A Deterministic Algorithm for Computing Divisors in an Interval

Liqiang Peng[1,2], Yao Lu[1,2,3(✉)], Noboru Kunihiro[3], Rui Zhang[1], and Lei Hu[1,2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100 093, China
{pengliqiang,r-zhang}@iie.ac.cn, hu@is.ac.cn

[2] Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing 100 093, China

[3] The University of Tokyo, Tokyo, Japan

Abstract. We revisit the problem of finding a nontrivial divisor of a composite integer when it has a divisor in an interval $[\alpha, \beta]$. We use Strassen's algorithm to solve this problem. Compared with Kim-Cheon's algorithms (Math Comp 84(291): 339–354, 2015), our method is a deterministic algorithm but with the same complexity as Kim-Cheon's probabilistic algorithm, and our algorithm does not need to impose that the divisor is prime. In addition, we can further speed up the theoretical complexity of Kim-Cheon's algorithms and our algorithm by a logarithmic term $\log(\beta - \alpha)$ based on the peculiar property of polynomial arithmetic we consider.

Keywords: Integer factorization · Divisors in an interval
Polynomial arithmetic

1 Introduction

RSA is the most widely deployed public-key cryptosystem. Its security relies on the difficulty of factoring large composite integer: if integer factorization is solved then RSA is broken. Factoring large numbers is long been believed as a mathematical hard problem in computational number theory. Now it is conjectured that integer factorization cannot be solved in polynomial-time without quantum computers.

However, even if integer factorization is indeed difficult to solve, one has to be very careful against the side-channel attacks, which is any attack based on information gained from the physical implementation of cryptosystems.

In this paper, we focus on the problem of integer factorization given the approximation of divisors. More precisely, we mainly focus on finding a nontrivial divisor of a composite integer N when it has a divisor in an interval $[\alpha, \beta]$.

It is clear that this problem can be solved in $\mathcal{O}(\beta - \alpha)$ time with trial division. However, based on the bit-size of parameters α and β, more efficient algorithms exist.

- For sufficiently small interval bit-size $\beta - \alpha$: Using Coppersmith's method [5] of finding small roots of modular polynomial equations, we can recover all divisors in the interval in polynomial time in $\log N$.
- For relatively small α and large β: Using Pollard's rho method [12], we can find a nontrivial divisor in $\mathcal{O}(\beta^{1/2})$ time.
- For large α and large $\beta - \alpha$: Using Kim-Cheon's algorithms [10], we can recover a nontrivial divisor in $\widetilde{\mathcal{O}}((\beta - \alpha)^{1/2})$ time.

Specifically, in [10], Kim and Cheon proposed two algorithms, one is probabilistic and the other is its deterministic version, for achieving birthday complexity in finding a divisor in an interval. Using their proposed algorithms, one can check the existence of prime divisors in the interval, and if they exist, one can find all such prime divisors.

Compared with Kim-Cheon's probabilistic algorithm, their deterministic algorithm is more complex, difficult to understand, and needs more time complexity. Besides, for the case of composite divisors, their probabilistic algorithm works well, but their deterministic algorithm fails. Therefore, Kim and Cheon posted as an open problem to design a deterministic algorithm for composite divisors.

1.1 Our Contributions

In this paper, we propose a deterministic algorithm to find a nontrivial divisor of a composite integer N when it has a divisor in an interval $[\alpha, \beta]$. Our deterministic algorithm has the same time complexity as Kim-Cheon's probabilistic algorithm, and also works for the case of composite divisors. In addition, we can further speed up the theoretical complexity of Kim-Cheon's algorithms and our algorithm by a logarithmic term $\log(\beta - \alpha)$ based on the peculiar property of polynomial arithmetic we consider.

Technically, recall that Kim-Cheon's algorithm reduces the target problem to solving a discrete logarithm problem over $(\mathbb{Z}/n\mathbb{Z})^*$, where n is an unknown divisor of the known integer N. We view the original problem from a different perspective: we relate the original problem to a variant of deterministic integer factorization problem, and then use Strassen's algorithm [13,14] to solve it. More precisely, let $p = \beta - x$ be a divisor of N in the interval $[\alpha, \beta]$, where $x \in [0, \beta - \alpha]$ is unknown. Then the problem of finding p can be transformed to computing $\gcd(N, \beta - x)$. Although x is unknown, we can use $\gcd\left(N, \prod_{i=0}^{\beta-\alpha}(\beta - i) \pmod{N}\right)$ to find p. Therefore, how to calculate $\prod_{i=0}^{\beta-\alpha}(\beta - i) \pmod{N}$ efficiently becomes the key point of the complexity.

