Certifying RSA

Saqib A. Kakvi, Eike Kiltz, and Alexander May

Faculty of Mathematics
Horst Görtz Institute for IT-Security
Ruhr University Bochum, Germany
{saqib.kakvi,eike.kiltz,alex.may}@rub.de

Abstract. We propose an algorithm that, given an arbitrary N of unknown factorization and prime $e \geq N^{\frac{1}{4}+\varepsilon}$, certifies whether the RSA function $\mathsf{RSA}_{N,e}(x) := x^e \bmod N$ defines a permutation over $\mathbb{Z}_N^*$ or not. The algorithm uses Coppersmith's method to find small solutions of polynomial equations and runs in time $O(\varepsilon^{-8} \log^2 N)$. Previous certification techniques required $e > N$.

1 Introduction

One of the most well known cryptographic primitives is the RSA function [25]. Given a public modulus N (which is usually the product of two primes) and an exponent e, it is defined as $\mathsf{RSA}_{N,e} : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$, $x \mapsto x^e \bmod N$. It is well known that the RSA function defines a permutation over the domain $\mathbb{Z}_N^*$ iff $\gcd(e, \varphi(N)) = 1$. Furthermore, with the right choice of parameters, the RSA function even defines a *trapdoor* permutation since the prime factorization of N allows to efficiently invert $\mathsf{RSA}_{N,e}$.

Trapdoor permutations have many applications to public-key cryptosystems and serve as a building block for (often quite complex) cryptographic protocols. In a large number of applications of trapdoor functions, the fact that the function is a permutation is required to be publicly verifiable. The importance of trapdoor permutations with an efficient permutation checking procedure was first noted by Bellare and Yung [2, 3], who called them *certified trapdoor permutations.* Certified trapdoor permutations are in particular important in scenarios where one party (for example, the prover) sends a description of a trapdoor permutation to another party (for example, the verifier). A dishonest prover may send a malicious description of a trapdoor function which is not a permutation. If this remains unnoticed by the verifier, it may allow the prover to cheat in the protocol. See Section 1.2 for a list of applications of certified trapdoor permutations.

RSA as a certified trapdoor permutation. The question whether the RSA function is a certified trapdoor permutation was first addressed by Bellare and Yung who wrote in [2, 3]:

In particular, RSA is (probably) not certified [...]. This is because [...] the (description of) the trapdoor permutation f includes a number which is a product of two primes, and there is (probably) no polynomial time procedure to test whether or not a number is a product of two primes.

To overcome this problem, Bellare and Yung showed that every trapdoor permutation can be *transformed* into a certified trapdoor permutation by presenting pre-images (under the function) of random elements specified in a common reference string (CRS), hence certifying that the function is a permutation. While this result is certainly interesting at a theoretical level, the Bellare-Yung transformation has two main disadvantages. First, it comes with an additional computational overhead (consisting of a number of evaluations of the function) and is therefore relatively inefficient. Second, in order to keep the same data structures one would rather prefer that the initial trapdoor function (e.g., RSA) can be certified directly, without any additional overhead such as a CRS or pre-images. Related transformations for RSA were proposed in [14, 6, 7].

Subsequently, two results were obtained about the direct certifiability of RSA, i.e., without using a CRS and expanding the public description. First, [5, 20] observed that if $e > N$ and e is prime, then the RSA function $\mathsf{RSA}_{N,e}$ is a certified permutation. (This is, since if e is a prime, then it can never divide $\varphi(N) < N$ and hence $\gcd(e, \varphi(N)) = 1$.) However, choosing $e > N$ is usually avoided in practice due to the costs for modular exponentiation. Second, Kiltz et al. [19] noted that if $e < N^{1/4}$, then $\mathsf{RSA}_{N,e}$ is a *lossy trapdoor permutation* [24] (under the phi-Hiding Assumption [5]) and hence it cannot be certified. This is because a lossy trapdoor permutation is the opposite of a certified trapdoor permutation: a honestly generated (N, e) with $N = pq$ and $\gcd(e, \varphi(N)) = 1$ cannot be efficiently distinguished from (N, e) for which $\mathsf{RSA}_{N,e}$ is many-to-1 and hence not a permutation.

