Journal of Theoretical and Applied Information Technology
15th August 2020. Vol.98. No 15

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

RSA PUBLIC KEY SOLVING TECHNIQUE BY USING GENETIC ALGORITHM

[1]DIAN RACHMAWATI, [2]HILDA AYU TAMARA, [3]SAJADIN SEMBIRING, [4]MOHAMMAD ANDRI BUDIMAN

[1,2,3,4]Departemen Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi,

Universitas Sumatera Utara, Jl. Universitas No. 9-A, Kampus USU, Medan 20155, Indonesia

E-mail: [1]dian.rachmawati@usu.ac.id

ABSTRACT

The Rivest Shamir Adleman (RSA) algorithm is one of the cryptographic algorithms that have a high level of security in the message security, and this is due to the difficulty of finding the prime number factor of a huge integer (factor n being the two main factors, p, and q so that it becomes n = p · q). Because it is assessed as safe in securing messages, researchers look for weaknesses in the RSA algorithm. The downside of this RSA algorithm is from the key solving technique. The algorithm for solving this key is sought and studied so that the secret key p and q can be known or cryptanalysis. The key solving technique that can be done is to use the heuristic method, which is a genetic algorithm. The process of finding factors from this public key is conducted algorithmically using a genetic algorithm. The genetic algorithm works by factoring the public key n randomly to generate the initial value of the chromosomal candidates p and q. After the initial population is formed, the chromosome will experience evaluation, selection, crossover, and mutation so that the best solution is found. The chromosomes are produced from one generation to a maximum of a predetermined generation. Furthermore, two figures of the results of the factorization must be done checking the prime number, in the study of the test method of prime numbers using the Lehmann algorithm. If the test result of the prime number is correct, then the two numbers are the secret key p and q of the RSA algorithm. The results of the research from solving the technique of public-key RSA algorithms using genetic algorithms suggest that the genetic algorithm can be used to break the public key of the RSA algorithm. The opportunity to find the secret key p and Q RSA algorithm is affected by the size of the pop size and maximum size of the generation, the larger the size of population size and maximum size of the generation, the larger the population size and maximum size of the generation then the higher the probability of the p and Q secret keys found. The size of the crossover Rate (PC) is best used to solve the problem of solving the RSA public key using the genetic algorithm is 25%-50%. While the size of the mutation rate (PM) is best used to resolve the problem of solving the RSA public key, using the genetic algorithm is 10%-20%.
Keywords: *RSA, Cryptography, Cryptanalysis, Heuristic, Genetic, Lehmann.*

1. INTRODUCTION

The Rivest, Shamir, Adleman (RSA) algorithm was published in 1978, the founder Ron Rivest, Adi Shamir, and Leonard Adleman. Currently, the RSA algorithm has been used in almost all areas related to computer network usage. The RSA algorithm has two key types: the public key and the secret key. The public key can be known, used, and learned by anyone. But the secret key is not to be recognized by anyone so that confidentiality can be maintained.

The RSA algorithm is one of the rated cryptographic algorithms to have a high level of security to secure the message, as it performs huge number factoring into its two prime factors (a factor of n Be the two prime factors p and q, so it becomes $n = p \cdot q$). Because it is rated safe in securing messages is not unlikely, the RSA algorithm also has drawbacks. The downside of this algorithm is sought and learned so that the secret keys p and q can be known. One way is to use the method of factorization (factoring the public key n).

The method of factorization is one of the methods used to perform attack techniques on RSA. This attack is also known as a cryptanalysis experiment. Cryptanalysis is a technique that is done both to find the lack or weakness of the

cryptographic system by finding the key as well as obtaining the original message (plaintext) [1].

In previous research (Zeni Fera Bhakti, 2013) titled Cryptoanalysis RSA with Elliptical Curves concluded that cryptanalysis RSA with an elliptical curve is a technique to solve the message by finding the factor of a number *n* by utilizing The rule of summing dots on the elliptic curve. This factoring aims to find a secret key [2]. The research (Ade Chandra, Mohammad Andri Budiman, Dian Rachmawati, 2017), titled On Factoring RSA Modulus using Taboo Search, concluded that taboo search is not an excellent candidate to determine *n* public key RSA widened key Its corresponding secrets *p* and *q*. Taboo Search also takes more time to count the *n* digits than the Pollard algorithm, which calculates the larger *n* digits in less time [3].

