e. $[z]_{\underline{N}} \overset{\text{def}}{=} [z]_N$ if $[z]_N \in [0, N/2)$ and $[z]_{\underline{N}} \overset{\text{def}}{=} [z]_N - N$ if $[z]_N \in (N/2, N)$). It is well known that such a c exists and can be computed efficiently (in time $O(n^2)$) using continued fractions (see, e.g. Lemma 16 in [25]).
– Observe that $[cx]_N = cx - \omega_c N$, where $\omega_c = \lfloor \frac{cx}{N} \rfloor$. Let $\widehat{x} = s_H \cdot 2^{n/2-w}$. Notice that $\widehat{x}$ approximates x within additive error $\Delta_x \leq 2^{n/2-w}$ and consequently the rational number $\frac{c\widehat{x}}{N}$ approximates $\frac{cx}{N}$ within additive error $\frac{|c|\Delta_x}{N} \leq \Delta_x / N^{1/2} \leq 2^{n/2-w}/2^{(n-1)/2} < 1$, where we have used the fact that $|c| < N^{1/2}$ and $w \geq 1$. It follows that $\omega_c \in \{\lfloor \frac{c\widehat{x}}{N} \rfloor, \lfloor \frac{c\widehat{x}}{N} \rfloor \pm 1\}$ (where the + sign applies if $c \geq 0$ and the − sign applies otherwise). So A obtains 2 candidates for ω_c.
– Using $L_r([cx]_N) = L_r(cx - \omega_c N) = L_r(L_r(c) \cdot L_r(x) - L_r(\omega_c N))$, A computes (with the known $s_L = L_r(x)$, c and N) 2 candidates for $L_r([cx]_N)$ from the 2 candidates for ω_c.
– Similarly, writing $[bcx]_N = [bc]_{\underline{N}} \cdot x - \omega_{bc} N$, with $\omega_{bc} = \lfloor \frac{[bc]_{\underline{N}} x}{N} \rfloor$, using $|[bc]_{\underline{N}}| < N^{1/2}$ we obtain $\omega_{bc} \in \{\lfloor \frac{[bc]_{\underline{N}} \widehat{x}}{N} \rfloor, \lfloor \frac{[bc]_{\underline{N}} \widehat{x}}{N} \rfloor \pm 1\}$ (with + sign if $[bc]_{\underline{N}} \geq 0$ and − sign otherwise), so A also computes 2 candidates for ω_{bc} and two corresponding candidates for $L_r([bcx]_N) = L_r([bc]_{\underline{N}} x - \omega_{bc} N) = L_r(L_r([bc]_{\underline{N}}) L_r(x) - \omega_{bc} N)$.
– Using $\widehat{x}$ and the 2 candidates for ω_c computed above, A computes two candidate approximations $c\widehat{x} - \omega_c N$ for $[cx]_N$. Since $\widehat{x}$ approximates x within additive error $\Delta_x \leq 2^{n/2-w}$ we have that $c\widehat{x} - \omega_c N$ approximates $[cx]_N$ within additive error $|c|\Delta_x \leq N^{1/2} 2^{(n-1)/2} / 2^{w-1/2} \leq N/2^{w-1}$ using $N \geq 2^{n-1}$.
– Similarly, using $\widehat{x}$ and the 2 candidates for ω_{bc} computed above, A computes two candidate approximations $[bc]_{\underline{N}} \widehat{x} - \omega_{bc} N$ for $[bcx]_N$, one of which has additive error $|[bc]_{\underline{N}}| \Delta_x \leq N/2^{w-1}$.
– Choose a uniformly random $a \in \mathbb{Z}_N^*$ and compute $y' = [(a^{-1}c)^e y]_N = [(a^{-1}cx)^e]_N$.
– Collecting all of the above information, A obtains 4 candidates for $(N, y' = [(a^{-1}cx)^e]_N, a, s_1 = L_r([cx]_N), u_1 = \widehat{M}_{N,w-1}([cx]_N), b' = [ab]_N, s_2 = L_r([bcx]_N), u_2 = \widehat{M}_{N,w-1}([bcx]_N))$. Note that this is a valid instance of $(n, e, r, w-1, w-1)$-FSRSA. Furthermore, it has almost exactly the correct distribution, since the triple $(x' = [a^{-1}cx]_N, a, b' = [ab]_N)$ is uniformly random in $\mathbb{Z}_N \times \mathbb{Z}_N^* \times \mathbb{Z}_N$ thanks to the uniformly random choice of $(x, a, b) \in \mathbb{Z}_N \times \mathbb{Z}_N^* \times \mathbb{Z}_N$. The FSRSA instance distribution is not exactly correct because here a is uniform on $\mathbb{Z}_N^*$ while it should be uniform on $\mathbb{Z}_N$. However, simple calculation shows that the statistical distance between the uniform distribution on $\mathbb{Z}_N^*$ and the uniform distribution on $\mathbb{Z}_N$ is negligible, namely $1 - \phi(N)/N = (p+q-1)/N \leq 4/2^{n/2}$.
– A runs A' on the above 4 candidate $(n, e, r, w-1, w-1)$-FSRSA instances. On one of those runs, A' outputs $x' = [a^{-1}cx]_N$ with probability at least $\eta - 4/2^{n/2}$, from which x is easily recovered as $x = [ac^{-1}x']_N$.

