Decimal Expansion from the Difference of Two Irrational Roots

Lomba Peneliti Belia Nasional 2020
Kinglouis Steven Tantra
SMA Santo Aloysius 1 Bandung
kinglouisteven@gmail.com

Introduction

Although technology advancements have made our lives easier and better, they pose several risks. One of the most fatal risks is the stealing of some private and important data. Therefore, the role of cyber security is really crucial, mainly in terms of providing stronger encryption and security systems. One of the strongest encryption still applicable until now is the RSA (Rivest – Shamir – Adleman) Algorithm, an asymmetrical algorithm or usually known as public key cryptography. This algorithm consists of 2 keys, one is the public key, deemed non-confidential, and the other is the private key, deemed confidential with its own role as a decryptor. Main strength of the RSA algorithm is the difficulties to factorize big integers to its prime factors.

Based on the RSA algorithm, this research is conducted to try a similar problem. RSA consists of 2 multiplicated prime integers, in this research, meanwhile, this research applies 2 non-square integers in which their square root is subtracted and the decimal expansion is taken.

Decimal expansion of every irrational number from roots of non-square integers is a unique sequence, therefore by subtracting 2 different irrational roots, it is hoped that it will result in a unique sequence too. The uniqueness of the series of these integers could be utilized for storing encrypted information. For instance, one integer is the information, and the other is the key. The bigger the integer and the longer decimal expansion of the root, it is hoped that the computational time to find both the initial integers (decrypting without key) will become longer. The main aim of this research is to determine the proper condition for resulting in a unique decimal sequence.

Research Methodology

Formulations used in this research are described as follows :

- a, b : a pair of positive non-square integers, where $a > b$ and $2 \leq \{a, b\} \leq k$
- k : upper limit for a and b, represented in the form of $k = 10^m$
- $p = \sqrt{a} - \sqrt{b}$
- S : fractional part of decimal expansion of p
- n : number of decimal digits of S

In this research, C++ programming language with MPFR (multiple precision floating-point reliable) library is used. The library could handle decimal expansion into high precision, for instance, more than 1000 digits.

On the first stage of the research, the value of a and b will be restricted, for instance, $a, b < k = 1000$ with the number of decimal digits $n = 10$. An algorithm will be made to list all possibilities of a and b. Nevertheless, a and b should be a non-square integer, therefore, a square number checker function will be made first. Afterwards, the value of S will be generated. The program will sort all pairs by the decimal expansion to find duplicates. If there exists the same decimal expansion from different pairs of a and b, therefore, no unique value of a and b exists. This phenomenon will be analyzed to find the relation between a and b which results in the value of p which is not unique.

On the second stage, a search algorithm to find back the value of a and b if n digits decimal expansion of p is known will be made. In this algorithm, the results do not need to be stored in memory, only the final result will. Thus, greater value of k and n could be tried.

The algorithm used is illustrated in these steps :

1) Input k and string S.
2) Search all possible values for a and b with different filters.
3) If the decimal expansion from $\sqrt{a} - \sqrt{b}$ match with S, the output will be a and b.
4) If no combinations of a and b give matching decimal expansion, the program outputs “not found”.

The time required to do the search will be observed. The value of n and k will be set to become larger and larger until the searching time becomes so long, for instance, one hour. If the result of a and b found by the algorithm is different from the original, it will be stored to be observed to see the relation of a and b resulting in the case. Time required to do the search will also be noted to find the correlation between n and k with time.

The approach used in this research is brute – force algorithm, finding one by one from all possibilities. Theoretically this algorithm will have the time complexity of O(k^2).

The results of this research are more likely to be illustrated in the form of a table consisting of the values of a and b resulting in a non-unique S to maintain readability. Moreover, it is possible that graphs will also be used to show the correlation between the addition of the value of n and k towards time.

Expected result from this research is that there exists a condition where the problem finally becomes practically unsolvable and requires a really long computation time. With the occurrence of the condition, this problem is hoped to be developed into a new strong encryption/decryption algorithm to become an alternative in addition to current used algorithms.

Result and Analysis

Several experiments have been conducted and the results are presented as follows :

For the first stage where the algorithm used is in purpose to sort all pairs by the decimal expansion to find duplicates. Were only the square number filter applied, there would be many combinations with identical decimals, as there are some groups which have the same square-free factor and the integers could be represented in the form $a = q^2 z$ and $b = r^2 z$ with q,r and z are square-free numbers, and there are many combinations of q and r which output the same result if a and b are square-rooted.