From the conclusion of both researches, researchers interested in researching RSA public key solving techniques to factoring the public key *n* with different methods of factoring with the purpose of heuristic genetic algorithm, expected result of solving the public key *n* RSA has a relatively less time to find the secret key *p* and *q*.

The algorithm used to perform cryptanalysis is genetics. Genetic algorithms are often used on practical issues that focus on searching for parameters or optimal solutions. Excess genetic algorithm is the ability to gain global optima in search of solutions, so it is often used in optimization. But in addition to the excess genetic algorithm has a weakness, that is to take a long time in the process of searching the solution [12].

With this research, in addition to obtaining a private RSA key using genetic algorithms, this study will also analyze pop size parameters, crossover rates, mutation rates, and maximum generation of genetic algorithms to perform cryptoanalysis on RSA algorithms.

2. CRYPTOGRAPHY

Cryptography comes from the Greek language Crypto and Graphia. Crypto means secret, and Graphia means writing. According to its terminological, cryptography is the science and art that is useful to protect a message that the sender will send to the recipient [4].

The cryptographic components consist of several components, such as:

1. Encryption
2. Decryption
3. Key
4. Ciphertext
5. Plaintext
6. Messages
7. Cryptanalysis

Based on the key that the cryptography is divided into two parts:

1. Symmetrical Cryptography

   Symmetrical cryptography is a conventional cryptographic algorithm that uses the same key for the encryption process and decryption process. Examples of symmetrical cryptographic algorithms are DES (Data Encryption Standard), Blowfish, Twofish, MARS, IDEA, 3DES (DES Applied 3 times), AES (Advanced Encryption Standard) called the original Rijndael.

2. Asymmetric Cryptography

   Asymmetric cryptography (public key) is an algorithm that uses different keys for encryption and decryption processes. The encryption key can be shared publicly and named as public key, while the decryption key is stored for self-use and is designated as a private key. Examples of algorithms using asymmetric keys are RSA (Rivest Shamir Adleman), Rabin, and ECC (Elliptic Curve Cryptography).

3. TEST THE PRIME NUMBER OF LEHMANN ALGORITHM

Lehmann's algorithm is required when a prime number has reached a large number and cannot be recalled for sure. Generally, prime numbers do not follow a definite pattern, and to know a prime number or not; it is not easy [10]. A number is said to be prime when the number is only consumable with the number 1 and the number itself [5]. The steps are as follows:

1. Generate a random number a, $1 < a < p$ (p is the number that is tested in prime).
2. Calculate the Legendre symbol:
   $$\alpha = a^{(p-1)/2} \bmod p$$
3. If $\alpha \neq 1$ and $\alpha \neq p - 1$, then p is not prime.
4. If $\alpha = 1$ and $\alpha = p - 1$, then the odds of prime with probabilistic $\geq 50\%$.

4. RSA ALGORITHM

The RSA algorithm includes asymmetric cryptography. RSA is one of the public key

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

cryptosystems. The RSA algorithm can factorize a huge number [4]. For that reason, RSA is considered safe. To generate two keys, selected two large random prime numbers [11].

The RSA algorithm was created by three researchers from MIT (Massachusetts Institute of Technology) in 1978, namely Ron Rivest, Adi Shamir, and Leonard Adleman. RSA is the most popular public-key RSA cryptographic system because it is the first system that can be used for confidentiality, key distribution, and digital signatures. The RSA algorithm has the following magnitudes:

1. p and q prime numbers (secret)
2. $n = p \cdot q$ (not secret)
3. $\phi(n) = (p-1)(q-1)$ (secret)
4. e (encryption key) (not secret)
5. d (decryption key) (secret)
6. m (plaintext) (secret)
7. c (ciphertext) (not secret)

The security of the RSA algorithm lies in the difficulty of factoring the non-prime number into the primes factor, which in this case $n = p \cdot q$. Once n was successfully factored into p and q, then $\phi(n) = (p-1)(q-1)$ could be calculated. Next, because the e encryption key is announced (not secret), then the decryption key d can be calculated from equation $d.\ e \equiv 1 \pmod{\phi(n)}$ [4].

Key generation processes are performed by the receiver. The RSA algorithm key generation steps are as follows.

1. Generate two prime numbers, i.e., p and q, with the condition $p \neq q$.
2. Calculate the value $n = p \cdot q$.
3. Calculate $\phi(n) = (p-1)(q-1)$.
4. Select the public key e randomly, with the condition:
   - $1 < e < \phi(n)$
   - GCD $(e, \phi(n)) = 1$
5. Generate secret key d, with condition $d.\ e \equiv 1 \pmod{\phi(n)}$.