To summarize, if $e < N^{1/4}$, then the RSA function is lossy and cannot be certified (unless the phi-hiding assumption is wrong); if $e > N$, then it is certified [5, 20]; if $N^{1/4} < e < N$, nothing is known and therefore generic NIZK proofs [3] have to be added to certify RSA.

1.1 Our Results

In this work we close the above gap by showing an efficient certification procedure that works for any prime exponent $e > N^{1/4}$. Concretely, we construct an algorithm that, given an arbitrary modulus N (with unknown factorization) and a prime $e \geq N^{1/4+\varepsilon}$, returns 1 iff $\mathsf{RSA}_{N,e}$ defines a permutation over $\mathbb{Z}_N^*$. The running time of the algorithm is $O(\varepsilon^{-8} \log^2(N))$ bit operations plus additional $O(\log^4 N)$ if e needs to be checked for primality.

Our Certification Algorithm. The idea of our new certification algorithm is as follows. The $\mathsf{RSA}_{N,e}$ function defines a permutation over $\mathbb{Z}_N^*$ iff e does not divide $\varphi(N)$. Hence given N, e, our goal is to identify if $\gcd(e, \varphi(N)) = 1$ or not. First, we use Coppersmith's algorithm [8, 21] to find prime divisors p of N in a specific range. Concretely, our algorithm $\mathsf{FindFactor}$ run with parameter β successfully identifies if a given prime $e > N^{1/4+\varepsilon}$ divides $p - 1$ iff there exists a divisor p of N in the range $[N^{\beta}, N^{\beta^2+1/4+\epsilon}]$. If we could assume that $N = pq$ is the product of two primes, both of size roughly $N^{1/2}$, then we could run $\mathsf{FindFactor}$ with parameter $\beta = 1/2$ to identify whether e divides $\varphi(N)$ or not.

However, the certification algorithm has to view N as an arbitrary integer with unknown factorization. If $N = pq$ with $p \approx N^{2/3}$ and $q \approx N^{1/3}$, then $\mathsf{FindFactor}$ run with parameter $\beta = 1/2$ does not work any more. To get around this, we run the $\mathsf{FindFactor}$ algorithm multiple times (with different parameters β) to check for various ranges of the prime factors of N. Our main technical contribution is to show that the number of invocations of $\mathsf{FindFactor}$ in our certification algorithm is $poly(\varepsilon)$ if $e \geq N^{1/4+\varepsilon}$.

Extensions. Our certification algorithm works only for prime e but it can be extended to the case where the factorization of $e = \prod e_i^{z_i}$ is known. In that case we can give an efficient certification procedure if $e_i \geq N^{1/4+\varepsilon}$, for all i. If, for one i, we have $e_i < N^{1/4}$, then $\mathsf{RSA}_{N,e}$ is (at least) e_i-to-1 (lossy) under the phi-hiding assumption. Extending our methods to work with arbitrary integers e of unknown factorization remains an open problem.

1.2 Certified Trapdoor Permutations and Applications

The only known candidate trapdoor permutations are the (factoring-based) Blum-Blum-Shub permutation [4], the RSA permutation [25], and Paillier [23]. Since the Blum-Blum-Shub function is lossy assuming one cannot distinguish $N = pq$ from $N = pqr$ [22, 12], the RSA trapdoor function is the most efficient certified trapdoor permutation currently known. Our results show that one can use RSA with prime $e = N^{1/4+\varepsilon}$ (rather than $e > N$) as a certified trapdoor permutation.

