symmetry

MDPI

Article

Small Private Exponent Attacks on RSA Using Continued Fractions and Multicore Systems

Hatem M. Bahig [1], Dieaa I. Nassr [1], Mohammed A. Mahdi [2] and Hazem M. Bahig [2,*]

[1] Department of Mathematics, Faculty of Science, Ain Shams University, Cairo 11566, Egypt
[2] Information and Computer Science Department, College of Computer Science and Engineering, University of Ha'il, Hail, Ha'il 81481, Saudi Arabia
* Correspondence: h.bahig@uoh.edu.sa

Abstract: The RSA (Rivest–Shamir–Adleman) asymmetric-key cryptosystem is widely used for encryptions and digital signatures. Let (n, e) be the RSA public key and d be the corresponding private key (or private exponent). One of the attacks on RSA is to find the private key d using continued fractions when d is small. In this paper, we present a new technique to improve a small private exponent attack on RSA using continued fractions and multicore systems. The idea of the proposed technique is to find an interval that contains $\phi(n)$, and then propose a method to generate different points in the interval that can be used by continued fraction and multicore systems to recover the private key, where ϕ is Euler's totient function. The practical results of three small private exponent attacks on RSA show that we extended the previous bound of the private key that is discovered by continued fractions. When n is 1024 bits, we used 20 cores to extend the bound of d by 0.016 for de Weger, Maitra-Sarkar, and Nassr et al. attacks in average times 7.67 h, 2.7 h, and 44 min, respectively.

Keywords: continued fractions; private exponent attack; RSA; Wiener's attack; integer factorization; multicore systems

check for updates

Citation: Bahig, H.M.; Nassr, D.I.; Mahdi, M.A.; Bahig, H.M. Small Private Exponent Attacks on RSA Using Continued Fractions and Multicore Systems. *Symmetry* **2022**, *14*, 1897. https://doi.org/10.3390/sym14091897

Academic Editors: Ioan Rașa, Takeshi Koshiba, Takeshi Koshiba, Yuan Ping and Yuri Borissov

Received: 5 August 2022
Accepted: 6 September 2022
Published: 10 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In 1978, Rivest, Shamir and Adleman [1] proposed the first asymmetric-key cryptosystem (RSA) for encryptions and digital signatures. Its security is based on the difficulty of factoring a large integer $n = p_1 p_2$ that is a product of two large prime numbers p_1 and p_2, with $p_1 > p_2$, of the same bit-sizes, i.e., $p_2 < p_1 < 2p_2$. Although there is a quantum algorithm that factors integers in polynomial time [2], there is no polynomial time algorithm for factoring integers in classical computers.

The RSA encryption process of a message x is computing $x^e \pmod n$, where (n, e) is the RSA public key. The RSA decryption process of the ciphertext y is computing $y^d \pmod n$, where d is the private key and satisfies that $ed - 1 = k\phi(n)$ for some integer k, where $\phi(n) = (p_1 - 1)(p_2 - 1)$ is Euler's totient function. The RSA encryption and decryption processes take times $O(\log e \log^2 n)$ and $O(\log d \log^2 n)$, respectively.

In order to speed up the decryption process, one might be tempted to use a small private exponent $d = n^{\delta}$,, i.e., δ is small. Wiener [3] showed that if $d < \frac{1}{3}n^{1/4}$, i.e., $\delta \leq 1/4$, then d is one of the denominators of the convergents of the continued fraction expansion of $\frac{e}{n}$, and thus RSA is insecure. Boneh and Durfee [4] used the lattice reduction to improve the bound of d to be $n^{0.292}$, where their method is based on Coppersmith's [5] technique to find small roots of modular polynomial equations.

Many other strategies [6–10] for improving the bound of d were inspired by Wiener's result. They mainly try to find an approximation of $\phi(n)$ better than n or to find a better lattice to recover large d.

For example, de Weger [6] used $n + 1 - 2\sqrt{n}$ as an estimation of $\phi(n)$ to recover d when $\delta < 3/4 - \beta$, where $p_1 - p_2 = n^{\beta}, 0.25 < \beta \leq 0.5$. Maitra and Sarkar [9] used

$n+1-\frac{3\sqrt{2}}{2}\sqrt{n}$ as an estimation of $\phi(n)$ to recover d when $|2p_2 - p_1|$ is small. Note that if $p_1 - p_2 = n^{\beta}, 0 \leq \beta \leq 0.25$, then Fermat's factoring method [6,11–13] factorizes n in polynomial time.

In order to unify small private exponent attacks on RSA and to determine a universal attack using continued fractions or lattices, the authors in [14,15] proposed concepts of the Wiener and Coppersmith intervals using continued fractions and lattices, respectively. An integer interval I is called Wiener's interval if each $m \in I$ satisfies Wiener's attack, i.e., $\left|\frac{e}{m} - \frac{k}{d}\right| < \frac{1}{2d^2}$. While an interval I is called Coppersmith's interval if each $m \in I$ satisfies that the tuple $(u_0, v_0) = (k, \phi(n) - m)$ is a root of the polynomial $F(u, v) = uv + mu + 1 \pmod{e}$.

In this paper, we are interested in improving the bound of d by:

1. Proposing an interval I that contains $\phi(n)$, Section 3. The proposed interval is not necessary a Wiener or Coppersmith interval. It is sufficient to find an approximation $m \in I$ of $\phi(n)$ such that $\left|\frac{e}{m} - \frac{k}{d}\right| < \frac{1}{2d^2}$, i.e., Wiener's attack using continued fraction succeeds.
2. Proposing a new strategy to search for $m \in I$ such that $\left|\frac{e}{m} - \frac{k}{d}\right| < \frac{1}{2d^2}$.
3. Using multicore systems to accelerate finding $m \in I$ such that $\left|\frac{e}{m} - \frac{k}{d}\right| < \frac{1}{2d^2}$. The interval I is divided into subintervals of the same length approximately. Then each core searches for such m in one subinterval. We choose that the number of subintervals is equal to the number of available cores.

We use the proposed strategy to study practically the possibility of attacking RSA when $d = n^{\delta} < \sqrt{n}$. Estimating a small interval that contains $\phi(n)$ is not simple. Therefore, we estimate the interval based on some conditions on the primes factors of n as we will see in Section 3. The practical study of the proposed method shows that we succeed to factor n with δ greater than previously discovered using continued fractions.

