Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/37/20230503

Parallel implementation of Wiener’s attack on RSA: Algorithm design and performance evaluation

Jiaming Lu

University of Toronto, Toronto, M1C 1A4, Canada

jiaming.lu@mail.utoronto.ca

Abstract. The robustness of the RSA cryptosystem is intrinsically tied to the choice of public and private keys. As pinpointed by M. Wiener, should the private decryption exponent, d, be improperly chosen-either disproportionately large or unduly small in relation to the public key n-an adversary could feasibly deduce the private keys within a practical time span. In this paper, the algorithm invented by Wiener is revisited and find a way to parallelize the attacking algorithm by using OpenMP. The algorithm is based on the proof made in Wiener’s article, that if $d < \frac{1}{3}n^{\frac{1}{4}}$, then the private key d is the denominator of one of the convergent of $\mathcal{CF}\left(\frac{e}{n}\right)$. Using a constant set of private keys (e,n), an extensive series of simulated attacks employing Wiener’s method was conducted on a laptop to determine the optimal thread count for executing the parallel algorithm. The results indicate that the algorithm’s execution speed can be enhanced by a factor of 1.5607 (rounded to five significant figures). Specifically, while the sequential version of the algorithm averaged a runtime of 3518939.04 nanoseconds, its parallel counterpart averaged 2240493.28 nanoseconds.

Keywords: RSA, Parallel Computing, OpenMP, Wiener’s Attack, Small Private Exponent Attack.

1. Introduction

The RSA cryptosystem, one of the most enduring encryption methods, remains widely used today. Since its inception in 1978 by Ron Rivest, Adi Shamir, and Leonard Adleman [1], numerous variations have emerged to meet the continually evolving demands of the industry. The implementation of the RSA cryptosystem is small and sweet, with three private keys (p, q, d) and two public keys $(e, n = pq)$. By using some modular exponentiation, messages can be transmitted securely after encryption. The RSA cryptosystem operates asymmetrically, utilizing distinct keys for encryption and decryption. Its security hinges on the challenging task of factoring substantial composite numbers, typically those exceeding 2048 bits in length.

Developing attacks on the RSA cryptosystem helps the reinforcement of the security of the system. Since the publishment of the RSA cryptosystem, the system has been analyzed by numerous researchers to find its vulnerability [2]. Past studies have shown that if the private keys or public keys of the RSA system are not chosen carefully, the cryptosystem is feasible to be broken by a third party in a reasonable amount of time [2-4]. However, once the flaw of the cryptosystem is known, improvement can be made and thus the system is more robust. This paper aims to accelerate the traditional algorithms in Wiener's attack [5], where the private key d is relatively small compared to the public key n.

It is natural to believe that the security of the RSA cryptosystem has a deep relationship with the bit lengths of the private keys (p, q) chosen - the longer of the bit length, the harder for attackers to factorize the public key n. Parallelism provides a shortcut for sequential algorithms with high demand in computing resource to reduce the time spend for execution. Therefore, implementing a parallel version of Wiener's attack helps attackers to utilize the computing resources and makes the attacker break the cryptosystem in a reasonable amount of time.

2. Preliminaries about RSA and Wiener's attack

2.1. Mathematical background

Cryptography is inextricably linked to the foundational principles of mathematics, particularly number theory. In this section, key definitions and significant theorems are presented to set a uniform notation and terminology for the subsequent discussion in this paper.

Definition 1 (Greatest common divisor). *For every pair of non-zero $a, b \in \mathbb{Z}$, the greatest common divisor of a, b, is written by $gcd\ (a, b)$, is the largest $c \in \mathbb{Z}$, such that $a = cd$ and $b = ce$ for some integers $d, e \in \mathbb{Z}$.*

Definition 2 (Coprime). *For every pair of $a, b \in \mathbb{Z}$, a and b are coprime if and only if* $\gcd(a, b) = 1$.

Definition 3 (Euler's totient function). *The Euler's totient function of a positive $n \in \mathbb{Z}$, is written by $\phi(n)$, is the number of integers $k \in [1, n]$ satisfying* $\gcd(k, n) = 1$.

