The Equivalence of Strong RSA and Factoring in the Generic Ring Model of Computation

Divesh Aggarwal, Ueli Maurer, Igor Shparlinski

The Equivalence of Strong RSA and Factoring in the Generic Ring Model of Computation*

Divesh Aggarwal[1], Ueli Maurer[1], and Igor Shparlinski[2]

[1] Department of Computer Science, ETH Zurich, Switzerland.
{divesha,maurer}@inf.ethz.ch
[2] Department of Computing, Macquarie University, NSW, Australia.
igor@comp.mq.edu.au

Abstract. Let N be the result of an RSA modulus generation, i.e., a random variable distributed according to some appropriate distribution over the set of products of two primes, such that factoring N is believed to be hard. The Strong RSA assumption states that, given an x chosen uniformly at random from $\mathbb{Z}_N$, it is computationally infeasible to compute a $y \in \mathbb{Z}_N$ and an $e \in \mathbb{N} \setminus \{1\}$ such that $y^e \equiv x \pmod{N}$. This assumption is important in cryptography and has been used to construct several cryptosystems.
Due to the lack of complexity-theoretic lower bound proofs for cryptographic problems in a general model of computation, it is a common practice in cryptography to give proofs of computational security in meaningful restricted models of computation.
Some examples of restricted models that are interesting in cryptography are the generic group model for proving lower bounds for the discrete logarithm problem and related problems, and the random oracle model for proving the soundness of protocols or hash function constructions. A generic model captures that an algorithm does not exploit the bit representation of the elements other than for testing equality. The security of the RSA public-key cryptosystem can be analyzed in the generic ring model.
In this paper, we prove that for almost all possible distributions of N, the problem of factoring N can be efficiently reduced to solving the Strong RSA problem on $\mathbb{Z}_N$ in the generic ring model of computation, where an algorithm can perform ring operations, inverse ring operations, and test equality.

1 Introduction

1.1 The Strong RSA Assumption

The RSA [12] public key cryptosystem is probably the most widely used public key cryptosystem. Its security relies on the assumption that, given an RSA public

* This version is an extended abstract of the paper.

key (n, e) and an $x \in \mathbb{Z}_n$, it is hard to compute a $y \in \mathbb{Z}_n$ such that $y^e \equiv x \pmod{n}$.

The *Strong RSA assumption* was introduced by Baric and Pfitzmann [2]. Cramer and Shoup [5] gave a signature scheme based on this assumption. This assumption differs from the RSA assumption in that it states that it is hard to compute e-th roots even if the adversary is allowed to choose e. Clearly, the RSA assumption is potentially a weaker assumption. A still weaker assumption is that factoring the modulus n is hard.

Before stating these assumptions, we define a non-negligible function. A function $f : \mathbb{N} \mapsto \mathbb{R}$ is called *non-negligible* if there exists $c > 0$ and $k_0 \in \mathbb{N}$ such that for all $k > k_0$, we have $|f(k)| > k^{-c}$.

We consider a random variable N that is chosen according to a certain probability distribution over a set of products of two primes. The set of primes for which the result of this paper holds will be discussed in Section 2.1. The *Strong RSA problem* is the problem of computing, given N and an element x chosen uniformly at random from $\mathbb{Z}_N$, $y \in \mathbb{Z}_N$ and $e \in \mathbb{N} \setminus \{1\}$ such that $y^e \equiv x \pmod{N}$. The assumptions are defined with respect to N and are formally stated below:

Strong RSA Assumption for N: There exists no probabilistic polynomial-time (PPT) algorithm that solves the Strong RSA problem for N with non-negligible probability.

RSA Assumption for N and e: There exists no PPT algorithm that, given N, an element x chosen uniformly at random from $\mathbb{Z}_N$, and an integer $e \in \mathbb{N} \setminus \{1\}$, computes $y \in \mathbb{Z}_N$ such that $y^e \equiv x \pmod{N}$ with non-negligible probability.