The organization of this paper is as follows. Section 2 includes a brief background on continued fractions and a review of some results on small private exponent attacks on RSA. In Section 3, we propose three intervals that contain $\phi(n)$ for three attacks on RSA. Each attack has different conditions on the prime factors p_1 and/or p_2. In Section 4, we present a new technique to search for m in the estimated intervals to find a good approximation of $\phi(n)$. Section 5 includes using multicore systems to study practically how the proposed technique can improve three attacks on RSA, i.e., extend the bound of δ in three attacks. The theoretical study of the complexity of the proposed attacks is presented in Section 6. The conclusion and future works are given in Sections 7.

2. Preliminaries

This section presents a definition of continued fractions, how to calculate continued fractions and some theorems and lemmas necessary in this paper.

Given a non-negative rational number r, a (finite) continued fraction expansion [16,17] of r (or simply we write $\mathcal{CF}(r)$) is an expression of the form:

$$r = r_1 + \frac{1}{r_2 + \frac{1}{r_3 + \cdots + \frac{1}{r_s}}}.$$

This expansion is denoted by $s - tuple$ of non-negative integers $[r_1, r_2, \ldots, r_s]$.

The following steps are a polynomial time algorithm [18] of order $O(log^2 y)$ for computing a unique $\mathcal{CF}(r)$ for the rational number $r = \frac{x}{y}$, where $x < y$ are two positive integers such that $gcd(x, y) = 1$:

- $r_0 = x/y$.
- Compute $r_i = \frac{1}{r_{i-1} - \lfloor r_{i-1} \rfloor}, 1 \leq i \leq s$, where $s \leq 2 \log y$ is the smallest value of i such that $\lfloor c_i \rfloor = c_i$.

- Return $[r_1, r_2, \ldots, r_s]$, where $r_s > 1$.
  The $\mathcal{CF}(r)$ is *infinite* in case of r is irrational number, i.e.,

$$r_1 + \frac{1}{r_2 + \frac{1}{r_3 + \cdots + \frac{1}{\ddots}}}.$$

In this case, we write the expansion as $[r_1, r_2, \ldots]$.

Theorem 1 ((Legendre) [19]). *Let λ be a real number, and u, v be two positive integers such that $gcd(u, v) = 1$. If*

$$\left|\lambda - \frac{u}{v}\right| < \frac{1}{2v^2},$$

then $\frac{u}{v}$ is a convergent of $\mathcal{CF}(\lambda)$.

Lemma 1 ([20,21]). *If n is a product of two primes p_1 and p_2 of the same size, then $n + 1 - \frac{3\sqrt{2}}{2}\sqrt{n} < \phi(n) < n + 1 - 2\sqrt{n}$.*

Theorem 2 ([6]). *Let $n = p_1 p_2$ be a product of two primes p_1, p_2 of the same size, with $p_1 > p_2$. Suppose that $1 < e, d < \phi(n)$ satisfy $ed \equiv 1 \pmod{\phi(n)}$ and $d = n^{\delta}$. Given n and e, the integer n can be factored in polynomial time in $\log n$ if*

$$\delta < \frac{3}{4} - \beta \text{ using continued fraction} \tag{1}$$

$$\delta < \frac{1}{6}(4\beta + 5) - \frac{1}{3}\sqrt{(4\beta + 5)(4\beta - 1)} \text{ using lattice} \tag{2}$$

where $p_1 - p_2 = n^{\beta}$.

Proposition 1 ([9]). *Suppose that l is a positive integer, and $n = p_1 p_2$ is a product of two primes p_1 and p_2. If $p_2 > \frac{2l+2}{4l+1} p_1$, then $\left|\frac{3}{\sqrt{2}}\sqrt{n} - (p_1 + p_2)\right| < \frac{l(2p_2 - p_1)^2}{(\frac{3}{\sqrt{2}} + 2)\sqrt{n}}$.*

Theorem 3 ([9]). *Let l be a positive integer, and $n = p_1 p_2$ be a product of two primes p_1 and p_2 with $p_2 > \frac{2l+2}{4l+1} p_1, 2p_2 - p_1 = n^{\theta}$, and $d = n^{\delta}$. Then n can be factored in polynomial time in $\log n$ when*

$$\delta < \frac{3}{4} - \theta - \tau \tag{3}$$

where $2\tau > (\log \frac{4l}{\frac{3}{\sqrt{2}} + 2})(\frac{1}{\log n})$.

Theorem 4 ([14]). *Let $(n = p_1 p_2, e)$, and $d = n^{\delta}$ be the public and private keys of RSA, respectively, where $p_1 > p_2$ and $2p_1 < n - \frac{9}{4}\sqrt{n}$. If $p_0 \geq \sqrt{n}$ is an approximation for p_1 such that*

$$|p_1 - p_0| \leq \frac{1}{8} n^{\alpha}, \alpha \leq \frac{1}{2}, \delta < \frac{1 - \alpha}{2} \tag{4}$$

Then $[n + 1 - \lambda_1, n + 1 - \lambda_2]$ is a Wiener's interval for (n, e), where

$$\lambda_1 = \begin{cases} p_0 + \frac{n}{p_0} + \frac{1}{8} n^{\alpha}, & p_0 \leq p_1; \\ p_0 + \frac{n}{p_0 - \frac{1}{8} n^{\alpha}}, & p_1 \leq p_0 \text{ and } \sqrt{n} \leq p_0 - \frac{1}{8} n^{\alpha}; \\ 2\sqrt{n} + \frac{1}{8} n^{\alpha}, & p_1 \leq p_0 \text{ and } p_0 - \frac{1}{8} n^{\alpha} < \sqrt{n}. \end{cases}$$

$$\lambda_2 = \begin{cases} p_0 + \frac{n}{p_0 + \frac{1}{8}n^\alpha}, & p_0 \le p_1; \\ \frac{n}{p_0} + p_0 - \frac{1}{8}n^\alpha, & p_1 \le p_0 \text{ and } \sqrt{n} \le p_0 - \frac{1}{8}n^\alpha; \\ \sqrt{n} + \frac{n}{\sqrt{n} + \frac{1}{8}n^\alpha}, & p_1 \le p_0 \text{ and } p_0 - \frac{1}{8}n^\alpha < \sqrt{n}. \end{cases}$$

3. Estimation of $\phi(n)$

The main problem of using CFs in small private exponent attacks of RSA is to find a good approximation of $\phi(n)$ to use it in Theorem 1. In this section, we estimate an interval I that contains $\phi(n)$, i.e., determine the lower and upper bounds of $\phi(n)$. In fact, estimating a small interval that contains $\phi(n)$ is not easy. It is known that computing $\phi(n)$ is computationally equivalent to factoring n. Thus, we try to estimate I based on some conditions on the prime factors p_1 and p_2 of n.