After the recipient generates the public key, then the recipient sends the key to the sender to do the message encryption process. The message encryption process is as follows:

1. Specify the message (plaintext) m to be sent and convert it to decimal number with ASCII encoding.
2. Get the recipient's public key, i.e., e and n.
3. Calculate c (ciphertext) with the formula:
   $$c = m^{e} \bmod n$$
4. Send ciphertext to the recipient.

The decryption of the message is performed by the message recipient so that the messages received are ciphertext converted to the original (plaintext) Form for readability. The decryption of the message is as follows:

1. Accept c (ciphertext) from the sender of the message.
2. Retrieve the secret key of the message recipients, i.e., d and n.
3. Decrypt c (ciphertext) to m (plaintext) with the formula:
   $$m = c^{d} \bmod n$$

5. CRYPTANALYSIS

Cryptanalysis is the science and art to break the ciphertext into plaintext without knowing the keys and algorithms used. Cryptanalysis is a study of mathematical techniques on experiments to defeat cryptographic methods [6]. Cryptanalysis is also a technique that is done both to find the lack or weakness of the cryptographic system by discovering the key as well as obtaining the original message (plaintext) [1].

Cryptanalysis is said to be successful if it can restore plaintext or find its key. In this case, it is assumed that a cryptanalyst already knows the algorithm used in encryption and decryption in detail. People who do cryptanalysis are called cryptanalysts.

6. HEURISTIC

G.J. Reiner (1997) says that a heuristic sense is a technique, an art and not a science. Thus, the heuristic does not have general regulations. Heuristic itself is often interpreted as a skill in discovering, identifying, or detailing a bibliography or classifying and caring for records [7].

A heuristic is also known as an approximate technique. The heuristic quest's primary purpose is to build a model that can be easily understood and can provide a suitable solution in reasonable computing time [8]. The heuristic search method has been widely used in business, economics, sports, environment, statistics, pharmaceuticals, and engineering problems that are difficult to solve.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

7. GENETIC ALGORITHM

The genetic algorithm is a computing algorithm inspired by the evolutionary theory that is then adopted into a computing algorithm to find solutions to a problem more naturally. This algorithm is a computing algorithm inspired by Darwin's evolutionary theory, stating that the survival of a creature influenced by the "Strong is the winning" rule.

For the first time, the genetic algorithm was introduced by John Holland in the early 1970s in New York, USA. The concept used in the genetic algorithm is to follow what is done by nature [9].

The problem of combination optimization is one of the applications of a genetic algorithm to get an optimal solution value to a problem that has many possible solutions. The parameters used in the genetic algorithm are as follows:

1. **A chromosome** is a solution that will be resurrected.
2. **The population** is a collection of chromosome-chromosome.
3. **Genes** are elements that compose a chromosome.
4. **The generation** is a chromosome-chromosome that will undergo changes sustainably.
5. **objective_function** is a chromosome-chromosome that is rated to have a better success rate on a generation than the previous generation of the problem to be resolved.
6. **Fitness** is a measure used to assess the success rate of a chromosome in a generation.
7. **Selection** is a process of selecting the chromosome that will be preserved because it has a good value of solutions for the next generation.
8. **Offspring** is a new chromosome-chromosome result of a crossover process.
9. **Crossover** is a process of intersecting between chromosomes in a generation.
10. **crossover_rate** is the total chromosome in a crossover population.
11. **A mutation** is a process of changing from one or more gene values in a chromosome.
12. **mutation_rate** is the total gene in a population that has a mutation.

How the genetic algorithm works are as follows:

1. Chromosome formation.
2. Initialize the chromosome.
   The chromosome initialization process is performed by providing an initial value to the chromosome with random numbers as much as the population you want to be resurrected.
3. The chromosome evaluation.
   The chromosome evaluation process is done by solving the problem according to the case that the solution wants to be searched by calculating the **objective_function** of the specified chromosome.
4. Chromosome selection.
   The chromosome selection process is divided into the following steps:
   4.1 Calculate the **fitness** function of each chromosome.
   4.2 Calculating the probability (**P[i]**) using the formula:
   P[i] = fitness[i] / total_fitness
   4.3 Calculates the cumulative value of probability (**C[i]**).
   4.4 The chromosome selection process uses a roulette wheel, with the condition:
   - Generate random number **R** in range 0-1.
   - If **R[k] < C[1]**, then select chromosome 1 as the parent.
   - If **C[k-1] < R < C[k]**, then select the chromosome to-k as the parent.