We now mention a number of cryptographic protocols that are using certified (rather than standard) trapdoor permutations as a building block. Most importantly, NIZK protocols for any NP-statement can be built from (doubly-enhanced) certified trapdoor permutations [11, 17, 15, 16]. Since the RSA trapdoor permutation is doubly-enhanced [17] we obtain simplified and more efficient NIZK protocols from the RSA assumption (with $e > N^{1/4}$), that do not suffer from the Bellare-Yung certification overhead. Apart from that, [10] used certified trapdoor permutations to construct ZAPS and verifiable PRFs; [13] to construct round-optimal blind signatures; [20, 1] to build sequential aggregate signatures. We stress that requiring the trapdoor permutation to be certified is not only an artifact of the security proofs. In almost all cases the use of a *lossy trapdoor permutation* leads to a concrete attack on the scheme. For example, the security of the RSA-based aggregate signatures scheme of [20] can be broken (assuming the Phi-Hiding Assumption) when instantiated with $e < N^{1/4}$ (e.g., using the common choices $e = 3$ or $e = 2^{16} + 1$). The same holds for the NIZK protocols for any NP statement [17]. Recently, [18] showed that a full-domain hash impossibility result by Coron [9] only holds if the trapdoor function is certified.

2 Definitions

2.1 Notation

We denote our security parameter as k. For all $n \in \mathbb{N}$, we denote by 1^n the n-bit string of all ones. For any element x in a set S, we use $x \in_R S$ to indicate that we choose x uniformly at random from S. We denote the set of prime numbers by $\mathbb{P}$ and the set of n-bit prime numbers by $\mathbb{P}_n$. We denote by $\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N : \gcd(x, N) = 1\}$ the multiplicative group modulo an integer N. All logarithms are base 2 unless otherwise stated.

2.2 Families of Permutations

Definition 1. *A family of permutations* $\mathsf{P} = (\mathsf{Gen}, \mathsf{Eval})$ *consists of the following two polynomial-time algorithms.*

1. *A probabilistic algorithm* Gen*, which on input* 1^k *outputs a public description pub which includes an efficiently sampleable domain* Dom_{pub}*.*
2. *A deterministic algorithm* Eval*, which on input pub and* $x \in \mathsf{Dom}_{pub}$*, outputs* $y \in \mathsf{Dom}_{pub}$*. We write* $f(x) = \mathsf{Eval}(pub, x)$*.*

We require that for all $k \in \mathbb{N}$ *and all pub output by* $\mathsf{Gen}(1^k)$*,* $\mathsf{Eval}(pub, \cdot)$ *defines a permutation over* Dom_{pub}*.*

Definition 1 extends to families of trapdoor permutations, where Gen additionally outputs a trapdoor *trap* which can be used by a deterministic polynomial-time algorithm Invert to compute $f^{-1}(y)$, for any $y \in \mathsf{Dom}_{pub}$.

We want to point out that $\mathsf{Eval}(pub, \cdot)$ is only required to be a permutation for correctly generated *pub* but not every bit-string *pub* yields a permutation. A family of permutations Π is said to be *certified* [3] if the fact that it is a permutation can be verified in polynomial time given *pub*.

Definition 2. $\mathsf{CP} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Certify})$ *is called a family of certified permutations if* $(\mathsf{Gen}, \mathsf{Eval})$ *is a family of permutations and* $\mathsf{Certify}$ *is a deterministic polynomial-time algorithm that, on input of* 1^k *and an arbitrary pub (potentially not generated by* Gen*), returns* 1 *iff* $\mathsf{Eval}(pub, \cdot)$ *defines a permutation over* Dom_{pub}*.*

Definition 2 also extends to families of certified trapdoor permutations.