Factoring Assumption for N: There exists no PPT algorithm that, given N, finds a non-trivial factor of N with non-negligible probability.

It is easy to see that if the Strong RSA assumption holds then the factoring assumption holds. However, it is not known whether the converse is true. In this paper, we show that this converse holds with respect to a restricted class of algorithms called generic algorithms. Our result thus subsumes the result of [1] and proves a more general result.

Generic algorithms are algorithms that do not exploit the representation of the elements. This class of algorithms was introduced by Shoup [13], based on the work of Nechaev [10]. They proved lower bounds on the complexity of computing discrete logarithms and some related problems in cyclic groups in the context of generic algorithms. Maurer [9] provided a simpler and more general model for analyzing representation-independent algorithms.

1.2 The Generic Ring Model of Computation

We give a brief description of the model of [9]. This is almost the same as the description used in [1] and is sufficient for our purpose. The model is characterized by a black-box **B** which can store values from a certain set T in internal

state variables $V_0, V_1, V_2, \cdots$. The initial state (the input of the problem to be solved) consists of the values of $[V_0, \ldots, V_\ell]$ for some positive integer ℓ, which are set according to some probability distribution over $T^{\ell+1}$.

The black box $\mathbf{B}$ allows two types of operations:

- *Computation operations.* For a set Π of operations on T, a computation operation consists of selecting $f \in \Pi$ (say t-ary) as well as the indices $i_1, \ldots, i_{t+1}$ of $t+1$ state variables. The black-box $\mathbf{B}$ computes $f(V_{i_1}, \ldots, V_{i_t})$ and stores the result in $V_{i_{t+1}}$.
- *Relation Queries.* For a set Σ of relations on T, a query consists of selecting a relation $\rho \in \Sigma$ (say t-ary) as well as the indices $i_1, \ldots, i_t$ of t state variables. The query is replied by the binary output $\rho(V_{i_1}, \ldots, V_{i_t})$ that takes the value 1 if the relation is satisfied, and 0 otherwise.

An algorithm in this model is characterized by its interactions with the black box $\mathbf{B}$. The algorithm inputs operations (computation operations and relation queries) to the black box, and the replies to the relation queries are returned to the algorithm. The complexity of an algorithm for solving any problem can be measured by the number of operations it performs on $\mathbf{B}$.

For this paper, the set T is $\mathbb{Z}_N$. Also $\ell = 1$ and V_0 is always set to be the unit element 1 of $\mathbb{Z}_N$ and V_1 is the value x that is the input to the algorithm. Also, N is assumed to be explicitly given to the algorithm. A *generic ring algorithm (GRA)* is an algorithm that is allowed only ring operations and equality queries, i.e., $\Pi = \{+, -, \cdot, /\}$ and $\Sigma = \{=\}$.

Note that division by non-invertible elements of $\mathbb{Z}_N$ is not defined. This can be modeled in the above generic ring model by having the black-box $\mathbf{B}$ send an "exception" bit b to the algorithm and leaving the corresponding state variable undefined whenever there is a division by a non-invertible element. For more details on this, refer to [1].

1.3 The Strong RSA Problem in the Generic Ring Model of Computation

The difficulty in formulating the Strong RSA problem in the generic ring model is that an algorithm for solving this problem should output e and y where e is not an element of the ring $\mathbb{Z}_N$ and hence cannot be computed in the black-box. If we model the problem such that the algorithm outputs e explicitly, then e does not depend on the input except possibly on the result of equality queries. The natural solution to this problem is to model e as an element of the ring $\mathbb{Z}_N$ which is interpreted as an integer in $\{0, \ldots, N-1\}$. In particular, for $u, v \in \mathbb{Z}_N$, the element $u^v \in \mathbb{Z}_N$ is defined as an element of $\mathbb{Z}_N$ obtained from raising u to the power corresponding to the integer representing v modulo N. The Strong RSA problem in our model can thus be stated as the problem of computing (y, e) such that $y^e \equiv x \pmod{N}$ when $V_0 = 1, V_1 = x$ and $\Pi = \{+, -, \cdot, /\}$ and $\Sigma = \{=\}$.