   4.5 Make the selected chromosome as a new selection result.
5. Crossover
   The first step in the chromosome crossover process is by specifying the cut-point method to be used; in the case of the study, the method used is a one-cut point, i.e., randomly selecting one position in the parent's chromosome then Exchanged genes. The next step is to choose the parent's chromosome randomly from the number of the chromosome that is experiencing a crossover; the total chromosome that has undergone a crossover is affected by the **crossover_rate** parameter.
6. Mutation
   The mutation process is done by replacing a single randomly selected gene with a new value obtained randomly as well. The Total chromosome that had a mutation in a population was determined by the **mutation_rate** parameter.
7. Repeat step 3 if you have not found the best solution of the chromosome-chromosome resulting from the previous generation until you find the best solution of a problem to look for a solution or to reach maximum generation has been determined.

Journal of Theoretical and Applied Information Technology
15th August 2020. Vol.98. No 15

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

8. GENERATE RSA ALGORITHM KEY

The process of generating a secret key and the public key from the RSA algorithm is done by clicking the Generate Key button. After that, the secret key and public key have been randomly generated, so it will appear in the TextBox with the value P = 239, q = 397, and n = 94883. The Generate secret key and public key test from the RSA algorithm can be seen in Figure 1 below.

Figure 1. Generate RSA Algorithm Key

As seen in Picture 1, the manual calculation is as follows from the result of the RSA key generating algorithm.

1. Generate two prime numbers, i.e., p and q, with the condition $p \neq q$. In the test system, the p and q values obtained are p = 239 and q = 397.

p	q

2. Calculate the value n by using the formula $n = p \cdot q$.
   Example:
   $n = p \cdot q$
   $n = 239 \cdot 397$
   $n = 94883$
   In system tests, the value of n = 94883 is obtained.

9. RSA PUBLIC KEY FACTORING USING GENETIC ALGORITHMS

Factoring public key n RSA algorithm using the genetic algorithm is done by inputting the public key n of the key generate the result on the menu RSA Key Generator, Pop Size, Crossover Rate, Mutation Rate, and Maximum Generation so that it will appear in the textbox with the value n = 94883, Pop Size = 10, Crossover Rate = 25%, Mutation Rate = 10%, and Maximum Generation = 1000. Next, click the Process button, then the public key of the RSA algorithm will be in a process using the method of factoring genetics algorithm so that the result will be displayed on the textbox with the value p = 239 and q = 397. Testing the public key factoring of the RSA algorithm using the genetic algorithm can be seen in Figure 2 below.

Figure 2. RSA Public Key Factoring Using Genetic Algorithms

From the result of the factoring of public key n RSA algorithms using the genetic algorithm, as seen in Figure 2, the manual calculation is as follows.

1. Chromosome formation.
   Because the solution is a p and q value, then the variable p, q, is used as a chromosome forming genes.

2. Initialize the chromosome.
   Specify the population number is 10, then:
   Chromosome[1] = [p ; q] = [97 ; 853]
   Chromosome[2] = [p ; q] = [563 ; 677]
   Chromosome[3] = [p ; q] = [751 ; 781]
   Chromosome[4] = [p ; q] = [773 ; 269]
   Chromosome[5] = [p ; q] = [589 ; 787]
   Chromosome[6] = [p ; q] = [369 ; 29]
   Chromosome[7] = [p ; q] = [787 ; 983]
   Chromosome[8] = [p ; q] = [853 ; 311]
   Chromosome[9] = [p ; q] = [887 ; 947]
   Chromosome[10] = [p ; q] = [241 ; 359]
3. The chromosome evaluation.
   The problem you want to solve is finding the value of the p and q variables that meet the equation **94883** = ***p . q***, then **objective_function(chromosome)** = | **94883** – **(*p . q*)** |.
   objective_function(*chromosome*[1]) = | 94883 – (97.853) | = 12142