Moreover, recently Chen and Nguyen [4] used a similar algorithm as Strassen's algorithm to solve Approximate Common Divisor Problem, the later was introduced by Howgrave-Graham [9] in CaLC 2001.

2 Preliminaries

Let a and b be integers. Let $\nu_a(b)$ denote the nonnegative integer such that $a^{\nu_a(b)} \mid b$ and $a^{\nu_a(b)+1} \nmid b$. Denote $[\alpha, \beta]$ as the set of all integers $\alpha \leq i \leq \beta$. Let $|\beta - \alpha|_2$ denote the bit-size of $\beta - \alpha$. We will use log for the binary (base 2) logarithm. Let $M(d)$ be the complexity of the multiplication of two polynomial with degree d [1]:

$$M(d) = \mathcal{O}(d \log d \log\log d).$$

In this paper, we consider the univariate polynomial $f(x) \in \mathbb{Z}_N[x]$ with N an arbitrary integer. We will use two polynomial arithmetic algorithms, $\mathbf{Alg}_{Poly}$ (compute a polynomial given as a product of d terms) and $\mathbf{Alg}_{MPE}$ (evaluate a univariate polynomial with degree d at d points), as subroutines. It is clear that we can solve them using $\mathcal{O}(d^2)$ additions and multiplications in $\mathbb{Z}_N$. However, there are classic algorithms with quasi-linear complexity operations in $\mathbb{Z}_N$ using a divide-and-conquer approach. Recently these two algorithms have been used in various area of public-key cryptanalysis [4,6,8]. We give the basic information of these two algorithms as follows:

$\mathbf{Alg}_{Poly}$: Takes integer N and d points (suppose that $a_0, \ldots, a_{d-1}$) as inputs; outputs a monic degree d polynomial over $\mathbb{Z}_N$ having d points as roots: $f(X) = \prod_{i=0}^{d-1}(X - a_i) \pmod N$. According to a classic result [1], the time complexity is $\mathcal{O}(\log d M(d))$ operations modulo N, and the storage requirement is $\mathcal{O}(d \log d)$ elements in $\mathbb{Z}_N$.

$\mathbf{Alg}_{MPE}$: Takes integer N, a polynomial $f(x)$ with degree d over $\mathbb{Z}_N$ and d points (suppose that $c_0, \ldots, c_{d-1}$) as inputs; outputs the evaluation of $f(x)$ at d input points: $f(c_0), \ldots, f(c_{d-1}) \pmod N$. According to a classic result [1], the time complexity is $\mathcal{O}(\log d M(d))$ operations modulo N, and the storage requirement is $\mathcal{O}(d \log d)$ elements in $\mathbb{Z}_N$.

3 Review Kim-Cheon's Algorithms

In this section, we will review Kim-Cheon's two algorithms: one is probabilistic and the other is its deterministic version. Their algorithms essentially work by solving the discrete logarithm problem over $(\mathbb{Z}/n\mathbb{Z})^*$, where n is an unknown divisor of the target composite integer N. Before given the full description of Kim-Cheon's algorithms, we would like to introduce a lemma from [10]:

Lemma 1. *There exists an algorithm FINDING which, given as input positive integers N, g, h, and δ with $1 < g, h < N$, $\gcd(gh, N) = 1$, outputs an integer $x \in [1, \delta]$ with $\gcd(g^x - h, N) > 1$ or shows that no such x exists in*

$$\mathcal{O}\left(M(\delta^{1/2}) \log \delta\right)$$

operations modulo N by using storage $\mathcal{O}(\delta^{1/2} \log \delta)$ elements in $\mathbb{Z}_N$.

We recall the *FINDING* algorithm, given as Algortihm 1.