Note that the run-time of A is bounded as $T \leq 4T' + O(n^2)$ and A succeeds with probability at least $\eta - 4/2^{n/2}$, as required. This completes the proof. $\square$

So, combining Theorems 2 and 3, we conclude:

Corollary 1. *For all $n \geq 2^9$, any (T, δ) distinguisher D for (n, e, r, ℓ)-*RSAPRG *can be converted into a $(T_{INV}, \epsilon_{INV})$ inversion algorithm A for the (n, e, r, w)-CopRSA problem (with $w = 3\log(2\ell/\delta) + 5$) with*

$$T_{INV} = 64 \cdot (\ell/\delta)^2 n \log(n) \cdot (T + O(\ell/r \log(e) n^2)) \text{ and } \epsilon_{INV} = \delta/9 - 4/2^{n/2}. \quad (6)$$

Remark. Fischlin and Schnorr [14] also outline an alternative security reduction (worked out in detail and optimized for the Rabin iteration function by Sidorenko and Schoenmakers [28]) for the (n, e, r, ℓ)-RSAPRG with $r > 1$ based on a general 'Computational XOR Lemma' [30,16]. However, this alternative reduction has an inherent exponential run-time factor 2^{2r} which we do not know how to eliminate, even using our stronger SSRSA assumption on RSA inversion.

5 Concrete Parameters and Estimated Performance

Using (6) we obtain an upper bound on the pseudorandom string length ℓ for a given security level (T, δ) and assumed expected run-time lower bound T_L for breaking the $(n, e, r, 3\log(2\ell/\delta) + 5)$-CopRSA problem. Recall that the latter is a $(1/e + \epsilon, e)$-SSRSA problem when

$$r/n = 1/2 - 1/e - \epsilon - (3\log(2\ell/\delta) + 5)/n, \quad (7)$$

and that $(1/e + \epsilon, e)$-SSRSA problem is conjectured to take time $T_L = \min(T_F(n), T_C(n, \epsilon))$, where $T_F(n)$ is a lower bound for factoring N and $T_C(n, \epsilon) = poly(n) \cdot 2^{\epsilon n}$ is the time for the Coppersmith attack on $(1/e + \epsilon, e)$-SSRSA. Asymptotically, we therefore have for any constant $\epsilon > 0$ that $T_L = T_F(n)$ since $T_F(n)$ is subexponential in n, so for any $\ell/\delta = poly(n)$ and $e \geq 3$ we can use $r/n = 1/2 - 1/e - \epsilon - o(1)$, i.e. $r = \Omega(n)$. The exact bound on r for a given modulus length n depends on the value of ϵ such that $T_F(n) = T_C(n, \epsilon)$. To estimate concrete values, we used the Number Field Sieve (NFS) factoring run-time model from [23] (we refer to the full version of the paper for more details [29]) – the results are summarised in Table 1.