Definition 4 (Congruence). *For every $a, b \in \mathbb{Z}$, and for some $n \in \mathbb{Z}$, a and b are congruent under modulo n, is written by $a \equiv b \ mod\ n$, if and only if there exists some $k \in \mathbb{Z}$ such that $a = kn + b$.*

Theorem 5 (Euler-Fermat's Theorem). *For every $a, b \in \mathbb{Z}$, and for some $n \in \mathbb{Z}$, if* $\gcd(a, n) = 1$, *then $a^{\phi(n)} \equiv 1 \ mod\ n$.*

Definition 5 (Divisibility). *For every integer $a, b \in \mathbb{Z}$, a is divisible by b, is written by $a \mid b$, if and only if there exists some integer $c \in \mathbb{Z}$ such that $a = bc$.*

Definition 6 (Floor function). *For every $a \in \mathbb{Z}$, the floor of a, written as $\lfloor a \rfloor$, is the greatest integer that is less for equal to a.*

2.2. Basics about RSA

2.2.1. Key generation. Suppose Alice wants to send message m to Bob by using the RSA cryptosystem, Bob will set up the system via following procedures:

a. Choose two large primes $p, q \in \mathbb{Z}$, and compute their product $n = pq$.
b. Choose a positive integer $d \in \mathbb{Z}$, such that $\gcd(d, \phi(n)) = 1$.
c. Find the multiplicative inverse e of d under modulo $\phi(n)$, i.e., $ed \equiv 1 \ mod\ \phi(n)$.
d. Publish the public keys (n, e), and keep secret keys (p, q, d) in private.

2.2.2. Encryption. For every message m that Alice sends to Bob, suppose m is coprime to the public key n, Alice can encrypt the message by:

$$c \equiv m^e \ mod\ n \tag{1}$$

And Alice will send the ciphertext c to Bob.

2.2.3. Decryption. When Bob receives the ciphertext c, he can decrypt it by:

$$m \equiv c^d \ mod\ n \tag{2}$$

The correctness of decryption process follows directly from applying the assumption of $\gcd(m, n) = 1$ and Euler-Fermat's Theorem.

2.3. Basics about Wiener's attack on RSA

Definition 7 (Continued fraction). *A continued fraction expansion of a rational number* $\frac{u}{v} \in \mathbb{Q}$*, sometime is abbreviated as* $\mathcal{CF}\left(\frac{u}{v}\right)$*, is an expansion of the form:*

$$\frac{u}{v} = x_0 + \frac{1}{x_1 + \frac{1}{\ddots + \frac{1}{x_n}}} \tag{3}$$

where x_0 *is an integer and for every integer* $i \in [1, n]$*,* x_i *is a positive integer.*

By using the Euclidean division algorithm, the continued fraction expansion of any rational number $\frac{u}{v}$ can be computed within $O(\log(\max(u, v)))$ time.

Example 8. *To construct the continued fraction expansion of* $\frac{2000}{2019}$*, by using the Euclidean division algorithm, we have:*

$$\begin{aligned} 2000 &= \mathbf{0} \times 2019 + 2000 \\ 2019 &= \mathbf{1} \times 2000 + 19 \\ 2000 &= \mathbf{105} \times 19 + 5 \\ 19 &= \mathbf{3} \times 5 + 4 \\ 5 &= \mathbf{1} \times 4 + 1 \\ 4 &= \mathbf{4} \times 1 \end{aligned}$$

Hence, the continued fraction expansion of $\frac{2000}{2012}$ *has coefficients:*

$$x_0 = 0,\ x_1 = 1,\ x_2 = 105,\ x_3 = 3,\ x_4 = 1,\ x_5 = 4.$$

Definition 9 (Convergent). *The convergent of continued fraction expansion of* $\frac{u}{v} \in \mathbb{Q}$ *is a series of approximation* c_i*, obtained by truncating the coefficients:*

$$c_0 = x_0,\ c_1 = x_0 + \frac{1}{x_0},\ c_2 = x_0 + \frac{1}{x_1 + \frac{1}{x_2}}, \cdots, c_n = x_0 + \frac{1}{x_1 + \frac{1}{\ddots + \frac{1}{x_n}}} \tag{4}$$

Where each c_i *is called the* i*-th convergent of the continued fraction expansion of* $\frac{u}{v}$.