As a consequence, eliminating integers having the same square-free factor procedure is applied as the second filter. The results are shown in Table 1. Moreover, another filter is tried, in which both a and b are non-squarefree numbers, where the results are shown in Table 2.

k	n	Total Combinations	Total Collisions
10	1	20	11
10	2	20	1
10	3	20	0
100	4	3945	725
100	5	3945	101
100	6	3945	9
100	7	3945	0
1000	6	467789	96809
1000	7	467789	11834
1000	8	467789	1185
1000	9	467789	110
1000	10	467789	15
1000	11	467789	0
10000	10	48981352	126291
10000	12	48981352	1362
10000	15	48981352	2
10000	16	48981352	0

Table 1 - Both a and b are Non-Square Integers with no Common Square-Free Factors.

k	n	Total Combinations	Total Collisions
10	1	21	11
10	2	21	2
10	3	21	0
100	4	1830	184
100	5	1830	22
100	6	1830	1
100	7	1830	0
1000	6	184528	16523
1000	7	184528	1919
1000	8	184528	188
1000	9	184528	9
1000	10	184528	0
1000	15	184528	0
10000	10	18498403	17485
10000	14	18498403	2
10000	15	18498403	0
15000	20	41582640	0

Table 2 - Both a and b Are Non-Squarefree Numbers

The total collisions shows the values S which are not unique. After the results of both filters have been analyzed, there is no correlation found between a and b which results in the same value of S. Moreover, after being checked, it could be seen that their decimal sequences after the n^{th} digit are not the same. Therefore, the phenomenon could be considered as a coincidence for their first n digits to be similar.

Total combinations compared from Table 1 and Table 2 have no significant difference for $m = 1$. Still, for $m \geq 2$, it shows a relation in which total combinations from Table 1 is roughly 2.5 to 2.6 times more than Table 2.

One of the most striking findings is when the value of n is increased by one, the total collisions will decrease to about 10%. For instance, observe table 1 cases. For $k = 1000$, $n = 7$, the total collisions value is 11834. For $k = 1000$, $n = 8$, the total collisions value is 1185, which is approximately 10% from the previous value of n. The same phenomenon is also observed in table 2.

Based on several observations towards Table 1, number of n of each m which results in 0 total collisions are :

- $m = 1, n = 3$
- $m = 2, n = 7$
- $m = 3, n = 11$

- $m = 4, n = 16$

Some slight differences are found in Table 2, as the value of m and n which results in 0 total collisions are :

- $m = 1, n = 3$
- $m = 2, n = 7$
- $m = 3, n = 10$
- $m = 4, n = 15$

Based on the pattern observed, for k to be ten times larger than the previous, the value of n needs to be increased possibly by 3 to 5 for resulting in 0 collisions.

As shown on Figure 1 and Figure 2, the value of n for some random value of m could be determined from the equations. For instance, if the value of m is 20, therefore, the value of n will be 84 or 85 if the first filter is applied and 77 if the second filter is applied.

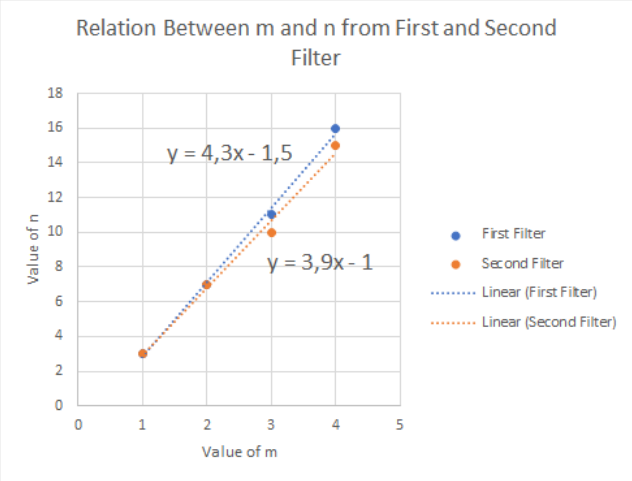

Figure 1 - Relation Between m and n from First and Second Filter

As have been predicted before the experiment has been conducted, this algorithm consumes a lot of memory, almost 4GB, and trying for $k > 10^4$ is a little bit risky due to the capacity of the author's laptop RAM.