1.4 Comparison With the Model of Damgård and Koprowski

The only result that looks at the Strong RSA problem with respect to generic algorithms is due to Damgård and Koprowski [6]. In [6], the authors show that no generic group algorithm(GGA) can, given an x chosen uniformly at random from a certain group G, efficiently compute $y \in G$ and $e \in \mathbb{Z}$ such that $y^e = x$ for a reasonable choice of the group G, where GGAs are algorithms that allow only the multiplication operation, which in our model would correspond to $\Pi = \{\cdot\}$ and $\Sigma = \{=\}$.

Their result is very restricted in the sense that they assume that e is computed outside the black-box and hence does not depend on the input except possibly on the result of equality queries. Thus, their result is not much more general than proving the hardness of the RSA problem in this model (that is, when e is fixed). In contrast, in our model, e can be computed in the black-box as an element of $\mathbb{Z}_N$ using a GRA which is given x as input.

In [6], they consider only GGAs while we prove the equivalence for GRAs. Since both the ring operations, multiplication and addition modulo N, are easy to perform on the ring $\mathbb{Z}_N$, it is more interesting to study the complexity of Strong RSA problem for GRAs rather than for GGAs.

1.5 Other Related Work

In addition to the result [6] mentioned in Section 1.4, there have been some other results studying the RSA and related problems with respect to generic algorithms.

Boneh and Venkatesan [3] showed that any straight-line program that factors N by making at most a logarithmic number of queries to an oracle solving the Low Exponent RSA (LE-RSA) problem, which is the RSA problem when the public exponent e is small, can be converted into a PPT algorithm for factoring N. This result is, however, of limited interest because the complexity of the factoring algorithm that does not use the oracle increases exponentially with the number of oracle queries made by the algorithm that uses the oracle.

Brown [4] showed that if factoring is hard then the LE-RSA problem is intractable for straight-line programs (SLPs), i.e., where $\Pi = \{+, -, \cdot, /\}$. More precisely, he proved that an efficient SLP for breaking LE-RSA can be transformed into an efficient factoring algorithm. Leander and Rupp [8] generalized the result of [4] to GRAs which, as explained earlier, can test for equality of elements in addition to the computation operations. Aggarwal and Maurer [1] further generalized these results to show that any GRA for computing e-th roots (for any e) can be transformed into an efficient factoring algorithm.

2 Preliminaries

2.1 Notation

Let n be some value taken by the random variable N.

By $\mathbb{Z}_n[X]$, we denote the ring of polynomials in x with coefficients in the ring $\mathbb{Z}_n$ of integers modulo n.

Let $\tau(k)$ denote the number of positive integer divisors of an integer $k \geq 2$. For $\eta > 0$, we define $\mathcal{L}_\eta$ as the set of primes ℓ with $\tau(\ell - 1) \leq (\log \ell)^{1+\eta}$.

Let $\pi^*_\eta(L) = \#(\mathcal{L}_\eta \cap [1, L])$ and let $\pi(L)$ be the number of primes $\ell \leq L$. The following result states that for any constant $\eta > 0$, $\pi^*_\eta(L)$ is almost as large as $\pi(L)$, i.e., $\mathcal{L}_\eta$ contains almost all primes. The proof is deferred to the full version.

Lemma 1. *For any fixed $\eta > 0$, we have*

$$\pi^*_\eta(L) = \pi(L)\left(1 - O\left((\log L)^{-\eta}\right)\right).$$

Thus, for any $\eta > 0$, the set $\mathcal{L}_\eta$ contains almost all sufficiently large primes that are used for RSA. In particular, it includes all *safe primes*, i.e., primes ℓ such that $(\ell - 1)/2$ is also a prime which are the primes most often used for RSA.