We remark that Definition 2 follows [20] and is slightly weaker than that of Bellare and Yung [3], where, for all inputs, the $\mathsf{Certify}$ algorithm is required to return 1 iff *pub* was generated by $\mathsf{Gen}(1^k)$, with some constant error probability (in the sense of a BPP algorithm).[1] In fact, it seems that the certification

[1] The difference between the two definitions can be explained for the case of RSA. Suppose the original Gen algorithm outputs $pub = (N = pq, e)$ with $\gcd(e, \varphi(N)) = 1$. This cannot define a certified permutation with respect to the Bellare-Yung definition since if $pub' = (N' = pqr, e')$ with $\gcd(e', \varphi(N')) = 1)$ then $pub \approx pub'$ under the 2-vs-3 prime assumption but pub' is never output by Gen. However, since $\gcd(N', e') = 1$, $\mathsf{RSA}_{N',e'}$ defines a permutation so there is some hope that it still meets Definition 2.

constructions by Bellare and Yung [3, Section 3] only meet our weaker definition which is, in particular, sufficient for their applications to NIZK for all NP languages.

2.3 RSA trapdoor permutation

In Figure 1 we give a description of a family of trapdoor permutations $\mathsf{RSA}_\gamma = (\mathsf{RSAGen}_\gamma, \mathsf{RSAEval}, \mathsf{RSAInvert})$, parametrized by some function $\gamma > 0$ (which controls the size of the exponent $e \approx N^\gamma$). The domain is defined as $\mathsf{Dom}_{pub} = \mathbb{Z}_N^*$.

algorithm $\mathsf{RSAGen}_\gamma(1^k)$	**algorithm** $\mathsf{RSAEval}(pk, x)$	**algorithm** $\mathsf{RSAInvert}(td, y)$
$p, q \in_R \mathbb{P}_{k/2}$ $N = pq$ repeat $e \in_R \mathbb{P}_{\gamma k}$ until $(gcd(e, \varphi(N)) = 1)$ $d = e^{-1} \mod \varphi(N)$ return $(pk = (N, e), td = d)$	return $x^e \mod N$	return $y^d \mod N$

Fig. 1. RSA permutation algorithms

3 RSA Certification Algorithm

In this section we will give a certification algorithm for the RSA trapdoor permutation RSA_γ from Section 2.3. Our algorithm can be derived from the following main theorem.

Theorem 3. *Let N be an integer of unknown factorization and $e < N$ be a prime integer such that $\gamma = \log_N e = \frac{1}{4} + \varepsilon$ and $gcd(e, N) = 1$. We can decide if $gcd(e, \varphi(N)) = 1$ or $gcd(e, \varphi(N)) = e$ in time $O(\varepsilon^{-8} \log^2 N)$.*

Proof. Let us write $N = \prod_{i=1}^{n} p_i^{z_i}$, with prime p_i. Therefore,

$$\varphi(N) = \prod_{i=1}^{n} p_i^{z_i - 1}(p_i - 1).$$

Since e is prime, we can only have $gcd(e, \varphi(N)) = 1$ or $gcd(e, \varphi(N)) = e$. In the last case, we must have $e | \varphi(N)$. If $e > N$ then we know that $gcd(e, \varphi(N)) = 1$ [20]. When $e < N$, then we need to perform some further checks.

Let us look at the case $e | \varphi(N)$. If $e | p_i^{z_i - 1}$ then $gcd(e, N) = e$, which contradicts the prerequisite that e and N are coprime. Hence we must have $e | (p_i - 1)$ for some i. Let us denote $p = p_i$. There exists an $x_0 \in \mathbb{N}$ s.t.

$$ex_0 + 1 = p.$$

Our goal is to recover x_0 and thus to find p. Notice that x_0 is a small root of the polynomial equation $f(x) = ex + 1$ modulo p.

This allows us to use Coppersmith's algorithm for finding small roots of modular polynomial equations.