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

objective_function(*chromosome*[2]) = | 94883 – (563.677) | = 286268
objective_function(*chromosome*[3]) = | 94883 – (751.781) | = 491648
objective_function(*chromosome*[4]) = | 94883 – (773.269) | = 113054
objective_function(*chromosome*[5]) = | 94883 – (589.787) | = 368660
objective_function(*chromosome*[6]) = | 94883 – (369.29) | = 84182
objective_function(*chromosome*[7]) = | 94883 – (787.983) | = 678738
objective_function(*chromosome*[8]) = | 94883 – (853.311) | = 170400
objective_function(*chromosome*[9]) = | 94883 – (887.947) | = 745106
objective_function(*chromosome*[10]) = | 94883 – (241.359) | = 8364

The average of objective_function are:

Average = (12142+286268+491648+ 113054+368660+84182+ 678738+170400+745106+ 8364) / 10
= 2958562 / 10
= 295856,2

4. Chromosome selection.
   - Calculate fungsi_fitness on each chromosome:

     fitness[1] = 1 / (1+objective_function[1]) = 1 / 12142 = 0,000082351972330
     fitness[2] = 1 / (1+objective_function[2]) = 1 / 286268 = 0,000003493217917
     fitness[3] = 1 / (1+objective_function[3]) = 1 / 491648 = 0,000002033971390
     fitness[4] = 1 / (1+objective_function[4]) = 1 / 113054 = 0,000008845252311
     fitness[5] = 1 / (1+objective_function[5]) = 1 / 368660 = 0,000002712519089
     fitness[6] = 1 / (1+objective_function[6]) = 1 / 84182 = 0,000011878882910
     fitness[7] = 1 / (1+objective_function[7]) = 1 / 678738 = 0,000001473320378
     fitness[8] = 1 / (1+objective_function[8]) = 1 / 170400 = 0,000005868510161
     fitness[9] = 1 / (1+objective_function[9]) = 1 / 745106 = 0,000001342089123
     fitness[10] = 1 / (1+objective_function[10]) = 1 / 8364 = 0,000119545726240
     Total_fitness = 0,000239545461850
   - Calculate the probability on each chromosome:

     P[1] = 0,000082351972330 / 0,000239545461850 = 0,343784314233529
     P[2] = 0,000003493217917 / 0,000239545461850 = 0,014582692948722
     P[3] = 0,000002033971390 / 0,000239545461850 = 0,008490961901148
     P[4] = 0,000008845252311 / 0,000239545461850 = 0,036925150835768
     P[5] = 0,000002712519089 / 0,000239545461850 = 0,011323608756385
     P[6] = 0,000011878882910 / 0,000239545461850 = 0,049589263007231
     P[7] = 0,000001473320378 / 0,000239545461850 = 0,006150483363617
     P[8] = 0,000005868510161 / 0,000239545461850 = 0,024498523645623
     P[9] = 0,000001342089123 / 0,000239545461850 = 0,005602648918528
     P[10] = 0,000119545726240 / 0,000239545461850 = 0,499052352389449
   - Calculate cumulative value probability:

     C[1] = 0,343784314233529
     C[2] = 0,343784314233529+ 0,014582692948722 = 0,358367007182251
     C[3] = 0,343784314233529+ 0,014582692948722+ 0,008490961901148 = 0,366857969083399
     C[4] = 0,343784314233529+ 0,014582692948722+ 0,008490961901148+ 0,036925150835768 = 0,403783119919167
     C[5] = 0,343784314233529+ 0,014582692948722+ 0,008490961901148+ 0,036925150835768+ 0,011323608756385 = 0,415106728675553
     C[6] = 0,343784314233529+ 0,014582692948722+ 0,008490961901148+ 0,036925150835768+ 0,011323608756385+ 0,049589263007231 = 0,464695991682784
     C[7] = 0,343784314233529+ 0,014582692948722+

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

0,008490961901148 +
0,036925150835768+
0,011323608756385+
0,049589263007231+
0,006150483363617
= 0,4708464750464
C[8] = 0,343784314233529+
0,014582692948722+
0,008490961901148+
0,036925150835768+
0,011323608756385+
0,049589263007231+
0,006150483363617+
0,024498523645623
= 0,495344998692023
C[9] = 0,343784314233529+
0,014582692948722+
0,008490961901148 +
0,036925150835768+
0,011323608756385+
0,049589263007231+
0,006150483363617+
0,024498523645623+
0,005602648918528
= 0,500947647610551
C[10] = 0,343784314233529+
0,014582692948722+
0,008490961901148 +
0,036925150835768+
0,011323608756385+
0,049589263007231+
0,006150483363617+
0,024498523645623+
0,005602648918528+
0,499052352389449
= 1

- Chromosome selection using roulette wheel on condition:
  - Generate random number **R** in range 0-1.
    R[1] = 0,146945683819682
    R[2] = 0,137258404929777
    R[3] = 0,332847856140159
    R[4] = 0,801088840142399
    R[5] = 0,965433054121879
    R[6] = 0,616090588558507
    R[7] = 0,96063804671198
    R[8] = 0,102306277073131
    R[9] = 0,139026307099977
    R[10] = 0,760375377144839
  - If **R[k] < C[1],** then select chromosome 1 as the parent.
  - If **C[k-1] < R < C[k],** then select the chromosome to-k as the parent.