In the following, we consider three cases for the prime factors p_1 and p_2 :

Attack 1: In [6], if $p_1 - p_2 = n^\beta$, $0.25 \le \beta \le 0.5$, then

$$\begin{aligned} n + 1 - \sqrt{n^{2\beta} + 4n} &\le n + 1 - \sqrt{(p_1 - p_2)^2 + 4n} \\ &= n + 1 - (p_1 + p_2) = \phi(n) \\ &< n - 2\sqrt{n} \end{aligned}$$

Thus,

$$I = [n + 1 - \sqrt{n^{2\beta} + 4n}, n - 2\sqrt{n}].$$

Attack 2: Using Proposition 1 and Theorem 3 if for a positive integer l, $\frac{2l+2}{4l+1}p_1$ and $2p_2 - p_1 = n^\theta$, $\delta < \frac{3}{4} - \theta - \tau, 2\tau > (\log \frac{4l}{\frac{3}{\sqrt{2}}+2})(\frac{1}{\log n})$, then

$$\left| \frac{3\sqrt{2}}{2}\sqrt{n} - (p_1 + p_2) \right| < \frac{l(2p_2 - p_1)^2}{(\frac{3\sqrt{2}}{2} + 2)\sqrt{n}}$$

Therefore,

$$n + 1 - \frac{3\sqrt{2}}{2}\sqrt{n} < \phi(n) < n + 1 - \frac{3\sqrt{2}}{2}\sqrt{n} + \frac{l(2p_2 - p_1)^2}{(\frac{3\sqrt{2}}{2} + 2)\sqrt{n}}$$

It is clear that if $\frac{l}{\frac{3\sqrt{2}}{2}+2} < n^\epsilon$, for a small value ϵ, then

$$I = [n + 1 - \frac{3\sqrt{2}}{2}\sqrt{n}, n + 1 - \frac{3\sqrt{2}}{2}\sqrt{n} + n^{2\theta - 0.5 + \epsilon}].$$

Attack 3: Based on the result in [22], an approximation p_0 of the prime factor p_1 may be obtained by some expectations in side-channel attacks. In [14], if $p_2 < p_1 < 2p_2$ and p_0 be an approximation of the prime factor p_1 where $|p_1 - p_0| \le \frac{1}{2}n^\alpha$, then $\phi(n)$ can be estimated to be in the interval

$$I = [n + 1 - (p_0 + \frac{n}{p_0}) - \frac{1}{2}n^\alpha, n + 1 - (p_0 + \frac{n}{p_0}) + \frac{1}{2}n^\alpha].$$

The proof is as follows:
Let $p_1 = cn^{1/2}$. Then $p_2 = \frac{1}{c}n^{1/2}$, where $1 < c < \sqrt{2}$. We have $p_1 + p_2 = \frac{c^2+1}{c}n^{1/2}$.
Since $|p_1 - p_0| \le \frac{1}{2}n^\alpha$, we have either $p_0 \le p_1 \le p_0 + \frac{1}{2}n^\alpha$ or $p_0 - \frac{1}{2}n^\alpha \le p_1 \le p_0$.

If $p_0 \leq p_1 \leq p_0 + \frac{1}{2}n^\alpha$, then $\frac{p_0}{\sqrt{n}} \leq c \leq \frac{p_0+\frac{1}{2}n^\alpha}{\sqrt{n}}$. Therefore,

$$p_0 + \frac{n}{p_0} \leq p_1 + p_2 \leq p_0 + \frac{1}{2}n^\alpha + \frac{n}{p_0 + \frac{1}{2}n^\alpha}.$$

Furthermore, if $p_0 - \frac{1}{2}n^\alpha \leq p_1 \leq p_0$, then $\frac{p_0-\frac{1}{2}n^\alpha}{\sqrt{n}} \leq c \leq \frac{p_0}{\sqrt{n}}$. Therefore,

$$p_0 - \frac{1}{2}n^\alpha + \frac{n}{p_0 - \frac{1}{2}n^\alpha} \leq p_1 + p_2 \leq p_0 + \frac{n}{p_0}.$$

Therefore, either $p_0 \leq p_1 \leq p_0 + \frac{1}{2}n^\alpha$ or $p_0 - \frac{1}{2}n^\alpha \leq p_1 \leq p_0$, we have

$$|p_1 + p_2 - (p_0 + \frac{n}{p_0})| \leq \frac{1}{2}n^\alpha.$$

Thus,

$$I = [n + 1 - (p_0 + \frac{n}{p_0}) - \frac{1}{2}n^\alpha, n + 1 - (p_0 + \frac{n}{p_0}) + \frac{1}{2}n^\alpha].$$

4. The Proposed Strategy

In this section, we propose a new strategy to search for $m \in I$, such that $\left|\frac{e}{m} - \frac{k}{d}\right| < \frac{1}{2d^2}$. In general, the proposed interval $I = [a, b]$ that contain $\phi(n)$ are large. Since I is large, it is not feasible in polynomial time to test all integers in I. The main problem is to determine the number of tested points, i.e., how many points are sufficient to find $\phi(n)$ or to stop the search. Testing a fixed number L of points in I has a problem: if L is small, then we may not find the solution. Otherwise, i.e., if L is very large, then the distance between two consecutive points may be small and the time to find a solution may be large if the solution is in the last parts of I. For this reason, we propose a new method to generate test points in I as follows (see Algorithm 1):

We first test whether k/d is a convergent of $\mathcal{CF}(e/a)$ or $\mathcal{CF}(e/b)$. If k/d is not a convergent of $\mathcal{CF}(e/a)$ or $\mathcal{CF}(e/b)$, we set the length $c = b - a$, $x_1 = a$, $x_2 = x_1 + c$, and then we repeat taking x as the midpoint between x_1 and x_2, i.e., $x = (x_1 + x_2)/2$ and check whether k/d is a convergent of $\mathcal{CF}(e/x)$. If not, we repeat the previous steps with the length $c = c/2$, change x_1 to be x_2 and x_2 to be $x_1 + c$ until the midpoint x is greater than b. For each new midpoint x, the counter is increased by 1 as long as it does not exceed the maximum number of iterations L. The loop is terminated either by:

1. Finding a solution, Lines 19–20 in Algorithm 1.
2. Exceed the maximum number of generated test points L, Line 13 in Algorithm 1. This number can be replaced by a maximum time to find a solution.
3. The number of round i, i.e., number of iterations in the first while loop (Line 13 in Algorithm 1), is $i > \lfloor \log_2(b - a) \rfloor$, i.e., $c \leq 1$. In this case, we exhausted most points in the interval and the total number of tested points is about

$$2 + \sum_{i=0}^{\lfloor \log_2(b-a) \rfloor} 2^i$$

which is large when $b - a$ is large.