Theorem 4 (Coppersmith). *Let N be an integer of unknown factorization, which has a divisor $p \geq N^{\beta}, 0 < \beta \leq 1$. Let $0 < \mu \leq \frac{1}{7}\beta$. Furthermore, let $f(x)$ be a univariate monic polynomial of degree δ. Then we can find all solutions x_0 for the equation:*

$$f(x_0) = 0 \bmod p \qquad \text{with } |x_0| \leq \frac{1}{2} N^{\frac{\beta^2}{\delta} - \mu}$$

This can be achieved in time $O(\mu^{-7}\delta^5 \log^2 N)$. The number of solutions x_0 is bounded by $O(\mu^{-1}\delta)$.

A proof can be found in [21].

We use Coppersmith's algorithm to find prime divisors p of N in a specific range as specified in the following lemma.

Lemma 5. *Let N be an integer of unknown factorization with divisor $p \geq N^{\beta}$ for some $\beta \in (0, 1]$. Let $\mu \in (0, \frac{\beta}{7}]$. Further, let $e = N^{\gamma}$ with $e | p-1$. Then there is an algorithm* FindFactor *that on input N, e, β, μ outputs p in time $O(\mu^{-7} \log^2 N)$ provided that*

$$p \leq N^{\beta^2 + \gamma - \mu}.$$

If FindFactor *cannot find a non-trivial factor of N, it outputs $\perp$.*

Proof. Since $e|p-1$, we have $ex_0 = p - 1$ for some $x_0 \in \mathbb{N}$. Thus the polynomial $f(x) = ex + 1$ has the root x_0 modulo p. Multiplication of $f(x)$ by e^{-1} modulo N gives us a monic polynomial with the same root modulo p. Let us bound the size of our desired root x_0. We have

$$x_0 = \frac{p-1}{e} < \frac{N^{\beta^2 + \gamma - \mu}}{N^{\gamma}} = N^{\beta^2 - \mu}.$$

Thus we can recover x_0 by Theorem 4 in time $O(\mu^{-7} \log^2 N)$. Also by Theorem 4, the number of candidates for x_0 is bounded by $O(\mu^{-1})$. For every candidate we check whether $\gcd(ex_0 + 1, N)$ gives us the divisor p. This can be done in time $O(\mu^{-1} \log^2 N)$, which concludes the proof.

Lemma 5 can be used to check whether $e|p - 1$ for some prime divisor p in the range $[N^{\beta}, N^{\beta^2 - \mu + \gamma}]$. Our goal is to check whether $e|p-1$ for some p in the entire range $[e, N]$, which we will call the target range.

Obviously $p \leq N$. Thus, we can set the upper bound to $\beta^2 + \gamma - \mu = 1$. This in turn implies a lower bound of $\beta = \sqrt{1 - (\gamma - \mu)}$. Hence, we can first search for a divisor p in the interval $[N^{\sqrt{1-(\gamma-\mu)}}, N]$. If we do not find a divisor p in this interval, then we know that any divisor p must satisfy $p \leq N^{\sqrt{1-(\gamma-\mu)}}$. This defines a new upper bound, and in turn a new lower bound.

In total, we cover the target range by a sequence of intervals $[N^{\beta_1}, N^{\beta_0}], \ldots, [N^{\beta_n}, N^{\beta_{n-1}}]$ where the β_i are defined by the recurrence relation

$$\beta_{i+1} = \max\{\sqrt{\beta_i - (\gamma - \mu)}, \gamma\} \text{ with } \beta_0 = 1.$$

Two examples of such an interval sequence are illustrated in Figure 2.

The following lemma shows that our recurrence reaches γ and thus covers the target range $[e, N]$ after a certain number of steps.

Lemma 6. *Let $\frac{1}{4} < \gamma - \mu < \gamma < 1$. Then the recurrence relation*

$$\beta_{i+1} = \max\{\sqrt{\beta_i - (\gamma - \mu)}, \gamma\} \textit{ with } \beta_0 = 1$$

satifies $\beta_k = \gamma$ for some $k \leq \left\lceil \frac{1-\gamma}{\gamma-\mu-\frac{1}{4}} \right\rceil + 1$.