Then:
Chromosome [1] = *Chromosome* [1]
Chromosome [2] = *Chromosome* [1]
Chromosome [3] = *Chromosome* [1]
Chromosome [4] = *Chromosome* [10]
Chromosome [5] = *Chromosome* [10]
Chromosome [6] = *Chromosome* [10]
Chromosome [7] = *Chromosome* [10]
Chromosome [8] = *Chromosome* [1]
Chromosome [9] = *Chromosome* [1]
Chromosome [10] = *Chromosome* [10]
Thus, the new chromosome selection results:
Chromosome [1] = [97 ; 853]
Chromosome [2] = [97 ; 853]
Chromosome [3] = [97 ; 853]
Chromosome [4] = [241 ; 359]
Chromosome [5] = [241 ; 359]
Chromosome [6] = [241 ; 359]
Chromosome [7] = [241 ; 359]
Chromosome [8] = [97 ; 853]
Chromosome [9] = [97 ; 853]
Chromosome [10] = [241 ; 359]

5. Crossover
crossover_rate (pc) = 25% = 0,25.
Generate random number **R** as many population
R[1] = 0,255045193366262
R[2] = 0,926347032155072
R[3] = 0,718925365115947
R[4] = 0,729613121007389
R[5] = 0,524259147012727
R[6] = 0,68750028157956
R[7] = 0,301913816622418
R[8] = 0,820768455891296
R[9] = 0,00218154210698863
R[10] = 0,267668404741058
The chromosome to-k will be selected as the parent if **R [K]** < **pc**, from the random number **R** above, then the one used as the parent is **chromosome [9]**.
Since the number of genes in a chromosome is only 2, then the cut-point crossover position is 1 for each chromosome.
offspring[9] = chromosome[9]
X chromosome[9]
= [97 ; 853] X [97 ; 853]
= [97 ; 853]
Thus the chromosome population after experiencing the crossover process becomes:
Chromosome [1] = [97 ; 853]
Chromosome [2] = [97 ; 853]

Chromosome [3] = [97 ; 853]
Chromosome [4] = [241 ; 359]
Chromosome [5] = [241 ; 359]
Chromosome [6] = [241 ; 359]
Chromosome [7] = [241 ; 359]
Chromosome [8] = [97 ; 853]
Chromosome [9] = [97 ; 853]
Chromosome [10] = [241 ; 359]

6. Mutation
   - Total_gene = (number of genes in chromosome) * total population
     = 2 * 10
     = 20
   - mutation_rate (pm) = 10% = 0,1
   - Number of mutations = 0,1 * 20
     = 2
   - Generate random numbers to select the position of the gene that has been mutations, then selected numbers 7 and 9. Thus the position of the selected 7th and 9th genes has a mutation.
   - A random number that is generated to replace the value of the gene is 733 and 431. Thus the chromosome population after experiencing the mutation process is:
     Chromosome [1] = [97 ; 853]
     Chromosome [2] = [97 ; 853]
     Chromosome [3] = [97 ; 853]
     Chromosome [4] = [**733** ; 359]
     Chromosome [5] = [**431** ; 359]
     Chromosome [6] = [241 ; 359]
     Chromosome [7] = [241 ; 359]
     Chromosome [8] = [97 ; 853]
     Chromosome [9] = [97 ; 853]
     Chromosome [10] = [241 ; 359]
   - **objective_function** after a generation are:
     objective_function(*chromosome*[1]) = | 94883 – (97.853) | = 12142
     objective_function(*chromosome*[2]) = | 94883 – (97.853) | = 12142
     objective_function(*chromosome*[3]) = | 94883 – (97.853) | = 12142
     objective_function(*chromosome*[4]) = | 94883 – (733.359) | = 168264
     objective_function(*chromosome*[5]) = | 94883 – (431.359) | = 59846
     objective_function(*chromosome*[6]) = | 94883 – (241.359) | = 836
     objective_function(*chromosome*[7]) = | 94883 – (241.359) | = 8364
     objective_function(*chromosome*[8]) = | 94883 – (97.853) | = 12142
     objective_function(*chromosome*[9]) = | 94883 – (97.853) | = 12142
     objective_function(*chromosome*[10]) = | 94883 – (241.359) | = 8364
     The average of objective_function are:
     Average = (12142+12142+12142+ 168264+59846+8364+ 8364+12142+12142+ 8364) / 10
     = 313912 / 10 = 31391,2