Theorem 10 (Euler-Wallis Theorem [6]***).*** *For any* $j \geq 1$*, the* j*-th convergent* $c_j = \frac{a_j}{b_j}$ *can be determined by the following recursive equation:*

$$\begin{array}{llll} a_{-2} = 0, & a_{-1} = 1, & a_j = x_j a_{j-1} + a_{j-2}, & \forall j \geq 0 \\ b_{-2} = 1, & b_{-1} = 0, & b_j = x_j b_{j-1} + b_{j-2}, & \forall j \geq 0 \end{array}$$

3. Wiener's attack

Wiener's attack is also known as the attack on low private exponent. It was firstly shown by M. Wiener [5] that if the private exponent $d < \frac{1}{3}n^{\frac{1}{4}}$, then there exists a linear running time algorithm that can be used to recover the private key d. The attack of Wiener's method relies on continued fraction expansion and convergent of the rational number $\frac{e}{n}$. In Wiener's research, he proved that if the following conditions are met:

a. $q < p < 2q$,
b. $0 < e < \phi(n)$,
c. $ed - k\phi(n) = 1$,
d. $d < \frac{1}{3}n^{\frac{1}{4}}$,

then the private exponent d is the denominator among one of the convergent of $\frac{e}{n}$.

However, there is research have shown that the bound $d < \frac{1}{3}n^{\frac{1}{4}}$ is not accurate, reference [7] offers a detailed experiment that Wiener's attack fails when

$$d = \left\lfloor \frac{1}{2} n^{\frac{1}{4}} \right\rfloor + 1$$

and ends the whole article with a conjecture that the accurate abound for Wiener's attack to succeed is $d < \frac{1}{18^{\frac{1}{4}}} n^{\frac{1}{4}}$. Moreover, there are variations on Wiener's attack that provides a larger bound for the private exponent [4, 8].

Table 1. Algorithm design.

Algorithm 1: Wiener's attack

Input: Two public keys (e, n)
Output: The private key (d, p, q) or $\perp$

```
1.  def attack(e, n):
2.    calculate the continued fraction expansion of $\frac{e}{n}$ via Euclidean division algorithm
3.    calculate the convergent of $\mathcal{CF}\left(\frac{e}{n}\right)$ by using Euler-Wallis Theorem
4.    for $0 \le i \le n$:
5.      if $a_i | (eb_i - 1)$:
6.        $\lambda_i = \frac{eb_i - 1}{a_i}$
7.        $S = n - \lambda_i + 1$
8.        find the roots $p', q'$ of the equation $x^2 - Sx + n = 0$
9.        if root exists AND $p' \times q' == n$:
10.         return $(d = b_i, p = p', q = q')$
11. return $\perp$
```

In lines 2-3 of Table 1, two arrays are initialized containing the continued fraction expansion and convergents of fraction $\frac{e}{n}$. Then, beginning at line 4, for each element c_i in the array of convergent, the convergent can be written as $c_i = \frac{a_i}{b_i}$, we check if $eb_i - 1$ is divisible by a_i. If divisible, then the root of equation $x^2 - Sx + n = 0$, where $S = n - \lambda_i + 1$ and $\lambda_i = \frac{eb_i - 1}{a_i}$, is possible to be the private key (p, q) [7]. The root of equation $x^2 - Sx + n = 0$ is given by the quadratic formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{5}$$

with $a = 1, b = -S, c = n$. It is important to check the if discriminant $\Delta = b^2 - 4ac$ is non-negative before taking the square root to prevent arithmetic error. If $\Delta > 0$, indicates root exists, then we test if the root obtained by the quadratic formula is the factor n. When the factorization is found, the denominator of the corresponding convergent in current is returned, otherwise enter the next iteration. After checking all the convergent within the array, if the private key is still not found, $\perp$, also known as bottom, is returned at line 11, to indicate the failure of the attack.