Proof. Since by definition $\gamma \leq \beta_i \leq 1$ for all i and $\mu > 0$, we have $\beta_i - (\gamma - \mu) > 0$ and therefore $\sqrt{\beta_i - (\gamma - \mu)}$ is defined in $\mathbb{R}$.

We now show by induction that the sequence of the β_i is monotone decreasing. Let us start with $\beta_1 < \beta_0$. Since $\beta_0 - (\gamma - \mu) < 1$, we have $\max\{\sqrt{\beta_0 - (\gamma - \mu)}, \gamma\} < 1$ and therefore $\beta_1 < \beta_0$.

Our inductive hypothesis is $\beta_i \leq \beta_{i-1}$ for all $i \leq n$. Now $\beta_n \leq \beta_{n-1}$ implies

$$\beta_n - (\gamma - \mu) \leq \beta_{n-1} - (\gamma - \mu)$$

and therefore by monotonicity of the square root function

$$\sqrt{\beta_n - (\gamma - \mu)} \leq \sqrt{\beta_{n-1} - (\gamma - \mu)}.$$

This yields $\max\{\sqrt{\beta_n - (\gamma - \mu)}, \gamma\} \leq \max\{\sqrt{\beta_{n-1} - (\gamma - \mu)}, \gamma\}$. Thus, $\beta_{n+1} \leq \beta_n$.

Since the sequence of the β_i is monotone decreasing and bounded below by γ, it converges. Now we show that we can upper bound the number $k-1$ of intervals $[\beta_i, \beta_{i-1}]$, $1 \leq i < k$ for which $\beta_i > \gamma$. This implies that our sequence stabilizes after k steps at the point $\beta_k = \gamma$.

Let us define a function $\Delta(\beta_{i-1}) = \beta_{i-1} - \beta_i \geq 0$, which gives us the length of the i^{th} interval. For $\beta_i > \gamma$ we obtain $\Delta(\beta_{i-1}) = \beta_{i-1} - \sqrt{\beta_{i-1} - (\gamma - \mu)}$. Since the first two derivatives of $\Delta(\beta)$ satisfy

$$\Delta'(\beta) = 1 - \tfrac{1}{2}(\beta - (\gamma - \mu))^{-\frac{1}{2}} \text{ and } \Delta''(\beta) = \tfrac{1}{4}(\beta - (\gamma - \mu))^{-\frac{3}{2}} > 0,$$

an easy computation shows that $\Delta(\beta)$ achieves its minimum at the point $\beta^{(0)} = \frac{1}{4} + \gamma - \mu$. Therefore, each interval length is of size at least

$$\Delta(\beta^{(0)}) = \gamma - \mu - \frac{1}{4}.$$

This in turn means that the number $k-1$ of intervals with $\beta_i > \gamma$ is at most

$$k - 1 \leq \left\lceil \frac{1-\gamma}{\gamma - \mu - \frac{1}{4}} \right\rceil,$$

which concludes the proof.

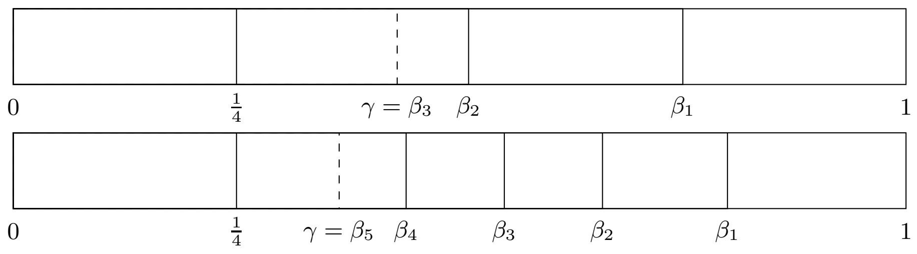

Fig. 2. Values Obtained for ($\gamma = 0.43, \mu = 0.06$) and ($\gamma = 0.365, \mu = 0.05$)

We continue the proof of Theorem 3. We can use the algorithm FindFactor from Lemma 5 with the parameters (N, e, β_i, μ) to test if there is any factor p such that $e|p-1$ in the sub-range $[N^{\beta_i}, N^{\beta_i^2-\mu+\gamma}]$. If we run FindFactor multiple times with the β_i values computed using the relation in Lemma 6, we can test the entire range as required.