The objective_function value decreases more than the objective_function value before experiencing the selection, crossover and mutation. This indicates that the chromosome or solution produced after one generation is better than the previous generation.

These chromosomes will continue to repeat the process of evaluation, selection, crossover, and mutation as in previous generations to find a chromosome or the best solution to a problem.

After 216 generations obtained the best chromosome are:

Chromosome = [239 ; 397]

If decode then:

p = 239 ; q = 397

If the equation is counted 94883 = p . q

94883 = 239 . 397

10. EXPERIMENTS AND DISCUSSIONS

The experiment was conducted on a Windows 10 Laptop with an Intel Core i3-6006U processor, 64-bit architecture, and 4096 MB of RAM. The Integrated Development Environment (IDE) used for encoding is Microsoft Visual Studio 2017 1.18.1049, and the programming language used in c#.

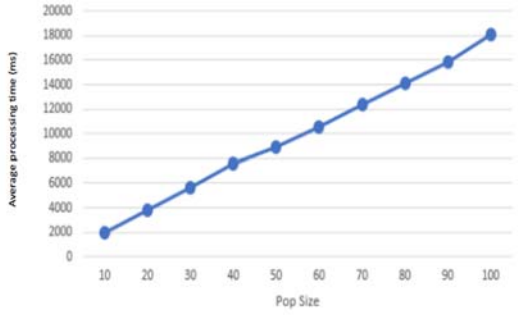

Figure 3. Chart Size Comparison of Pop Size with The Processing Time

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 3 can be seen graphs that show the relationship between the size of Pop size rather than straight to the processing time. This indicates that the bigger the size of the Pop Size then the time used to run the program also the longest.

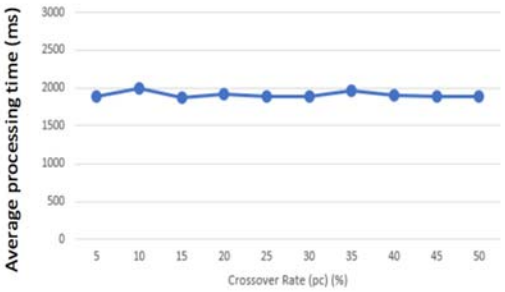

Figure 4. Chart Size Comparison Crossover Rate (pc) with Time Processing

Figure 4 can be seen as a graph showing the relationship between the size of the Crossover Rate (pc) does not affect the processing time. This indicates that the larger the size of the Crossover Rate (pc) then the time used to run the program is not always the longest. Likewise, the smaller the size of the Crossover Rate (pc) then the time used to run the program is not always faster.

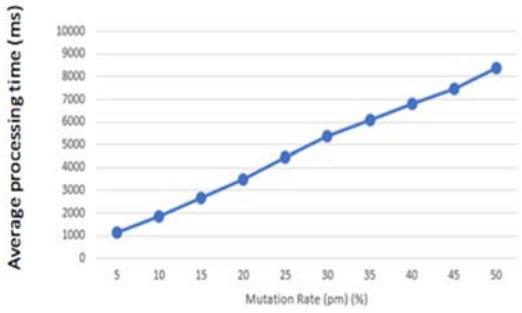

Figure 5. Chart Size Comparison Mutation Rate (pm) with Processing Time

In Figure 5 can be seen as a graph showing the relationship between the size of Mutation Rate (pm) rather than straight to the processing time. This indicates that the larger the size of the Mutation Rate (pm) then the time used to run the program is also getting longer.