Algorithm 1. $x \leftarrow FINDING(N, g, h, \delta)$

```
Input: Positive integers $N, g, h$ and $\delta$ with $1 < g, h < N$, $\gcd(gh, N) = 1$.
Output: An integer $x \in [1, \delta]$ satisfying $\gcd(g^x - h, N) > 1$.
 1: Set $L := \lceil \delta^{1/2} \rceil$.
 2: Compute the polynomial
        $$F(X) = \prod_{0 \le i \le L-1} (X - hg^i) \bmod N$$
    using Algorithm $\mathbf{Alg}_{Poly}$.
 3: Evaluate $F(X)$ at multiple points $g^{jL}$ for all $1 \le j \le L$ using Algorithm $\mathbf{Alg}_{MPE}$
 4: $j := 1$
 5: while $j \le L$ do
 6:    $d_j = \gcd(F(g^{jL}), N)$
 7:    if $d_j > 1$ then
 8:       Find the great $u$ satisfying $\gcd(g^{jL} - hg^u, N) > 1$.
 9:       Output $x := jL - u$ and stop.
10:    end if
11:    $j := j + 1$
12: end while
13: Output "there is no such $x$" and stop.
```

The complexity of Algorithm *FINDING* mainly relies on the complexity of $\mathbf{Alg}_{Poly}$ and $\mathbf{Alg}_{MPE}$, thus the overall complexity is $\mathcal{O}\left(\log \delta M(\delta^{1/2})\right)$ operations modulo N with using storage $\mathcal{O}(\delta^{1/2} \log \delta)$ elements in $\mathbb{Z}_N$.

Now we review Kim-Cheon's probabilistic algorithm for computing a nontrivial divisor of a composite integer N, given as Algortihm 2.

Algortihm 2 takes $\mathcal{O}\left(M((\beta - \alpha)^{1/2}) \log(\beta - \alpha)\right)$ operations modulo N. The storage requirement is $\mathcal{O}((\beta - \alpha)^{1/2} \log(\beta - \alpha))$ elements in $\mathbb{Z}_N$. In [10], Kim and Cheon showed that Algortihm 2 succeeds with a probability of at least 1/2.

Kim-Cheon's Deterministic Algorithm. Since we do not know exactly how many a's are to be tested or how to choose a to split N in Algortihm 2, hence, the algorithm works probabilistically. Therefore, Kim and Cheon proposed a deterministic algorithm to overcome this problem, the key tool of their deterministic algorithm was the distribution of smooth numbers, which was originally used for devising a deterministic primality test under some condition by Konyagin and Pomerance [11]. We omit the details of their algorithm here, instead, we refer to [10]. Obviously, Kim-Cheon's probabilistic algorithm performs better than their deterministic algorithm.

4 Our Deterministic Algorithm

In this section, we propose a deterministic algorithm to find a nontrivial divisor of a composite integer N when it has a divisor in an interval $[\alpha, \beta]$. Our algorithm

Algorithm 2. Kim-Cheon's probabilistic algorithm for computing a nontrivial divisor of a composite integer N

```
Input: A composite integer $N$ with unknown factorization and an interval $[\alpha, \beta]$.
Output: A nontrivial divisor of $N$ when it has a divisor in an interval $[\alpha, \beta]$.
 1: Choose an integer $a$ uniformly at random in $\{2, \ldots, N-1\}$.
 2: if $\gcd(a, N) > 1$ then
 3:    output $\gcd(a, N)$ and stop.
 4: end if
 5: Compute $x_a \in [1, \beta-\alpha]$ such that $d = \gcd(a^{x_a} - a^{\beta-1} \mod N, N) > 1$ by applying
    subalgorithm FINDING (Alg.1).
 6: if there is no such $x_a$ then
 7:    output "$N$ has no prime divisor in the interval $[\alpha, \beta]$)" and stop.
 8: end if
 9: if $d < N$ then
10:    output $d$ and stop.
11: end if
12: if $d = N$ and $y_a := \beta - 1 - x_a$ is even then
13:    $i := 1$
14:    while $i \leq \nu_2(y_a)$ do
15:       compute $d_i = \gcd(a^{y_a/2^i} - 1, N)$
16:       if $1 < d_i < N$ then
17:          output $d_i$ and stop
18:       end if
19:       $i := i + 1$
20:    end while
21: end if
22: Output "failure" and stop.
```

has the same time complexity as Kim-Cheon's probabilistic algorithm, and also works for the case of composite divisors.

4.1 Algorithmic Details

Now we show how to reduce the target problem to a variant of integer factorization problem. Let p be the divisor of N in the interval $[\alpha, \beta]$. At first, we can write p as

$$p = \beta - x$$

where x is an unknown variable satisfying $0 \leq x \leq \beta - \alpha$. Then in this case, we are given one exact multiple $N (N \equiv 0 \bmod p)$ and one integer $\beta = p + x$, and the goal is to learn the divisor p. Here, we do not require that p is prime.