We now discuss the choice of the parameter μ. Lemma 5 gives us the condition $\mu \leq \beta_i/7$ for all values of i. We know from the proof of Lemma 6 that $\gamma \leq \beta_i$ for all values of i. Hence it is sufficient to pick μ such that $\mu \leq \gamma/7$.

Furthermore, from Lemma 6 we have the condition $\mu < \gamma - \frac{1}{4}$. It is easy to verify that both conditions

$$\mu \leq \gamma/7 \text{ and } \mu < \gamma - \tfrac{1}{4}$$

are satisfied by the choice $\mu := \frac{1}{7}(\gamma - \frac{1}{4}) = \frac{1}{7}\varepsilon$ for all $\gamma > \frac{1}{4}$.

We give the whole algorithm GCDDecide for deciding whether $\gcd(e, \phi(N)) = 1$ in Figure 3.

```
algorithm GDCDecide($N, e$)
if ($e > N$) then return 1
$\gamma = \log_N e$, $\varepsilon = \gamma - \frac{1}{4}$
if ($\varepsilon \leq 0$) then return $\perp$
$\mu = \frac{1}{7}\varepsilon$
$\beta_0 = 1, i = 0$
while($\beta_i >= \gamma$)
    if (FindFactor($N, e, \beta_i, \mu$) $\neq \perp$) return $e$
    $i++$
    $\beta_i = \max\{\sqrt{\beta_{i-1} - (\gamma - \mu)}, \gamma\}$
wend
return 1
```

```
algorithm RSACertify($N, e$)
if (!PRIME($e$)) then return $\perp$
if ($gcd(e, N)! = 1$) then return $\perp$
if (GCDDecide($N, e$)$! = 1$)
    then return false
else return true
```

Fig. 3. GCD Decision and RSA Certification algorithms

It remains to determine the running time $t_{\mathsf{GCDDecide}}$ of GCDDecide. We know from Lemma 6 that we need at most $\lceil (1-\gamma)/(\gamma - \mu - \frac{1}{4}) \rceil + 1$ iterations of

FindFactor, which can be bounded as

$$\left\lceil \frac{1-\gamma}{\gamma-\mu-\frac{1}{4}} \right\rceil + 1 \leq \left\lceil \frac{1}{6\mu-\frac{1}{4}} \right\rceil + 1 = O(\mu^{-1}) = O(\varepsilon^{-1}).$$

Since each iteration takes time $O(\mu^{-7} \log^2 N)$, we obtain

$$t_{\mathsf{GCDDecide}} = O(\mu^{-8} \log^2 N) = O(\varepsilon^{-8} \log^2 N).$$

This concludes the proof of Theorem 3.

We now describe our full certification algorithm RSACertify that certifies the RSA trapdoor permutation RSA_γ from Section 2.3, for $\gamma = 1/4 + \varepsilon$. Note that we assume in Theorem 3 that e is prime and that $gcd(e, N) = 1$. If we want to check these prerequisites, we have an additional overhead of $O(\log^4 N)$ for the primality test on e and $O(\log^2 N)$ for the GCD computation. The complete certification algorithm RSACertify is described in Figure 3. The total running time of RSACertify, denoted by $t_{\mathsf{RSACertify}}$, is given by the expression

$$\begin{aligned} t_{\mathsf{RSACertify}} &= O(\log^4 N) + O(\log^2 N) + t_{\mathsf{GCDDecide}} \\ &= O(\max\{\log^4 N, \varepsilon^{-8} \log^2 N\}). \end{aligned}$$

Let $\mathsf{CRSA}_\gamma = (\mathsf{RSAGen}_\gamma, \mathsf{RSAEval}, \mathsf{RSAInvert}, \mathsf{RSACertify})$, as described in Figures 1 and 3, where γ controls the size of $e \approx N^\gamma$. By Theorem 3 we can see that, for any $\gamma = 1/4 + 1/poly(k)$, CRSA_γ defines a family of certified trapdoor permutations with respect to Definition 2.