Let n be the product of primes p and q. Without loss of generality, we can assume that $q < p$. In this paper, we assume κ to be the security parameter and the term non-negligible is with respect to κ. We define a set $\mathcal{N}_\eta$ as follows:

$$\mathcal{N}_\eta = \{n = p \cdot q \ : \ p, q \text{ are primes}, 2^\kappa \leq n < 2^{\kappa+1}, \ \ q < p, \text{ and } p, q \in \mathcal{L}_\eta\}\ .$$

The set of RSA moduli considered in this paper are elements of the set $\mathcal{N}_\eta$.

2.2 Definitions: SLPs and GRAs

In this section, we define two classes of algorithms, namely Straight Line Programs (SLPs) and Generic Ring Algorithms(GRAs). These definitions and the corresponding interpretation is taken from [1] and the reader should refer to [1] for a more detailed description.

An SLP corresponds to $\Pi = \{+, -, \cdot, /\}$ and $\Sigma = \{\}$ in the model of [9] and is defined as follows:

Definition 1. An L-step *straight-line program (SLP)* S is a sequence of $L - 1$ triples $(i_2, j_2, \circ_2)$, $\ldots$, $(i_L, j_L, \circ_L)$ such that $0 \leq i_k, j_k < k$ and $\circ_k \in \{+, -, \cdot, /\}$.

The interpretation of an SLP is that if x is considered a variable, then the SLP S computes a sequence of rational functions $f_2, \ldots, f_L$, where, $f_0 = 1$, $f_1 = x$ and $f_k = f_{i_k} \circ_k f_{j_k}$ for $2 \leq k \leq L$. The rational functions f_i can be interpreted as a pair of polynomials (a_i, b_i) representing the numerator and the denominator of f_i.

A (randomized) GRA is a generalization of an SLP and is defined as follows:

Definition 2. An L-step *(randomized) GRA* G is an algorithm that, in the kth step, for $2 \leq k \leq L$, outputs an operation of the form $(i_k, j_k, \circ_k)$, where $0 \leq i_k, j_k < k$ and $\circ_k \in \{+, -, \cdot, /, =\}$ and if $\circ_k$ is $=$, it takes an input bit (as the result of the equality query).[3]

3 Generic Strong RSA vs Factoring

3.1 The Statement of the Main Result

As mentioned earlier, in this paper we restrict our attention to the case where the adversary is only allowed to use a GRA to solve the Strong RSA problem. We refer to the Strong RSA assumption in this case as the Generic Strong RSA assumption for the random variable N:

Generic Strong RSA Assumption for N: For any L bounded by a polynomial in the security parameter, there exists no L-step randomized GRA thatsolves the Strong RSA problem for N with non-negligible probability.

Theorem 1. *Let $\eta > 0$ be fixed. Let N be a random variable taking values in the set $\mathcal{N}_\eta$. If the Factoring Assumption for N holds, then the Generic Strong RSA Assumption for N holds.*

It should be noted that this theorem is proved for any distribution of N over the set $\mathcal{N}_\eta$. Also, n is a value taken by the random variable N.

3.2 The Proof for Straight-Line Programs

Here we obtain a version of Theorem 1 in the case when a randomized GRA is replaced by an SLP in the definition of the Generic Strong RSA Assumption.

The following result is a slight generalization of [1, Lemma 6] and we need this to prove the result for straight line programs.

Lemma 2. *For all $\mu > 0$ and $L \in \mathbb{N}$, there exists an algorithm that takes as input n and, for every L-step SLP S that computes the rational function $f(x) \in \mathbb{Z}_n[x]$ not identically 0 such that $\Pr_{x \in_R \mathbb{Z}_n}(f(x) \equiv 0 \pmod{n}) \geq \mu$, returns a factor of n in expected time $O(L^4/\mu)$.*

This proof is almost the same as the proof of [1, Lemma 6].