3.1. Implementation details

The Wiener's attack in its parallel form was developed using C/C++ in conjunction with OpenMP (Open Multi-Processing) to facilitate multi-threaded computations [9]. Additionally, the GMP library was employed to manage arithmetic operations on integers with unfettered precision [10]. By prefixing the attacking algorithm's fourth line with "#pragma omp parallel for schedule(dynamic)", a cluster of threads is spawned. These threads collaboratively execute commands within the loop, allocating tasks dynamically without adhering to any predetermined sequence.

4. Analysis of the result

To measure the execution time of the program, high resolution clock in chrono library is used to record the start time T_s and finish time T_f of the attack function [11]. The execution time is defined the difference between start and finish time.

$$T_{execution} = T_f - T_s \tag{6}$$

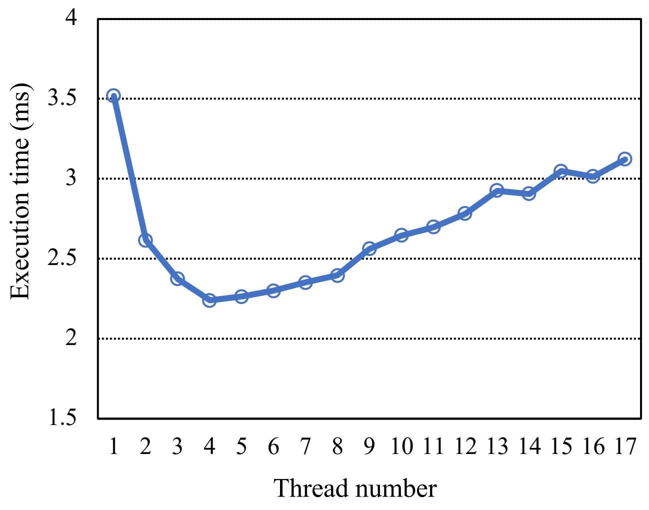

Figure 1. Running time of Wiener's attack against thread number.

The experiment was performed on Intel(R) Core (TM) i7-6700HQ CPU @ 2.60GHz with Manjaro operating system [12]. The public key used was:

e=30749686305802061816334591167284030734478031427751495527922388099381921172620
569310945418007467306454160014597828390709770861577479329793948103408489494025272
8344735558548350441533749785544144163050122676439578389986486511007054468759795736
75767605387333733876537528353237076626094553367977134079292593746416875606876735717
905892280664538346000950343671655257046364067221469807138232820446015769882472160
5518400529219303579883343066591202531147906384964800923619515365764272957894291974
8359785965797783236891253476110026906550935134505075894367465105341998256109443225
8103614830448382949765459939698951824447818497599,

n=10996616399290324377064345629609375913073751033373648335234548864343261420103
0629970207047930115652268531222079508230987041869779760776072105738457123387124961
036111210544028669181361694095594938869077306417325203381820822917059651429857093
3886188184372826248579275512858115426852692297055941663704261521288959019147099020
3736565257573020189736113951881616474622873341028359523640598595841449137230187871
8635708605256444921222945267625853091126691358833453283744166617463257821375566155
6758684520324019617278143144813434677022999494079356023893421835362228425569066570
01984320973035314726867840698884052182976760066141.

To bolster the dependability of experimental outcomes, a shell script was devised to execute the attack repeatedly, subsequently calculating the average of the data amassed. In each iteration, Wiener's attack was undertaken with varying thread counts, spanning from 1 to 17. As shown in Figure 1. Results indicate peak performance at four threads, registering an average runtime of 2,240,493.28 nanoseconds, a maximum of 2,566,219 nanoseconds, and a minimum of 2,144,689 nanoseconds. In juxtaposition with the sequential rendition of Wiener's attack – which recorded an average runtime of 3,518,939.04

nanoseconds, a peak at 4,919,243 nanoseconds, and a trough at 3,209,481 nanoseconds – the parallel version, when operating on four threads, manifested a performance enhancement by a factor of 1.5607. Notably, the data's standard deviation was minimized (78,107.77506) at this four-thread configuration. As shown in Figure 2.