Next we give our algorithm based on Strassen's algorithm [13,14] for solving the integer factorization problem. It is clear that

$$p = \gcd\left(N, \prod_{i=0}^{\beta-\alpha} (\beta - i) \pmod{N}\right)$$

The key problem is how to calculate $\prod_{i=0}^{\beta-\alpha}(\beta - i) \pmod N$ faster.

To calculate faster, we require the degree of polynomial be a power of two. Let $|\beta - \alpha|_2 = l$. Therefore, we focus on

$$p = \gcd\left(N, \prod_{i=0}^{2^l-1}(\beta - i) \pmod N\right)$$

Set $l^* = \lceil l/2 \rceil$, we can rewrite it as

$$\prod_{i=0}^{2^l-1}(\beta - i) \pmod N = \prod_{i=0}^{2^{l^*-(l \bmod 2)}-1} \prod_{j=0}^{2^{l^*}-1}(\beta - 2^{l^*}i - j) \pmod N$$

We define the polynomial $f_j(x)$ of degree j modulo integer N:

$$f_j(x) = \prod_{k=0}^{j-1}(\beta - x - k) \pmod N$$

Therefore, we have

$$\prod_{i=0}^{2^l-1}(\beta - i) \pmod N = \prod_{i=0}^{2^{l^*-(l \bmod 2)}-1} f_{2^{l^*}}(2^{l^*}i) \pmod N$$

which means

$$p = \gcd\left(N, \prod_{i=0}^{2^{l^*-(l \bmod 2)}-1} f_{2^{l^*}}(2^{l^*}i) \pmod N\right)$$

We need to compute the polynomial $f_{2^{l^*}}(x)$ explicitly and evaluate this polynomial at $2^{l^*-(l \bmod 2)}$ points, which can fortunately be done using $\mathbf{Alg}_{Poly}$ and $\mathbf{Alg}_{MPE}$. We give a full description of our algorithm as follows.

In our algorithm, the condition $d = 1$ means that there is no divisor in the interval $[\alpha, \beta]$ and if $1 < d \leq \beta$, d is the divisor what we want. However, if there are more than one divisors in the interval $[\alpha, \beta]$, we will obtain that $d > \beta$. According to the Strassen's algorithm, for this case we can use a trick of computing greatest common divisor based on a product tree to determine which $f_{2^{l^*}}(2^{l^*}k)$, where $1 \leq k \leq 2^{l^*-(l \bmod 2)}$ has only one divisor. Algorithm 4 gives a brief description of this trick. Note that, if it is still that $\gcd(N, f_{2^{l^*}}(2^{l^*}k)) > \beta$ which means there are still more than one divisors of N fall in the same interval $[\beta - 2^{l^*}(k+1) + 1, \beta - 2^{l^*}k]$, we can further use same trick as Algorithm 4 to construct a product tree based on the following expression

$$f_{2^{l^*}}(2^{l^*}k) = \prod_{i=0}^{2^{l^*}-1}(\beta - 2^{l^*}k - i) \pmod N.$$

Algorithm 3. Our deterministic algorithm for computing a nontrivial divisor of a composite integer N

```
Input: A composite integer $N$ with unknown factorization and an interval $[\alpha, \beta]$.
Output: A nontrivial divisor of $N$ when it has a divisor in an interval $[\alpha, \beta]$.
 1: Set $l^* = \lceil |\beta - \alpha|_2 / 2 \rceil$.
 2: Compute the polynomial $f_{2^{l^*}}(x)$ using $\mathbf{Alg}_{Poly}$.
 3: Evaluate $f_{2^{l^*}}(x)$ at multiple points $2^{l^*}k$ for all $1 \le k \le 2^{l^* - (l \bmod 2)}$ using
    $\mathbf{Alg}_{MPE}$.
 4: Compute $d = \gcd(N, f_{2^{l^*}}(1) f_{2^{l^*}}(2) \cdots f_{2^{l^*}}(2^{l^* - (l \bmod 2)}) \bmod N)$.
 5: if $d = 1$ then
 6:     output "there is no divisor in interval $[\alpha, \beta]$" and stop.
 7: end if
 8: if $1 < d \le \beta$ then
 9:     output d and stop.
10: end if
11: if $\beta < d \le N$ then
12:     compute a divisor in an interval $[\alpha, \beta]$, using Algorithm 4.
13: end if
```

Then the divisor in the interval $[\alpha, \beta]$ can be finally determined.