Now, we prove the main technical result of this section. It shows that any SLP that, given x, computes a y and an e such that $y^e \equiv x \pmod{n}$ for some $e > 1$, can be used to factor n.

[3] The choice of the operation at each step may be randomized.

Lemma 3. *For all $\eta > 0$, $\alpha > 0$ and $L \in \mathbb{N}$, there exists an algorithm that takes as input n and an L-step SLP S, and satisfies the following property. If S computes the rational functions $P_1(x)/Q_1(x)$ and $P_2(x)/Q_2(x)$ such that the cardinality of the set* [4]

$$\left\{ x \in \mathbb{Z}_n \ : \ \left(\frac{P_1(x)}{Q_1(x)}\right)^{P_2(x)/Q_2(x)} \equiv x \pmod{n} \ \wedge \ P_2(x) \not\equiv Q_2(x) \pmod{n} \right\}$$

is at least $3n(\log n)^{-\alpha}$, then the algorithm returns a factor of n in expected time $O\left(\log^{2\alpha+2\eta+3} n \left(L^4 + \log^4 n\right)\right)$.

Proof. We define two events E_1 and E_2.

- Let E_1 be the event that there exists $c \in \mathbb{Z}_n$ $(c \neq 1)$ such that

$$\#\left\{ x \in \mathbb{Z}_n \ : \ \frac{P_2(x)}{Q_2(x)} - c \equiv 0 \pmod{n} \right\} \geq \frac{n}{(\log n)^{\alpha+1+\eta}} \tag{1}$$

- Let E_2 be the event that there exists $c \in \mathbb{Z}_n$ $(c \neq 1)$ such that

$$\#\left\{ x \in \mathbb{Z}_n \ : \ \frac{P_2(x)}{Q_2(x)} - c \equiv 0 \pmod{q} \right\} \geq \frac{2n}{(\log n)^{\alpha+1+\eta}} \tag{2}$$

For an event E, we use $\overline{E}$ to denote the complementary event of E.

We split the proof into three cases that are mutually exhaustive. In the first two cases, we give an algorithm for factoring n, and in the third case, we show that the inequality in the statement of Lemma 3 is not satisfied.

CASE 1: E_1 holds.
In this case, $P_2(x)/Q_2(x)$ is a constant c with non-negligible probability, i.e., $P_2(x) - cQ_2(x) \equiv_n 0$ with non-negligible probability. Thus, if $P_2(x) - cQ_2(x)$ is not identically zero, then, using Lemma 2, we can obtain a factor of n. If $P_2(x) - cQ_2(x)$ is identically zero, then $P_1(x)^c - xQ_1(x)^c \equiv_n 0$ with non-negligible probability and thus, we can use Lemma 2 for the polynomial $P_1(x)^c - xQ_1(x)^c$, thus obtaining a factor of n.

CASE 2: The events $\overline{E_1}$ and E_2 hold.
In this case, as can be seen from the definitions of E_1 and E_2, the expression $P_2(X)/Q_2(X)$ behaves differently modulo p and modulo q for a random X. Thus, it can be shown that $\gcd\left(P_2(X_1)/Q_2(X_1) - P_2(X_2)/Q_2(X_2),\ n\right)$ for randomly chosen X_1, X_2 returns a factor of n with non-negligible probability. The details of the proof of CASE 1 and CASE 2 is deferred to the full version of the paper.

[4] Note that $P_2(x)/Q_2(x)$ is understood as being computed modulo n.

CASE 3: The event $\overline{E_2}$ holds.

In this case, we bound the cardinality of the set

$$\left\{x \in \mathbb{Z}_n \ : \ \left(\frac{P_1(x)}{Q_1(x)}\right)^{P_2(x)/Q_2(x)} \equiv x \mod n, \ P_2(x) \not\equiv Q_2(x) \mod n\right\}.$$

Let us consider the natural homomorphism from $\mathbb{Z}_n$ to $\mathbb{Z}_q$ and define

$$R = \max_{c \in Z_q} \# \left\{x \in \mathbb{Z}_q \ : \ \frac{P_2(x)}{Q_2(x)} \equiv c \pmod q\right\} \ \leq \ 2q(\log n)^{-(\alpha+1+\eta)} \ , \quad (3)$$

where the inequality follows from the fact that $\overline{E_2}$ holds.