For the second stage, where a search algorithm to find back the value of a and b with $n = 15$, the relation between k and time is illustrated on Table 3 and Figure 2. When $a, b < 10000$, only a single solution is found, consistent with the previous results. Meanwhile, it is also tried with larger a, b (last row of Table 3), which is not known if there are any collisions or not. It turned out that there is still a single solution for the test number (a = 19404, b = 10647) Nevertheless, to ensure the upper limit for no collisions more cases should be tried, which require an incredibly long time ($T \sim 2 \times 10^5$ s = 2.3 days just for 1 case with m = 5) unless a faster algorithm is developed, which is subject for future works.

As shown on Figure 2, as k increases, the time also fits with a second order polynomial graph in respect with the value of k. Consequently, it is concluded that time increases in polynomial form and could be explained by analyzing the algorithm used to perform the experiment.

In the algorithm, total numbers checked from 1 to k is $k(k-1)/2$, minus the square integers and integer pairs having the same square-free factor. Nevertheless, it will have no effect as only $k(k-1)/2$ factor will be considered in the count. The term of $k(k-1)/2$ is usually considered as k^2 in determining time complexity. Therefore, the algorithm has the time complexity of O(k^2), as expected from Brute-Force Algorithm.

Even though the time fits with a second order polynomial graph with respect to k, it is exponential with respect to m.

k	Time required to find back a and b (seconds)
100	0
1000	11
10000	1349
20000	5985

Table 3 - Relation Between the Value of k vs Time

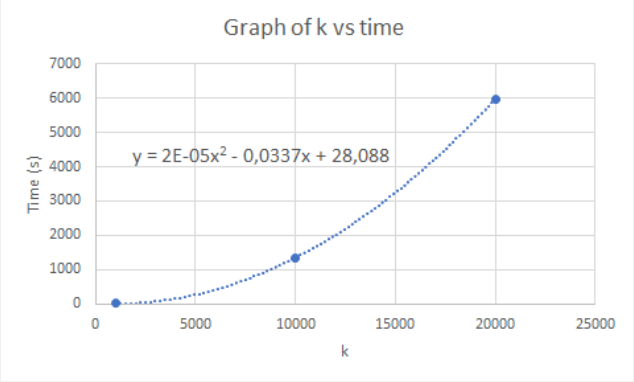

Figure 2 - Relation Between the Value of k vs Time

Conclusion

A research has been conducted to explore the decimal expansion from the difference of two irrational roots. Based on the data taken on the experiment, evidence is shown that this problem could be categorized as a hard problem. Should the problem develop more, there will be a likeability for this problem to become a new alternative for a strong encryption / decryption algorithm.

Future Work

More research should be conducted to apply larger integers, or to develop a better algorithm to replace the current Brute-Force algorithm. A functional encryption and decryption algorithm could be developed as a proof of concept as well.

References

Munir,Rinaldi. 2016. Matematika Diskrit. Bandung;Penerbit Informatika.

Rosen,Kenneth H. 2012. Discrete Mathematics and Its Applications, Seventh Edition. New York; McGraw-Hill.

Cormen,Thomas H., Charles E. Liserson, Ronald L. Rivest, Clifford Stein. 2009. Introduction to Algorithms, Third Edition. Cambridge; The MIT Press.

Ireland,David. 2020. RSA Algorithm. https://www.di-mgt.com.au/rsa_alg.html. Accessed on 6th July 2020.

Lake,Josh. 2018. What is RSA encryption and how does it work?. https://www.comparitech.com/blog/information-security/rsa-encryption/. Accessed on 6th July 2020.

Munir, Rinaldi. 2018. Algoritma RSA. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2017-2018/Algoritma-RSA-(2018).pdf. Accessed on 6th July 2020.

Mann,Adam. 2019. What Are Irrational Numbers?. https://www.livescience.com/irrational-numbers.html. Accessed on 6th July 2020.

Dehal,Aditya. 2019. An Introduction to the Time Complexity of Algorithms. https://www.freecodecamp.org/news/time-complexity-of-algorithms/. Accessed on 6th July 2020.

Speciner,Mike. 2019. What is the time complexity for an RSA algorithm? What is a solution with the derivation of the complexity?. https://www.quora.com/What-is-thetime-complexity-for-an-RSA-algorithm-What-is-a-solution-with-the-derivation-of-thecomplexity. Accessed on 6th July 2020.

Ikatan Alumni Tim Olimpiade Komputer Indonesia. Analisis Kompleksitas. https://github.com/ia-toki/training-gate-id-pdf/raw/master/pemrograman-dasarcpp_08-analisis-kompleksitas.pdf. Accessed on 6th July 2020.

Inverse Symbolic Calculator. http://wayback.cecm.sfu.ca/projects/ISC/ISCmain.html. Accessed on 13th July 2020.