Now, we analyze the complexity of Algorithm 3. The complexity of $\mathbf{Alg}_{Poly}$ and $\mathbf{Alg}_{MPE}$ takes $\mathcal{O}\left(\log(\beta - \alpha) M((\beta - \alpha)^{1/2})\right)$ operations modulo N and the storage requirement is $\mathcal{O}((\beta - \alpha)^{1/2} \log(\beta - \alpha))$ elements in $\mathbb{Z}_N$. In addition, we need GCD computations at most $2 \log(\beta - \alpha)^{1/2}$ times and $\mathcal{O}((\beta - \alpha)^{1/2})$ multiplications on modulo N. Therefore, the complexity of our algorithm mainly relies on the complexity of $\mathbf{Alg}_{Poly}$ and $\mathbf{Alg}_{MPE}$, just like Kim-Cheon's probabilistic algorithm our deterministic algorithm takes $\mathcal{O}\left(\log(\beta - \alpha) M((\beta - \alpha)^{1/2})\right)$ operations modulo N.

4.2 Logarithmic Speedup

The complexity of Kim-Cheon's algorithms and our algorithm mainly relies on $\mathbf{Alg}_{Poly}$ and $\mathbf{Alg}_{MPE}$. However, since the peculiar property of these polynomials we consider, hence more efficient algorithms exist. Thus, we can speed up the theoretical complexity of Kim-Cheon's algorithms and our algorithm by a logarithmic term $\log(\beta - \alpha)$.

Revisiting Kim-Cheon's Algorithms. In Algortihm 1, they want to compute the polynomial $F(X) = \prod_{0 \le i \le L-1}(X - hg^i) \bmod N$ and evaluate $F(x)$ at points $g^L, g^{2L}, \ldots, g^{L^2}$. Notice that both (hg^i) and (g^{iL}) are geometric progressions, hence we can use more efficient algorithm of Bostan et al. [3] to compute polynomial interpolation and polynomial evaluation at a geometric progression. Bostan gave his pseudocode in [2]. This technique can speed up the overall complexity of Kim-Cheon's algorithms by a logarithmic term $\log(\beta - \alpha)$.

Algorithm 4. RecursiveFinding(N, A)

```
Input: A composite integer $N$ and a set of numbers $\{a_1, \dots, a_n\}$.
Output: A nontrivial divisor of $N$ in the interval $[\alpha, \beta]$.
 1: $n' := \lceil n/2 \rceil$
 2: Compute $d = \gcd(N, \prod_{i=1}^{n'} a_i)$
 3: if $1 < d \le \beta$ then
 4:     output $d$ and stop.
 5: end if
 6: if $d = 1$ then
 7:     RecursiveFinding($N, \{a_{n'+1}, \dots, a_n\}$)
 8: end if
 9: if $\beta < d \le N$ then
10:     RecursiveFinding($N, \{a_1, \dots, a_{n'}\}$)
11: end if
```

Revisiting Our Algorithm. Likewise, our deterministic algorithm can also been improved by using a smarter way to calculate the evaluation of function $f_{2^{l^*}}(x)$ at $2^{l^*-(l \bmod 2)}$ points. We use Chen-Nguyen's technique, which based on Bostan, Gaudry and Schost's result [3], to speed up Algortihm 3.

More specifically, Bostan, Gaudry and Schost's result can be described as follows:

Theorem 1 *(Theorem 5 of [3]). Let a, b be in ring $\mathbb{R}$ and d be in $\mathbb{N}$ such that $\boldsymbol{d}(a, b, d)$ is invertible, with $\boldsymbol{d}(a, b, d) = b \cdot 2 \cdots d \cdot (a - db) \cdots (a + db)$, and suppose that the inverse of $\boldsymbol{d}(a, b, d)$ is known. Let $F(x)$ be in $\mathbb{R}[X]$ of degree at most d and $r \in \mathbb{R}$. Given $F(r), F(r + b), \dots, F(r + db)$, one can compute $F(r + a), F(r + a + b), \dots, F(r + a + db)$ in time $2M(d) + \mathcal{O}(d)$ time and space $\mathcal{O}(d)$. Here, $M(d)$ is the time of multiplying two polynomial of degree at most d.*

Define set $S(k_1, \dots, k_j) := \{\sum_{i=1}^{j} p_{k_i} 2^{k_i} \mid p_{k_i} \in \{0, 1\}\}$. Suppose that we already have the evaluation of $f_{2^j}(x)$ at points $S(k_{l-j+1}, \dots, k_l)$, if we can calculate the evaluation of $f_{2^{j+1}}(x)$ at points $S(k_{l-j}, \dots, k_l)$, then with each iteration, we can evaluate the $f_{2^{l^*}}(x)$ at $2^{l^*-(l \bmod 2)}$ points closer until $j = 2^{l^*}$.