Figure 1 shows the idea of generating uniformly distributed 2^i test points in I for a round i, where $c = \lfloor (b - a)/2^i \rfloor$, $i \geq 0$.

For example, let $n = 802117 = 761 * 991$ be an RSA modulus. We have I = [800218, 800326]. Figure 2 shows the generated test points in I for rounds $i = 0, 1$, and 2, i.e., we repeat the second while loop (Lines 17–25) of Algorithm 1 three times $i = 0, 1, 2$. In Figure 3,

we show the generation of the first fifty-five test points in the first 6 rounds (the sixth round is not completed).

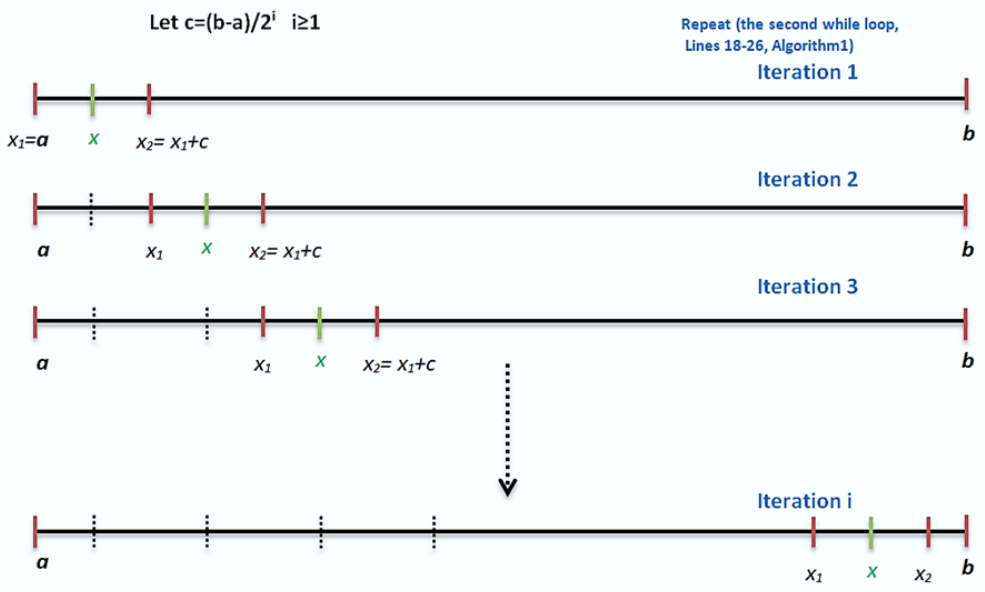

Figure 1. generating test points in a round i.

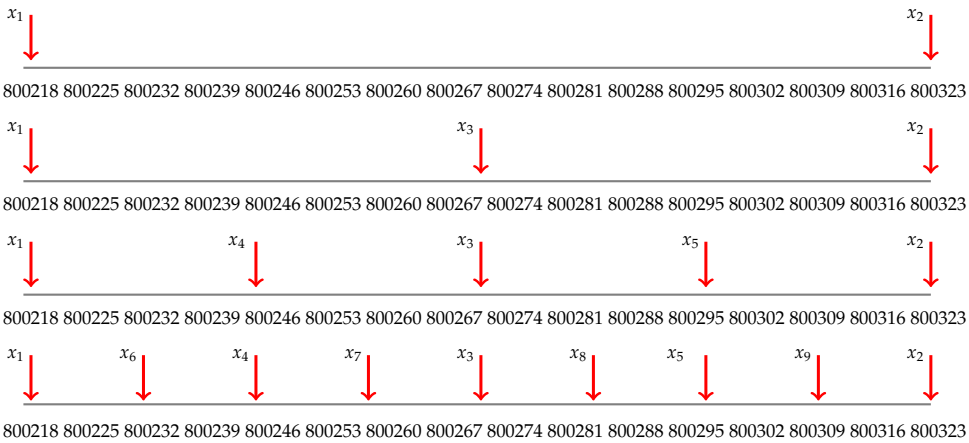

Figure 2. Generating test points in $I = [800218, 800323]$, with rounds $0, 1$, and 2.

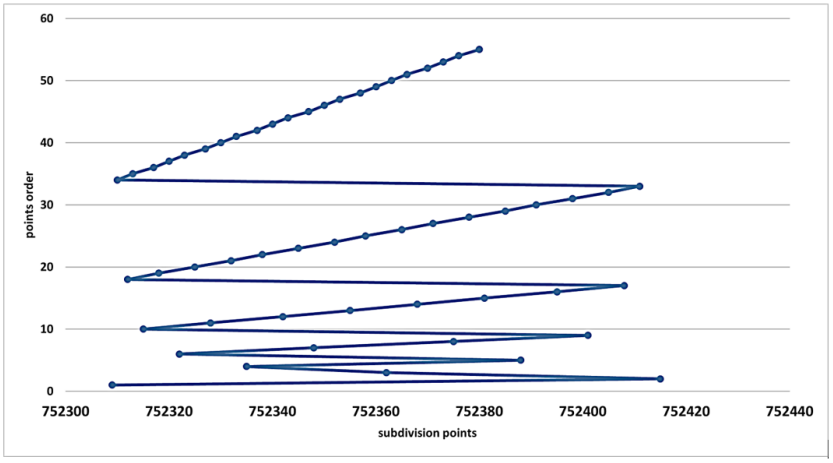

Figure 3. The scatter of test points.

Algorithm 1: Search for d

Input: RSA public key (e, n), an interval $I = [a, b]$, where $\phi(n) \in I$, and a maximum number L of points.
Output: the private exponent d, or 0.
Begin
if *k/d is a convergent of $\mathcal{CF}(e/a)$* **then**
    return d
else
    if *k/d is a convergent of $\mathcal{CF}(e/b)$* **then**
        return d
    else
        $i = 0$
        $c = b - a$
        $counter = 0$
        while *$counter < L$ and $c > 1$* **do**
            $x_1 = a$
            $x_2 = x_1 + c$
            $x = \lfloor (x_1 + x_2)/2 \rfloor$
            while *$x < b$ and $counter < L$* **do**
                $counter = counter + 1$
                if *k/d is a convergent of $\mathcal{CF}(e/x)$* **then**
                    return d
                end if
                $x_1 = x_2$
                $x_2 = x_1 + c$
                $x = \lfloor (x_1 + x_2)/2 \rfloor$
            end while
            $c = \lfloor c/2 \rfloor$
            $i = i + 1$
        end while
        return 0 – not found
    end if
end if
end

5. Implementation

In this section, we present the implementation of the proposed attack. The implementation is written in C/C++, compiled with GNU C++ Compiler, and run on an Intel(R) Xeon(R) E5645 CPU 2.40GHz running the Ubuntu operating system. We used GMP package [23], a free library for arbitrary precision arithmetic, and OpenMP (Open Multi-Processing) [24] to support multiprocessing programming in C.