Each $x = 0, \ldots, n-1$ can be uniquely written as $m + pk$ with $m = 0, \ldots, p-1$ and $k = 0, \ldots, q-1$. We also note that for any polynomial P, $P(m+pk) \equiv P(m) \pmod p$. Thus, the congruence $\left(\frac{P_1(x)}{Q_1(x)}\right)^{P_2(x)/Q_2(x)} \equiv x \pmod n$, taken modulo p is

$$\left(\frac{P_1(m)}{Q_1(m)}\right)^{\frac{P_2(m+pk)}{Q_2(m+pk)}} \equiv m \pmod p, \qquad 0 \leq m \leq p-1, \ \ 0 \leq k \leq q-1. \quad (4)$$

The number of solutions with $m = 0$ is at most q. Now, we bound the number of solutions if $m \neq 0$. In this case, for (4) to hold, $P_1(m) \not\equiv 0$ and $Q_1(m) \not\equiv 0 \pmod p$. Hence, in order to give an upper bound on the cardinality of $\mathcal{X}$, we also assume that $P_1(m)$ and $Q_1(m)$ are not 0 modulo p.

We now fix a primitive root g modulo p. Then, for all m, there exist u_m and v_m such that

$$m \equiv g^{u_m} \pmod p, \ \ \frac{P_1(m)}{Q_1(m)} \equiv g^{v_m} \pmod p, \qquad 0 \leq u_m, v_m \leq p-2 \ .$$

From (4), we get that

$$v_m \cdot \frac{P_2(m+pk)}{Q_2(m+pk)} \equiv u_m \pmod{p-1}. \quad (5)$$

Let $\quad t_m = \gcd(p-1, v_m, u_m) \quad$ and $\quad s_m = \frac{p-1}{t_m}$.

Then, (5) is equivalent to

$$\frac{v_m}{t_m} \cdot \frac{P_2(m+pk)}{Q_2(m+pk)} \equiv \frac{u_m}{t_m} \pmod{s_m} \ . \quad (6)$$

Clearly, if $\gcd(v_m/t_m, s_m) > 1$, then the above congruence has no solutions. Therefore, $\gcd(v_m/t_m, s_m) = 1$ must hold. Then, by multiplying (6) by $(v_m/t_m)^{-1}$ modulo s_m, it leads to a congruence of the type

$$\frac{P_2(m+pk)}{Q_2(m+pk)} \equiv w_m \pmod{s_m}, \quad (7)$$

with some integer w_m depending only on m and not on k. Thus $P_2(m+pk)/Q_2(m+pk)$ can take at most one value modulo s_m for a fixed m. Also, $P_2(m+pk)/Q_2(m+pk)$ takes a unique value modulo p for a fixed m. Thus, by the Chinese Remainder Theorem, $P_2(m+pk)/Q_2(m+pk)$ takes at most one value modulo ps_m. This implies (using $q < p$) that, for a fixed m, there are at most $\frac{pq}{ps_m} = \frac{qt_m}{p-1} \le t_m$ possible values for $P_2(m+pk)/Q_2(m+pk)$. Note that $m+pk$ is uniformly random in $\mathbb{Z}_q$ if m is fixed and k is chosen uniformly at random in $\mathbb{Z}_q$. So, from (5), $P_2(m+pk)/Q_2(m+pk)$ takes any of these possible values for at most R possible values of k.
Since $t_m \mid \gcd(u_m, p-1)$, $t_m = t$ for at most $(p-1)/t$ values of m.
Collecting together the values m such that $t_m = t$ for all $t \mid p-1$ and counting q possible values of x for $m = 0$, we obtain that