Acknowledgements

Saqib Kakvi and Eike Kiltz were funded by a Sofja Kovalevskaja Award of the Alexander von Humboldt Foundation and the German Federal Ministry for Education and Research.

References

1. M. Bellare, C. Namprempre, and G. Neven. Unrestricted aggregate signatures. In L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, editors, *ICALP 2007: 34th International Colloquium on Automata, Languages and Programming*, volume 4596 of *Lecture Notes in Computer Science*, pages 411–422. Springer, July 2007.
2. M. Bellare and M. Yung. Certifying cryptographic tools: The case of trapdoor permutations. In E. F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 442–460. Springer, Aug. 1993.
3. M. Bellare and M. Yung. Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation. *Journal of Cryptology*, 9(3):149–166, 1996.
4. L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, May 1986.

5. C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414. Springer, May 1999.
6. J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In M. J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 413–430. Springer, Aug. 1999.
7. D. Catalano, D. Pointcheval, and T. Pornin. Trapdoor hard-to-invert group isomorphisms and their application to password-based authentication. *Journal of Cryptology*, 20(1):115–149, Jan. 2007.
8. D. Coppersmith. Finding a small root of a univariate modular equation. In U. M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 155–165. Springer, May 1996.
9. J.-S. Coron. On the exact security of full domain hash. In M. Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer, Aug. 2000.
10. C. Dwork and M. Naor. Zaps and their applications. In *41st Annual Symposium on Foundations of Computer Science*, pages 283–293. IEEE Computer Society Press, Nov. 2000.
11. U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *FOCS*, pages 308–317, 1990.
12. D. M. Freeman, O. Goldreich, E. Kiltz, A. Rosen, and G. Segev. More constructions of lossy and correlation-secure trapdoor functions. In P. Q. Nguyen and D. Pointcheval, editors, *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 279–295. Springer, May 2010.
13. S. Garg, V. Rao, A. Sahai, D. Schröder, and D. Unruh. Round optimal blind signatures. In *CRYPTO*, pages 630–648, 2011.
14. R. Gennaro, D. Micciancio, and T. Rabin. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In *ACM CCS 98: 5th Conference on Computer and Communications Security*, pages 67–72. ACM Press, Nov. 1998.
15. O. Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.
16. O. Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.
17. O. Goldreich. Basing non-interactive zero-knowledge on (enhanced) trapdoor permutations: The state of the art. In *Studies in Complexity and Cryptography*, pages 406–421. 2011.
18. S. A. Kakvi and E. Kiltz. Optimal security proofs for full domain hash, revisited. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 537–553. Springer, April 2012.
19. E. Kiltz, A. O'Neill, and A. Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 295–313. Springer, Aug. 2010.
20. A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In C. Cachin and J. Camenisch, editors,

Advances in Cryptology – EUROCRYPT 2004, volume 3027 of *Lecture Notes in Computer Science*, pages 74–90. Springer, May 2004.

21. A. May. Using LLL-reduction for solving RSA and factorization problems. In *The LLL Algorithm*, Information Security and Cryptography, pages 315–348. Springer, 2010.
22. P. Mol and S. Yilek. Chosen-ciphertext security from slightly lossy trapdoor functions. In P. Q. Nguyen and D. Pointcheval, editors, *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 296–311. Springer, May 2010.
23. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, May 1999.
24. C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In R. E. Ladner and C. Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 187–196. ACM Press, May 2008.
25. R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.