Our estimates indicate that we can (with $n = 6144$ bit and $e = 8$) achieve a rate around 19300 cycles/byte (0.87 Mbit/s with 2.1 GHz clock) on a Pentium 4 Processor, outputting more than 2^{30} bits with provable 2^{70} instructions distinguishing run-time (under the $(1/e + \epsilon, e)$-SSRSA assumption). This seems to be close to practical requirements of some stream cipher applications (it is several hundred times faster than the basic Blum-Blum Shub generator outputting one bit per iteration with the same modulus length). Compared to the recent provably secure QUAD PRG construction [3] (based on the 'MQ' problem), our PRG seems to have a lower throughput, although it is difficult to make a fair comparison since unlike our figures above, the performance figures reported in [3]

Table 1. Estimate of achievable performance for provable $T = 2^{70}$ instructions distinguishing time to achieve advantage $\delta = \frac{1}{100}$, using $e = 8, 9$ (assuming hardness of the CopRSA SSRSA problem) and $e = 2$ (assuming hardness of FSRSA problem - see Section 6). Throughput ('Thrpt') columns are estimated throughput based on Wei Dai's Crypto++ benchmarks page [13] (for Pentium 4 2.1GHz processor) and extrapolation assuming classical arithmetic.

n **(bit)**	$\log(\ell)$	**Rate,**$e=8$ **(bit/mult)**	**Thrpt (Mbit/s)**	**Rate,**$e=9$ **(bit/mult)**	**Thrpt (Mbit/s)**	**Rate,**$e=2$ **(bit/mult)**	**Thrpt (Mbit/s)**
3072	9.3	341	1.68	267	1.31	660	3.2
4096	18.0	460	1.28	360	1.00	899	2.5
5120	25.4	581	1.03	454	0.80	1140	2.0
6144	32.0	702	0.87	549	0.67	1383	1.7

(between 3000 and 4500 cycles/byte on Pentium 4) are for a 'practical' choice of parameters, smaller than those for which the security proof can be applied. A possible advantage of our construction is its significantly smaller static parameters (i.e. non-secret parameters defining the pseudorandom generator) of length $n \approx 5$ kbit, while in [3] the static parameters are longer than 1 Mbit (this might allow our construction to be implemented with less code memory requirements). On the other hand, our construction has a longer state and is based on the hardness of factoring so is insecure against potential future quantum attacks, while the MQ problem in [3] may be secure even against such attacks.

6 Potential Improvements

6.1 Cryptanalysis of the FS-RSA Problem

As observed in Section 4, the (n, e, r, k, l)-FSRSA problem, although not well-known, gives a more direct proof of security for the RSA PRG than the SSRSA problem. In this section we describe a 'Coppersmith-type' lattice attack on (n, e, r, k, l)-FSRSA (which we believe is essentially optimal) and show that it is likely to succeed only when $r/n \geq 1/2 - (k + l)/(2n) - 1/(2e)$. This value of r/n is larger by about $1/(2e) + (\max(k, l)/n - (k + l)/(2n))$ than that the largest value for which the corresponding SSRSA problem in Section 4 is secure, leading to improved throughput for the RSA PRG by using this stronger assumption.

The attack on (n, e, r, k, l)-FSRSA problem works as follows. First we reduce the problem to solving two modular equations in two small unknowns z_1 and z_2. Namely, given $(y = [x^e]_N, a \in_R \mathbb{Z}_N, s_1 = L_r([ax]_N), u_1 = \widehat{M}_{N,k}([ax]_N), b \in_R \mathbb{Z}_N, s_2 = L_r([bx]_N), u_2 = \widehat{M}_{N,l}([bx]_N))$, we have