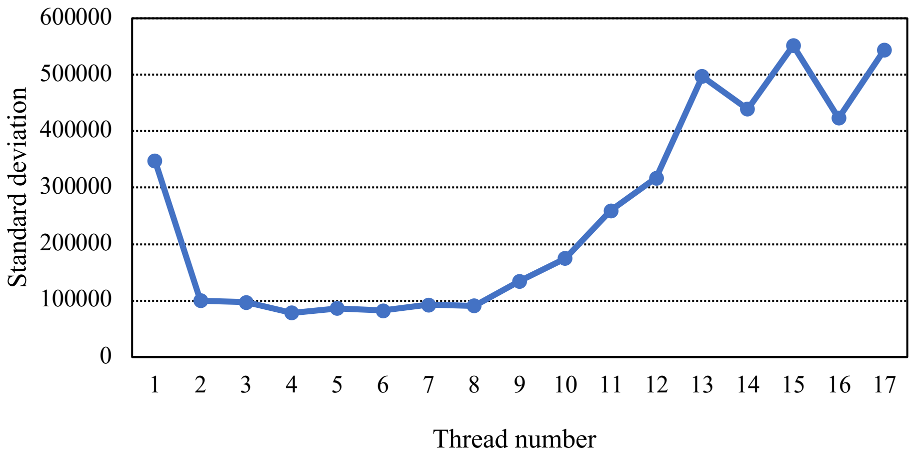

Figure 2. Standard deviation of data collected against thread number.

5. Conclusion

Upon the development of a sequential algorithm, there often arises a compelling interest to explore its parallel version and subsequently assess its performance. This study delves into the intriguing task of parallelizing the time-honored Wiener's attack algorithm, leveraging the capabilities of the OpenMP API. The findings are illuminating: the parallelization using OpenMP indeed augments the performance of Wiener's attack, with the most optimal results surfacing when the thread count stands at four. As avenues for future research unfold, there lies a rich potential in employing diverse multi-threading APIs such as MPI and the thread library intrinsic to C/C++. By doing so, a comprehensive evaluation can be made, comparing the running times across these parallel implementations and providing valuable insights into the optimal methodology for such cryptographic attacks.

References

[1] Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120-126.

[2] Boneh, D. (1999). Twenty years of attacks on the RSA cryptosystem. Notices of the AMS, 46(2), 203-213.

[3] d de la Fe, S., Park, H. B., Sim, B. Y., Han, D. G., & Ferrer, C. (2021). Profiling attack against RSA key generation based on a Euclidean algorithm. Information, 12(11), 462.

[4] Susilo, W., Tonien, J., & Yang, G. (2020). A generalised bound for the Wiener attack on RSA. Journal of Information Security and Applications, 53, 102531.

[5] Wiener, M. J. (1990). Cryptanalysis of short RSA secret exponents. IEEE Transactions on Information theory, 36(3), 553-558.

[6] Hardy, G. H., & Wright, E. M. (1979). An introduction to the theory of numbers. Oxford university press.

[7] Susilo, W., Tonien, J., & Yang, G. (2019, May). The Wiener attack on RSA revisited: a quest for the exact bound. In Australasian Conference on Information Security and Privacy (pp. 381-398). Cham: Springer International Publishing.

[8] Steinfeld, R., Contini, S., Wang, H., & Pieprzyk, J. (2005). Converse results to the Wiener attack on RSA. In Public Key Cryptography-PKC 2005: 8th International Workshop on Theory and

Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005. Proceedings 8 (pp. 184-198). Springer Berlin Heidelberg.
[9] Lewis, T. (n.d.). Home. OpenMP. https://www.openmp.org/.
[10] The GNU MP Bignum Library. (n.d.). Gmplib.org. https://gmplib.org.
[11] Cppreference.com. (2017). Date and time utilities. https://en.cppreference.com/w/cpp/chrono.
[12] Manjaro. (2011). #FREE OPERATING SYSTEM FOR EVERYONE. https://manjaro.org/.