The implementation considers the three attacks described in Section 3. If $\phi(n)$ is expected to be in an interval $I = [a, b]$, then we distribute I on 20 threads. Let $S = \{a_0 = a, a_1, \ldots, a_{20} = b\}$ be the set of end points of the 20 sub-intervals of $[a, b]$. Then thread number i, $1 \leq i \leq 20$, independently runs Algorithm 1 on the sub-interval $[a_{i-1}, a_i[$. The size of the RSA modulus n conducted in the experimental study was 1024 bits, where each prime factor has 512 bits and was generated randomly. For most studied cases, the number of tested n is 100. The maximum value of test points was $L = 10^7$.

The First Attack: We consider the first attack in Section 3. We assume that $e \approx n$, $d = n^\delta$ and $p_1 - p_2 = n^\beta$, i.e., $\phi(n) \in [n + 1 - \sqrt{n^{2\beta} + 4n}, n + 1 - 2\sqrt{n}]$. Based on Equations (1) and (2), we study the performance of using the proposed technique to attack RSA when β in the range $0.36 \sim 0.5$. For each selected value of β, we study the possibility of attack for different values of δ.

Table 1 shows the average execution time and the (ceiling of) the average number of tested points of running the attack using single core and 20-cores. For $\beta = 0.5, 0.46, 044, 0.4$, and 0.36 we study δ in the ranges 0.256∼0.268, 0.296∼0.308, 0.316∼0.328, 0.356∼0.368, 0.396∼0.408, respectively. All values of δ in the table are greater than the bound of de Weger [6] using continued fractions, Equation (1). This means that the proposed method using continued fractions and 20 cores succeeded to extend the bound of d. Furthermore, the results in the table show that δ is in the range of de Weger's results [6] using lattice, Equation (2) The parallel (multicore) implementation of the attack speeds up the sequential implementation by 18.1 on average.

Table 1. Performance of the first attack ($p_1 - p_2 = n^\beta$ and $d = n^\delta$) using 20 cores where t is the average of execution time (seconds) and l is the ceiling of the average number of test points.

β	δ	Nu. Samples	One Thread	20 Threads	Speedup (in Time)
0.5	0.256	100	$t = 0.043$, l =94	$t = 0.002$, $l = 4$	21.5
	0.26	100	$t = 7.77$, $l = 15877$	$t = 0.42$, $l = 856$	18.5
	0.264	100	$t = 2002.6$, $l = 3097665$	$t = 111.7$, $l = 112900$	17.9
	0.268	20	-	$t = 39739.6$, $l = 500482$	
0.46	0.296	100	$t = 0.063$, $l = 118$	$t = 0.003$, $l = 7$	21
	0.3	100	$t = 10.854$, $l = 23867$	$t = 0.5781$, $l = 1153$	19
	0.304	100	$t = 3296.8$, $l = 3660146$	$t = 173.93$, $l = 137063$	18.9
	0.308	20	-	$t = 57539.5$, $l = 555210$	
0.44	0.316	100	$t = 0.031$, $l = 63$	$t = 0.002$, $l = 4$	15.5
	0.32	100	$t = 5.22$, $l = 11104$	$t = 0.28$, $l = 576$	18.6
	0.324	100	$t = 1000.23$, $l = 1406319$	$t = 56.68$, $l = 57944$	17.6
	0.328	20	-	$t = 13288.79$, $l = 172930$	
0.4	0.356	100	$t = 0.034$, $l = 71$	$t = 0.002$, $l = 4$	17
	0.36	100	$t = 5.024$, $l = 12147$	$t = 0.28$, $l = 574$	17.9
	0.364	100	$t = 1926.7$, $l = 2158924$	$t = 115.7$, $l = 85716$	16.6
	0.368	20	-	$t = 10148.54$, $l = 130304$	
0.36	0.396	100	$t = 0.069$, $l = 68$	$t = 0.004$, $l = 3$	17.25
	0.4	100	$t = 5.37$, $l = 12993$	$t = 0.29$, $l = 605$	18.5
	0.404	100	$t = 1319.3$, $l = 2102342$	$t = 77.32$, $l = 79480$	17.0
	0.408	20	-	$t = 17501.6$, $l = 222104$	

The Second Attack: We consider the second attack in Section 3. We assume that $e \approx n$, $d = n^\delta$ and $2p_2 - p_1 \leq n^\theta$, i.e., $\phi(n) \in [n + 1 - \frac{3\sqrt{2}}{2}\sqrt{n}, n + 1 - \frac{3\sqrt{2}}{2}\sqrt{n} + n^{2\theta - 0.5 + \epsilon}]$. Based on Equation (3), we study the performance of using multicore systems to attack RSA when θ in the range $0.36 \sim 0.46$. For each selected values of θ, we study the possibility of attack for different values of δ.

Table 2 shows the average execution time and the (ceiling of the) average number of tested points of running the attack using single core and 20-cores. For $\theta = 0.46, 0.44, 0.4$, and 0.36, we study δ in the ranges 0.296∼0.308, 0.316∼0.328, 0.356∼0.368 and 0.396∼0.408, respectively.

All values of δ in the table are greater than the bound of Maitra-Sarkar [9] using continued fractions, i.e., $\delta < 3/4 - \theta - \tau$. This means that the proposed method using continued fractions and 20 cores succeeded to extend the bound of d. The parallel (multicore) implementation of the attack speeds up the sequential implementation by 17.3 on average

The Third Attack: we consider the third attack in Section 3. We assume that an approximation p_0 of p_1 is obtained where $|p_0 - p_1| < \frac{1}{2}n^\alpha$, i.e.,

$$\phi(n) \in [n + 1 - (p_0 + \frac{n}{p_0}) - \frac{1}{2}n^\alpha, n + 1 - (p_0 + \frac{n}{p_0}) + \frac{1}{2}n^\alpha].$$

We study the performance of using multicore systems to attack RSA when $\alpha > 0.25$, and and δ as in Equation (4). We choose α in the range 0.36∼0.46. For each selected value of α, we study the possibility of the attack for different values of δ.