$$x^e \equiv y \pmod N, \tag{8}$$

$$[ax]_N = s_1 + \bar{z}_1' \cdot 2^r; |[ax]_N - u_1| \leq N/2^k \tag{9}$$

and

$$[bx]_N = s_2 + \bar{z}_2' \cdot 2^r; |[bx]_N - u_2| \leq N/2^l \tag{10}$$

where $\bar{z}'_1 < N/2^r$ and $\bar{z}'_2 < N/2^r$ consist of the $n-r$ MS bits of $[ax]_N$ and $[bx]_N$, respectively. Let $\widehat{z}_1 = \lfloor \frac{u_1 - s_1}{2^r} \rfloor$. From (9) we conclude that $|\bar{z}'_1 - \widehat{z}_1| \leq |(\frac{[ax]_N - s_1}{2^r}) - (\frac{u_1 - s_1}{2^r})| + 1 \leq N/2^{r+k} + 1 \leq N/2^{r+k-1}$ (for $2^{r+k} < N$) and hence letting $\bar{z}_1 = \bar{z}'_1 - \widehat{z}_1$ we obtain $[ax]_N = (s_1 + 2^r \widehat{z}_1) + 2^r \bar{z}_1$ where integer $|\bar{z}_1| < N/2^{r+k-1}$. Similarly, from (10) we obtain $[bx]_N = (s_2 + 2^r \widehat{z}_2) + 2^r \bar{z}_2$ where integer $|\bar{z}_2| < N/2^{r+l-1}$ (for $2^{r+l} \leq N$) and $\widehat{z}_2 = \lfloor (u_2 - s_2)/2^r \rfloor$. Treating the last two equations for $[ax]_N$ and $[bx]_N$ as congruences modulo N, we eliminate the unknown variable x (by multiplying the second congruence by $[ab^{-1}]_N$ and subtracting from the first) to obtain a single linear polynomial $f(z_1, z_2)$ in two variables z_1, z_2, having the desired small unknowns $\bar{z}_1, \bar{z}_2$ as a zero modulo N (i.e. $f(\bar{z}_1, \bar{z}_2) \equiv 0 \pmod{N}$), namely:

$$f(z_1, z_2) = \alpha \cdot z_1 + z_2 + \beta, \tag{11}$$

where $\alpha = [-ab^{-1}]_N$ and $\beta = [-a^{-1}b2^{-r}(s_1 + 2^r \widehat{z}_1) + 2^{-r}(s_2 + 2^r \widehat{z}_2)]_N$ are known. Also, substituting $x \equiv a^{-1}(s_1 + 2^r \widehat{z}_1) + 2^r a^{-1} \widehat{z}_1 \pmod{N}$ into (8) we obtain a degree e univariate polynomial in z_1 having the small unknown $\bar{z}_1$ as a zero modulo N (i.e. $g(\bar{z}_1) \equiv 0 \pmod{N}$):

$$g(z_1) = (z_1 + \widehat{\alpha})^e - \widehat{\beta}, \tag{12}$$

where $\widehat{\alpha} = [2^{-r}s_1 + \widehat{z}_1]_N$ and $\widehat{\beta} = [-(a2^{-r})^e y]_N$ are known. To find the small zero $(\bar{z}_1, \bar{z}_2)$ of (11) and (12) we use the bivariate modular polynomial lattice method of Coppersmith [11] as simplified by Howgrave-Graham [20] and used in many subsequent works. Namely, for an integer m we use the polynomials $f(z_1, z_2)$ and $g(z_1)$ to construct the following family of polynomials $h_{i,k}(z_1, z_2)$ indexed by a pair of integers $i = 0, 1, \ldots, me$ (which we refer to as the 'block index') and $k = 0, \ldots, i$ (which we call the 'inner index') for each block i:

$$h_{i,k}(z_1, z_2) = N^{me - (i - k + \lfloor \frac{k}{e} \rfloor)} z_1^{[k]_e} g(z_1)^{\lfloor \frac{k}{e} \rfloor} f(z_1, z_2)^{i-k}. \tag{13}$$

Observe that each of the polynomials $h_{i,k}(z_1, z_2)$ has $(\bar{z}_1, \bar{z}_2)$ as a zero modulo N^{me}, because $f(\bar{z}_1, \bar{z}_2)^{i-k} \equiv 0 \pmod{N^{i-k}}$ and $g(\bar{z}_1)^{\lfloor \frac{k}{e} \rfloor} \equiv 0 \pmod{N^{\lfloor \frac{k}{e} \rfloor}}$.