$$\begin{aligned}
\#\mathcal{X} &\le q + \sum_{t|p-1} \frac{p-1}{t} \cdot tR \;=\; q + (p-1)R \sum_{t|p-1} 1 \\
&= q + (p-1)R\tau(p-1) \;\le\; q + 2(p-1)q(\log n)^{-(\alpha+1+\eta)}(\log n)^{1+\eta} \\
&< q + 2n(\log n)^{-\alpha} < 3n(\log n)^{-\alpha} ,
\end{aligned}$$

where the third last inequality follows from the fact that $p \in \mathcal{L}_\eta$ and using (3), and the last inequality follows from the fact that $q < \sqrt{n}$ (since $p > q$) and $\sqrt{n}$ is larger than $(\log n)^\alpha$ for n large enough. However, this contradicts our assumption.

□

3.3 Generalizing the Proof from SLPs to GRAs

In this section we state how to obtain a factoring algorithm that, given access to a GRA that solves the Strong RSA problem in $\mathbb{Z}_n$, outputs either a factor of n or an SLP that solves the Strong RSA problem in $\mathbb{Z}_n$ with almost the same success probability.

Definition 3. For a GRA $\mathcal{G}$, let $\lambda_n(\mathcal{G})$ denote the probability that $\mathcal{G}$, when run on an x chosen uniformly at random from $\mathbb{Z}_n$, computes a pair y, e such that $y^e \equiv x \pmod{n}$.

Note that $\lambda_n(S)$ for an SLP S is implicitly defined by the above definition, since an SLP is a special type of a GRA.

Lemma 4. *For all $\varepsilon > 0$ and $L \in \mathbb{N}$, there exists an algorithm of time complexity $O((L/\varepsilon)^{5/2})$ that, given access to an L-step randomized GRA $\mathcal{G}$ such that $\lambda_n(\mathcal{G}) = \mu$, with probability $\mu/2 - \varepsilon$, either outputs a factor of n or an L-step SLP S such that $\lambda_n(S) \ge \mu/2$.*

The proof of Lemma 4 is the same as the proof of [1, Lemma 8]. Note that the statement of Lemma 4 is different from [1] because of different definitions of $\lambda_n(\mathcal{G})$, but this does not change anything in the proof.

Combining Lemma 4 and Lemma 3, we can obtain the proof of Theorem 1. The details of this proof can be found in the full version of the paper.

References

1. D. Aggarwal and U. Maurer. Breaking RSA generically is equivalent to factoring. In *EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 36-53.
2. N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 480-494.
3. D. Boneh and R. Venkatesan. Breaking RSA may be easier than factoring. In *EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 59-71.
4. D. R. L. Brown. Breaking RSA may be as difficult as factoring. In *Cryptology ePrint Archive*, Report 205/380, 2006.
5. R. Cramer and V. Shoup. Signature schemes based on the Strong RSA assumption. In *6th ACM Conference on Computer and Communications Security*, pages 46-52. Nov 1999.
6. I. Damgård and M. Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In *EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 256-271.
7. T. Jager and J. Schwenk. On the analysis of cryptographic assumptions in the generic ring model. In *ASIACRYPT 2009* volume 5912 of *Lecture Notes in Computer Science*, pages 399-416.
8. G. Leander and A. Rupp. On the equivalence of RSA and factoring regarding generic ring algorithms. In *ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 241-251.
9. U. Maurer. Abstract models of computation in cryptography. In *Cryptography and Coding 2005*, volume 3796 of *Lecture Notes in Computer Science*, pages 1-12.
10. V. I. Nechaev. Complexity of a deterministic algorithm for the discrete logarithm. In *Mathematical Notes*, volume 55, no. 2, pages 91-101, 1994.
11. K. Prachar. *Primzahlverteilung*, Springer-Verlag, Berlin, 1957.
12. R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. In *Communications of the ACM*, volume 21, pages 120-126, 1978.
13. V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 256-266.