Table 2. Performance of the second attack ($2p_2 - p_1 = n^{\theta}$ and $d = n^{\delta}$) using 20 cores where t is the average execution time (seconds) and l is the ceiling of the average number of test points.

θ	δ	Nu. Samples	One Thread	20 Threads	Speedup (in Time)
0.46	0.296	100	$t = 0.035,\ l = 53$	$t = 0.002,\ l = 4$	17.5
	0.3	100	$t = 3.69,\ l = 6639$	$t = 0.23,\ l = 486$	16.0
	0.304	100	$t = 973.7,\ l = 394798$	$t = 61.38,\ l = 15819$	15.8
	0.308	20	-	$t = 20716.3,\ l = 266455$	
0.44	0.316	100	$t = 0.020,\ l = 38$	$t = 0.001,\ l = 2$	20
	0.32	100	$t = 1.91,\ l = 4488$	$t = 0.10,\ l = 207$	19.1
	0.324	100	$t = 543.2,\ l = 213884$	$t = 30.19,\ l = 7812$	17.9
	0.328	20	-	$t = 7901.3,\ l = 101421$	
0.4	0.356	100	$t = 0.019,\ l = 34$	$t = 0.001,\ l = 2$	19
	0.36	100	$t = 1.50,\ l = 3096$	$t = 0.09,\ l = 202$	16.6
	0.364	100	$t = 501.4,\ l = 220657$	$t = 27.43,\ l = 7077$	18.2
	0.368	20	-	t =6355.7 $l = 82008$	
0.36	0.396	100	$t = 0.019,\ l = 37$	$t = 0.001,\ l = 2$	19
	0.4	100	$t = 1.52,\ l = 2551$	$t = 0.10,\ l = 203$	15.2
	0.404	100	$t = 464.9,\ l = 311345$	$t = 32.83,\ l = 8331$	14.1
	0.408	20	-	$t = 4624.7,\ l = 58978$	

Table 3 shows the average execution time and the (ceiling of the) average number of tested points of running the attack using single core and 20-cores. For $\alpha = 0.46, 0.44, 0.4$, and 0.36 we study δ in the ranges 0.274∼0.286, 0.284∼0.296, 0.304∼0.316, and 0.324∼0.336, respectively.

All values of δ in the table are greater than the bound of Equation (4) using continued fractions. This means that the proposed method using continued fractions and 20 cores succeeded to extend the bound of d. The parallel (multicore) implementation of the attack speeds up the sequential implementation by 14.9 on average.

Table 3. Performance of the third attack ($p_2 < p_1 < 2p_2$, $|p_0 - p_1| = \frac{1}{2}n^{\alpha}$ and $d = n^{\delta}$) using 20 cores where t is the average of execution time (seconds) and l is the ceiling of the average number of test points.

α	δ	Nu. Samples	One Thread	20 Threads	Speedup (in Time)
0.46	0.274	100	$t = 0.013,\ l = 26$	$t = 0.001,\ l = 2$	13
	0.278	100	$t = 0.44,\ l = 1121$	$t = 0.03,\ l = 53$	14.6
	0.282	100	$t = 91.64,\ l = 57602$	$t = 5.189,\ l = 1342$	17.6
	0.286	20	-	$t = 966.6,\ l = 13539$	
0.44	0.284	100	$t = 0.012,\ l = 21$	$t = 0.001,\ l = 2$	12
	0.288	100	$t = 0.28,\ l = 610$	$t = 0.02\ \ l = 35$	14
	0.292	100	$t = 250.3,\ l = 144888$	$t = 13.49,\ l = 3480$	18.5
	0.296	20	-	$t = 3324.8,\ l = 41458$	
0.4	0.304	100	$t = 0.015,\ l = 31$	$t = 0.001,\ l = 2$	15
	0.308	100	$t = 0.50,\ l = 1441$	$t = 0.03,\ l = 71$	16.6
	0.312	100	$t = 144.0,\ l = 97241$	$t = 8.39,\ l = 2185$	17.1
	0.316	20	-	$t = 1818.5,\ l = 24390$	
0.36	0.324	100	$t = 0.012,\ l = 25$	$t = 0.001,\ l = 2$	12
	0.328	100	$t = 0.27,\ l = 515$	$t = 0.02,\ l = 34$	13.5
	0.332	100	$t = 58.12,\ l = 36556$	$t = 3.81,\ l = 997$	15.2
	0.336	20	-	$t = 4342.7,\ l = 52888$	

Table 4 shows the upper bound of δ for the proposed attacks and previous attacks [6,9,14] using continued fractions. The proposed attack raises the previous bound of δ by η. As we can see from Tables 1–3, the value of η depends on the number of generated test points in I. The execution times required to complete the attacks depend on the number of cores, type of attack, and η. For example, if $\eta = 0.016$, then the execution time to find the private key for the third attack (Table 3) is 44 min on average, while the execution times are 7.67 h for the first attack (Table 1) and 2.7 h for the second attack (Table 2).

Table 4. Comparison in the upper bound of δ between the proposed and previous attacks using continued fractions, where η is a small positive number.

Conditions of Attacks	Bound of δ	Our Result
$\lvert p_1 - p_2 \rvert \leq n^{\beta}$ $0.25 \leq \beta \leq 0.5$	[6] $\delta < 3/4 - \beta$	$\delta < 3/4 - \beta + \eta$
$\lvert 2p_2 - p_1 \rvert \leq n^{\theta}$ $0.25 \leq \theta \leq 0.5$	[9] $\delta < 3/4 - \theta$	$\delta < 3/4 - \theta + \eta$
$p_0 \geq \sqrt{n}$ is an approximation for p_1 $\lvert p_1 - p_0 \rvert \leq \frac{1}{8} n^{\alpha}, \alpha \leq \frac{1}{2}$	[14] $\delta < \frac{1-\alpha}{2}$	$\delta < \frac{1-\alpha}{2} + \eta$

6. Complexity Analysis

Let m_0 be an approximation for $\phi(n)$. In the following lemma, we show the relationship between the difference $m_0 - \phi(n)$ and the upper bounds of e and d.