It follows that any integer linear combination of the polynomials $h_{i,k}(z_1, z_2)$ also has $(\bar{z}_1, \bar{z}_2)$ as a zero modulo N^{me}. Let $B_1 = N/2^{r+k-1}$ and $B_2 = N/2^{r+l-1}$ denote the upper bounds derived above on $|\bar{z}_1|$ and $|\bar{z}_2|$, respectively. We set up a lattice $\mathcal{L}$ to search for linear combinations of the polynomials $h_{i,k}(z_1, z_2)$, which have sufficiently small coefficients such that they have $(\bar{z}_1, \bar{z}_2)$ as a zero over the integers, not just modulo N^{me}. Given two such linearly independent polynomials we can take their resultant to obtain a single univariate polynomial equation in z_1 over the integers which is easy to solve. The square basis matrix $\mathcal{B}_{\mathcal{L}}$ for lattice $\mathcal{L}$ has rows and columns indexed by pairs of integers (i, k), where the (i', k')th column of the (i, k)th row of $\mathcal{B}_{\mathcal{L}}$ contains the coefficient of the monomial $z_1^{k'} z_2^{i'-k'}$ in the polynomial $h_{i,k}(B_1 z_1, B_2 z_2)$. With this ordering, $\mathcal{B}_{\mathcal{L}}$ is in lower diagonal form and its determinant $\det(\mathcal{L})$ is the product of the diagonal elements of $\mathcal{B}_{\mathcal{L}}$. Some straightforward calculations (see full paper [29]) show that $\det(\mathcal{L}) = N^{me \cdot d(me) - W(m,e)} (B_1 B_2)^{D(me)/2}$, where $d(me) = \frac{1}{2}(me+1)(me+2)$ is the dimension of $\mathcal{L}$, $D(me) = \frac{e^3}{3}m^3 + O(m^2)$ and $W(m,e) = \frac{1}{2}D(me) + \frac{e^2}{6}m^3 + O(m^2)$. Let $h_1(z_1, z_2)$ and $h_2(z_1, z_2)$ denote the polynomials corresponding to the first two vectors in the reduced basis of $\mathcal{L}$ returned by LLL on input $\mathcal{B}_{\mathcal{L}}$. Using Lemma 1, we can show (see full paper [29]) that h_1 and h_2 will have a common zero over $\mathbb{Z}$ if the following condition is satisfied:

$$2^{d(me)/2}\det(\mathcal{L})^{\frac{1}{d(me)-1}} < \frac{N^{me}}{\sqrt{d(me)}}. \tag{14}$$

Plugging the expression for $\det(\mathcal{L})$ into this condition, we obtain $(B_1B_2)^{1/2} < N^{\frac{W(m,e)-me}{D(me)}}/\gamma(me)$, where the factor $\gamma(me) \stackrel{\text{def}}{=} (\sqrt{d(me)}2^{d(me)/2})^{\frac{d(me)-1}{D(me)}}$ is independent of n and so is of order $O(N^{o(1)})$ as n increases. For increasing parameter m, the leading m^3 terms dominate, and hence the ratio $\frac{W(m,e)-me}{D(me)}$ approaches asymptotically the value $\frac{1}{2} + \frac{e^2/6}{e^3/3} = \frac{1}{2} + \frac{1}{2e}$. So the attack success condition becomes $(B_1B_2)^{1/2} < N^{1/2+1/(2e)-o(1)}$ for large n and m. Using $B_1 = \frac{N}{2^{r+k-1}}$ and $B_2 = \frac{N}{2^{r+l-1}}$ and $N < 2^n$ we obtain the asymptotic attack success bound

$$\frac{r}{n} > 1/2 - 1/(2e) - \frac{(k+l)}{2n} + o(1). \tag{15}$$

Although the attack is heuristic (in the sense that resultant of h_1 and h_2 may be a zero polynomial), our numerical experiments (see [29]) suggest that the attack works in practice. We conjecture that bound (15) is essentially optimal for 'Coppersmith-type' lattice attacks on (n, e, r, k, l)-FSRSA (see [29]).