The key technique is how to calculate the evaluation of $f_{2^{j+1}}(x)$ at points $S(k_{l-j}, \dots, k_l)$ using Theorem 1. For every $X \in S(k_{l-j}, \dots, k_l)$, we have

$$f_{2^{j+1}}(X) = f_{2^j}(X) \cdot f_{2^j}(X + 2^{j+1})$$

We can easily calculate $f_{2^j}(X)$ and $f_{2^j}(X + 2^{j+1})$ using Theorem 1, and evaluate $f_{2^{j+1}}(x)$ at points $S(k_{l-j}, \dots, k_l)$.

Note that, our algorithm does not need to impose that the divisor in the interval is prime. However, if we impose that the divisor is prime, we can use the method of [7], proposed by Costa and Harvey, to further speed up the theoretical complexity by removing some elements in the interval that do not contribute any useful information.

5 Conclusion

In this paper we revisit the problem of finding a nontrivial divisor of a composite integer N when it has a divisor in an interval $[\alpha, \beta]$. We present a deterministic algorithm to solve this problem, and our algorithm has the same complexity with Kim-Cheon's probabilistic algorithm. Besides, based on the special structure of polynomial, we give a method to speed up the theoretical complexity of Kim-Cheon's algorithm and our algorithm by a logarithmic term $\log(\beta - \alpha)$.

Acknowledgements. This research was supported the National Natural Science Foundation of China (Grants 61702505, 61472417, 61732021, 61772520), National Cryptography Development Fund (MMJJ20170115, MMJJ20170124) and the Fundamental Theory and Cutting Edge Technology Research Program of Institute of Information Engineering, CAS (Grants Y7Z0341103, Y7Z0321102), JST CREST Grant Number JPMJCR14D6, JSPS KAKENHI Grant Number 16H02780.

References

1. Bluestein, L.I.: A linear filtering approach to the computation of the discrete fourier transform. IEEE Trans. Electroacoust. **18**, 451–466 (1970)
2. Bostan, A.: Algorithmique efficace pour des opérations de base en calcul formel. Ph.D. thesis (2003). École polytechnique (in English)
3. Bostan, A., Gaudry, P., Schost, E.: Linear recurrences with polynomial coefficients and application to integer factorization and Cartier-Manin operator. SIAM J. Comput. **36**(6), 1777–1806 (2007)
4. Chen, Y., Nguyen, P.Q.: Faster algorithms for approximate common divisors: breaking fully-homomorphic-encryption challenges over the integers. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 502–519. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_30
5. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. J. Cryptol. **10**(4), 233–260 (1997)
6. Coron, J.-S., Joux, A., Mandal, A., Naccache, D., Tibouchi, M.: Cryptanalysis of the RSA subgroup assumption from TCC 2005. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 147–155. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_9
7. Costa, E., Harvey, D.: Faster deterministic integer factorization. Math. Comput. **83**(285), 339–345 (2014)
8. Fouque, P.-A., Tibouchi, M., Zapalowicz, J.-C.: Recovering private keys generated with weak PRNGs. In: Stam, M. (ed.) IMACC 2013. LNCS, vol. 8308, pp. 158–172. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45239-0_10
9. Howgrave-Graham, N.: Approximate integer common divisors. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 51–66. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44670-2_6
10. Kim, M., Cheon, J.H.: Computing prime divisors in an interval. Math. Comp. **84**(291), 339–354 (2015)
11. Konyagin, S., Pomerance, C.: On primes recognizable in deterministic polynomial time. In: Graham, R.L., Nešetřil, J. (eds.) The mathematics of Paul Erdős I. Springer, Heidelberg (1997)

12. Pollard, J.M.: Monte Carlo methods for index computation (mod p). Math. Comp. **32**(143), 918–928 (1978)
13. Pollard, J.M.: Theorems on factorization and primality testing. In: Proceedings of the Cambridge Philosophical Society, vol. 76, pp. 521–528 (1974)
14. Strassen, V.: Einige Resultate *ü*ber Berechnungskomplexit*ä*t. Jber. Deutsh. Math. -Verein. **78**(1), 1–8 (1976/1977)