Lemma 2. *Let n be a positive composite integer, $e = n^{\xi}$, $d = n^{\delta}$ and $e, d < \phi(n)$, where $ed = 1 + k\phi(n)$. If $|m - \phi(n)| = n^{\gamma}$ and $8(1+\epsilon) < n^{2-(\xi+\gamma+2\delta)}$ where $m, \phi(n) > n/2$, then k/d is a convergent of $\mathcal{CF}(e/m)$, i.e.,*

$$\left| \frac{e}{m} - \frac{k}{d} \right| < \frac{1}{2d^2}$$

Proof. We have

$$\begin{aligned} \left| \frac{e}{m} - \frac{k}{d} \right| &= \left| \frac{1 + k\phi(n) - km}{md} \right| \\ &< \left| \frac{2k(n^{\gamma} + 1)}{nd} \right| \end{aligned} \tag{5}$$

Also, we have $k = \frac{ed-1}{\phi(n)} < \frac{2ed}{n}$, $k < 2n^{\xi+\delta-1}$. Therefore, Equation (5) leads to

$$\begin{aligned} \left| \frac{e}{m} - \frac{k}{d} \right| &< 4n^{\xi-2}(n^{\gamma} + 1) \\ &< 4(1+\epsilon)n^{\xi+\gamma-2} \\ &< \frac{1}{2} n^{-2\delta} = \frac{1}{2d^2} \end{aligned}$$

□

Suppose that $\phi(n)$ is in an interval $[a, b]$, i.e., $a \leq \phi(n) \leq b$. We show, in the following theorem, the relationship between the length $b - a$ of this interval and the running time to retrieve the private exponent d.

Theorem 5. *Let $(n, e = n^{\xi})$ be a public key of RSA and $d = n^{\delta}$ be the corresponding private exponent. Suppose that we can estimate $\phi(n) \in [a, b]$ for two known values a and b and we divide $[a, b]$ into $S - 1$ subintervals of the same size such that*

$$\frac{b-a}{S} \leq \frac{1}{4(1+\epsilon)} n^{2-(\xi+2\delta)}$$

for a small value ϵ. Then d can be obtained in time in $l \log n$.

Proof. Let $\{m_1, m_2, \cdots, m_S\}$ be points of a subdivision for the interval $[a, b]$ where $m_{i+1} - m_i = \lfloor \frac{b-a}{S} \rfloor$ for $i = 1, 2, \ldots, S-1$. We test for every m_i whether k/d is a convergent of $\mathcal{CF}(e/m_i)$. Let m_{i_0} satisfies that

$$\left|m_{i_0} - \phi(n)\right| \leq |m_i - \phi(n)|, 1 \leq i \leq S.$$

Thus, $|m_{i_0} - \phi(n)| \leq \lfloor \frac{b-a}{2S} \rfloor$. Thus, we have

$$\begin{aligned} |m_{i_0} - \phi(n)| &\leq \frac{b-a}{2S} \\ &\leq \frac{1}{8(1+\epsilon)} n^{2-(\xi+2\delta)} \end{aligned}$$

Let $\frac{b-a}{2S} = n^{\gamma}$, for some real number γ. Then, we have $8(1+\epsilon) < n^{2-(\xi+\gamma+2\delta)}$. By Lemma 2, k/d is a convergent of $\mathcal{CF}(e/m_{i_0})$. Since computing $\mathcal{CF}(e/m_i)$ takes a polynomial time in $\log n$, so to test all e/m_i, $i = 1, 2, \cdots, S$, we need a time of order $S \log n$. □

Theorem 5 shows that the complexity of the proposed method depends on the size of S besides the length of I.

7. Conclusions and Future Works

The RSA cryptosystem is used in the most popular security products and protocols in use today. We have presented a new technique to improve a small private exponent attack on RSA. We have successfully raised the upper bound of the private exponent d by $\eta = 0.016$ using continued fractions and multicore systems for three small private exponent attacks in RSA: de Weger [6], Maitra-Sarkar [9], and Nassr et al. [14]. The average execution times for the attacks are 7.67 h, 2.7 h, and 44 min, respectively. These results were obtained using 20 cores and for n with 1024 bits. The execution time and the value of η can be improved by

1. Finding a shorter interval for $\phi(n)$,, i.e., finding better lower and upper bounds of $\phi(n)$. In particular, when the prime factors p_1 and p_2 satisfy some conditions as in the three attacks.
2. Improving test points generation to find a value close to $\phi(n)$. We have presented a new strategy (Algorithm 1) to generate such points.
3. Increasing the number of cores.

Increasing the number of cores is necessary to complete the attack in a reasonable time, but it is expected that increasing the number of cores only will not increase η dramatically since the proposed interval for $\phi(n)$ is not small.

The results presented in the paper can be extended to different variations of RSA such as [25–30]. The results can also be applied to different attacks [4,31] on the private exponent of RSA that use lattices instead of continued fractions. It is also possible to use cloud systems (with thousands of cores) to implement the attacks.

Thus, interesting research questions raised by this study are (1) how to get better lower and upper bounds of $\phi(n)$? (2) how to improve test point generation.

Author Contributions: Conceptualization, H.M.B. (Hatem M. Bahig) and D.I.N.; methodology, H.M.B. (Hatem M. Bahig), D.I.N. and H.M.B. (Hazem M. Bahig); software, D.I.N. and M.A.M.; validation, H.M.B. (Hatem M. Bahig), D.I.N. and H.M.B. (Hazem M. Bahig); formal analysis, H.M.B. (Hatem M. Bahig) and D.I.N.; data curation, D.I.N. and H.M.B. (Hatem M. Bahig); writing—original draft preparation, H.M.B. (Hatem M. Bahig) and D.I.N.; writing—review and editing, H.M.B. (Hazem M. Bahig); visualization, H.M.B. (Hatem M. Bahig), D.I.N. and M.A.M.; supervision, H.M.B. (Hazem M. Bahig); project administration, H.M.B. (Hazem M. Bahig); funding acquisition, H.M.B. (Hazem M. Bahig). All authors have read and agreed to the published version of the manuscript.