6.2 Using Even Exponents

Assuming Hardness of FSRSA Problem. If we assume that the attack of the previous section is optimal so the (n, e, r, k, l)-FSRSA problem is hard when the bound (15) is violated, then we can allow r/n to approach $1/4$ even for $e = 2$, with only one modular squaring required per iteration. It is shown in [14] that with appropriate modifications to the proof, Lemma 4 holds also for $e = 2$ if we replace the iteration function $x \to [x^e]_N$ by the 'absolute Rabin function' $f_a(x) = |x^2|_N \stackrel{\text{def}}{=} \min([x^2]_N, N - [x^2]_N)$, choose $N = pq$ to be a *Blum* RSA modulus with $p \equiv q \equiv 3 \pmod 4$, and choose the PRG seed $x_0 \in_R M_N$, where $M_N \stackrel{\text{def}}{=} \mathbb{Z}_N^*(+1) \cap (0, N/2)$, and $\mathbb{Z}_N^*(+1)$ denotes the subset of elements of $\mathbb{Z}_N^*$ having Jacobi symbol $+1$. Since f_a permutes the set M_N, the proof of Lemma 3 holds as well. Refer to Table 1 for performance of this PRG variant, where it is assumed that the best attack on (n, e, r, k, l)-FSRSA with $r/n = 1/2 - 1/(2e) - \frac{(k+l)}{2n} + \epsilon$ takes time $\min(T_F(n), 2^{\epsilon n})$, where $T_F(n)$ is the time needed to factor N. We stress however that this assumption is new and needs further study.

Assuming Hardness of SSRSA Problem. Our reduction (Theorem 3) from the CopRSA to FSRSA problem also extends with some small modifications to the case of even e (see [29]). For $e = 8$, it actually gives better rate than the best odd exponent assuming the hardness of SSRSA ($e = 9$) – see Table 1.

7 Conclusion

We have shown that an efficient variant of the RSA PRG is provably secure assuming the hardness of a well-studied variant of the RSA inversion problem in which some of the plaintext bits are known.

We see two avenues for further improvement. Even using the FSRSA assumption in Section 6, the PRG rate which we can prove secure is $r = (1/2 - 1/(2e) - \epsilon - o(1))n$ for 'small' ϵ. Can this rate be improved using a different proof (but a similar inversion assumption) up to $r = (1 - 1/e - \epsilon - o(1))n$? The other question is whether the factor ℓ^2 in the reduction run-time factor $O((\ell/\delta)^2 n \log(n))$ can be significantly reduced.

Finally we remark that besides generic applications of PRGs, our result can also be applied to prove security of an efficient semantically secure (IND-CPA) RSA-based public key encryption scheme, assuming the hardness of the SSRSA one-wayness problem (see [29]). An interesting open problem is to construct additional efficient cryptographic primitives based on this problem.

Acknowledgements. The authors would like to thank Scott Contini and Igor Shparlinski for enlightening discussions and encouragement and the anonymous referees for their useful comments. This work was supported by Australian Research Council Discovery Grants DP0663452, DP0451484 and DP0665035.

References

1. W. Alexi, B. Chor, O. Goldreich, and C.P. Schnorr. RSA and Rabin Functions: Certain Parts Are as Hard as the Whole. *SIAM Journal on Computing*, 17(2):194–209, 1988.
2. M. Ben-Or, B. Chor, and A. Shamir. On the Cryptographic Security of Single RSA Bits. In *Proc. 15-th STOC*, pages 421–430, New York, 1983. ACM Press.
3. C. Berbain, H. Gilbert, and J. Patarin. QUAD: a Practical Stream Cipher with Provable Security. In *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 109–128, Berlin, 2006. Springer-Verlag.
4. S.R. Blackburn, D. Gomez-Perez, J. Gutierrez, and I.E. Shparlinski. Reconstructing Noisy Polynomial Evaluation in Residue Rings. *Journal of Algorithms.* (To Appear).
5. S.R. Blackburn, D. Gomez-Perez, J. Gutierrez, and I.E. Shparlinski. Predicting Nonlinear Pseudorandom Number Generators. *Mathematics of Computation*, 74:1471–1494, 2004.
6. L. Blum, M. Blum, and M. Shub. A Simple Unpredictable Pseudo-Random Number Generator. *SIAM Journal on Computing*, 15:364–383, 1986.
7. M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, 13:850–864, 1984.
8. D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d less than $N^{0.292}$. *IEEE Trans. on Info. Theory*, 46(4):1339–1349, 2000.
9. D. Boneh, S. Halevi, and N.A. Howgrave-Graham. The Modular Inversion Hidden Number Problem. In *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 36–51, Berlin, 2001. Springer-Verlag.