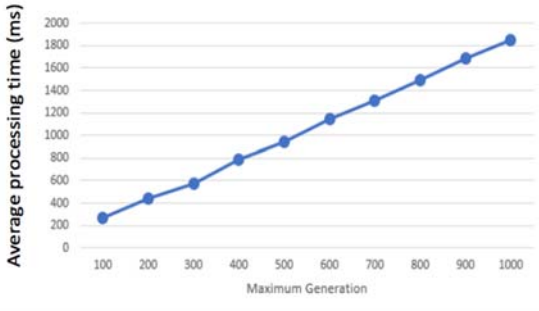

Figure 6. Chart Size Comparison Maximum Generation with Time Processing

Figure 6 can be seen as a graph that shows the relationship between Maximum Generation size is directly proportional to process time. This indicates that the larger the size of the Maximum Generation then the time used to run the program is also longer.

Based on the experiment, the heuristic method of the genetic algorithm can be used to break the public key of the RSA algorithm. The processing time required to get the RSA secret key is affected by the sizes of Pop, Crossover Rate (pc), Mutation Rate (pm), and Maximum Generation.

11. CONCLUSIONS

Based on the tests that have been done on solving the public key RSA algorithm using a method heuristic genetic algorithm, then the conclusion obtained by the author is as follows:

1. The heuristic method of genetic algorithms can be used to break the public key of RSA algorithms.
2. The processing time required to obtain the RSA secret key is affected by the size of the Pop size, Crossover Rate (pc), Mutation Rate (pm), and Maximum Generation.
3. The relationship between the Pop Size is directly proportional to the processing time. This shows that the bigger the size of the Pop Size then the time used to run the program also the longest.
4. The relationship between the size of the Crossover Rate (pc) does not affect the processing time. This indicates that the larger the size of the Crossover Rate (pc) then the time used to run the program is not always the longest. Likewise, the smaller the size of the Crossover Rate (pc) then the time used to run the program is not always faster.
5. The relationship between the size of the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Mutation Rate (pm) is proportional to the processing time. This indicates that the larger the Mutation Rate (pm) then the time used to run the program is also getting longer.

6. The relationship between Maximum Generation size is proportional to the processing time. This indicates that the larger the size of the Maximum Generation then the time used to run the program is also longer.
7. The opportunity to discover the secret key *p* and *q* of the RSA algorithm is influenced by the size of Pop Size and Maximum Generation size. The bigger the size of the Pop Size and Maximum Generation size, the greater the opportunity to be found the secret key *p* and *q*.
8. Size Crossover Rate (pc) is best used to resolve the problem of solving the RSA public key using the genetic algorithm is 25%-50%.
9. Size Mutation Rate (pm) is best used to resolve the problem of solving the RSA public key using a genetic algorithm is 10%-20%.

REFERENCES

[1] Munir, Rinaldi. "*Serangan (Attack) Terhadap Kriptografi*". Bahan Kuliah Ke-2 IF5054 Kriptografi. Program Studi Teknik Informatika Institut Teknologi Bandung. 2004.

[2] Bhakti, Zeni Fera. "*Kriptoanalisis RSA dengan Kurva Eliptik*". Skripsi. 2010.

[3] Candra, Ade., et al. "On Factoring the RSA Modulus Using Tabu Search." *Journal of Computing and Applied Informatics (JoCAI) Vol. 01, No. 1, 2017, 11-21*. 2017.

[4] Ariyus, Dony. "*Pengantar Ilmu Kriptografi: Teori, Analisis, dan Implementasi*". Yogyakarta: Penerbit Andi. 2008.

[5] Scheiner, Bruce. "*Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C*." United States of America: Wiley. 1996.

[6] Mollin, Richard A. "*An Introduction to Cryptography Second Edition*." United States of America: Taylor and Francis Group, LLC. 2007.

[7] Abdurrahman, Dudung. "*Metode Penelitian Sejarah*". Jakarta: Logos Wacana Ilmu. 1999.

[8] Salhi, Saïd. "*Heuristic Search the Emerging Science of Problem Solving*." Canterbury, United Kingdom: Springer Nature. 2017.

[9] Desiani, Anita., Muhammad Arhami. "*Konsep Kecerdasan Buatan*". Yogyakarta: Penerbit Andi. 2006.

[10] M A Budiman et al 2018 J. Phys.: Conf. Ser. 978 012090

[11] Dian Rachmawati and Yeni Rosalin Munthe 2018 J. Phys.: Conf. Ser. 1090 012062

[12] Windarto, Eridani, U. D. Purwati, A comparison of continuous genetic algorithm and particle swarm optimization in parameter estimation of Gompertz growth model, AIP Conference Proceedings, 2084 (2019), 020017.