Funding: This research has been funded by Scientific Research Deanship at University of Ha'il—Saudi Arabia through project number RG-21 124.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors are grateful to the referees for their valuable comments and remarks. The authors would like to acknowledge the support provided by Scientific Research Deanship at University of Ha'il—Saudi Arabia through project number RG-21 124.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rivest, R.; Shamir, A.; Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [CrossRef]
2. Shor, P. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [CrossRef]
3. Wiener, M. Cryptanalysis of short RSA secret exponents. *IEEE Trans. Inf. Theory* **1990**, *36*, 553–558. [CrossRef]
4. Boneh, D.; Durfee, G. Cryptanalysis of RSA with private key d less than $N^{0.292}$. *IEEE Trans. Inf. Theory* **2000**, *46*, 1339–1349. [CrossRef]
5. Coppersmith, D. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *J. Cryptol.* **1997**, *10*, 233–260. [CrossRef]
6. De Weger, B. Cryptanalysis of RSA with Small Prime Difference. *Appl. Algebra Eng. Commun. Comput.* **2002**, *13*, 17–28. [CrossRef]
7. Blömer, J.; May, A. A Generalized Wiener Attack on RSA. In *Public Key Cryptography—PKC 2004, Proceedings of the 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, 1–4 March 2004*; Lecture Notes in Computer Science (Volume 2947); Springer: Berlin/Heidelberg, Germany, 2004; pp. 1–13.
8. Jochemsz, E.; May, A. A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants. In *Advances in Cryptology—ASIACRYPT 2006, Proceedings of the 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, 3–7 December 2006*; Lecture Notes in Computer Science (Volume 4284); Springer: Berlin/Heidelberg, Germany, 2006; pp. 267–282.
9. Maitra, S.; Sarkar, S. Revisiting Wiener's Attack—New Weak Keys in RSA. In *Information Security, Proceedings of the 11th International Conference, ISC 2008, Taipei, Taiwan, 15–18 September 2008*; Lecture Notes in Computer Science (Volume 5222); Springer: Berlin/Heidelberg, Germany, 2008; pp. 228–243.
10. Nitaj, A.; Ariffin, M.R.K.; Nassr, D.I.; Bahig, H.M. New Attacks on the RSA Cryptosystem. In *Progress in Cryptology—AFRICACRYPT 2014, Proceedings of the P7th International Conference on Cryptology in Africa, Marrakesh, Morocco, 28–30 May 2014*; Springer International Publishing: Cham, Switzerland, 2014; pp. 178–198.
11. Bahig, H.M.; Mahdi, M.A.; Alutaibi, K.A.; AlGhadhban, A.; Bahig, H.M. Performance Analysis of Fermat Factorization Algorithms. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 51–60. [CrossRef]
12. Bahig, H.M.; Bahig, H.M.; Kotb, Y. Fermat Factorization using a Multi-Core System. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*. [CrossRef]
13. Bahig, H.M. Speeding Up Fermat's Factoring Method using Precomputation. *Ann. Emerg. Technol. Comput.* **2022**, *6*, 51–60. [CrossRef]
14. Nassr, D.I.; Bahig, H.M.; Bhery, A.; Daoud, S.S. A new RSA vulnerability using continued fractions. In Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications, Doha, Qatar, 31 March–4 April 2008; pp. 694–701.
15. Bahig, H.M.; Nassr, D.I.; Bhery, A.; Nitaj, A. A Unified Method for Private Exponent Attacks on RSA Using Lattices. *Int. J. Found. Comput. Sci.* **2020**, *31*, 207–231. [CrossRef]
16. Jones, W.B.; Thron, W.J. Continued Fractions: Analytic Theory and Applications. In *Encyclopedia of Mathematics and Its Applications*; Cambridge University Press: Cambridge, UK, 1984; pp. 17–26
17. Cuyt, A.A.; Petersen, V.; Verdonk, B.; Waadeland, H.; Jones, W.B. *Handbook of Continued Fractions for Special Functions*, 1st ed.; Springer: Dordrecht, The Netherlands, 2008.
18. Steinfeld, R.; Contini, S.; Pieprzyk, J.; Wang, H. *Converse Results to the Wiener Attack on RSA*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2005; pp. 184–198.
19. Stein, W. *Elementary Number Theory: Primes, Congruences, and Secrets: A Computational Approach*; Undergraduate Texts in Mathematics; Springer: Berlin/Heidelberg, Germany, 2008.
20. Dujella, A. Continued fractions and RSA with small secret exponent. *Tatra Mt. Math. Publ.* **2004**, *29*, 101–112.
21. May, A. New RSA Vulnerabilities Using Lattice Reduction Methods. Ph.D. Thesis, University of Paderborn, Paderborn, Germany, 2003. Available online: http://www.cs.uni-paderborn.de/uploads/tx_sibibtex/bp.pdf (accessed on 6 March 2022).
22. Kaedi, S.; Doostari, M.; Ghaznavi-Ghoushchi, M.B.; Yusefi, H. A New Side-Channel Attack on Reduction of RSA CRT Montgomery Method Based. *J. Circuits Syst. Comput.* **2020**, *30*, 2150038. [CrossRef]
23. GNU MP. The GNU Multiple Precision Arithmetic Library, 6.2.1 ed. 2020. Available online: http://gmplib.org/ (accessed on 1 August 2022).

24. OpenMP Architecture Review Board. OpenMP Application Program Interface Version 5.2. 2021. Available online: https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5-2.pdf (accessed on 1 August 2022).
25. Bahig, H.; Bhery, A.; Nassr, D. Cryptanalysis of Multi-Prime RSA with Small Prime Difference. In *Information and Communications Security, Proceedings of the 14th International Conference, ICICS 2012, Hong Kong, China, 29–31 October 2012*; Lecture Notes in Computer Science (Volume 7618); Springer: Berlin/Heidelberg, Germany, 2012; pp. 33–44.
26. Nassr, D.I.; Anwar, M.; Bahig, H.M. Improving small private exponent attack on the Murru-Saettone cryptosystem. *Theor. Comput. Sci.* **2022**, *923*, 222–234. [CrossRef]
27. Bunder, M.; Nitaj, A.; Susilo, W.; Tonien, J. A generalized attack on RSA type cryptosystems. *Theor. Comput. Sci.* **2017**, *704*, 74–81 [CrossRef]
28. Nitaj, A.; Kamel Ariffin, M.R.; Hanisah Adenan, N. N.; Azman Abu, N. Classical Attacks on a Variant of the RSA Cryptosystem. In *Progress in Cryptology—LATINCRYPT 2021, Proceeding of the 7th International Conference on Cryptology and Information Security in Latin America, Bogotá, Colombia, 6–8 October 2021*; Lecture Notes in Computer Science (Volume 12912); Springer: Berlin/Heidelberg, Germany, 2021; pp. 151–167.
29. Nitaj, A.; Kamel Ariffin, M.R.; Hanisah Adenan, N. N.; Chien Lau, T. S.; Chen, J. Security Issues of Novel RSA Variant. *IEEE Access* **2022**, *10*, 53788–53796. [CrossRef]
30. Abd Ghafar, A.H.; Kamel Ariffin, M.R.; Asbullah, M.A. A New LSB Attack on Special-Structured RSA Primes. *Symmetry* **2020**, *12*, 838 [CrossRef]
31. Durfee, G. Cryptanalysis of RSA Using Algebraic and Lattice Methods. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 2002. Available online: http://theory.stanford.edu/~gdurf/durfee$-$thesis$-$phd.pdf (accessed on 6 March 2022).