10. D. Catalano, R. Gennaro, N. Howgrave-Graham, and P. Nguyen. Paillier's Cryptosystem Revisited. In *Proc. CCS '01*, New York, November 2001. ACM.
11. D. Coppersmith. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *J. of Cryptology*, 10:233–260, 1997.
12. D. Coppersmith. Finding Small Solutions to Low Degree Polynomials. In *CALC '01*, volume 2146 of *LNCS*, pages 20–31, Berlin, 2001. Springer-Verlag.
13. W. Dai. *Crypto++ 5.2.1 Benchmarks*, 2006. `http://www.eskimo.com/~weidai/benchmarks.html`.
14. R. Fischlin and C.P. Schnorr. Stronger Security Proofs for RSA and Rabin Bits. *Journal of Cryptology*, 13:221–244, 2000.
15. R. Gennaro. An Improved Pseudo-Random Generator Based on the Discrete-Logarithm Problem. *Journal of Cryptology*, 18:91–110, 2005.
16. O. Goldreich. *Foundations of Cryptography, Volume I.* Cambridge University Press, Cambridge, 2003.
17. O. Goldreich and V. Rosen. On the Security of Modular Exponentiation with Application to the Construction of Pseudorandom Generators. *J. of Cryptology*, 16:71–93, 2003.
18. S. Goldwasser and S. Micali. Probabilistic Encryption. *J. of Computer and System Sciences*, 28(2):270–299, 1984.
19. S. Goldwasser, S. Micali, and P. Tong. Why and How to Establish a Private Code on a Public Network. In *Proc. FOCS '82*, pages 134–144. IEEE Computer Society Press, 1982.
20. N. Howgrave-Graham. Finding Small Roots of Univariate Polynomials Revisited. In *Cryptography and Coding*, volume 1355 of *LNCS*, pages 131–142, Berlin, 1997. Springer-Verlag.
21. R. Impagliazzo and M. Naor. Efficient Cryptographic Schemes Provably as Secure as Subset Sum. *Journal of Cryptology*, 9:199–216, 1996.
22. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 261:515–534, 1982.
23. A.K. Lenstra and E.R. Verheul. Selecting Cryptographic Key Sizes. *J. of Cryptology*, 14:255–293, 2001.
24. S. Micali and C.P. Schnorr. Efficient, Perfect Polynomial Random Number Generators. *J. of Cryptology*, 3:157–172, 1991.
25. P. Q. Nguyen and I. E. Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *J. Cryptology*, 15:151–176, 2002.
26. P. Q. Nguyen and J. Stern. The Two Faces of Lattices in Cryptology. In *Cryptography and Lattices*, volume 2146 of *LNCS*, pages 146–180, Berlin, 2001. Springer-Verlag.
27. S. Patel and G. Sundaram. An Efficient Discrete Log Pseudo Random Generator. In *CRYPTO '98*, volume 1462 of *LNCS*, pages 304–317, Berlin, 1998. Springer-Verlag.
28. A. Sidorenko and B. Schoenmakers. Concrete Security of the Blum-Blum-Shub Pseudorandom Generator. In *Cryptography and Coding 2005*, volume 3796 of *LNCS*, pages 355–375, Berlin, 2005. Springer-Verlag.
29. R. Steinfeld, J. Pieprzyk, and H. Wang. On the Provable Security of an Efficient RSA-Based Pseudorandom Generator. Cryptology ePrint Archive, Report 2006/206, 2006. `http://eprint.iacr.org/2006/206`.
30. U.V. Vazirani and V.V. Vazirani. Efficient and Secure Pseudo-Random Number Generation. In *Proc. FOCS '84*, pages 458–463. IEEE Computer Society Press, 1982.